ORACLE UNIVERSITY

SQL Fundamentals

Activity Guide
X95174GC10
Edition 1.0 | May 2016

Learn more from Oracle University at **oracle.com/education/**

ORACLE

Authors

Apoorva Srinivas and Puja Singh

Technical Contributors and Reviewers

Nancy Greenberg, Suresh Rajan, Satyajit Ranganathan and Gururaj Bs

This book was published using: Oracle Tutor

Table of Contents

Practices for Lesson 1: Introduction

Chapter 1

Practices for Lesson 1

There are no practices for this lesson.

Practices for Lesson 2: Relational Database Overview

Chapter 2

Practices for Lesson 2: Overview

Practice Overview

In this practice, you learn about relational database concepts.

In some of the practices, there may be exercises that are prefaced with the phrases “If you have time” or “If you want an extra challenge.” Work on these exercises only if you have completed all the other exercises within the allocated time and would like an additional challenge to your skills.

Perform the practices slowly and precisely. There can be any number of solutions for the practices. You can experiment with saving and running command files. If you have any questions at any time, ask your instructor.

Practice 2-1: Relational Database Overview

Overview

This is the first of many practices in this course. The solutions (if you require them) can be found at the end of each practice. The practices are intended to cover most of the topics that are presented in the corresponding lesson.

In this practice, you learn to identify entities, attributes, and their corresponding tables, rows, and columns. You also learn to identify unique identifiers and the corresponding primary keys from the given scenarios.

Tasks

1. Match the ERD elements to their corresponding database elements.

Analysis	Design
1. Attribute	a. Column
2. Entity	b. Foreign key
3. ER Model	c. Physical design
4. Instance	d. Primary key
5. Primary UID	e. Row
6. Relationship	f. Table
7. Secondary UID	g. Unique key

2. The goal of this practice is to recognize attributes for an entity.

   The three entities—SONG, EVENT, and CUSTOMER—play a role in DJ business and are listed as the first three column headings in the following table. The fourth column contains a list of attributes. Use an X or a check mark to indicate that the attribute could belong to one or more of the entities listed. For example, could Title be an attribute for Song, for Event, and/or for Customer?

SONG	EVENT	CUSTOMER	
			Title
			Description
			Venue
			First Name
			Phone Number
			Release Date
			Last Name
			Type
			Email Address

3. For each entity, select the attribute that could be the entity's unique identifier.

   Entity: `STUDENT`
   Attributes: student ID, first name, last name, address

   Entity: `MOVIE`
   Attributes: title, date released, producer, director

   Entity: `LOCKER`
   Attributes: size, location, number

4. Identify the tables from the given scenario:

   Book.com is an online virtual store where customers can browse the catalog and select products of interest.

   a. Every book has a title, ISBN, year, and price. The store also keeps information about the author and publisher for each book.
   b. For authors, the database keeps the name, the address, and the URL of their home page.
   c. For publishers, the database keeps the name, address, phone number, and URL of their website.
   d. The store has several warehouses, each of which has a code, address, and phone number.
   e. Each warehouse stocks several books. A book may be stocked at multiple warehouses.
   f. The database records the number of copies of a book stocked at various warehouses.
   g. The bookstore keeps the name, address, email ID, and phone number of its customers.
   h. A customer owns several shopping carts. A shopping cart is identified by a Shopping_Cart_ID and contains several books.
   i. Some shopping carts may contain more than one copy of a book. The database records the number of copies of each book in any shopping cart.
   j. At the time of checkout, more information will be needed to complete the transaction. Usually, the customer will be asked to fill or select a billing address, a shipping address, a shipping option, and payment information such as a credit card number. An email notification is sent to the customer as soon as the order is placed.

Solution 2-1: Relational Database Overview

1. The ERD elements matched to their corresponding database elements are:

Analysis	Design
1. Attribute	a. Column
2. Entity	b. Table
3. ER Model	c. Physical design
4. Instance	d. Row
5. Primary UID	e. Primary Key
6. Relationship	f. Foreign Key
7. Secondary UID	g. Unique key

2. The goal of this practice is to recognize attributes for an entity.

   The three entities—`SONG`, `EVENT`, and `CUSTOMER`—play a role in DJ business and are listed as the first three column headings in the following table. The fourth column contains a list of attributes. Use an X or a check mark to indicate that the attribute could belong to one or more of the entities listed. For example, could Title be an attribute for Song, for Event, and/or for Customer?

SONG	EVENT	CUSTOMER	
X			Title
X	X		Description
	X		Venue
		X	First Name
		X	Phone Number
X			Release Date
		X	Last Name
X	X		Type
		X	Email Address

3. For each entity, select the attribute that could be the entity's unique identifier.

   Entity: `STUDENT`
   Attributes: student ID, first name, last name, address
   UID: student ID

   Entity: `MOVIE`
   Attributes: title, date released, producer, director
   UID: A combination of title and date released, or an artificial UID such as movie ID

   Entity: `LOCKER`
   Attributes: size, location, number
   UID: number

4. One possible solution for the given scenario is:

Table Name: BOOKS

Column	Datatype
Book_ID	VARCHAR2
Book_Name	VARCHAR2
Author_ID	VARCHAR2
Price	NUMBER
Publisher_ID	VARCHAR2

Table Name: PUBLISHER

Column	Datatype
Publisher_ID	VARCHAR2
Publisher_Name	VARCHAR2
Publisher_Address	VARCHAR2
Publisher_URL	VARCHAR2

Table Name: AUTHOR

Column	Datatype
Author_ID	VARCHAR2
Author_Name	VARCHAR2
Author_Address	VARCHAR2
Author_URL	NUMBER

Table Name: CUSTOMER

Column Name	Data type
Customer_ID	VARCHAR2
Customer_Name	VARCHAR2
Street_Address	VARCHAR2
City	VARCHAR2
Phone_Number	VARCHAR2
Credit_Card_Number	VARCHAR2
Email_Address	VARCHAR2

Table Name: CREDIT_CARD_DETAILS

Column Name	Data type
Credit_Card_Number	VARCHAR2
Credit_Card_Type	VARCHAR2
Expiry_Date	DATE

Table Name: ORDER_DETAILS

Column	Data type
Order_ID	NUMBER
Customer_ID	VARCHAR2
Shipping_Type	VARCHAR2
Date_of_Purchase	DATE
Shopping_Cart_ID	NUMBER

Table Name: PURCHASE_HISTORY

Column	Data type
Customer_ID	VARCHAR2
Order_ID	NUMBER

Table Name: SHIPPING_TYPE

Column	Data type
Shipping_Type	VARCHAR2
Shipping_Price	NUMBER

Table Name: WAREHOUSE

Column Name	Data type
Code	Number
Address	VARCHAR2
Phone	Number

Table Name: BOOK_STOCK

Column Name	Data type
Book_ID	Number
Code	Number
No_Of_Copies	Number

Table Name: SHOPPING_CART

Column	Data type
Shopping_Cart_ID	NUMBER
Book_ID	VARCHAR2
Date	DATE
Quantity	NUMBER

Note: Student solutions will not be this detailed at this point in the course.

Practices for Lesson 3: Database Storage Structures

Chapter 3

Practices for Lesson 3: Overview

Practice Overview

In this practice, you learn about database storage structures by solving a crossword puzzle and answering multiple-choice questions.

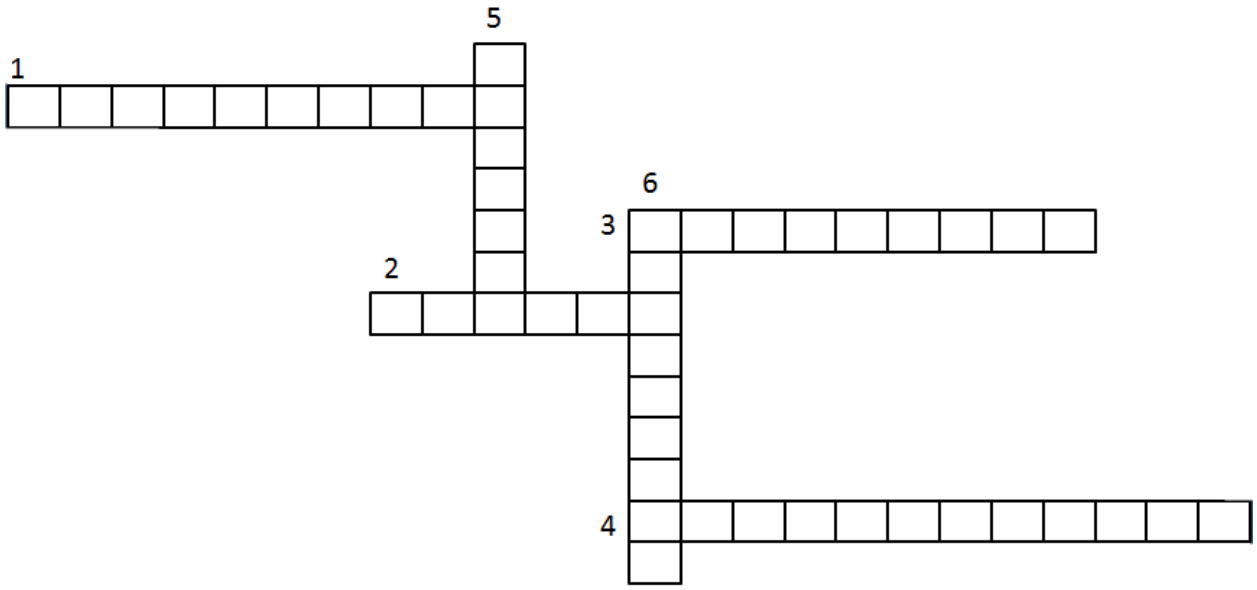
Practice 3-1: Database Storage Structures

Overview

In this practice, you use the clues provided to solve the crossword puzzle and answer some multiple-choice questions on the database storage structures discussed in the lesson.

Tasks

1. Solve the crossword.



ACROSS

1: The primary logical storage structures of any Oracle database

2: Logical unit of database storage space allocation made up of contiguous data blocks

3: The Oracle Database physically stores tablespace data here.

4: Small binary files that record the physical structure of the database

DOWN

5: A set of extents that have been allocated for a specific type of data structure

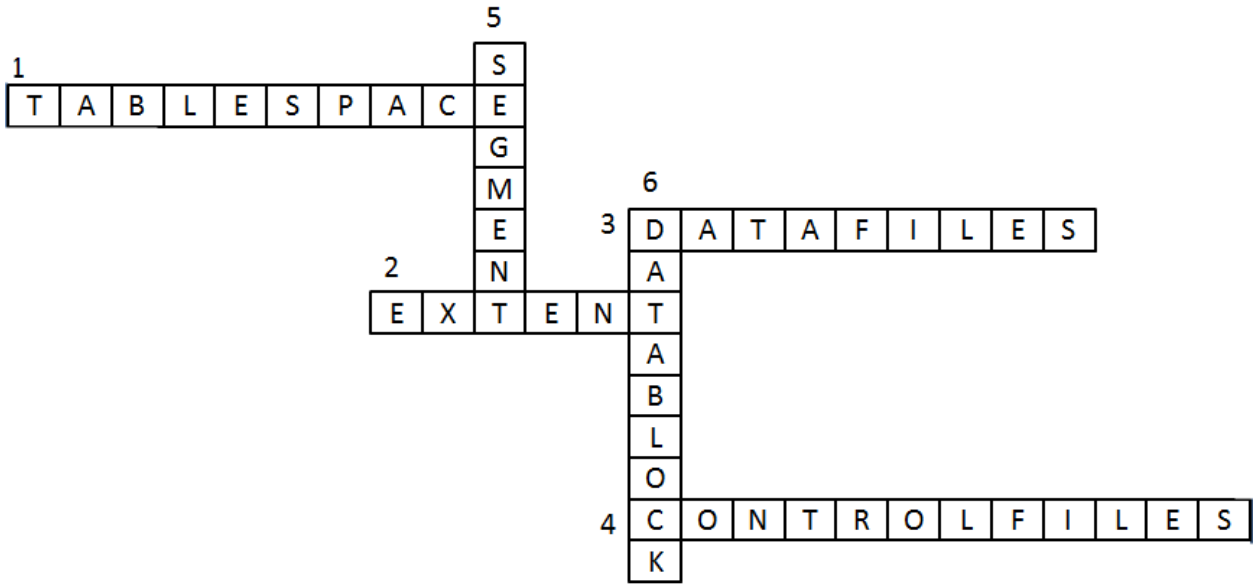
6: Smallest logical storage unit of a database

2. Select the best answer:
   - A single _____ represents a specific number of bytes on the physical hard disk.
     i. Segment
     ii. Data File
     iii. Data Block
     iv. Control File

- The first data block of every segment contains a directory of the _____ in the segment.
  i. Data Files
  ii. Extents
  iii. Control Files
  iv. Redo Files
- `SYSTEM` and `SYSAUX` are _____.
  i. Segments
  ii. Tablespaces
  iii. Data Files
  iv. Redo Logs
- _____ contain information about the database name and the database unique identifier (DBID).
  i. Data Files
  ii. Extents
  iii. Control Files
  iv. Redo Files

Solution 3-1: Database Storage Structures

1. The solution to the crossword puzzle based on the clues is provided as follows:



2. The solution is highlighted:
   a. A single _____ represents a specific number of bytes on the physical hard disk.
      i. Segment
      ii. Data File
      iii. Data Block
      iv. Control File
   b. The first data block of every segment contains a directory of the _____ in the segment.
      i. Data Files
      ii. Extents
      iii. Control Files
      iv. Redo Files
   c. `SYSTEM` and `SYSAUX` are _____.
      i. Segments
      ii. Tablespaces
      iii. Data Files
      iv. Redo Logs

d. _____ contain information about the database name and the database unique identifier (DBID).

i. Data Files
ii. Extents
iii. Control Files
iv. Redo Files

Practices for Lesson 4: Introduction to SQL

Chapter 4

Practices for Lesson 4: Overview

Practice Overview

In these practices, you identify information resources for SQL Developer, execute SQL statements by using SQL Developer, and examine data in the class schema. Specifically, you:

- Install and start SQL Developer
- Create a new database connection
- Browse the Academic (AD) schema tables
- Set a SQL Developer preference

Note

- All written practices use Oracle SQL Developer as the development environment. Although it is recommended that you use Oracle SQL Developer, you can also use SQL*Plus that is available in this course.
- For any query, the sequence of rows retrieved from the database may differ from the screenshots shown.

Practice 4-1: Accessing SQL Developer Resources

Overview

In this practice, you view a demonstration on introduction to the SQL Developer interface. Also, you navigate to the SQL Developer home page and browse helpful information about the tool.

Tasks

1. Access the SQL Developer home page.
   a. Access the online SQL Developer Home Page, which is available at: http://www.oracle.com/technology/products/database/sql_developer/index.html
   b. Bookmark the page for easier access in future.
2. Access the SQL Developer tutorial, which is available online at: http://www.oracle.com/webfolder/technetwork/tutorials/obe/db/sqldev/r40/sqldev4.0_GS/sqldev4.0_GS.html

   Review the following sections and associated demonstrations:
   a. Overview
   b. Creating a Database Connection
   c. Accessing Data

Solution 4-1: Accessing SQL Developer Resources

1. Access the SQL Developer home page.
   a. Access the online SQL Developer Home Page, which is available at: http://www.oracle.com/technology/products/database/sql_developer/index.html

   Note: The screenshots in this course reflect the 4.1.3 version of SQL Developer. However, the online SQL Developer Home Page points to the latest version of SQL Developer that is available for download.

   b. Bookmark the page for easier access in future.

2. Access the SQL Developer tutorial, which is available online at: http://www.oracle.com/webfolder/technetwork/tutorials/obe/db/sqldev/r40/sqldev4.0_GS/sqldev4.0_GS.html

   Then review the following sections and associated demos:

   a. Overview

   b. Creating a Database Connection

Practice 4-2: Installing SQL Developer

Overview

In this practice, you install the latest version of SQL Developer on your local machine. If you already have SQL Developer 4.1.3 installed on your machine, this practice need not be executed.

Assumptions

- The latest version of SQL Developer 4.1.3 is not yet installed on your local machine.

Tasks

1. Access the SQL Developer Home Page.
   a. Access the SQL Developer Home Page, which is available at: http://www.oracle.com/technology/products/database/sql_developer/index.html
   b. Click Download, to download the latest version of SQL Developer.
2. Install the SQL Developer on your local machine.

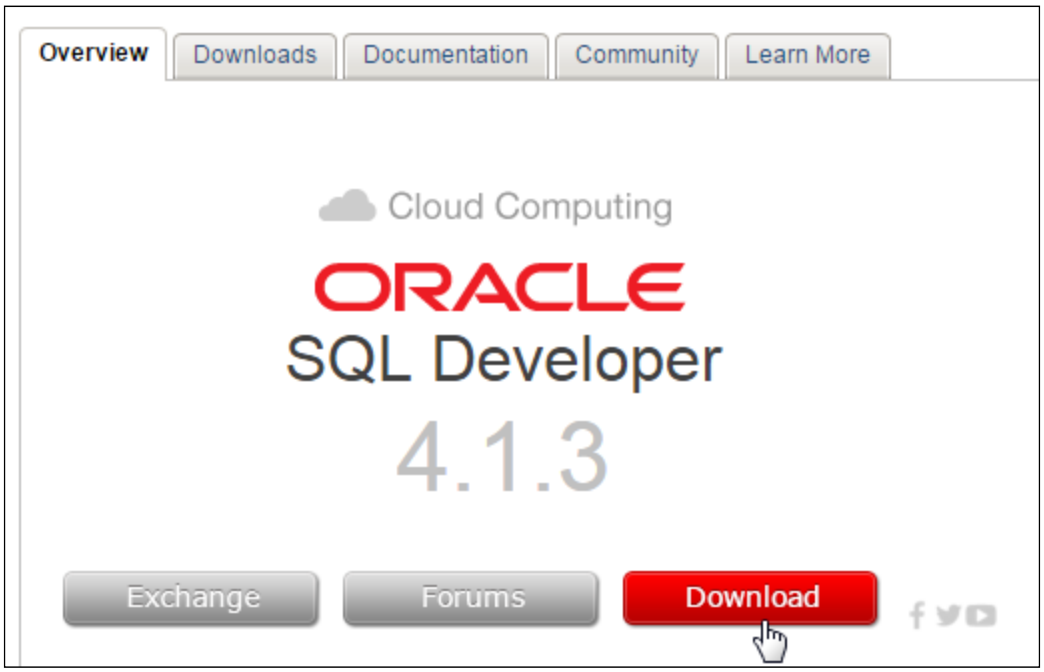
Solution 4-2: Installing SQL Developer

Overview

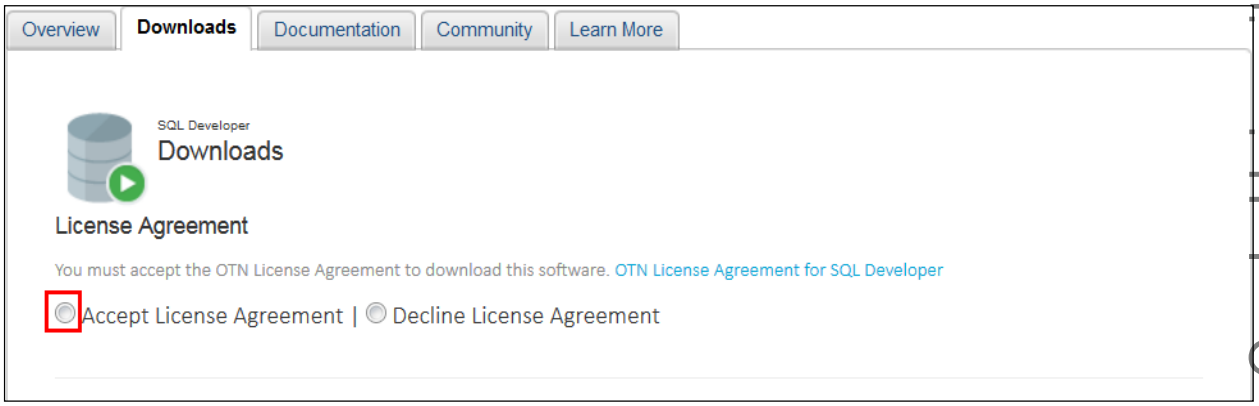
In this solution, you install the latest version of SQL Developer on your local machine. If you already have SQL Developer 4.1.3 installed on your machine, this solution need not be executed.

Steps

1. Access the SQL Developer Home Page.
   a. Access the SQL Developer Home Page, which is available at: http://www.oracle.com/technology/products/database/sql_developer/index.html
   b. Click **Download**, to download the latest version of SQL Developer.

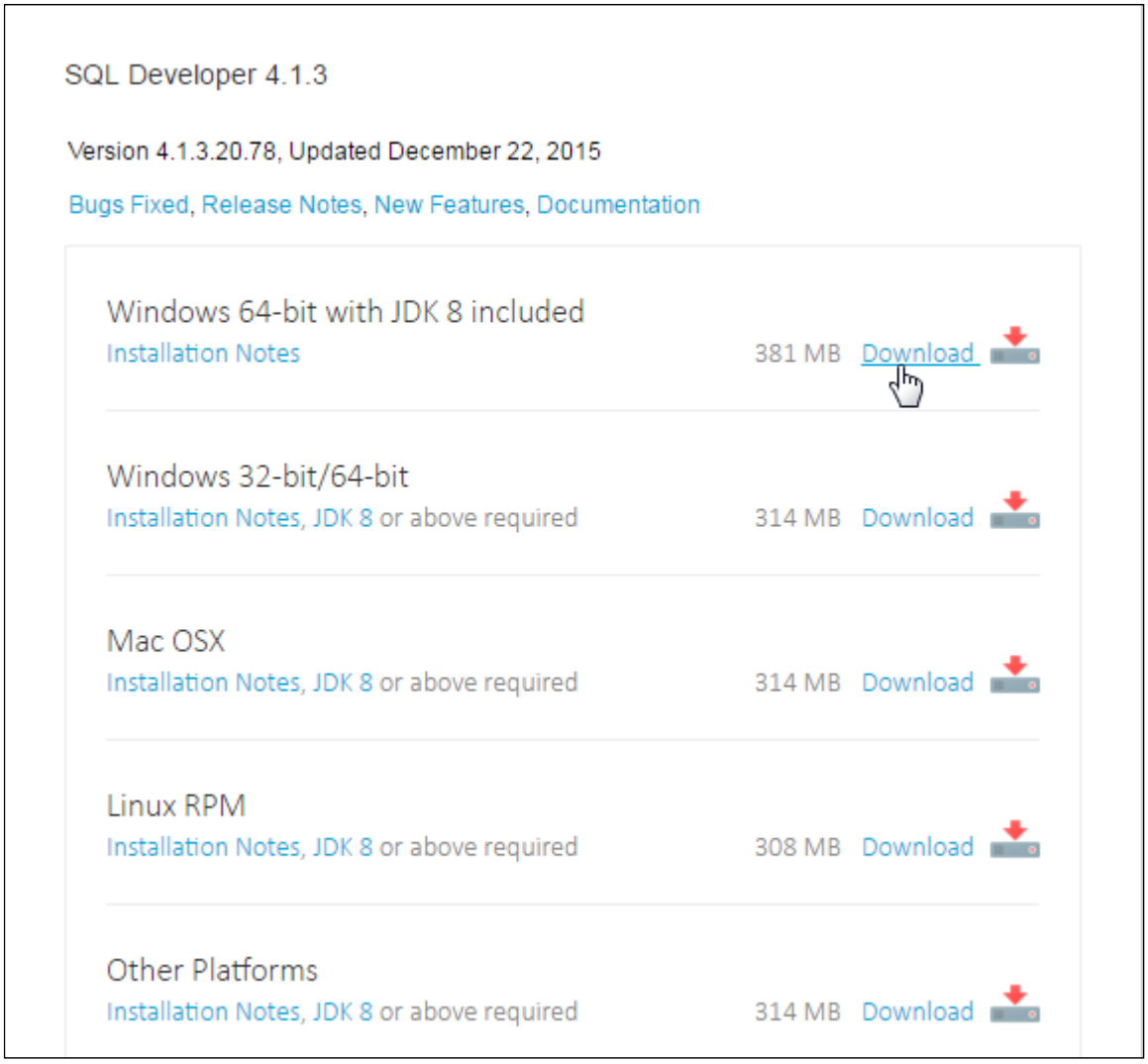


   c. Select **Accept License Agreement.**



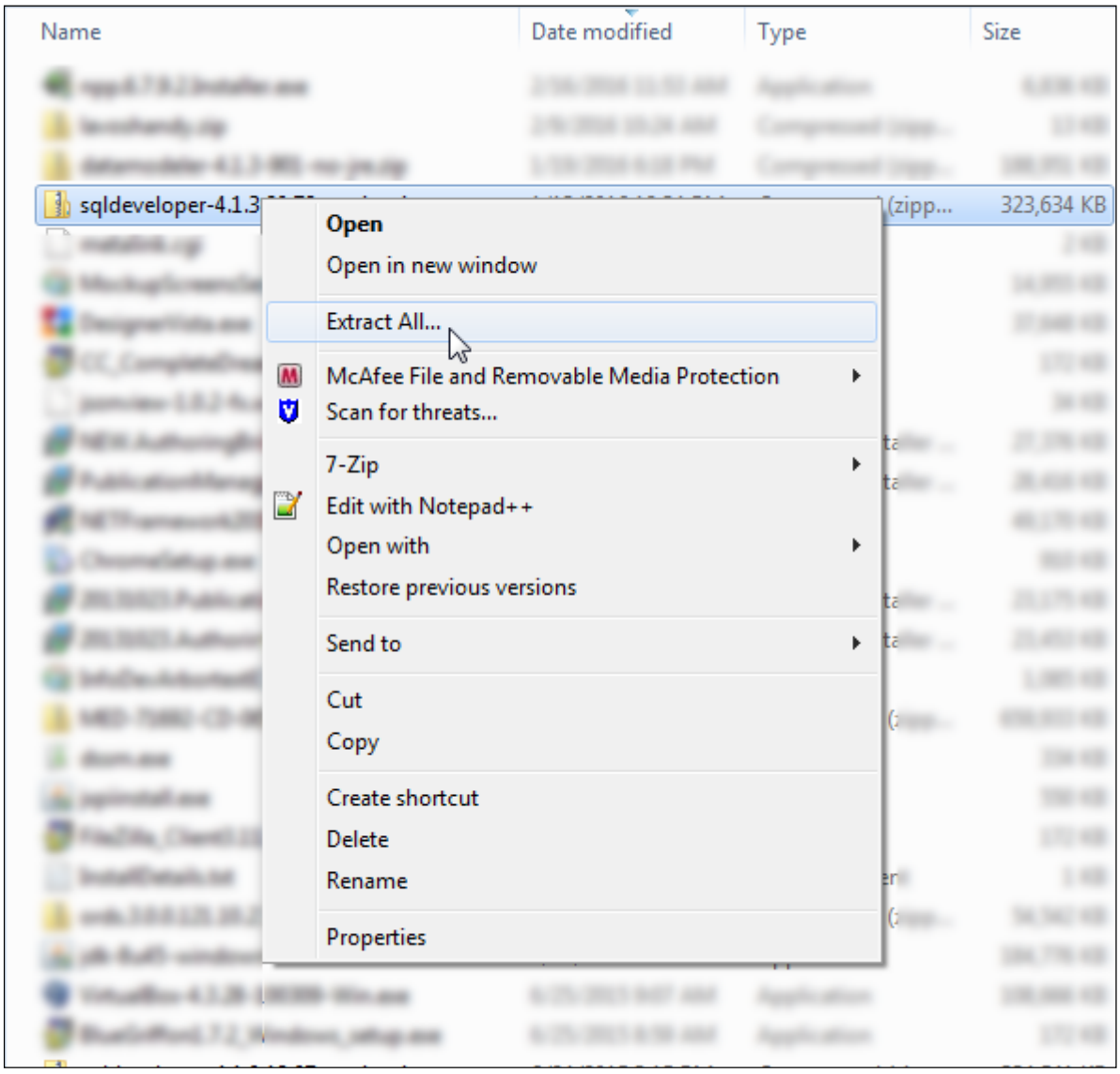
d. Click the **Download** link provided for the Operating System that is applicable to your local machine.

Note: If you are downloading for a Windows 64-bit machine, you can choose the appropriate download link depending on whether you have JDK 8 installed in your system or not.



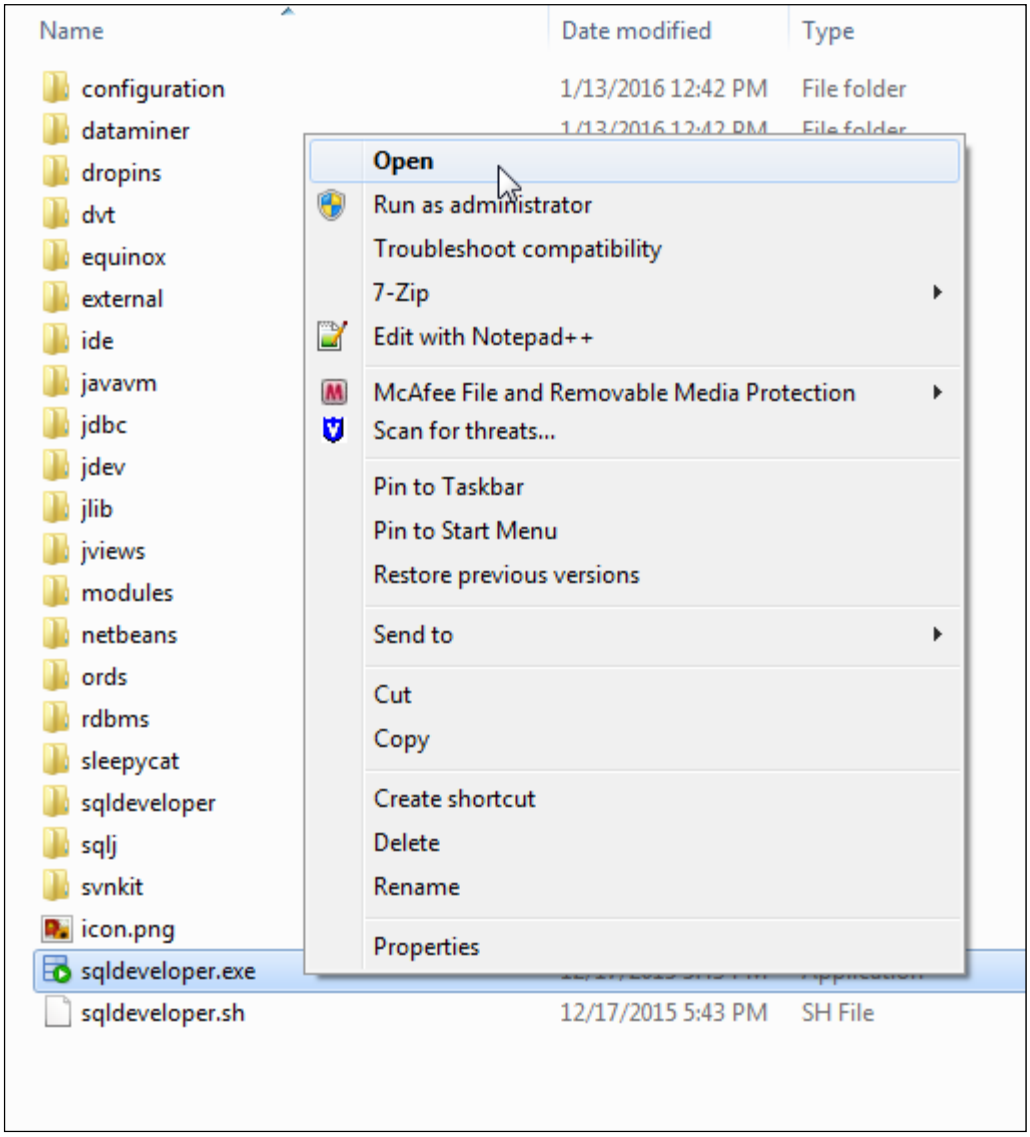
e. Browse to a folder on your machine, where you want to save the downloaded .zip file, and click **OK**.

2. Install SQL Developer on your local machine.
   a. Browse to the folder where you downloaded the SQL Developer software in Step 1, and extract the files.

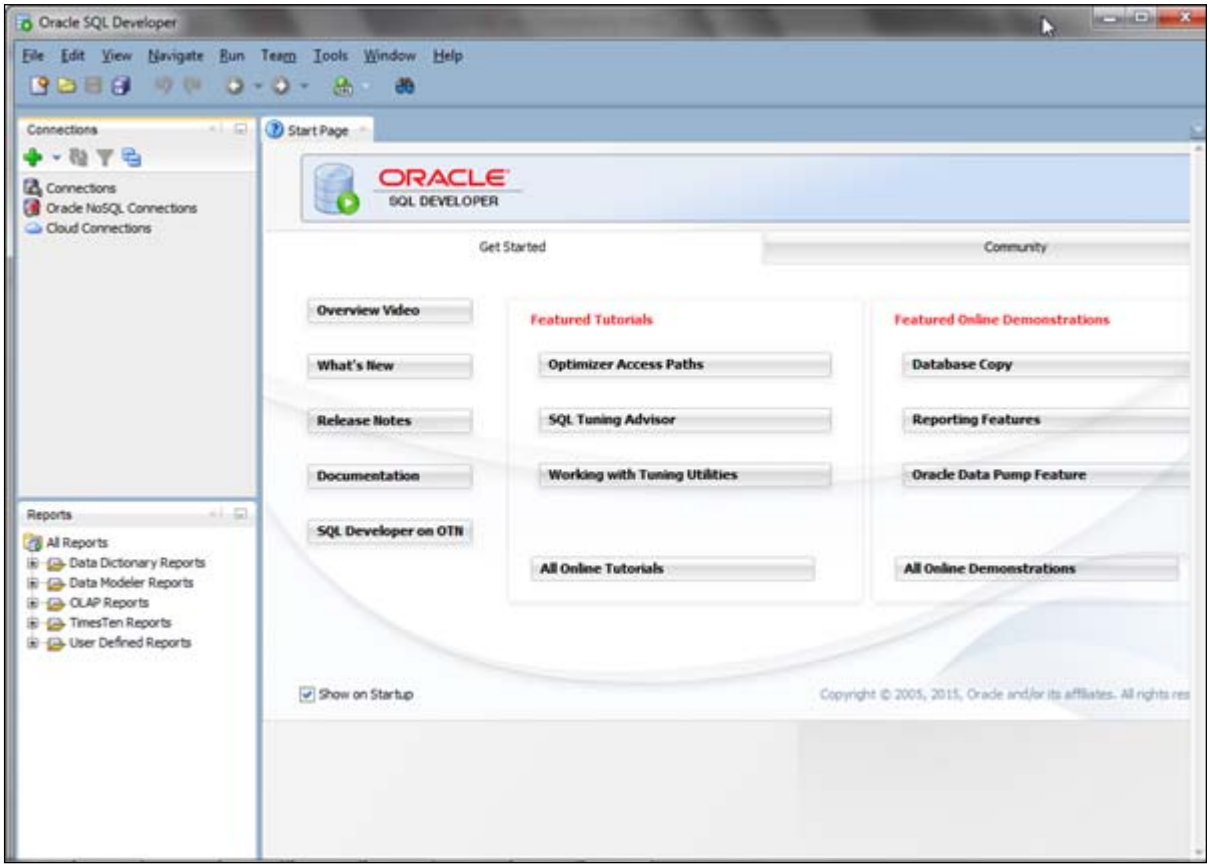


   b. Open the extracted folder, locate a folder named **sqldeveloper**, and open it.

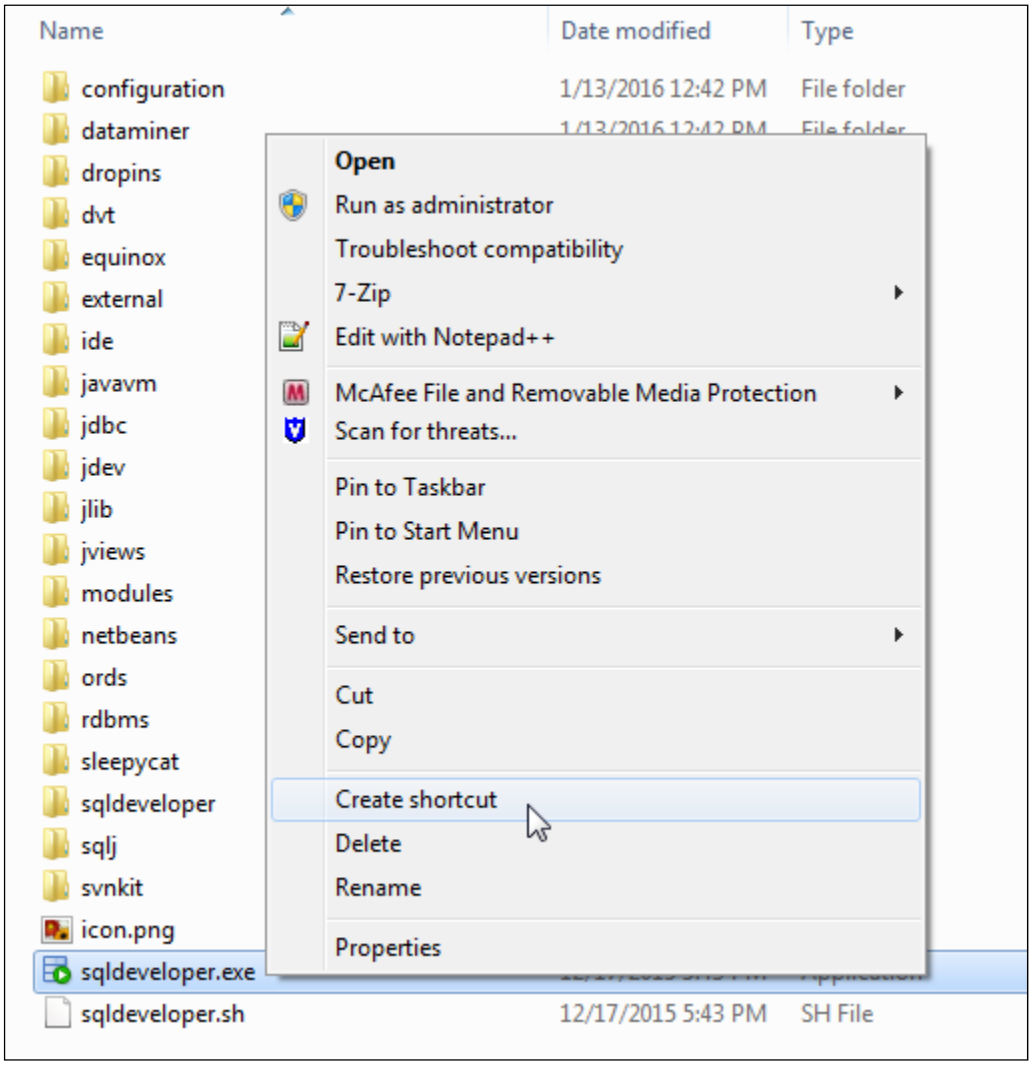
c. Locate sqldeveloper.exe, right-click **sqldeveloper.exe**, and select **Open**.



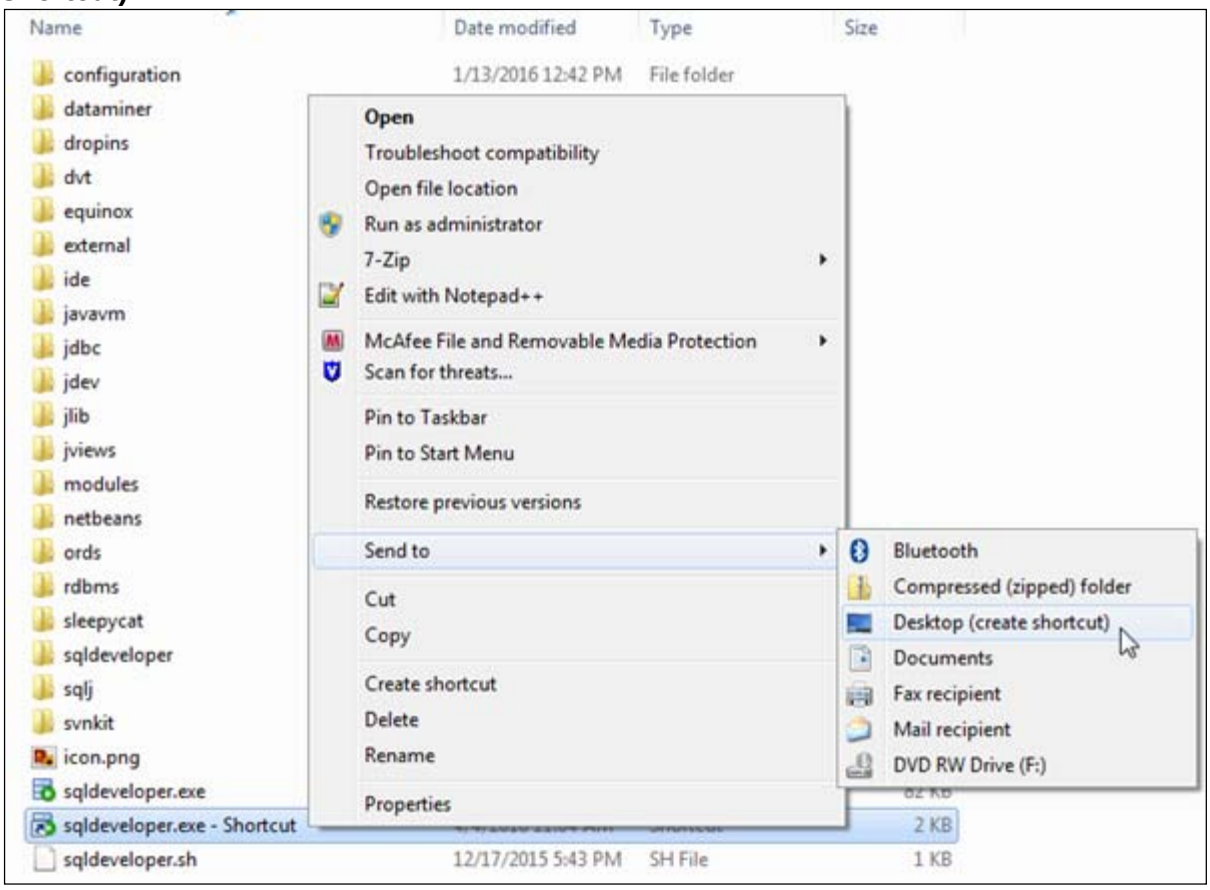
d. SQL Developer opens with a welcome page as follows:



e. For easier access in future, you can add a shortcut to SQL Developer on your desktop. Go back to the **sqldeveloper** folder again, right-click **sqldeveloper.exe**, and select **Create Shortcut**.



f. Right-click **sqldeveloper.exe – Shortcut**, select **Send to**, and select **Desktop (create shortcut)**.



Practice 4-3: Getting Started

1. Start SQL Developer.
2. Create a database connection by using the following information (**Hint:** Select the Save Password check box.):
   a. Connection Name: MyConnection
   b. Username: `ora<n>` (Replace `<n>` with the value in your username.)
   c. Password: `ora<n>` (Replace `<n>` with the value in your username.)
   d. Hostname: <`The Public IP address for your Database Cloud service instance`>
   e. Port: 1521
   f. Service name: `PDB1.<Identity-Domain>.oraclecloud.internal`
3. Test the new connection. If the Status is Success, connect to the database by using this new connection.
   a. In the Database Connection window, click the Test button.
      Note: The connection status appears in the lower-left corner of the window.
   b. If the Status is Success, click the Connect button.
4. Browse the tables in the Connections Navigator.
   a. In the Connections Navigator, view the objects that are available to you in the Tables node. Verify that the following tables are present:
      `AD_STUDENT_COURSE_DETAILS`
      `AD_STUDENT_DETAILS`
      `AD_STUDENT_ATTENDANCE`
      `AD_PARENT_INFORMATION`
      `AD_COURSE_DETAILS`
      `AD_DEPARTMENT`
      `AD_EXAM_DETAILS`
      `AD_EXAM_TYPE`
      `AD_EXAM_RESULTS`
      `AD_ACADEMIC_SESSION`
      `AD_FACULTY_DETAILS`
      `AD_FACULTY_LOGIN_DETAILS`
      `AD_FACULTY_COURSE_DETAILS`
5. Browse the structure of the `AD_STUDENT_ATTENDANCE` table and display its data.
   a. Expand the MyConnection connection by clicking the plus symbol next to it.
   b. Expand the Tables icon by clicking the plus symbol next to it.
   c. Display the structure of the `AD_STUDENT_ATTENDANCE` table.
6. Use the Data tab to view data in the `AD_STUDENT_ATTENDANCE` table.
   Note: Take a few minutes to familiarize yourself with the data, or refer to Appendix A, which provides the description and data for all the tables in the AD schema that you will use in this course.

7. Set your script pathing preference to `/home/oracle/labs/sql`.
   a. Select Tools > Preferences > Database > Worksheet.
   b. Enter the value in the “Select default path to look for scripts” field.

Solution 4-3: Getting Started

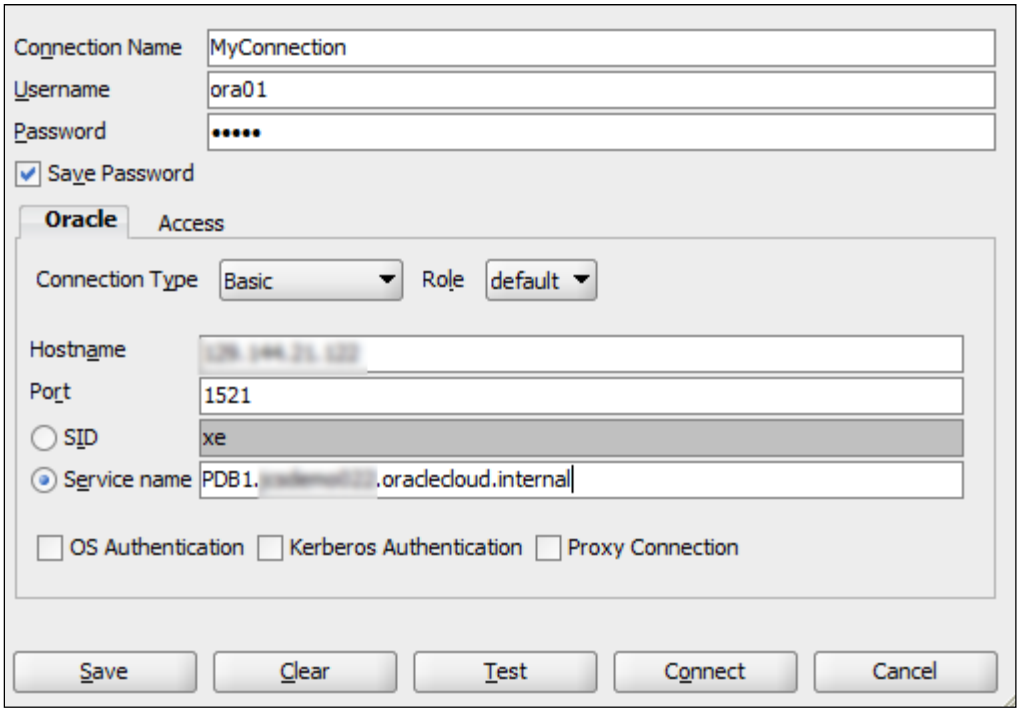
1. Start SQL Developer.
   Click the SQL Developer icon on your desktop.

2. Create a database connection by using the following information (**Hint:** Select the Save Password check box.):
   a. Connection Name: MyConnection
   b. Username: `ora<n>`
   c. Password: `ora<n>`
   d. Hostname: <`The Public IP address for your Database Cloud service instance>`
   e. Port: 1521
   f. Service name: `PDB1.<Identity-Domain>.oraclecloud.internal`

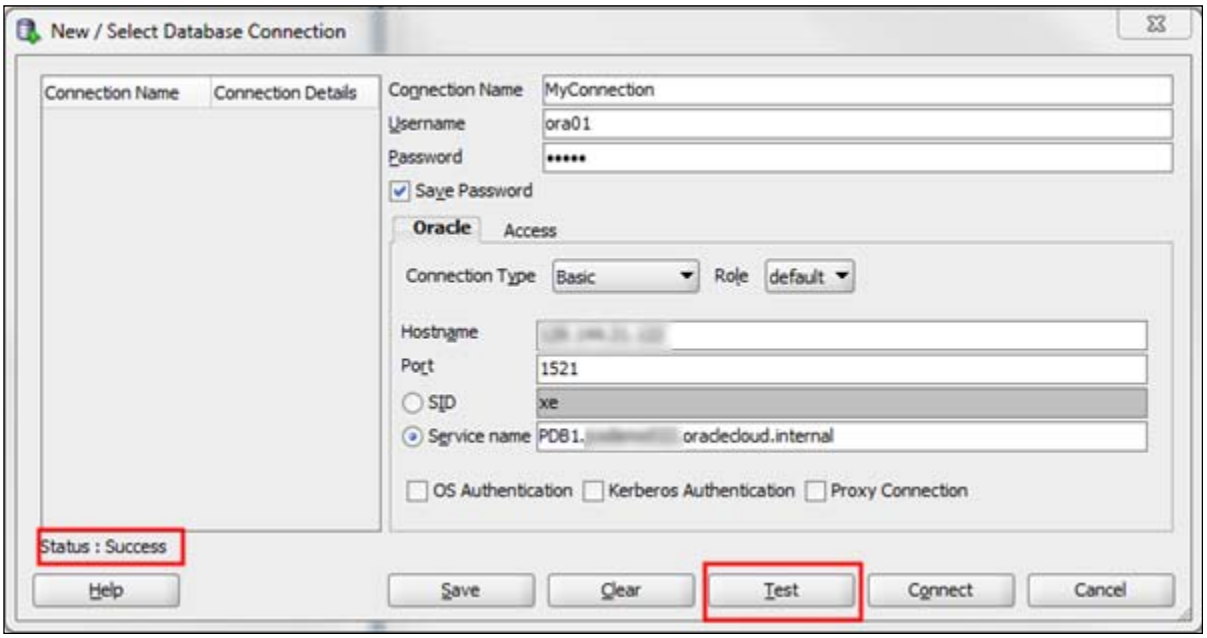
Right-click the Connections node on the Connections tab and select **New Connection**.

Result: The New/Select Database Connection window appears.

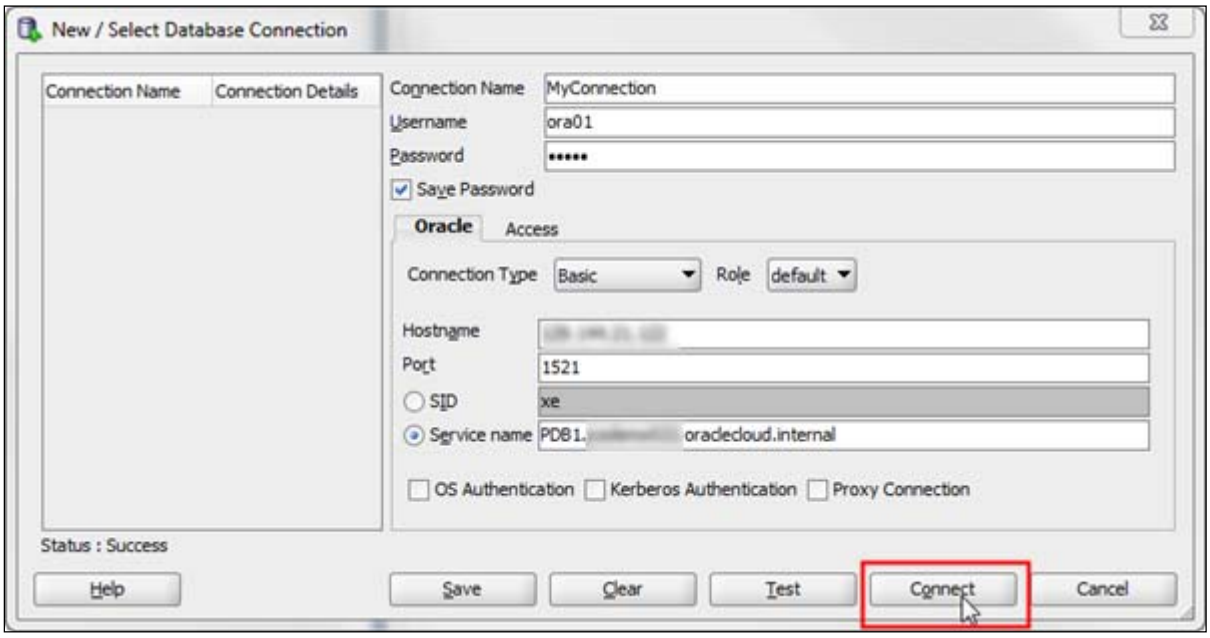
Use the preceding information to create the new database connection. In addition, select the Save Password check box, for example:



3. Test the new connection. If the Status is Success, connect to the database by using this new connection.
   a. In the Database Connection window, click the Test button.
      Note: The connection status appears in the lower-left corner of the window.

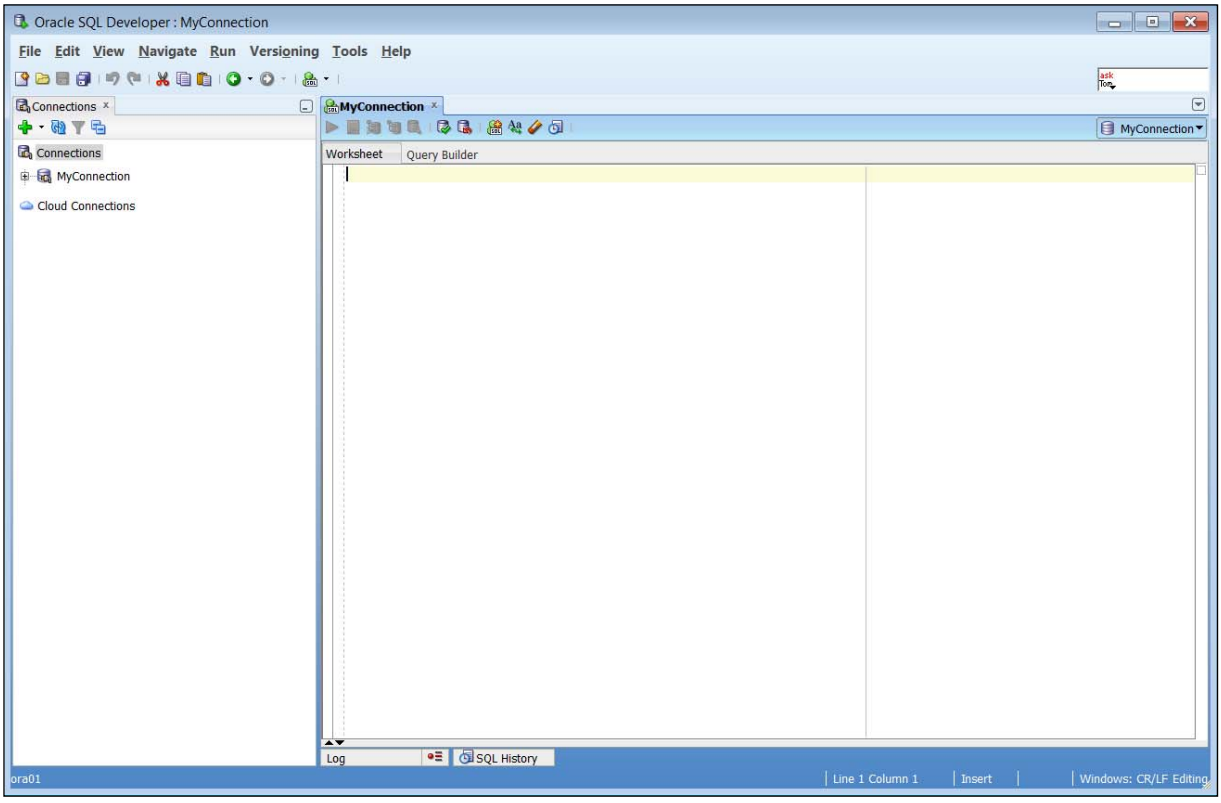


   b. If the Status is Success, click the Connect button.

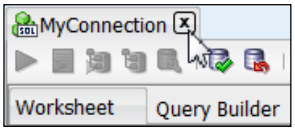


Note: To display the properties of an existing connection, right-click the connection name on the Connections tab and select Properties from the shortcut menu.

When you create a connection, a SQL Worksheet for that connection opens automatically.

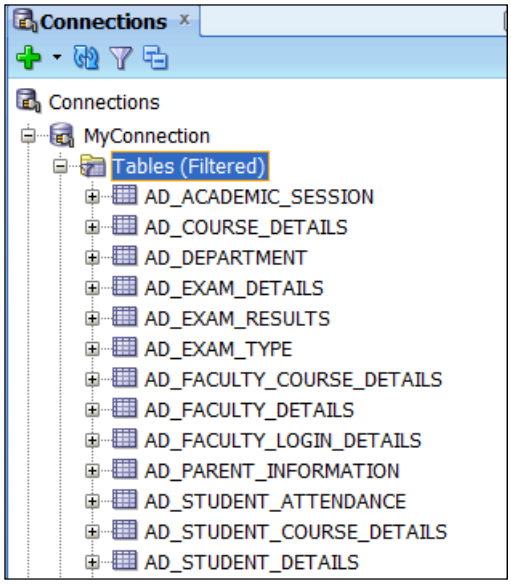


To close any SQL Worksheet tab, click X on the tab, as follows:



4. Browse the tables in the Connections Navigator.
   a. In the Connections Navigator, view the objects that are available to you in the Tables node. Verify that the following tables are present:

      AD_STUDENT_COURSE_DETAILS
      AD_STUDENT_DETAILS
      AD_STUDENT_ATTENDANCE
      AD_PARENT_INFORMATION
      AD_COURSE_DETAILS
      AD_DEPARTMENT
      AD_EXAM_DETAILS
      AD_EXAM_TYPE
      AD_EXAM_RESULTS
      AD_ACADEMIC_SESSION
      AD_FACULTY_DETAILS
      AD_FACULTY_LOGIN_DETAILS
      AD_FACULTY_COURSE_DETAILS

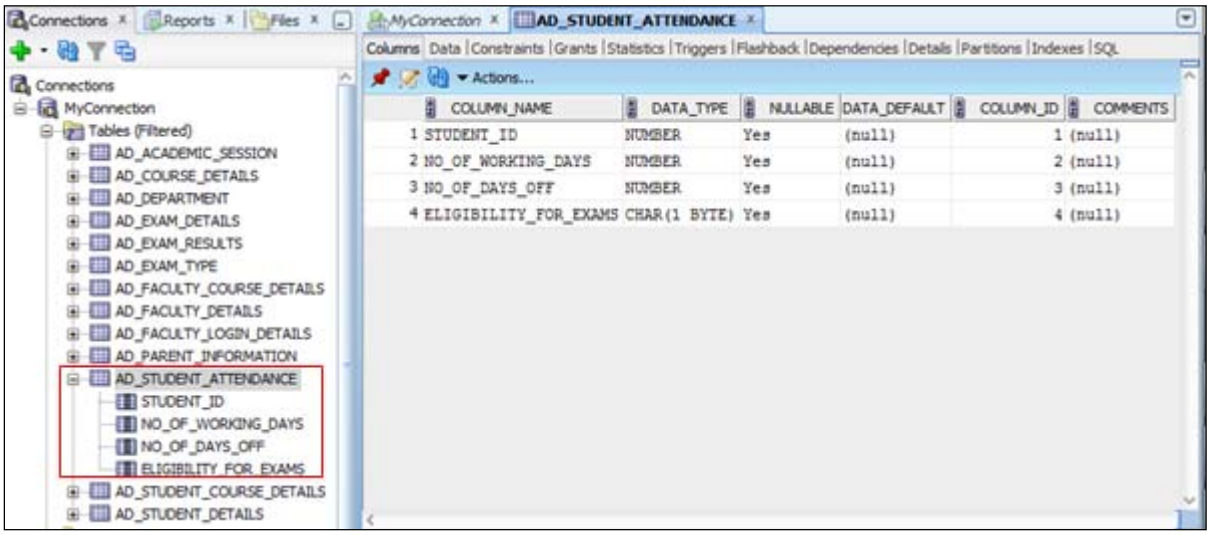


5. Browse the structure of the `AD_STUDENT_ATTENDANCE` table and display its data.
   a. Expand the MyConnection connection by clicking the plus symbol next to it.
   b. Expand Tables by clicking the plus symbol next to it.
   c. Display the structure of the `AD_STUDENT_ATTENDANCE` table.

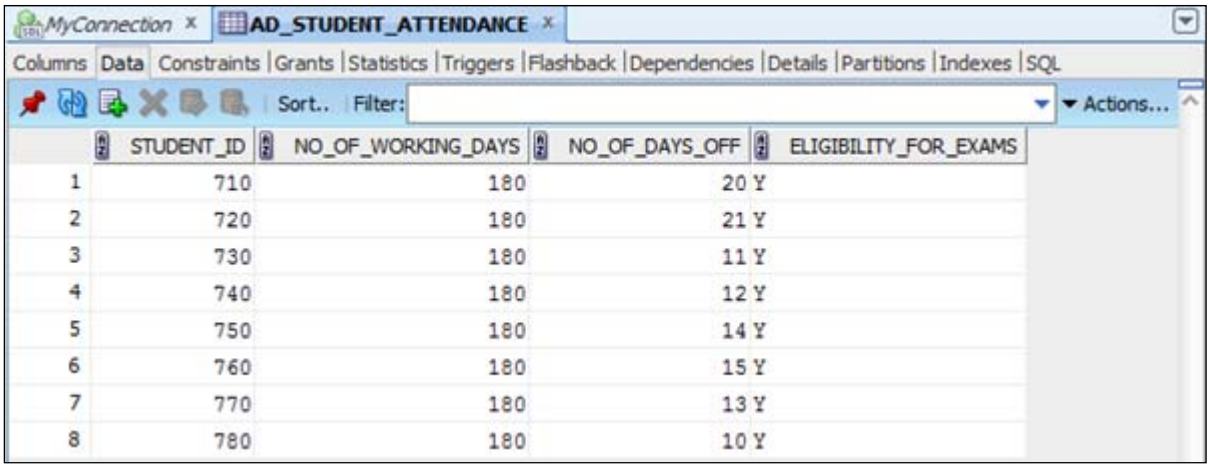
   Drill down on the `AD_STUDENT_ATTENDANCE` table by clicking the plus symbol next to it.

   Click the `AD_STUDENT_ATTENDANCE` table.

   Result: The Columns tab displays the columns in the `AD_STUDENT_ATTENDANCE` table as follows:

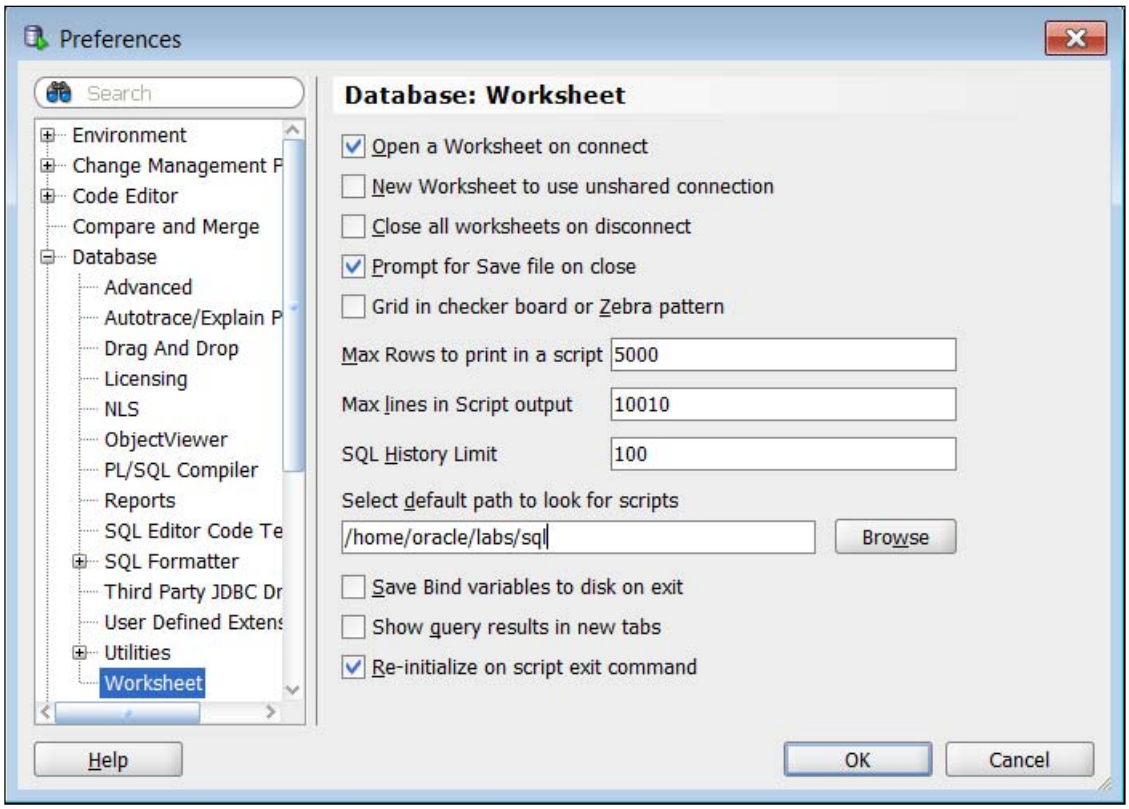


6. Use the Data tab to view the data in the `AD_STUDENT_ATTENDANCE` table.
   Result: The `AD_STUDENT_ATTENDANCE` table data is displayed as follows:



	STUDENT_ID	NO_OF_WORKING_DAYS	NO_OF_DAYS_OFF	ELIGIBILITY_FOR_EXAMS
1	710	180	20	Y
2	720	180	21	Y
3	730	180	11	Y
4	740	180	12	Y
5	750	180	14	Y
6	760	180	15	Y
7	770	180	13	Y
8	780	180	10	Y

7. Set your script pathing preference to `/home/oracle/labs/sql`.
   a. Select Tools > Preferences > Database > Worksheet.
   b. Enter the value in the “Select default path to look for scripts” field, and then, click OK.
      Note: To view the number of rows selected, enable the feedback option and set it to 1.
      ```
      set feedback on;
      set feedback 1;
      ```



Practices for Lesson 5: Retrieving Data by Using the SQL `SELECT` Statement

Chapter 5

Practices for Lesson 5: Overview

Practice Overview

This practice covers the following topics:

- Selecting all data from different tables
- Describing the structure of tables
- Performing arithmetic calculations and specifying column names

Practice 5-1: Retrieving Data by Using the SQL SELECT Statement

Overview

In this practice, you write simple SELECT queries. The queries cover most of the SELECT clauses and operations that you learned in the lesson.

Task 1

Test your knowledge:

1. The following SELECT statement executes successfully:

```
SELECT student_id, first_name, student_reg_year AS Admission
FROM   ad_student_details;
```

   True/False

2. The following SELECT statement executes successfully:

```
SELECT *
FROM   ad_course_details;
```

   True/False

3. There are four coding errors in the following statement. Can you identify them?

```
SELECT    student_id, first_name
Admission + 2  COURSE EXPIRY
FROM      ad_student_details;
```

Task 2

Note the following points before you begin with the practices:

- Save all your practice files at the following location:
  `/home/oracle/SQL_labs/labs`
- Enter your SQL statements in a SQL Worksheet. To save a script in SQL Developer, make sure that the required SQL Worksheet is active, and then from the File menu, select Save As to save your SQL statement as a `lab_<lessonno>_<stepno>.sql` script. When you modify an existing script, make sure that you use Save As to save it with a different file name.
- To run a query, click the Execute Statement icon in the SQL Worksheet. Alternatively, you can press F9. For DML and DDL statements, use the Run Script icon or press F5.
- After you have executed the query, make sure that you do not enter your next query in the same worksheet. Open a new worksheet.

You have been hired as a SQL programmer for Acme University. Your first assignment is to create some reports based on the data from the Academic tables.

4. Your first task is to determine the structure of the `AD_COURSE_DETAILS` table and its contents.

```
DESCRIBE AD_COURSE_DETAILS
Name          Null     Type
------------- -------- ------------
COURSE_ID     NOT NULL NUMBER
COURSE_NAME            VARCHAR2(50)
SESSION_ID             NUMBER
DEPARTMENT_ID          NUMBER
```

	COURSE_ID	COURSE_NAME	SESSION_ID	DEPARTMENT_ID
1	190	PRINCIPLES OF ACCOUNTING	100	10
2	191	INTRODUCTION TO BUSINESS LAW	100	10
3	192	COST ACCOUNTING	100	10
4	193	STRATEGIC TAX PLANNING FOR BUSINESS	100	10
5	194	GENERAL BIOLOGY	200	20
6	195	CELL BIOLOGY	200	20
7	196	INTRODUCTION TO PLANT PHYSIOLOGY	200	20
8	197	MARINE BIOLOGY	200	20
9	198	SIMULATION AND MODELING	300	30
10	199	WEB PROGRAMMING	300	30
11	187	DATA STRUCTURES	300	30
12	188	OOAD	300	30
13	189	COLLEGE READING	100	40
14	176	BUSINESS WRITING	200	40
15	175	AMERICAN LITERATURE	300	40

5. Your task is to determine the structure of the `AD_STUDENT_DETAILS` table and its contents.
   a. Determine the structure of the `AD_STUDENT_DETAILS` table.

```
DESCRIBE AD_STUDENT_DETAILS
Name             Null     Type
---------------- -------- -------------
STUDENT_ID       NOT NULL NUMBER
FIRST_NAME                VARCHAR2(50)
PARENT_ID                 NUMBER
STUDENT_REG_YEAR          DATE
EMAIL_ADDR                VARCHAR2(100)
```

   b. The University wants a query to display the student ID, first name, parent ID, and registration date for each student, with the student ID appearing first. Provide an alias `REGISTRATION` for the `STUDENT_REG_YEAR` column. Save your SQL statement to a file named `lab_05_5b.sql` so that you can dispatch this file to the respective department. Test your query in the `lab_05_5b.sql` file to ensure that it runs correctly.

Note: After you have executed the query, make sure that you do not enter your next query in the same worksheet. Open a new worksheet.

	STUDENT_ID	FIRST_NAME	PARENT_ID	REGISTRATION
1	720	JACK	600	01-JAN-14
2	740	RHONDA	620	01-SEP-14
3	750	ROBERT	610	01-MAR-14
4	760	JEANNE	610	01-MAR-14
5	770	MILLS	630	01-APR-15
6	710	NINA	630	01-JAN-13
7	780	NATHAN	640	01-JAN-16
8	730	NOAH	640	01-JUN-14

6. The University wants a query to display all unique exam names from the `AD_EXAM_DETAILS` table.

	NAME
1	SPRING SESSION EXAM
2	SUMMER SESSION EXAM
3	FALL SESSION EXAM

Task 3

If you have time, complete the following exercises:

7. The University wants more descriptive column headings for its report on students. Copy the statement from `lab_05_5b.sql` to a new SQL Worksheet. Name the columns `Student #`, `Student`, `Parent Information`, and `Registered On`, respectively. Then run the query again.

	Student #	Student	Parent Information	Registered On
1	720	JACK	600	01-JAN-14
2	740	RHONDA	620	01-SEP-14
3	750	ROBERT	610	01-MAR-14
4	760	JEANNE	610	01-MAR-14
5	770	MILLS	630	01-APR-15
6	710	NINA	630	01-JAN-13
7	780	NATHAN	640	01-JAN-16
8	730	NOAH	640	01-JUN-14

8. The University has requested a report of all courses and their course IDs. Display the course ID concatenated with the course name (separated by a comma and space) and name the column `Course ID and Title`.

	Course ID and Title
1	190, PRINCIPLES OF ACCOUNTING
2	191, INTRODUCTION TO BUSINESS LAW
3	192, COST ACCOUNTING
4	193, STRATEGIC TAX PLANNING FOR BUSINESS
5	194, GENERAL BIOLOGY
6	195, CELL BIOLOGY
7	196, INTRODUCTION TO PLANT PHYSIOLOGY
8	197, MARINE BIOLOGY
9	198, SIMULATION AND MODELING
10	199, WEB PROGRAMMING
11	187, DATA STRUCTURES
12	188, OOAD
13	189, COLLEGE READING
14	176, BUSINESS WRITING
15	175, AMERICAN LITERATURE

If you want an extra challenge, complete the following exercise:

9. To familiarize yourself with the data in the `AD_EXAM_DETAILS` table, create a query to display all the data from that table. Separate each column output by a comma. Name the column `THE_OUTPUT`.

	THE_OUTPUT
1	500,MCE,12-SEP-15,FALL SESSION EXAM
2	510,SA,15-SEP-15,FALL SESSION EXAM
3	520,LAB,18-SEP-15,FALL SESSION EXAM
4	530,ESS,21-MAR-16,SPRING SESSION EXAM
5	540,OB,25-MAR-16,SPRING SESSION EXAM
6	550,TF,02-APR-16,SPRING SESSION EXAM
7	560,FIB,26-MAY-16,SUMMER SESSION EXAM
8	570,SA,30-MAY-16,SUMMER SESSION EXAM

Solution 5-1: Retrieving Data by Using the SQL SELECT Statement

Task 1

Test your knowledge:

1. The following SELECT statement executes successfully:

```
SELECT student_id, first_name, student_reg_year AS Admission
FROM   ad_student_details;
```

True/False

2. The following SELECT statement executes successfully:

```
SELECT *
FROM   ad_course_details;
```

True/False

3. There are four coding errors in the following statement. Can you identify them?

```
SELECT    student_id, first_name
Admission '+' 2  COURSE EXPIRY
FROM      ad_student_details;
```

- **The AD_STUDENT_DETAILS table does not contain a column called Admission. The column is called STUDENT_REG_YEAR.**
- **The addition operator is + without quotes, not '+' as shown in line 2.**
- **The COURSE EXPIRY alias cannot include spaces. The alias should read COURSE_EXPIRY or should be enclosed within double quotation marks.**
- **A comma is missing after the FIRST_NAME column.**

Task 2

You have been hired as a SQL programmer for Acme University. Your first assignment is to create some reports based on the data from the Academic tables.

4. Your first task is to determine the structure of the AD_COURSE_DETAILS table and its contents.
   a. To determine the AD_COURSE_DETAILS table structure:

```
DESCRIBE AD_COURSE_DETAILS;
```

   b. To view the data contained in the AD_COURSE_DETAILS table:

```
SELECT *
FROM   AD_COURSE_DETAILS;
```

5. Your task is to determine the structure of the AD_STUDENT_DETAILS table and its contents.
   a. Determine the structure of the AD_STUDENT_DETAILS table.

```
DESCRIBE AD_STUDENT_DETAILS;
```

   b. The University wants a query to display the student ID, first name, parent ID, and registration date for each student, with the student ID appearing first. Provide an alias REGISTRATION for the STUDENT_REG_YEAR column. Save your SQL statement to a file named lab_05_5b.sql so that you can dispatch this file to the respective department. Test your query in the lab_05_5b.sql file to ensure that it runs correctly.

```
SELECT student_id, first_name, parent_id, student_reg_year
REGISTRATION
FROM AD_STUDENT_DETAILS;
```

6. The University wants a query to display all unique exam names from the AD_EXAM_DETAILS table.

```
SELECT DISTINCT NAME
FROM AD_EXAM_DETAILS;
```

Task 3

If you have time, complete the following exercises:

7. The University wants more descriptive column headings for its report on students. Copy the statement from lab_05_5b.sql to a new SQL Worksheet. Name the columns Student #, Student, Parent Information, and Registered On, respectively. Then run the query again.

```
SELECT student_id "Student #", first_name "Student",
       parent_id "Parent Information", student_reg_year
"Registered On"
FROM   AD_STUDENT_DETAILS;
```

8. The University has requested a report of all courses and their course IDs. Display the course ID concatenated with the course name (separated by a comma and space) and name the column Course ID and Title.

```
SELECT course_id||', '||course_name "Course ID and Title"
FROM   ad_course_details;
```

If you want an extra challenge, complete the following exercise:

9. To familiarize yourself with the data in the AD_EXAM_DETAILS table, create a query to display all the data from that table. Separate each column output by a comma. Name the column THE_OUTPUT.

```
SELECT exam_id || ',' || exam_type || ',' || start_date
       || ',' || name
       THE_OUTPUT
FROM   ad_exam_details;
```

Practices for Lesson 6: Restricting and Sorting Data

Chapter 6

Practices for Lesson 6: Overview

Practice Overview

This practice covers the following topics:

- Selecting data and changing the order of the rows that are displayed
- Restricting rows by using the `WHERE` clause
- Sorting rows by using the `ORDER BY` clause
- Using substitution variables to add flexibility to your SQL `SELECT` statements

Practice 6-1: Restricting and Sorting Data

Overview

In this practice, you build reports by using statements that use the `WHERE` clause and the `ORDER BY` clause. You make the SQL statements more reusable and generic by including the ampersand substitution.

Task

The University needs your assistance in creating some queries.

1. Because of shortage in attendance of students, the University needs a report that displays the student ID and number of days off for students who have been absent for more than 15 days. Save your SQL statement as a file named `lab_06_01.sql`. Run your query.

	STUDENT_ID	NO_OF_DAYS_OFF
1	710	20
2	720	21

2. Open a new SQL Worksheet. Create a report that displays the course name and department ID for course ID 199. Run the query.

	COURSE_NAME	DEPARTMENT_ID
1	WEB PROGRAMMING	30

3. The University needs to find information about high-scoring and low-scoring students. Create a report to display the student ID and marks for any student whose marks are not in the range 65 through 90. Save your SQL statement as `lab_06_03.sql`.

	STUDENT_ID	MARKS
1	720	91
2	720	97
3	750	60
4	750	97
5	760	60
6	770	91
7	770	99
8	710	91
9	780	56
10	780	61

4. Create a report to display the student ID, first name, and registration date for students with the first names of Robert and Nina. Order the query in ascending order by registration date.

	STUDENT_ID	FIRST_NAME	STUDENT_REG_YEAR
1	710	NINA	01-JAN-13
2	750	ROBERT	01-MAR-14

5. Display the course name and department ID of all courses in department 20 or 40 in ascending alphabetical order by `course_name`.

	COURSE_NAME	DEPARTMENT_ID
1	AMERICAN LITERATURE	40
2	BUSINESS WRITING	40
3	CELL BIOLOGY	20
4	COLLEGE READING	40
5	GENERAL BIOLOGY	20
6	INTRODUCTION TO PLANT PHYSIOLOGY	20
7	MARINE BIOLOGY	20

6. Modify `lab_06_03.sql` to display the student ID and marks of students whose marks are between 65 and 90, and have a course ID 199 or 189. Label the columns `Student #` and `Semester Marks`, respectively. Save `lab_06_03.sql` as `lab_06_06.sql` again. Run the statement in `lab_06_06.sql`.

	Student #	Semester Marks
1	740	76
2	750	85
3	730	87

7. The University needs a report that displays the first name and registration date of all students who registered in 2014.

	FIRST_NAME	STUDENT_REG_YEAR
1	JACK	01-JAN-14
2	RHONDA	01-SEP-14
3	ROBERT	01-MAR-14
4	JEANNE	01-MAR-14
5	NOAH	01-JUN-14

8. Create a report to display the first name and parent ID of all students who do not have an email address.

	FIRST_NAME	PARENT_ID
1	RHONDA	620
2	JEANNE	610
3	NATHAN	640

9. Create a report to display the first name, registration date, and email address of all students who have a valid email address. Sort the data in descending order of registration date and email address.
   Use the column's numeric position in the `ORDER BY` clause.

	FIRST_NAME	STUDENT_REG_YEAR	EMAIL_ADDR
1	MILLS	01-APR-15	mills@email.com
2	NOAH	01-JUN-14	noah@email.com
3	ROBERT	01-MAR-14	robert@email.com
4	JACK	01-JAN-14	jack@email.com
5	NINA	01-JAN-13	nina@email.com

10. Members of the University want to have more flexibility with the queries that you are writing. They would like a report that displays the student ID and marks of students who scored more than a number that the user specifies after a prompt. Save this query to a file named `lab_06_10.sql`. If you enter 75 when prompted, the report displays the following results:

	STUDENT_ID	MARKS
1	720	91
2	720	97
3	720	89
4	740	76
5	750	97
6	750	78
7	750	85
8	760	79
9	770	91
10	770	99
11	710	91
12	710	77
13	730	87
14	730	85

11. The University wants to run reports based on department. Create a query that prompts the user for a department ID, and generates the course ID, course name, and session ID for that department's courses. The University wants the ability to sort the report on a selected column. You can test the data with the following values:
    department_id = 30, sorted by course_name:

	COURSE_ID	COURSE_NAME	SESSION_ID
1	187	DATA STRUCTURES	300
2	188	OOAD	300
3	198	SIMULATION AND MODELING	300
4	199	WEB PROGRAMMING	300

department_id = 20, sorted by course_id:

	COURSE_ID	COURSE_NAME	SESSION_ID
1	194	GENERAL BIOLOGY	200
2	195	CELL BIOLOGY	200
3	196	INTRODUCTION TO PLANT PHYSIOLOGY	200
4	197	MARINE BIOLOGY	200

department_id = 40, sorted by session_id:

	COURSE_ID	COURSE_NAME	SESSION_ID
1	189	COLLEGE READING	100
2	176	BUSINESS WRITING	200
3	175	AMERICAN LITERATURE	300

If you have time, complete the following exercises:

12. Display the first names of all students where the second letter of the name is “O.”

	FIRST_NAME
1	ROBERT
2	NOAH

13. Display the first names of all students who have both an “a” and an “n” in their names.

	FIRST_NAME
1	RHONDA
2	JEANNE
3	NINA
4	NATHAN
5	NOAH

If you want an extra challenge, complete the following exercises:

14. Display the course ID and course name for all courses whose department IDs are either 10 or 40, and whose session IDs are not equal to 200 or 300.

	COURSE_ID	COURSE_NAME
1	190	PRINCIPLES OF ACCOUNTING
2	191	INTRODUCTION TO BUSINESS LAW
3	192	COST ACCOUNTING
4	193	STRATEGIC TAX PLANNING FOR BUSINESS
5	189	COLLEGE READING

15. Modify `lab_06_06.sql` to display the student ID, exam ID, course ID, and marks for all students whose marks are 70%. Save `lab_06_06.sql` as `lab_06_15.sql` again. Rerun the statement in `lab_06_15.sql`.

	Student #	Exam Code	Course Code	Score
1	740	570	197	70
2	760	510	192	70

Solution 6-1: Restricting and Sorting Data

The University needs your assistance in creating some queries.

1. Because of shortage in attendance of students, the University needs a report that displays the student ID and number of days off for students who have been absent for more than 15 days. Save your SQL statement as a file named `lab_06_01.sql`. Run your query.

```
SELECT  student_id, no_of_days_off
FROM    ad_student_attendance
WHERE   no_of_days_off > 15;
```

2. Open a new SQL Worksheet. Create a report that displays the course name and department ID for course ID 199. Run the query.

```
SELECT  course_name, department_id
FROM    ad_course_details
WHERE   course_id = 199;
```

3. The University needs to find information about high-scoring and low-scoring students. Create a report to display the student ID and marks for any student whose marks are not in the range 65 through 90. Save your SQL statement as `lab_06_03.sql`.

```
SELECT  student_id, marks
FROM    ad_exam_results
WHERE   marks NOT BETWEEN 65 AND 90;
```

4. Create a report to display the student ID, first name, and registration date for students with the first names of Robert and Nina. Order the query in ascending order by registration date.

```
SELECT   student_id, first_name, student_reg_year
FROM     ad_student_details
WHERE    first_name IN ('ROBERT', 'NINA')
ORDER BY student_reg_year;
```

5. Display the course name and department ID of all courses in department 20 or 40 in ascending alphabetical order by `course_name`.

```
SELECT   course_name, department_id
FROM     ad_course_details
WHERE    department_id IN (20, 40)
ORDER BY course_name ASC;
```

6. Modify `lab_06_03.sql` to display the student ID and marks of students whose marks are between 65 and 90, and have a course ID 199 or 189. Label the columns `Student #` and `Semester Marks`, respectively. Save `lab_06_03.sql` as `lab_06_06.sql` again. Run the statement in `lab_06_06.sql`.

```
SELECT   student_id "Student #", marks "Semester Marks"
FROM     ad_exam_results
WHERE    marks  BETWEEN 65 AND 90
AND      course_id IN (199, 189);
```

7. The University needs a report that displays the first name and registration date of all students who registered in 2014.

```
SELECT   first_name, student_reg_year
FROM     ad_student_details
WHERE    student_reg_year >= '01-JAN-14' AND student_reg_year <
'01-JAN-15';
```

8. Create a report to display the first name and parent ID of all students who do not have an email address.

```
SELECT   first_name, parent_id
FROM     ad_student_details
WHERE    email_addr IS NULL;
```

9. Create a report to display the first name, registration date, and email address of all students who have a valid email address. Sort the data in descending order of registration date and email address.
   Use the column's numeric position in the `ORDER BY` clause.

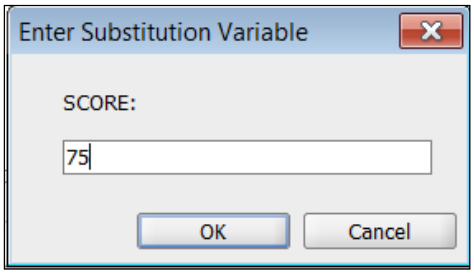
```
SELECT   first_name, student_reg_year, email_addr
FROM     ad_student_details
WHERE    email_addr IS NOT NULL
ORDER BY 2 DESC, 3 DESC;
```

10. Members of the University want to have more flexibility with the queries that you are writing. They would like a report that displays the student ID and marks of students who scored more than a number that the user specifies after a prompt. Save this query to a file named `lab_06_10.sql`.

    Enter `75` when prompted:

```
SELECT  student_id, marks
FROM    ad_exam_results
WHERE   marks > &score;
```

    Enter `75` when prompted for a value in a dialog box. Click OK.



11. The University wants to run reports based on department. Create a query that prompts the user for a department ID, and generates the course ID, course name, and session ID for that department's courses. The University wants the ability to sort the report on a selected column. You can test the data with the following values:

    manager_id = 30, sorted by course_name

    manager_id = 20, sorted by course_id

    manager_id = 40, sorted by session_id

```
SELECT course_id, course_name, session_id
FROM ad_course_details
WHERE department_id = &dept_num
ORDER BY &order_col;
```

If you have the time, complete the following exercises:

12. Display the first names of all students where the second letter of the name is "o."

```
SELECT   first_name
FROM     ad_student_details
WHERE    first_name LIKE '_O%';
```

13. Display the first names of all students who have both an "a" and an "n" in their names.

```
SELECT   first_name
FROM     ad_student_details
WHERE    first_name LIKE '%A%'
AND      first_name LIKE '%N%';
```

If you want an extra challenge, complete the following exercises:

14. Display the course ID and course name for all courses whose department IDs are either 10 or 40, and whose session IDs are not equal to 200 or 300.

```
SELECT   course_id, course_name
FROM     ad_course_details
WHERE    department_id IN (10, 40)
AND      session_id NOT IN (200, 300);
```

15. Modify `lab_06_06.sql` to display the student ID, exam ID, course ID, and marks for all students whose marks are 70%. Save `lab_06_06.sql` as `lab_06_15.sql` again. Rerun the statement in `lab_06_15.sql`.

```
SELECT   student_id "Student #", exam_id "Exam Code", course_id
         "Course Code", marks "Score"
FROM     ad_exam_results
WHERE    marks = 70;
```

Practices for Lesson 7: Using Single-Row Functions to Customize Output

Chapter 7

Practices for Lesson 7: Overview

Practice Overview

This practice covers the following topics:

- Writing a query that displays the current date
- Creating queries that require the use of numeric, character, and date functions
- Performing calculations of years and months of a course completed by a student

Practice 7-1: Using Single-Row Functions to Customize Output

Overview

This practice provides a variety of exercises by using the different functions that are available for the character, number, and date data types. Remember that for nested functions, the results are evaluated from the innermost function to the outermost function.

Tasks

1. Write a query to display the system date. Label the column `Date`.

   Note: If your database is remotely located in a different time zone, the output will be the date for the operating system on which the database resides.

	Date
1	18-FEB-16

2. The University needs a report to display the student ID, course ID, marks, and marks increased by 15.5% (expressed as a whole number) for each student. Label the column `New Score`. Save your SQL statement in a file named `lab_07_02.sql`.
3. Run your query in the `lab_07_02.sql` file.

	STUDENT_ID	COURSE_ID	MARKS	New Score
1	720	190	91	105
2	720	193	97	112
3	720	191	89	103
4	740	195	69	80
5	740	197	70	81
6	740	199	76	88
7	750	192	60	69
8	750	175	97	112
9	750	176	78	90
10	750	189	85	98
11	760	188	65	75
12	760	187	60	69
13	760	190	79	91
14	760	192	70	81
15	770	188	91	105
16	770	187	99	114
17	710	176	91	105
18	710	175	77	89
19	780	192	56	65
20	780	193	61	70
21	780	194	65	75
22	730	199	87	100
23	730	198	85	98

4. Modify your query in `lab_07_02.sql` to add a column that subtracts the old marks from the new marks. Label the column `Increase`. Save the contents of the file as `lab_07_04.sql`. Run the revised query.

	STUDENT_ID	COURSE_ID	MARKS	New Score	Increase
1	720	190	91	105	14
2	720	193	97	112	15
3	720	191	89	103	14
4	740	195	69	80	11
5	740	197	70	81	11
6	740	199	76	88	12
7	750	192	60	69	9
8	750	175	97	112	15
9	750	176	78	90	12
10	750	189	85	98	13
11	760	188	65	75	10
12	760	187	60	69	9
13	760	190	79	91	12
14	760	192	70	81	11
15	770	188	91	105	14
16	770	187	99	114	15
17	710	176	91	105	14
18	710	175	77	89	12
19	780	192	56	65	9
20	780	193	61	70	9
21	780	194	65	75	10
22	730	199	87	100	13
23	730	198	85	98	13

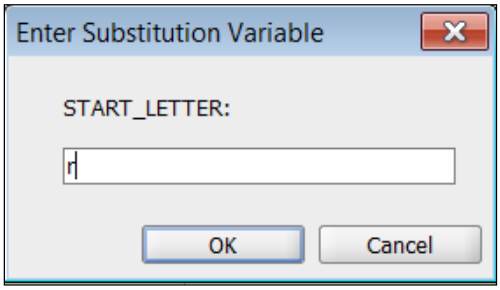
5. Perform the following tasks:
   a. Write a query that displays the first name (with the first letter in uppercase and all the other letters in lowercase) and the length of the first name for all students whose name starts with the letters “J,” “R,” or “M.” Give each column an appropriate label. Sort the results by the students’ first names.

	Name	Length
1	Jack	4
2	Jeanne	6
3	Mills	5
4	Rhonda	6
5	Robert	6

   b. Rewrite the query so that the user is prompted to enter the letter that the first name starts with. For example, if the user enters “`N`” (capitalized) when prompted for a letter, the output should show all students whose first name starts with the letter “N.”

	Name	Length
1	Nathan	6
2	Nina	4
3	Noah	4

c. Modify the query such that the case of the letter that is entered does not affect the output. The entered letter must be capitalized before being processed by the `SELECT` query.



	Name	Length
1	Rhonda	6
2	Robert	6

If you have time, complete the following exercises:

6. The University wants to find the number of months of a course completed by each student. For each student, display the first name and calculate the number of months between today and the date on which the student registered. Label the column as `MONTHS_COMPLETED`. Order your results by the number of months completed. The number of months must be rounded to the closest whole number.

Note: Because this query depends on the date when it is executed, the values in the `MONTHS_COMPLETED` column will differ for you.

	FIRST_NAME	MONTHS_COMPLETED
1	NATHAN	2
2	MILLS	11
3	RHONDA	18
4	NOAH	21
5	ROBERT	24
6	JEANNE	24
7	JACK	26
8	NINA	38

7. Create a query to display the exam name and exam type for all available exams. Format the exam type to be 15 characters long, left-padded with the * symbol. Label the column `EXAM_CODE`.

	EXAM_NAME	EXAM_CODE
1	Multiple Choice Exams	***********MCE
2	TRUE AND FALSE Exams	***********TF
3	FILL IN THE BLANKS Exams	***********FIB
4	ESSAY Exams	***********ESS
5	SHORT ANSWER Exams	***********SA
6	PROBLEM SOLVING Exams	***********PS
7	LAB Exams	***********LAB
8	OPEN BOOK Exams	***********OB

8. Create a query that displays students' IDs, and indicates the number of marks scored with asterisks. Each asterisk signifies 10 marks. Sort the data in descending order of marks. Label the column `STUDENTS_AND_THEIR_MARKS`.

	STUDENT_ID	STUDENTS_AND_THEIR_MARKS
1	770	********
2	720	********
3	750	********
4	720	********
5	710	********
6	770	********
7	720	*******
8	730	*******
9	750	*******
10	730	*******
11	760	******
12	750	******
13	710	******
14	740	******
15	760	******
16	740	******
17	740	*****
18	760	*****
19	780	*****
20	780	*****
21	750	*****
22	760	*****
23	780	****

9. Create a query to display the first name and the number of weeks enrolled into courses for all students whose email addresses are NULL. Label the number of weeks column as `WEEKS_COMPLETED`. Truncate the number of weeks value to 0 decimal places. Show the records in descending order of the student's tenure.

   Note: The `WEEKS_COMPLETED` value will differ because it depends on the date on which you run the query.

	FIRST_NAME	WEEKS_COMPLETED
1	JEANNE	102
2	RHONDA	76
3	NATHAN	7

Solution 7-1: Using Single-Row Functions to Customize Output

1. Write a query to display the system date. Label the column `Date`.

   Note: If your database is remotely located in a different time zone, the output will be the date for the operating system on which the database resides.

```
SELECT  sysdate "Date"
FROM    dual;
```

2. The University needs a report to display the student ID, course ID, marks, and marks increased by 15.5% (expressed as a whole number) for each student. Label the column `New Score`. Save your SQL statement in a file named `lab_07_02.sql`.

```
SELECT  student_id, course_id, marks,
        ROUND(marks * 1.155, 0) "New Score"
FROM    ad_exam_results;
```

3. Run your query in the file `lab_07_02.sql`.

```
SELECT  student_id, course_id, marks,
        ROUND(marks * 1.155, 0) "New Score"
FROM    ad_exam_results;
```

4. Modify your query in `lab_07_02.sql` to add a column that subtracts the old marks from the new marks. Label the column `Increase`. Save the contents of the file as `lab_07_04.sql`. Run the revised query.

```
SELECT  student_id, course_id, marks,
        ROUND(marks * 1.155, 0) "New Score",
        ROUND(marks * 1.155, 0) - marks "Increase"
FROM    ad_exam_results;
```

5. Perform the following tasks:

   a. Write a query that displays the first name (with the first letter in uppercase and all the other letters in lowercase) and the length of the first name for all students whose name starts with the letters “J,” “R,” or “M.” Give each column an appropriate label. Sort the results by the students’ first names.

```
SELECT  INITCAP(first_name) "Name",
        LENGTH(first_name) "Length"
FROM    ad_student_details
WHERE   first_name LIKE 'J%'
OR      first_name LIKE 'R%'
OR      first_name LIKE 'M%'
ORDER BY first_name;
```

b. Rewrite the query so that the user is prompted to enter the letter that the first name starts with. When prompted, enter "N."

```
SELECT  INITCAP(first_name) "Name",
        LENGTH(first_name) "Length"
FROM    ad_student_details
WHERE   first_name LIKE '&start_letter%'
ORDER BY first_name;
```

c. Modify the query such that the case of the letter that is entered does not affect the output. The entered letter must be capitalized before being processed by the SELECT query.

```
SELECT  INITCAP(first_name) "Name",
LENGTH(first_name) "Length"
FROM    ad_student_details
WHERE   first_name LIKE UPPER('&start_letter%' )
ORDER BY first_name;
```

If you have time, complete the following exercises:

6. The University wants to find the number of months of a course completed by each student. For each student, display the first name and calculate the number of months between today and the date on which the student registered. Label the column as MONTHS_COMPLETED. Order your results by the number of months completed. The number of months must be rounded to the closest whole number.

   Note: Because this query depends on the date when it is executed, the values in the MONTHS_COMPLETED column will differ for you.

```
SELECT first_name, ROUND(MONTHS_BETWEEN(
       SYSDATE, student_reg_year)) MONTHS_COMPLETED
FROM   ad_student_details
ORDER BY months_completed;
```

7. Create a query to display the exam name and exam type for all available exams. Format the exam type to be 15 characters long, left-padded with the * symbol. Label the column EXAM_CODE.

```
SELECT exam_name,
       LPAD(exam_type, 15, '*') EXAM_CODE
FROM   ad_exam_type;
```

8. Create a query that displays students' IDs, and indicates the number of marks scored with asterisks. Each asterisk signifies 10 marks. Sort the data in descending order of marks. Label the column STUDENTS_AND_THEIR_MARKS.

```
SELECT student_id,
       rpad(' ', marks/10, '*')
              STUDENTS_AND_THEIR_MARKS
FROM  ad_exam_results
ORDER BY marks DESC;
```

9. Create a query to display the first name and the number of weeks enrolled into courses for all students whose email addresses are NULL. Label the number of weeks column as `WEEKS_COMPLETED`. Truncate the number of weeks value to 0 decimal places. Show the records in descending order of the student's tenure.

   Note: The `WEEKS_COMPLETED` value will differ because it depends on the date when you run the query.

```
SELECT first_name, trunc((SYSDATE-student_reg_year)/7) AS
WEEKS_COMPLETED
FROM   ad_student_details
WHERE  email_addr is NULL
ORDER BY WEEKS_COMPLETED DESC;
```

Practices for Lesson 8: Using Conversion Functions

Chapter 8

Practices for Lesson 8: Overview

Practice Overview

This practice covers creating queries that use the `TO_CHAR` and `TO_DATE` functions.

Practice 8-1: Using Conversion Functions and Conditional Expressions

Overview

This practice provides a variety of exercises that use the `TO_CHAR` and `TO_DATE` functions.

Tasks

1. Create a report that produces the following information for each faculty member: `<faculty name>` earns `<salary>`monthly but wants `<3 times salary.>`. Label the column `Dream Salary`.

	Dream Salary
1	JILL MILLER earns $24,000.00 monthly but wants $72,000.00.
2	JAMES BORG earns $17,000.00 monthly but wants $51,000.00.
3	LYNN BROWN earns $8,800.00 monthly but wants $26,400.00.
4	ARTHUR SMITH earns $11,500.00 monthly but wants $34,500.00.
5	SALLY JONES earns $20,850.00 monthly but wants $62,550.00.

2. Display each student's first name, registration date, and course review date, which is the first Monday after six months of course registration. Label the column `REVIEW`. Format the dates to appear in a format that is similar to "Monday, the Thirty-First of July, 2015."

	FIRST_NAME	STUDENT_REG_YEAR	REVIEW
1	JACK	01-JAN-14	Monday, the Seventh of July, 2014
2	RHONDA	01-SEP-14	Monday, the Second of March, 2015
3	ROBERT	01-MAR-14	Monday, the Eighth of September, 2014
4	JEANNE	01-MAR-14	Monday, the Eighth of September, 2014
5	MILLS	01-APR-15	Monday, the Fifth of October, 2015
6	NINA	01-JAN-13	Monday, the Eighth of July, 2013
7	NATHAN	01-JAN-16	Monday, the Fourth of July, 2016
8	NOAH	01-JUN-14	Monday, the Eighth of December, 2014

3. Create a query that displays students' first names and contact information. If a student does not have an email address, show "No Email Address." Label the column `CONTACT_INFO`.

	FIRST_NAME	CONTACT_INFO
1	JACK	jack@email.com
2	RHONDA	No Email Address
3	ROBERT	robert@email.com
4	JEANNE	No Email Address
5	MILLS	mills@email.com
6	NINA	nina@email.com
7	NATHAN	No Email Address
8	NOAH	noah@email.com

4. The University wants to felicitate students who have scored good marks with a cash prize equal to marks. Write a query to display the student ID, marks, and cash prize for students who have scored more than 90. Format the cash prize column to appear in a format such as “$90.” Rename the column as `PRIZE_AMOUNT`.

	STUDENT_ID	MARKS	PRIZE_AMOUNT
1	720	91	$91
2	720	97	$97
3	750	97	$97
4	770	91	$91
5	770	99	$99
6	710	91	$91

Solution 8-1: Using Conversion Functions and Conditional Expressions

1. Create a report that produces the following information for each faculty member: <faculty name> earns <salary>monthly but wants <3 times salary.>. Label the column Dream Salary.

```
SELECT  faculty_name || ' earns '
        || TO_CHAR(salary, 'fm$99,999.00')
        || ' monthly but wants '
        || TO_CHAR(salary * 3, 'fm$99,999.00')
        || '.' "Dream Salary"
FROM    ad_faculty_details;
```

2. Display each student's first name, registration date, and course review date, which is the first Monday after six months of course registration. Label the column REVIEW. Format the dates to appear in a format that is similar to "Monday, the Thirty-First of July, 2015."

```
SELECT first_name, student_reg_year,
       TO_CHAR(NEXT_DAY(ADD_MONTHS(student_reg_year,
6),'MONDAY'),
       'fmDay, "the" Ddspth "of" Month, YYYY') REVIEW
FROM    ad_student_details;
```

3. Create a query that displays students' first names and contact information. If a student does not have an email address, show "No Email Address." Label the column CONTACT_INFO.

```
SELECT first_name,
       NVL(TO_CHAR(email_addr), 'No Email Address') CONTACT_INFO
FROM   ad_student_details;
```

4. The University wants to felicitate students who have scored good marks with a cash prize equal to marks. Write a query to display the student ID, marks, and cash prize for students who have scored more than 90. Format the cash prize column to appear in a format such as "$90." Rename the column as PRIZE_AMOUNT.

```
SELECT student_id, marks, TO_CHAR(marks, '$99') PRIZE_AMOUNT
FROM ad_exam_results
WHERE marks >= 90;
```

Practices for Lesson 9: Using Conditional Expressions

Chapter 9

Practices for Lesson 9: Overview

Practice Overview

This practice covers creating queries that use conditional expressions such as `CASE`, searched `CASE`, and `DECODE`.

Practice 9-1: Using Conditional Expressions

Overview

This practice provides exercises using conditional expressions such as `CASE`, searched `CASE`, and `DECODE`.

Tasks

1. Using the `CASE` function, write a query that displays the nature of all the different types of exams, based on the value of the `EXAM_TYPE` column in the `AD_EXAM_TYPE` table. Use the following data:

Exam Type	***Nature of Exam***
MCE	OBJECTIVE
TF	OBJECTIVE
FIB	OBJECTIVE
ESS	SUBJECTIVE
SA	SUBJECTIVE
PS	ANALYTICAL
LAB	PRACTICAL

For any other exam type, mention `NOT PERMITTED`.

	EXAM_TYPE	NATURE OF EXAM
1	ESS	SUBJECTIVE
2	FIB	OBJECTIVE
3	LAB	PRACTICAL
4	MCE	OBJECTIVE
5	OB	NOT PERMITTED
6	PS	ANALYTICAL
7	SA	SUBJECTIVE
8	TF	OBJECTIVE

2. Using the searched `CASE` expression, report on students' exam results as shown below. Use the `MARKS` column of the `AD_EXAM_RESULTS` table.

Marks	Grade Remark
<60	Fail
>60 and <70	Satisfactory
>70 and <80	Good
>80 and <90	Very Good
>90	Excellent

	STUDENT_ID	MARKS	GRADE REMARK
1	720	91	EXCELLENT
2	720	97	EXCELLENT
3	720	89	VERY GOOD
4	740	69	SATISFACTORY
5	740	70	SATISFACTORY
6	740	76	GOOD
7	750	60	SATISFACTORY
8	750	97	EXCELLENT
9	750	78	GOOD
10	750	85	VERY GOOD
11	760	65	SATISFACTORY
12	760	60	SATISFACTORY
13	760	79	GOOD
14	760	70	SATISFACTORY
15	770	91	EXCELLENT
16	770	99	EXCELLENT
17	710	91	EXCELLENT
18	710	77	GOOD
19	780	56	FAIL

…

20	780	61	SATISFACTORY
21	780	65	SATISFACTORY
22	730	87	VERY GOOD
23	730	85	VERY GOOD

3. Using the searched `DECODE` syntax, redo the step 1 conditional report to show the following output:

	EXAM_TYPE	NATURE OF EXAM
1	ESS	SUBJECTIVE
2	FIB	OBJECTIVE
3	LAB	PRACTICAL
4	MCE	OBJECTIVE
5	OB	NOT PERMITTED
6	PS	ANALYTICAL
7	SA	SUBJECTIVE
8	TF	OBJECTIVE

Solution 9-1: Using Conditional Expressions

1. Using the CASE function, write a query that displays the nature of all the different types of exams, based on the value of the EXAM_TYPE column in the AD_EXAM_TYPE table. Use the following data:

Exam Type	*Nature of Exam*
MCE	OBJECTIVE
TF	OBJECTIVE
FIB	OBJECTIVE
ESS	SUBJECTIVE
SA	SUBJECTIVE
PS	ANALYTICAL
LAB	PRACTICAL

For any other exam type, mention NOT PERMITTED.

```
SELECT exam_type, CASE exam_type
                WHEN 'MCE' THEN 'OBJECTIVE'
                WHEN 'TF'   THEN 'OBJECTIVE'
                WHEN 'FIB'  THEN 'OBJECTIVE'
                WHEN 'ESS'   THEN 'SUBJECTIVE'
                WHEN 'SA'  THEN 'SUBJECTIVE'
                 WHEN 'PS'  THEN 'ANALYTICAL'
                 WHEN 'LAB'  THEN 'PRACTICAL'
                ELSE 'NOT PERMITTED'  END  "NATURE OF EXAM"
FROM ad_exam_type;
```

2. Using the searched CASE expression, report on students' exam results as shown below. Use the MARKS column of the AD_EXAM_RESULTS table.

Marks	Grade Remark
<60	Fail
>60 and <70	Satisfactory
>70 and <80	Good
>80 and <90	Very Good
>90	Excellent

```
SELECT student_id, marks, CASE
      WHEN marks < 60  THEN 'FAIL'
      WHEN marks BETWEEN 60 AND 70  THEN 'SATISFACTORY'
      WHEN marks BETWEEN 70 and 80  THEN 'GOOD'
      WHEN marks BETWEEN 80 and 90  THEN 'VERY GOOD'
      WHEN marks BETWEEN 90 and 100 THEN 'EXCELLENT'
      ELSE 'ERROR'  END  "GRADE REMARK"
FROM ad_exam_results;
```

3. Using the searched DECODE syntax, redo the step 1 conditional report to show the following output:

Exam Type	*Nature of Exam*
MCE	OBJECTIVE
TF	OBJECTIVE
FIB	OBJECTIVE
ESS	SUBJECTIVE
SA	SUBJECTIVE
PS	ANALYTICAL
LAB	PRACTICAL

For any other exam type, mention NOT PERMITTED.

```
SELECT exam_type, DECODE(exam_type,
                'MCE','OBJECTIVE',
                'TF', 'OBJECTIVE',
                'FIB', 'OBJECTIVE',
                'ESS', 'SUBJECTIVE',
                'SA',  'SUBJECTIVE',
                'PS', 'ANALYTICAL',
                'LAB', 'PRACTICAL',
                'NOT PERMITTED') "NATURE OF EXAM"
FROM ad_exam_type;
```

Practices for Lesson 10: Reporting Aggregated Data Using the Group Functions

Chapter 10

Practices for Lesson 10: Overview

Practice Overview

This practice covers the following topics:

- Writing queries that use group functions
- Grouping by rows to achieve multiple results
- Restricting groups by using the `HAVING` clause

Practice 10-1: Reporting Aggregated Data Using the Group Functions

Overview

After completing this practice, you should be familiar with using the group functions and selecting groups of data.

Tasks

Determine the validity of the following statements. Circle either True or False.

1. Group functions work across many rows to produce one result per group.
   True/False
2. Group functions include nulls in calculations.
   True/False
3. The `WHERE` clause restricts rows before inclusion in a group calculation.
   True/False

The University needs the following reports:

4. Find the highest, lowest, and average marks of all the students across all the exams conducted for all the courses. Label the columns `Highest`, `Lowest`, and `Average`, respectively. Run the query.

	Highest	Lowest	Average
1	99	56	78.173913043478260869565217391304347826 09

5. Write a query to display the lowest, highest, and average marks obtained by students in each exam. Order the results in ascending order of the `exam_id`. Run the query.

	EXAM_ID	Highest	Lowest	Average
1	500	91	60	80.6666666666666666666666666666666666666667
2	510	97	70	81.3333333333333333333333333333333333333333
3	520	97	56	79
4	530	89	60	72
5	540	87	65	77
6	550	99	99	99
7	560	91	69	80
8	570	76	70	73

6. Write a query to display the number of students in each course. Use the table `AD_STUDENT_COURSE_DETAILS`. Order the results in ascending order of the `course_id`.

	COURSE_ID	COUNT(*)
1	175	2
2	176	2
3	187	2
4	188	2
5	189	1
6	190	2
7	191	1
8	192	3
9	193	2
10	194	1
11	195	1
12	197	1
13	198	1
14	199	2

7. Determine the courses for which the average marks in each exam was greater than 85.

	EXAM_ID	COURSE_ID	AVG(MARKS)
1	540	199	87
2	510	175	87
3	500	176	91
4	520	193	97
5	530	191	89
6	560	188	91
7	550	187	99
8	500	190	91

8. Find the difference between the highest and lowest salaries of the faculty members. Label the column `DIFFERENCE`. Use the table `AD_FACULTY_DETAILS`.

	DIFFERENCE
1	15200

9. Create a query that displays the maximum average marks obtained in a course across all the exams.

	MAX(AVG(MARKS))
1	89

10. Create a query to display the lowest marks obtained in `COURSE_ID`s 190, 191, and 192. Display the marks only if the lowest marks is greater than 75.

	COURSE_ID	MIN(MARKS)
1	190	79
2	191	89

Solution 10-1: Reporting Aggregated Data by Using the Group Functions

Determine the validity of the following statements. Circle either True or False.

1. Group functions work across many rows to produce one result per group.
   True/False

2. Group functions include nulls in calculations.
   True/**False**

3. The `WHERE` clause restricts rows before inclusion in a group calculation.
   True/False

The University needs the following reports:

4. Find the highest, lowest, and average marks of all the students across all the exams conducted for all the courses. Label the columns `Highest`, `Lowest`, and `Average`, respectively. Run the query.

```
SELECT MAX(marks) "Highest",
       MIN(marks) "Lowest",
       AVG(marks) "Average"
FROM   ad_exam_results;
```

5. Write a query to display the lowest, highest, and average marks obtained by students in each exam. Order the results in ascending order of the `exam_id`. Run the query.

```
SELECT exam_id, MAX(marks) "Highest",
                MIN(marks) "Lowest",
                AVG(marks) "Average"
FROM   ad_exam_results
GROUP BY exam_id
ORDER BY exam_id;
```

   Note: You can use the `ROUND()` function to round the average marks results.

6. Write a query to display the number of students in each course. Use the table `AD_STUDENT_COURSE_DETAILS`. Order the results in ascending order of the `course_id`.

```
SELECT course_id, COUNT(*)
FROM   ad_student_course_details
GROUP BY course_id
ORDER BY course_id;
```

7. Determine the courses for which the average marks in each exam was greater than 85.

```
SELECT exam_id, course_id, AVG(marks)
FROM   ad_exam_results
GROUP BY exam_id, course_id
HAVING AVG(marks) > 85;
```

8. Find the difference between the highest and lowest salaries of the faculty members. Label the column DIFFERENCE. Use the table AD_FACULTY_DETAILS.

```
SELECT   MAX(salary) - MIN(salary) DIFFERENCE
FROM     ad_faculty_details;
```

9. Create a query that displays the maximum average marks obtained in a course across all the exams.

```
SELECT   MAX(AVG(marks))
FROM     ad_exam_results
GROUP BY course_id;
```

10. Create a query to display the lowest marks obtained in COURSE_IDs 190, 191, and 192. Display the marks only if the lowest marks is greater than 75.

```
SELECT    course_id, MIN(marks)
FROM      ad_exam_results
WHERE     course_id in (190,191,192)
GROUP BY  course_id
HAVING    min(marks)>75
ORDER BY  min(marks)
```

Practices for Lesson 11: Retrieving Data from Multiple Tables Using Joins

Chapter 11

Practices for Lesson 11: Overview

Practice Overview

This practice covers the following topics:

- Joining tables using an equijoin
- Performing outer and self-joins
- Adding conditions

Practice 11-1: Retrieving Data from Multiple Tables Using Joins

Overview

This practice is intended to give you experience in extracting data from multiple tables by using the SQL:1999–compliant joins.

Tasks

1. Write a query for the University to produce the department names and the course names under each department. Use a `NATURAL JOIN` to produce the results.

	DEPARTMENT_NAME	COURSE_NAME
1	ACCOUNTING	COST ACCOUNTING
2	ACCOUNTING	INTRODUCTION TO BUSINESS LAW
3	ACCOUNTING	PRINCIPLES OF ACCOUNTING
4	ACCOUNTING	STRATEGIC TAX PLANNING FOR BUSINESS
5	BIOLOGY	INTRODUCTION TO PLANT PHYSIOLOGY
6	BIOLOGY	CELL BIOLOGY
7	BIOLOGY	GENERAL BIOLOGY
8	BIOLOGY	MARINE BIOLOGY
9	COMPUTER SCIENCE	SIMULATION AND MODELING
10	COMPUTER SCIENCE	WEB PROGRAMMING
11	COMPUTER SCIENCE	DATA STRUCTURES
12	COMPUTER SCIENCE	OOAD
13	LITERATURE	COLLEGE READING
14	LITERATURE	BUSINESS WRITING
15	LITERATURE	AMERICAN LITERATURE

2. The University needs a report of all the departments with their corresponding courses and the names of their head of department (`HOD`). Use the `USING` clause.

	DEPARTMENT_NAME	COURSE_NAME	HOD
1	ACCOUNTING	COST ACCOUNTING	MARK SMITH
2	ACCOUNTING	INTRODUCTION TO BUSINESS LAW	MARK SMITH
3	ACCOUNTING	PRINCIPLES OF ACCOUNTING	MARK SMITH
4	ACCOUNTING	STRATEGIC TAX PLANNING FOR BUSINESS	MARK SMITH
5	BIOLOGY	INTRODUCTION TO PLANT PHYSIOLOGY	DAVE GOLD
6	BIOLOGY	CELL BIOLOGY	DAVE GOLD
7	BIOLOGY	GENERAL BIOLOGY	DAVE GOLD
8	BIOLOGY	MARINE BIOLOGY	DAVE GOLD
9	COMPUTER SCIENCE	SIMULATION AND MODELING	LINDA BROWN
10	COMPUTER SCIENCE	WEB PROGRAMMING	LINDA BROWN
11	COMPUTER SCIENCE	DATA STRUCTURES	LINDA BROWN
12	COMPUTER SCIENCE	OOAD	LINDA BROWN
13	LITERATURE	COLLEGE READING	ANITA TAYLOR
14	LITERATURE	BUSINESS WRITING	ANITA TAYLOR
15	LITERATURE	AMERICAN LITERATURE	ANITA TAYLOR

3. The University needs a report of the courses that are being conducted in the `SUMMER` session. Use `session_id` as 300.

	COURSE_NAME	SESSION_NAME
1	SIMULATION AND MODELING	SUMMER SESSION
2	WEB PROGRAMMING	SUMMER SESSION
3	DATA STRUCTURES	SUMMER SESSION
4	OOAD	SUMMER SESSION
5	AMERICAN LITERATURE	SUMMER SESSION

.

4. Create a report to display faculty names and the courses they teach. Note that this query requires you to use three-way joins because the data about courses, faculty, and the faculty_id :course_id is available in `AD_COURSE_DETAILS`, `AD_FACULTY_DETAILS`, and `AD_FACULTY_COURSE_DETAILS`, respectively. Run the query.

Query Result

SQL | All Rows Fetched: 15 in 0.344 seconds

	COURSE_ID	COURSE_NAME	FACULTY_NAME
1	190	PRINCIPLES OF ACCOUNTING	JILL MILLER
2	191	INTRODUCTION TO BUSINESS LAW	JILL MILLER
3	192	COST ACCOUNTING	JILL MILLER
4	193	STRATEGIC TAX PLANNING FOR BUSINESS	JILL MILLER
5	194	GENERAL BIOLOGY	JAMES BORG
6	195	CELL BIOLOGY	JAMES BORG
7	196	INTRODUCTION TO PLANT PHYSIOLOGY	LYNN BROWN
8	197	MARINE BIOLOGY	LYNN BROWN
9	198	SIMULATION AND MODELING	ARTHUR SMITH
10	199	WEB PROGRAMMING	ARTHUR SMITH
11	187	DATA STRUCTURES	ARTHUR SMITH
12	188	OOAD	ARTHUR SMITH
13	189	COLLEGE READING	ARTHUR SMITH
14	176	BUSINESS WRITING	SALLY JONES
15	175	AMERICAN LITERATURE	SALLY JONES

5. There is a mentorship policy for junior faculty. The mentors guide junior faculty members with their teaching experiences and instructional methodologies. Write a query to list the names of faculty members along with their mentors. Note that senior faculty members may not have a mentor assigned.

	Faculty	FACULTY#	Mentor	MENTOR#
1	ARTHUR SMITH	830	JILL MILLER	800
2	SALLY JONES	840	JILL MILLER	800
3	LYNN BROWN	820	JAMES BORG	810

6. Create a report that displays the `student_id` and `first_name` of those students who have secured between 60 and 70 marks in any exam that was conducted. Also, report the `exam_id` for which the students secured the marks.

	STUDENT_ID	FIRST_NAME	EXAM_ID	MARKS
1	740	RHONDA	560	69
2	740	RHONDA	570	70
3	750	ROBERT	500	60
4	760	JEANNE	540	65
5	760	JEANNE	510	70
6	760	JEANNE	530	60
7	780	NATHAN	530	61
8	780	NATHAN	530	65

7. List the names of the departments and their corresponding course names. Include those departments that have still not launched any course.

	DEPARTMENT_NAME	COURSE_NAME
1	ACCOUNTING	COST ACCOUNTING
2	ACCOUNTING	INTRODUCTION TO BUSINESS LAW
3	ACCOUNTING	PRINCIPLES OF ACCOUNTING
4	ACCOUNTING	STRATEGIC TAX PLANNING FOR BUSINESS
5	BIOLOGY	INTRODUCTION TO PLANT PHYSIOLOGY
6	BIOLOGY	CELL BIOLOGY
7	BIOLOGY	GENERAL BIOLOGY
8	BIOLOGY	MARINE BIOLOGY
9	COMPUTER SCIENCE	SIMULATION AND MODELING
10	COMPUTER SCIENCE	WEB PROGRAMMING
11	COMPUTER SCIENCE	DATA STRUCTURES
12	COMPUTER SCIENCE	OOAD
13	LITERATURE	COLLEGE READING
14	LITERATURE	BUSINESS WRITING
15	LITERATURE	AMERICAN LITERATURE
16	BUSINESS MANAGEMENT	(null)

Solution 11-1: Retrieving Data from Multiple Tables Using Joins

1. Write a query for the University to produce the department names and the course names under each department. Use a `NATURAL JOIN` to produce the results.

```
SELECT department_name, course_name
FROM   ad_department
NATURAL JOIN  ad_course_details;
```

2. The University needs a report of all the departments with their corresponding courses and the names of their head of department (`HOD`). Use the `USING` clause.

```
SELECT department_name, course_name, hod
FROM   ad_department
JOIN   ad_course_details
USING (department_id);
```

3. The University needs a report of the courses that are being conducted in the `SUMMER` session. Use `session_id` as 300.

```
SELECT c.course_name, s.session_name
FROM   ad_course_details c JOIN ad_academic_session s
ON     (c.session_id = s.session_id)
WHERE s.session_id = 300;
```

4. Create a report to display faculty names and the courses they teach. Note that this query requires you to use three-way joins because the data about courses, faculty, and the faculty_id :course_id is available in `AD_COURSE_DETAILS`, `AD_FACULTY_DETAILS`, and `AD_FACULTY_COURSE_DETAILS`, respectively. Run the query.

```
SELECT a.course_id, b.course_name, c.faculty_name
FROM   ad_faculty_course_details a JOIN ad_course_details b
ON     (a.course_id = b.course_id)
JOIN   ad_faculty_details c
USING  (faculty_id);
```

5. There is a mentorship policy for junior faculty. The mentors guide junior faculty members with their teaching experiences and instructional methodologies. Write a query to list the names of faculty members along with their mentors. Note that senior faculty members may not have a mentor assigned.

```
SELECT f.faculty_name "Faculty", f.faculty_id "FACULTY#",
       m.faculty_name "Mentor", m.faculty_id  "MENTOR#"
FROM   ad_faculty_details f JOIN ad_faculty_details m
ON     (f.mentor_id = m.faculty_id);
```

6. Create a report that displays the `student_id` and `first_name` of those students who have secured between 60 and 70 marks in any exam that was conducted. Also, report the `exam_id` for which the students secured the marks.

```
SELECT a.student_id, a.first_name, b.exam_id, b.marks
FROM   ad_student_details a JOIN ad_exam_results b
ON     (a.student_id = b.student_id)
AND    (b.marks BETWEEN 60 AND 70);
```

7. List the names of the departments and their corresponding course names. Include those departments that have still not launched any course.

```
SELECT department_name, course_name
FROM   ad_department
LEFT OUTER JOIN   ad_course_details
USING (department_id);
```

Practices for Lesson 12: Using the Set Operators

Chapter 12

Practices for Lesson 12: Overview

Practice Overview

In this practice, you create reports by using the following:

- `UNION` operator
- `INTERSECT` operator
- `MINUS` operator

Practice 12-1: Using the Set Operators

Overview

In this practice, you write queries by using the set operators UNION/UNION ALL, INTERSECT, and MINUS.

Tasks

1. The University needs a list of COURSE_IDs that do not have any students enrolled. Use the set operators to create this report.

	COURSE_ID
1	196

2. The University needs a list of COURSE_IDs and COURSE_NAMEs that do not have any students enrolled. Use the set operators to create this report.

	COURSE_ID	COURSE_NAME
1	196	INTRODUCTION TO PLANT PHYSIOLOGY

3. Produce a list of all students who are enrolled in COURSE_ID 190 and 193 and their course information. Display the STUDENT_ID, FIRST_NAME, COURSE_ID and COURSE_NAME by using the set operators.

	STUDENT_ID	FIRST_NAME	COURSE_ID	COURSE_NAME
1	720	JACK	190	PRINCIPLES OF ACCOUNTING
2	720	JACK	193	STRATEGIC TAX PLANNING FOR BUSINESS
3	760	JEANNE	190	PRINCIPLES OF ACCOUNTING
4	780	NATHAN	193	STRATEGIC TAX PLANNING FOR BUSINESS

4. Create a report that lists the course_id and course_name of all accounting department courses that are scheduled in the spring session. Note that the ACCOUNTING DEPARTMENT_ID is 10 and the spring SESSION_ID is 100.

	COURSE_ID	COURSE_NAME
1	190	PRINCIPLES OF ACCOUNTING
2	191	INTRODUCTION TO BUSINESS LAW
3	192	COST ACCOUNTING
4	193	STRATEGIC TAX PLANNING FOR BUSINESS

5. Merge the records from the AD_COURSE_DETAILS and the AD_DEPARTMENT table. Display null for the columns that are not matching in the compound query.
   - List COURSE_NAME and DEPARTMENT_IDs of all courses from the AD_COURSE_DETAILS table

- List `DEPARTMENT_ID`s and `DEPARTMENT_NAME`s of all departments from the `AD_DEPARTMENT` table

	COURSE_NAME	DEPARTMENT_ID	DEPT_NAME
1	PRINCIPLES OF ACCOUNTING	10	(null)
2	INTRODUCTION TO BUSINESS LAW	10	(null)
3	COST ACCOUNTING	10	(null)
4	STRATEGIC TAX PLANNING FOR BUSINESS	10	(null)
5	GENERAL BIOLOGY	20	(null)
6	CELL BIOLOGY	20	(null)
7	INTRODUCTION TO PLANT PHYSIOLOGY	20	(null)
8	MARINE BIOLOGY	20	(null)
9	SIMULATION AND MODELING	30	(null)
10	WEB PROGRAMMING	30	(null)
11	DATA STRUCTURES	30	(null)
12	OOAD	30	(null)
13	COLLEGE READING	40	(null)
14	BUSINESS WRITING	40	(null)
15	AMERICAN LITERATURE	40	(null)
16	(null)	10	ACCOUNTING
17	(null)	20	BIOLOGY
18	(null)	30	COMPUTER SCIENCE
19	(null)	40	LITERATURE
20	(null)	50	BUSINESS MANAGEMENT

Solution 12-1: Using the Set Operators

1. The University needs a list of COURSE_IDs that do not have any students enrolled. Use the set operators to create this report.

```
SELECT course_id
FROM   ad_course_details
MINUS
SELECT course_id
FROM ad_student_course_details;
```

2. The University needs a list of COURSE_IDs and COURSE_NAMEs that do not have any students enrolled. Use the set operators to create this report.

```
SELECT course_id,course_name
FROM   ad_course_details
MINUS
SELECT x.course_id,y.course_name
FROM  ad_student_course_details x
JOIN  ad_course_details y
ON    x.course_id = y.course_id;
```

3. Produce a list of all students who are enrolled in COURSE_ID 190 and 193 and their course information. Display the STUDENT_ID, FIRST_NAME, COURSE_ID, and COURSE_NAME by using the set operators.

```
SELECT z.student_id, z.first_name, x.course_id, x.course_name
FROM ad_course_details x
JOIN ad_student_course_details y
ON y.course_id = x.course_id
JOIN ad_student_details z
ON y.student_id = z.student_id
WHERE y.course_id=190
UNION
SELECT z.student_id, z.first_name, x.course_id, x.course_name
FROM ad_course_details x
JOIN ad_student_course_details y
ON y.course_id = x.course_id
JOIN ad_student_details z
ON y.student_id = z.student_id
WHERE y.course_id=193;
```

4. Create a report that lists the course_id and course_name of all accounting department courses that are scheduled in the spring session. Note that the ACCOUNTING DEPARTMENT_ID is 10 and the spring SESSION_ID is 100.

.

```
SELECT course_id, course_name
FROM ad_course_details
WHERE session_id =100
INTERSECT
SELECT course_id, course_name
FROM ad_course_details
WHERE department_id=10;
```

5. Merge the records from the AD_COURSE_DETAILS and the AD_DEPARTMENT table. Display null for the columns that are not matching in the compound query.
   - List COURSE_NAME and DEPARTMENT_IDs of all courses from the AD_COURSE_DETAILS table
   - List DEPARTMENT_IDs and DEPARTMENT_NAMEs of all departments from the AD_DEPARTMENT table

```
SELECT course_name, department_id, to_char(null) dept_name
FROM ad_course_details
UNION ALL
SELECT to_char(null),department_id,department_name
FROM ad_department;
```

Note: This is just to demonstrate how to match the columns in the select queries in case some columns in the tables are not common.

Practices for Lesson 13: Using Subqueries to Solve Queries

Chapter 13

Practices for Lesson 13: Overview

Practice Overview

This practice covers the following topics:

- Creating subqueries to query values based on unknown criteria
- Using subqueries to find out the values that exist in one set of data and not in another

Practice 13-1: Using Subqueries to Solve Queries

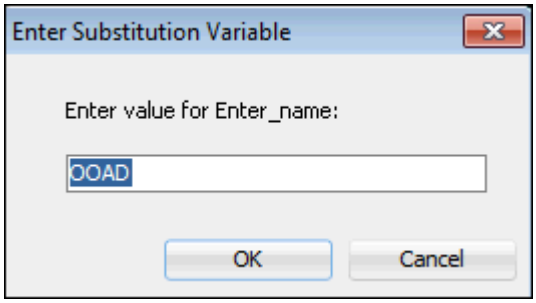
Overview

In this practice, you write complex queries using nested `SELECT` statements.

For practice questions, you may want to create the inner query first. Make sure that it runs and produces the data that you anticipate before you code the outer query.

Tasks

1. The University needs a query that prompts for a course name. The query then displays the names of all the courses available in the same department as the entered course (excluding that course). For example, if you enter `OOAD`, find all the courses available in the same department as the OOAD course. Use `UNDEFINE <variable name>` to undefine the substitution variable each time the query is run.



	COURSE_NAME	SESSION_ID
1	SIMULATION AND MODELING	300
2	WEB PROGRAMMING	300
3	DATA STRUCTURES	300

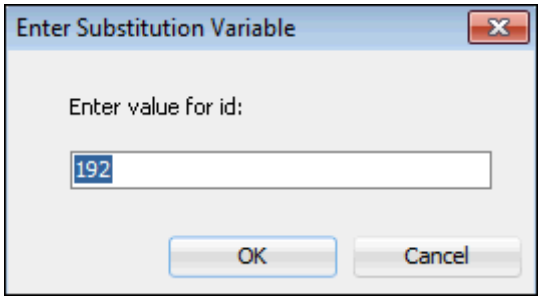
2. Create a report that displays the `faculty_id`, `faculty_name`, and `salary` of all the faculty members who earn more than the average salary. Sort the results in ascending order by salary.

	FACULTY_ID	FACULTY_NAME	SALARY
1	810	JAMES BORG	17000
2	840	SALLY JONES	20850
3	800	JILL MILLER	24000

3. Write a query that displays the examination result details of students who study courses ending with "`OGY`". Use the `AD_EXAM_RESULTS` table to list the `course_id`, `exam_id`, `student_id`, and `marks`. Run your query.

	COURSE_ID	EXAM_ID	STUDENT_ID	MARKS
1	194	530	780	65
2	195	560	740	69
3	197	570	740	70

4. The University needs a report that prompts for a `course_id`. It displays the names of the students who have enrolled in that course and the year of their registration. Use `UNDEFINE <variable name>` to undefine the substitution variable each time the query is run.



	FIRST_NAME	STUDENT_REG_YEAR
1	ROBERT	01-MAR-14
2	JEANNE	01-MAR-14
3	NATHAN	01-JAN-16

5. Create a report for the University that displays the salary and the names of all the faculty members who are mentored by the faculty member `JILL MILLER`.

	FACULTY_NAME	SALARY
1	ARTHUR SMITH	11500
2	SALLY JONES	20850

6. Write a query to display the `student_id` and `marks` of all students who appeared for the multiple-choice exams and scored more than the average marks scored in all the exams.

	STUDENT_ID	MARKS
1	710	91
2	720	91

Solution 13-1: Using Subqueries to Solve Queries

1. The University needs a query that prompts for a course name. The query then displays the names of all the courses available in the same department as the entered course (excluding that course). For example, if you enter `OOAD`, find all the courses available in the same department as the OOAD course. Use `UNDEFINE <variable name>` to undefine the substitution variable each time the query is run.

```
--Execute the UNDEFINE command to remove a variable

UNDEFINE Enter_name

-- Execute the below SELECT statements to retrieve the values
from AD_COURSE_DETAILS table

SELECT course_name, session_id
FROM   ad_course_details
WHERE  department_id = (SELECT department_id
                        FROM   ad_COURSE_details
                        WHERE  course_name = '&&Enter_name')
AND    course_name <> '&Enter_name';
```

   Note: `UNDEFINE` and `SELECT` are individual queries; execute them one after the other or press Ctrl + A + F9 to run them together.

2. Create a report that displays the `faculty_id`, `faculty_name`, and `salary` of all the faculty members who earn more than the average salary. Sort the results in ascending order by salary.

```
SELECT faculty_id, faculty_name, salary
FROM   ad_faculty_details
WHERE  salary > (SELECT AVG(salary)
                 FROM   ad_faculty_details)
ORDER BY salary;
```

3. Write a query that displays the examination result details of students who study courses ending with "`OGY`". Use the `AD_EXAM_RESULTS` table to list the `course_id`, `exam_id`, `student_id`, and `marks`. Run your query.

```
SELECT course_id, exam_id, student_id, marks
FROM   ad_exam_results
WHERE  course_id IN (SELECT course_id
                     FROM   AD_course_details
                     WHERE  course_name like '%OGY');
```

4. The University needs a report that prompts for a `course_id`. It displays the names of the students who have enrolled in that course and the year of their registration. Use `UNDEFINE <variable name>` to undefine the substitution variable each time the query is run.

```
--Execute the UNDEFINE command to remove a variable

UNDEFINE id

-- Execute the below SELECT statements to retrieve the values
from AD_STUDENT_DETAILS table
SELECT first_name, student_reg_year
FROM   ad_student_details
WHERE  student_id in( SELECT student_id
                      FROM   ad_student_course_details
                      WHERE  course_id = &&id);
```

5. Create a report for the University that displays the salary and the names of all the faculty members who are mentored by the faculty member `JILL MILLER`.

```
SELECT faculty_name, salary
FROM   ad_faculty_details
WHERE  mentor_id = (SELECT faculty_id
                    FROM   ad_faculty_details
                    WHERE  faculty_name = 'JILL MILLER');
```

6. Write a query to display the student_id and marks of all students who appeared for the multiple-choice exams and scored more than the average marks scored in all the exams.

```
SELECT student_id, marks
FROM   ad_exam_results
WHERE  exam_id IN (SELECT exam_id
                   FROM  ad_exam_details
                   WHERE  exam_type = 'MCE')
AND    marks > (SELECT AVG(marks)
                FROM   ad_exam_results);
```

Practices for Lesson 14: Introduction to Data Manipulation Language

Chapter 14

Practices for Lesson 14: Overview

Practice Overview

This practice covers the following topics:

- Inserting rows into tables
- Updating and deleting rows in a table
- Controlling database transactions

Practice 14-1: Introduction to Data Manipulation Language

Overview

The University wants you to create SQL statements to insert, update, and delete faculty data. As a prototype, you use the `MY_FACULTY` table before giving the statements to the University.

Notes

- For all the DML statements, use the Run Script icon (or press F5) to execute the query. Thus, you get to see the feedback messages in the Script Output pane. For `SELECT` queries, continue to use the Execute Statement icon or press F9 to get the formatted output in the Results pane.

Tasks

1. Create a table called `MY_FACULTY` by running the `lab_14_01.sql`. The script is located in the `SQL_labs\labs` folder.
2. Describe the structure of the `MY_FACULTY` table to identify the column names.

```
Name          Null     Type
------------- -------- ------------
FACULTY_ID    NOT NULL NUMBER(4)
FACULTY_NAME           VARCHAR2(50)
SALARY                 NUMBER(9,2)
```

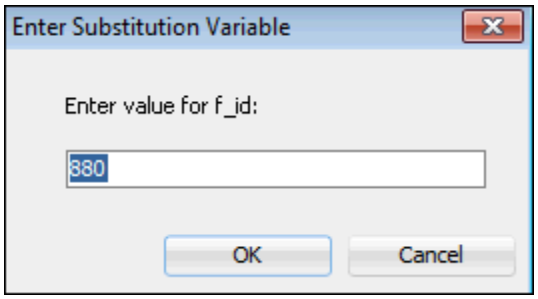
3. Create an `INSERT` statement to add the *first row* of data to the `MY_FACULTY` table from the following sample data. Do not list the columns in the `INSERT` clause. *Do not enter all rows yet.*

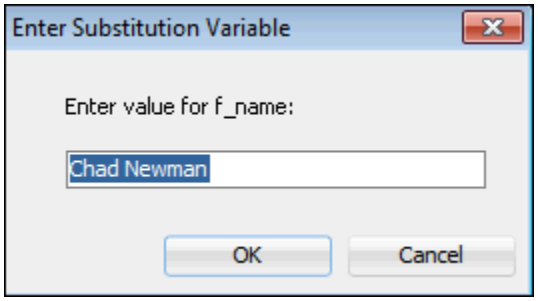
FACULTY_ID	FACULTY_NAME	SALARY
850	Ralph Patel	8950
860	Betty Dancs	8600
870	Ben Biri	11000
880	Chad Newman	7500
890	Audrey Ropeburn	15500
900	Max Hamilton	12000

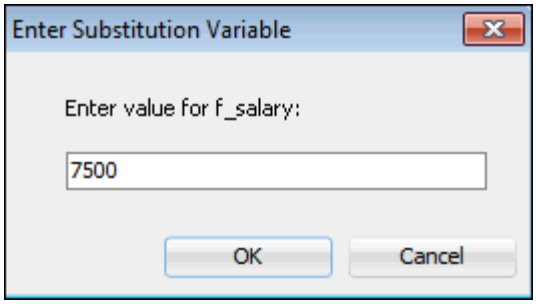
4. Populate the `MY_FACULTY` table with the second row of the sample data from the preceding list. This time, list the columns explicitly in the `INSERT` clause.
5. Confirm your addition to the table.

	FACULTY_ID	FACULTY_NAME	SALARY
1	850	Ralph Patel	8950
2	860	Betty Dancs	8600

6. Write an `INSERT` statement in a dynamic reusable query to load the remaining rows into the `MY_FACULTY` table. The query should prompt for all the columns (`FACULTY_ID`, `FACULTY_NAME`, and `SALARY`). Run the query three times to insert the remaining rows (Do not insert the last row.). The prompt windows to enter the fourth row are shown as example:







7. Confirm your additions to the table.

	FACULTY_ID	FACULTY_NAME	SALARY
1	850	Ralph Patel	8950
2	860	Betty Dancs	8600
3	870	Ben Biri	11000
4	880	Chad Newman	7500
5	890	Audrey Ropeburn	15500

8. Make the data additions permanent.

Update and delete data in the `MY_FACULTY` table.

9. Change the name of faculty member 870 to Ben Drexler.
10. Change the salary to $10000 for all faculty members who have a salary less than $9000.

11. Verify your changes to the table.

	FACULTY_ID	FACULTY_NAME	SALARY
1	850	Ralph Patel	10000
2	860	Betty Dancs	10000
3	870	Ben Drexler	11000
4	880	Chad Newman	10000
5	890	Audrey Ropeburn	15500

12. Delete Betty Dancs from the `MY_FACULTY` table.
13. Confirm your changes to the table.

	FACULTY_ID	FACULTY_NAME	SALARY
1	850	Ralph Patel	10000
2	870	Ben Drexler	11000
3	880	Chad Newman	10000
4	890	Audrey Ropeburn	15500

14. Commit all pending changes.

Control the data transactions in the `MY_FACULTY` table.

Note: Perform steps 15-22 in one session only.

15. Populate the table with the last row of the sample data listed in step 3 by using the statement that you used in step 6. Run the statement.
16. Confirm your addition to the table.

	FACULTY_ID	FACULTY_NAME	SALARY
1	850	Ralph Patel	10000
2	870	Ben Drexler	11000
3	880	Chad Newman	10000
4	890	Audrey Ropeburn	15500
5	900	Max Hamilton	12000

17. Mark an intermediate point in the processing of the transaction.
18. Delete all the rows from the `MY_FACULTY` table.

```
SAVEPOINT step_16
5 rows deleted.
```

19. Confirm that the table is empty.
20. Discard the most recent `DELETE` operation without discarding the earlier `INSERT` operation.
21. Confirm that the new row is still intact.

	FACULTY_ID	FACULTY_NAME	SALARY
1	850	Ralph Patel	10000
2	870	Ben Drexler	11000
3	880	Chad Newman	10000
4	890	Audrey Ropeburn	15500
5	900	Max Hamilton	12000

22. Make the data addition permanent.

Solution 14-1: Introduction to Data Manipulation Language

Insert data into the MY_FACULTY table.

1. Create a table called MY_FACULTY by running the lab_14_01.sql. The script is located in the SQL_labs\labs folder.

```
CREATE TABLE my_faculty
(faculty_id NUMBER(4) CONSTRAINT my_faculty_id_pk PRIMARY KEY,
faculty_name VARCHAR2(50),
salary NUMBER(9,2));
```

2. Describe the structure of the MY_FACULTY table to identify the column names.

```
DESCRIBE my_faculty
```

3. Create an INSERT statement to add the first row of data to the MY_FACULTY table from the following sample data. Do not list the columns in the INSERT clause.

FACULTY_ID	FACULTY_NAME	SALARY
850	Ralph Patel	8950
860	Betty Dancs	8600
870	Ben Biri	11000
880	Chad Newman	7500
890	Audrey Ropeburn	15500
900	Max Hamilton	12000

```
INSERT INTO my_faculty
  VALUES (850, 'Ralph Patel', 8950);
```

4. Populate the MY_FACULTY table with the second row of the sample data from the preceding list. This time, list the columns explicitly in the INSERT clause.

```
INSERT INTO my_faculty (faculty_id, faculty_name, salary)
  VALUES (860, 'Betty Dancs', 8600);
```

5. Confirm your additions to the table.

```
SELECT   *
FROM     my_faculty;
```

6. Write an INSERT statement in a dynamic reusable query to load the remaining rows into the MY_FACULTY table. The query should prompt for all the columns (FACULTY_ID, FACULTY_NAME, and SALARY). Run the following INSERT statement three times (Do not insert the last row of the sample data).

```
INSERT INTO my_faculty
VALUES (&f_id, '&f_name', &f_salary);
```

7. Confirm your additions to the table.

```
SELECT   *
FROM my_faculty;
```

8. Make the data additions permanent.

```
COMMIT;
```

Update and delete data in the MY_FACULTY table.

9. Change the name of faculty member 870 to Ben Drexler.

```
UPDATE  my_faculty
SET     faculty_name = 'Ben Drexler'
WHERE   faculty_id = 870;
```

10. Change the salary to $10000 for all faculty members with a salary less than $9000.

```
UPDATE  my_faculty
SET     salary = 10000
WHERE   salary < 9000;
```

11. Verify your changes to the table.

```
SELECT  *
FROM    my_faculty;
```

12. Delete Betty Dancs from the MY_FACULTY table.

```
DELETE
FROM  my_faculty
WHERE faculty_name = 'Betty Dancs';
```

13. Confirm your changes to the table.

```
SELECT  *
FROM    my_faculty;
```

14. Commit all pending changes.

```
COMMIT;
```

Control the data transactions in the MY_FACULTY table.

Note: Perform steps 15-22 in one session only.

15. Populate the table with the last row of the sample data listed in step 3 by using the statement that you created in step 6. Run the statement.

```
INSERT INTO my_faculty
VALUES (&f_id, '&f_name', &f_salary);
```

16. Confirm your addition to the table.

```
SELECT  *
FROM    my_faculty;
```

17. Mark an intermediate point in the processing of the transaction. Make sure Autocommit is set to off in SQL Developer before creating the SAVEPOINT. In the **Tools** menu, select **Preferences**. In the Preferences dialog box, expand **Database** and select **Advanced**. In the right pane, confirm that the Autocommit option is not selected. Click **OK**.

```
SAVEPOINT step_16;
```

18. Delete all the rows from the MY_FACULTY table.

```
DELETE
FROM  my_faculty;
```

19. Confirm that the table is empty.

```
SELECT *
FROM   my_faculty;
```

20. Discard the most recent DELETE operation without discarding the earlier INSERT operation.

```
ROLLBACK TO step_16;
```

21. Confirm that the new row is still intact.

```
SELECT *
FROM   my_faculty;
```

22. Make the data addition permanent.

```
COMMIT;
```

Practices for Lesson 15: Introduction to Data Definition Language

Chapter 15

Practices for Lesson 15: Overview

Practice Overview

This practice covers the following topics:

- Creating new tables
- Verifying that tables exist
- Defining various table and column constraints

Practice 15-1: Introduction to Data Definition Language

Overview

In this practice, you create new tables by using the `CREATE TABLE` statement. You also confirm that the new table was added to the database. Additionally, you learn to define various constraints.

Notes

- For all the DDL and DML statements, click the Run Script icon (or press F5) to execute the query in SQL Developer. Thus, you get to see the feedback messages in the Script Output pane. For `SELECT` queries, continue to click the Execute Statement icon or press F9 to get the formatted output in the Results pane.
- **Note:** Save your lab scripts in the SQL_labs>labs folder.

Tasks

1. Create the `DEPT` table based on the following table instance chart. You can either run the `CREATE TABLE` statement from the SQL Worksheet, or save the statement as `lab_15_01.sql` script and then execute the statement in the script to create the table. Confirm that the table is created.

Column Name	DEPT_ID	DEPARTMENT_NAME	HOD
Key Type	Primary Key		
Data Type	NUMBER	VARCHAR2	VARCHAR2
Length	7	50	50

```
Name            Null     Type
--------------- -------- ------------
DEPT_ID         NOT NULL NUMBER(7)
DEPARTMENT_NAME          VARCHAR2(50)
HOD                      VARCHAR2(50)
```

2. Create a copy of the same table as `COPY_DEPT`. This time create the `PRIMARY KEY` as a table constraint and add the `NOT NULL` constraint to `DEPARTMENT_NAME`.

```
Name            Null     Type
--------------- -------- ------------
DEPT_ID         NOT NULL NUMBER(7)
DEPARTMENT_NAME NOT NULL VARCHAR2(50)
HOD                      VARCHAR2(50)
```

3. Create the COURSES table based on the following table instance chart. Run the CREATE TABLE statement in the SQL Worksheet or save the statement as lab_15_03.sql script, and then execute the statement in the script to create the table. Confirm that the table is created.

Column Name	COURSE_ID	COURSE_NAME	DURATION	DEPT_ID	START_DATE
Key Type	PRIMARY KEY				
FK Table				DEPT	
FK Column				DEPT_ID	
Data Type	NUMBER	VARCHAR2	NUMBER	NUMBER	DATE
Length	7	50	4	7	
CHECK			>0 AND <24 MONTHS		
DEFAULT					SYSDATE

```
Name         Null     Type
------------ -------- ------------
COURSE_ID    NOT NULL NUMBER(7)
COURSE_NAME           VARCHAR2(50)
DURATION              NUMBER(4)
DEPT_ID               NUMBER(7)
START_DATE            DATE
```

Solution 15-1: Introduction to Data Definition Language

1. Create the DEPT table based on the following table instance chart. You can either run the CREATE TABLE statement from the SQL Worksheet, or save the statement as lab_15_01.sql script and then execute the statement in the script to create the table. Confirm that the table is created.

Column Name	DEPT_ID	DEPARTMENT_NAME	HOD
Key Type	Primary Key		
Data Type	NUMBER	VARCHAR2	VARCHAR2
Length	7	50	50

```
CREATE TABLE DEPT
(dept_id NUMBER(7) CONSTRAINT department_id_pk PRIMARY KEY,
department_name VARCHAR2(50),
hod VARCHAR2(50));
```

To confirm that the table was created and to view its structure, issue the following command:

```
DESCRIBE dept;
```

2. Create a copy of the same table as COPY_DEPT. This time create the PRIMARY KEY as a table constraint and add the NOT NULL constraint to DEPARTMENT_NAME.

```
CREATE TABLE COPY_DEPT
(dept_id NUMBER(7),
department_name VARCHAR2(50) NOT NULL,
hod VARCHAR2(50),
CONSTRAINT dpt_id_pk PRIMARY KEY(dept_id));
```

Note: Functionally, a column-level constraint is the same as a table-level constraint. Also note that a NOT NULL constraint can only be defined at the column level.

3. Create the COURSES table based on the following table instance chart. Run the CREATE TABLE statement in the SQL Worksheet or save the statement as lab_15_03.sql script, and then execute the statement in the script to create the table. Confirm that the table is created.

Column Name	COURSE_ID	COURSE_NAME	DURATION	DEPT_ID	START_DATE
Key Type	PRIMARY KEY				
Nulls/Unique					
FK Table				DEPT	
FK Column				DEPT_ID	
Data Type	NUMBER	VARCHAR2	NUMBER	NUMBER	DATE
Length	7	50	4	7	
CHECK			>0 AND <24 MONTHS		
DEFAULT					SYSDATE

```
CREATE TABLE COURSES(
course_id NUMBER(7) CONSTRAINT course_pk PRIMARY KEY,
course_name VARCHAR2(50),
duration NUMBER(4) CONSTRAINT dur_check CHECK(duration > 0 AND
duration < 24),
dept_id NUMBER(7) CONSTRAINT courses_dept_fk REFERENCES
dept(dept_id),
start_date DATE DEFAULT SYSDATE);
```

To confirm that the table was created and to view its structure:

```
DESCRIBE courses;
```

Practices for Lesson 16: Managing Tables Using DML Statements

Chapter 16

Practices for Lesson 16: Overview

Practice Overview

This practice covers the following topics:

- Creating a new table by using the `CREATE TABLE AS` syntax
- Verifying that tables exist
- Altering tables
  - Adding columns
  - Dropping columns
- Setting a table to read-only status
- Dropping tables

Practice 16-1: Managing Tables Using DML Statements

Overview

In this practice, you create a new table by using the `CREATE TABLE AS subquery` statement, and confirm that the new table was added to the database. You also use the `ALTER TABLE` command to modify table columns. Additionally, you learn to set the status of a table as `READ ONLY`, and then revert to `READ/WRITE`.

Notes

- To complete this practice, you must have completed the previous practice, Practice 15-1: Introduction to Data Definition Language. If you have not completed the previous practice, run the `sol_15.sql` script located in the `SQL_labs\soln` folder to create the required tables. **Make sure you uncomment the code before you run the script.**
- For all the DDL and DML statements, click the Run Script icon (or press F5) to execute the query in SQL Developer. Thus, you get to see the feedback messages in the Script Output pane. For `SELECT` queries, continue to click the Execute Statement icon or press F9 to get the formatted output in the Results pane.

Tasks

1. Modify the `COURSES` table. Add a column named `ANNUAL_FEES` of the `NUMBER` data type, with precision 9 and scale 2. Confirm your modification.

```
Table COURSES altered.

Name         Null     Type
------------ -------- -------------
COURSE_ID    NOT NULL NUMBER(7)
COURSE_NAME           VARCHAR2(50)
DURATION              NUMBER(4)
DEPT_ID               NUMBER(7)
START_DATE            DATE
ANNUAL_FEES           NUMBER(9,2)
```

2. Modify the `DEPT` table to allow for longer department names. Set the maximum size to 100. Confirm your modification.

```
Table DEPT altered.

Name            Null     Type
--------------- -------- --------------
DEPT_ID         NOT NULL NUMBER(7)
DEPARTMENT_NAME          VARCHAR2(100)
HOD                      VARCHAR2(50)
```

3. Drop the START_DATE column from the COURSES table. Confirm your modification by checking the description of the table.

```
Table COURSES altered.

Name         Null     Type
------------ -------- ------------
COURSE_ID    NOT NULL NUMBER(7)
COURSE_NAME           VARCHAR2(50)
DURATION              NUMBER(4)
DEPT_ID               NUMBER(7)
ANNUAL_FEES           NUMBER(9,2)
```

4. Create the COURSE_DETAIL table based on the structure of the AD_COURSE_DETAILS table. Include only the COURSE_ID and COURSE_NAME columns. Name the columns in your new table as ID and NAME, respectively. Include the courses that belong to department_id 20. View the structure of the table.

```
Name Null     Type
---- -------- ------------
ID   NOT NULL NUMBER
NAME          VARCHAR2(50)
```

5. View and verify the data in the COURSE_DETAIL table.

	ID	NAME
1	194	GENERAL BIOLOGY
2	195	CELL BIOLOGY
3	196	INTRODUCTION TO PLANT PHYSIOLOGY
4	197	MARINE BIOLOGY

6. Alter the status of the COURSE_DETAIL table to read-only.

7. Try to add a column SESSION_ID of number data type to the COURSE_DETAIL table.

   Note: You will get the “Update operation not allowed on table” error message. You will not be allowed to add any column to the table because it is assigned a read-only status.

```
Error starting at line : 2 in command -
ALTER TABLE course_detail
ADD session_id number(4)
Error report -
SQL Error: ORA-12081: update operation not allowed on table "ORA02"."COURSE_DETAIL"
12081. 00000 -  "update operation not allowed on table \"%s\".\"%s\""
*Cause:    An attempt was made to update a read-only materialized view.
*Action:   No action required. Only Oracle is allowed to update a
           read-only materialized view.
```

8. Revert the COURSE_DETAIL table to read/write status. Now try to add the same column again.

   Now, because the table is assigned a READ WRITE status, you will be allowed to add a column to the table.

   You should get the following messages:

```
Table COURSE_DETAIL altered.

Name        Null     Type
----------- -------- ------------
ID          NOT NULL NUMBER
NAME                 VARCHAR2(50)
SESSION_ID           NUMBER(4)
```

9. Drop the DEPT, COURSES, and COURSE_DETAIL tables.

Solution 16-1: Managing Tables Using DML Statements

1. Modify the COURSES table. Add a column named ANNUAL_FEES of the NUMBER data type, with precision 9 and scale 2. Confirm your modification.

```
ALTER TABLE courses
      ADD annual_fees NUMBER(9,2);

DESCRIBE courses;
```

2. Modify the DEPT table to allow for longer department names. Set the maximum size to 100. Confirm your modification.

```
ALTER TABLE dept
MODIFY (department_name VARCHAR2(100));

DESCRIBE dept;
```

3. Drop the START_DATE column from the COURSES table. Confirm your modification by checking the description of the table.

```
ALTER TABLE courses
     DROP COLUMN start_date;
 DESCRIBE courses;
```

4. Create the COURSE_DETAIL table based on the structure of the AD_COURSE_DETAILS table. Include only the COURSE_ID and COURSE_NAME. Name the columns in your new table ID and NAME, respectively. Include the courses that belong to department_id 20. View the structure of the table.

```
CREATE TABLE course_detail(id, name) AS
  SELECT  course_id, course_name
  FROM    ad_course_details where department_id=20;
DESCRIBE course_detail;
```

5. View and verify the data in the COURSE_DETAIL table.

```
SELECT * FROM COURSE_DETAIL;
```

6. Alter the status of the COURSE_DETAIL table to read-only.

```
ALTER TABLE COURSE_DETAIL READ ONLY;
```

7. Try to add a column SESSION_ID of number data type to the COURSE_DETAIL table.

   Note: You will get the “Update operation not allowed on table” error message. You will not be allowed to add any column to the table because it is assigned a read-only status.

```
ALTER TABLE course_detail
ADD session_id NUMBER(4);
```

8. Revert the COURSE_DETAIL table to the read/write status. Now try to add the same column again. Now, because the table is assigned a READ WRITE status, you will be allowed to add a column to the table.

```
ALTER TABLE course_detail READ WRITE;

ALTER TABLE course_detail
ADD session_id NUMBER(4);

DESCRIBE course_detail;
```

9. Drop the COURSES, DEPT, and COURSE_DETAIL tables.

   Note: You can even drop a table that is in READ ONLY mode. To test this, alter the table again to READ ONLY status, and then issue the DROP TABLE command. The tables will be dropped. Also, if you try to drop the DEPT table first, then you get the "unique/primary keys in table referenced by foreign keys" error; this is because the COURSES table’s DEPT_ID column references DEPT(DEPT_ID).

```
DROP TABLE courses;
DROP TABLE dept;
DROP TABLE course_detail;
```

Practices for Lesson 17: Introduction to Data Dictionary Views

Chapter 17

Practices for Lesson 17: Overview

Practice overview

This practice covers the following topics:

- Querying the dictionary views for table and column information
- Querying the dictionary views for constraint information

Practice 17-1: Introduction to Data Dictionary Views

Overview

In this practice, you query the dictionary views to find information about the objects in your schema.

Tasks

1. Query the `USER_TABLES` data dictionary view to see information about the tables that you own.

	TABLE_NAME
1	REGIONS
2	COUNTRIES
3	LOCATIONS
4	DEPARTMENTS
5	JOBS
6	EMPLOYEES
7	JOB_HISTORY
8	JOB_GRADES
9	AD_ACADEMIC_SESSION
10	AD_COURSE_DETAILS
11	AD_DEPARTMENT
12	AD_EXAM_DETAILS
13	AD_EXAM_RESULTS
14	AD_EXAM_TYPE
15	AD_FACULTY_COURSE_DETAILS
16	AD_FACULTY_DETAILS
17	AD_FACULTY_LOGIN_DETAILS
18	AD_PARENT_INFORMATION
19	AD_STUDENT_ATTENDANCE
20	AD_STUDENT_COURSE_DETAILS
21	AD_STUDENT_DETAILS

...

2. Query the `ALL_TABLES` data dictionary view to see information about all the tables that you can access. Exclude the tables that you own.

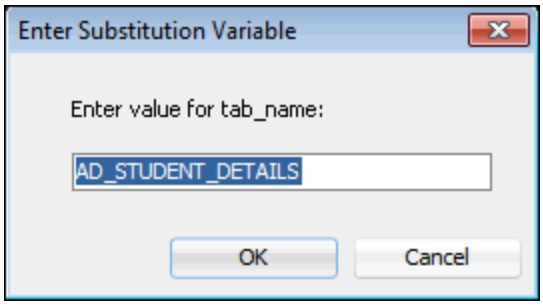
   Note: Your list may not exactly match the following list:

	TABLE_NAME	OWNER
1	DUAL	SYS
2	SYSTEM_PRIVILEGE_MAP	SYS
3	TABLE_PRIVILEGE_MAP	SYS
4	USER_PRIVILEGE_MAP	SYS
5	STMT_AUDIT_OPTION_MAP	SYS
6	AUDIT_ACTIONS	SYS
7	WRR$_REPLAY_CALL_FILTER	SYS
8	HS_BULKLOAD_VIEW_OBJ	SYS
9	HS$_PARALLEL_METADATA	SYS
10	HS_PARTITION_COL_NAME	SYS
11	HS_PARTITION_COL_TYPE	SYS

   ...

114	OL$NODES	SYSTEM
115	OL$HINTS	SYSTEM
116	OL$	SYSTEM
117	PLAN_TABLE$	SYS
118	WRI$_HEATMAP_TOPN_DEP2	SYS
119	WRI$_HEATMAP_TOPN_DEP1	SYS
120	WRI$_ADV_ASA_RECO_DATA	SYS
121	PSTUBTBL	SYS

3. For a specified table, write a query that reports the column names, data types, and data types' lengths, as well as whether nulls are allowed. Prompt the user to enter the table name. For example, if the user enters `AD_STUDENT_DETAILS`, the following output results:



	COLUMN_NAME	DATA_TYPE	DATA_LENGTH	NULLABLE
1	STUDENT_ID	NUMBER	22	N
2	FIRST_NAME	VARCHAR2	50	Y
3	PARENT_ID	NUMBER	22	Y
4	STUDENT_REG_YEAR	DATE	7	Y
5	EMAIL_ADDR	VARCHAR2	100	Y

4. Query the data dictionary to find the constraint names, constraint types, check conditions, name of the unique constraint that the foreign key references, and status for constraints on the `AD_STUDENT_DETAILS` table.

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_CONSTRAINT_NAME	STATUS
1	AD_STUDENT_DETAILS_FK	R	(null)	AD_PARENT_INFORMATION_PK	ENABLED
2	AD_STUDENT_DETAILS_PK	P	(null)	(null)	ENABLED
3	SYS_C0014021	U	(null)	(null)	ENABLED
4	SYS_C0014020	C	"STUDENT_ID" IS NOT NULL	(null)	ENABLED

5. Query the `USER_CONS_COLUMNS` view to get a report on all the tables that you own, their column_names, and the constraint_names.

   Note: The query output will vary depending on the practice activities done by you.

	TABLE_NAME	CONSTRAINT_NAME	COLUMN_NAME
1	AD_FACULTY_DETAILS	SYS_C0014018	FACULTY_ID
2	AD_FACULTY_DETAILS	AD_FACULTY_DETAILS_PK	FACULTY_ID
3	DEPARTMENTS	DEPT_LOC_FK	LOCATION_ID
4	AD_FACULTY_COURSE_DETAILS	AD_FACULTY_COURSE_DETAILS_FK	FACULTY_ID
5	EMPLOYEES	EMP_EMAIL_NN	EMAIL
6	EMPLOYEES	EMP_EMAIL_UK	EMAIL
7	EMPLOYEES	EMP_DEPT_FK	DEPARTMENT_ID
8	AD_EXAM_DETAILS	SYS_C0014016	EXAM_TYPE
9	AD_EXAM_DETAILS	AD_EXAM_DETAILS_FK	EXAM_TYPE

…

57	JOB_HISTORY	JHIST_START_DATE_NN	START_DATE
58	JOB_HISTORY	JHIST_DATE_INTERVAL	START_DATE
59	JOB_HISTORY	JHIST_EMP_ID_ST_DATE_PK	START_DATE
60	JOB_HISTORY	JHIST_END_DATE_NN	END_DATE
61	JOB_HISTORY	JHIST_DATE_INTERVAL	END_DATE
62	AD_STUDENT_DETAILS	SYS_C0014021	EMAIL_ADDR
63	EMPLOYEES	EMP_EMP_ID_PK	EMPLOYEE_ID
64	AD_COURSE_DETAILS	AD_COURSE_DETAILS_FKV2	DEPARTMENT_ID
65	JOB_HISTORY	JHIST_DEPT_FK	DEPARTMENT_ID

Solution 17-1: Introduction to Data Dictionary Views

Solution

1. Query the data dictionary to see information about the tables you own.

```
SELECT table_name
   FROM   user_tables;
```

2. Query the dictionary view to see information about all the tables that you can access. Exclude tables that you own.

   Note: Enter the appropriate username in the query.

```
SELECT table_name, owner
  FROM   all_tables
  WHERE  owner <>'ORAxx';
```

3. For a specified table, write a query that reports the column names, data types, and data types' lengths, as well as whether nulls are allowed. Prompt the user to enter the table name.

```
SELECT column_name, data_type, data_length, nullable
FROM   user_tab_columns
WHERE  table_name = UPPER('&tab_name');
```

   Execute the query and enter `AD_STUDENT_DETAILS` as the table name.

4. Query the data dictionary to find the constraint names, constraint types, search conditions, name of the unique constraint that the foreign key references, and status for constraints on the `AD_STUDENT_DETAILS` table. You must use the `USER_CONSTRAINTS` view to obtain all this information.

```
SELECT constraint_name, constraint_type,
       search_condition, r_constraint_name, status
FROM   user_constraints
WHERE  table_name = 'AD_STUDENT_DETAILS';
```

5. Query the `USER_CONS_COLUMNS` view to get a report on all the tables that you own, their column_names, and the constraint_names.

   Note: Enter the appropriate username in the query.

   Note: The query output will vary depending on the practice activities done by you.

```
SELECT table_name, constraint_name, column_name
FROM   user_cons_columns
WHERE  owner = 'ORAxx';
```

Practices for Lesson 18: Creating Views

Chapter 18

Practices for Lesson 18: Overview

Practices Overview

This practice covers the following topics:

- Creating a simple view
- Creating a complex view
- Creating a view with a check constraint
- Attempting to modify data in the view
- Querying the data dictionary for view information
- Removing views

Practice 18-1: Creating Views

Overview

This lesson's practice provides you with a variety of exercises in creating, using, and removing views.

Tasks

1. The University wants to hide the salary of the faculty in the `AD_FACULTY_DETAILS` table. Create a view called `FACULTY_VU` based on `faculty_id`, `faculty_name`, and `mentor_id` from the `AD_FACULTY_DETAILS` table. The heading for the `faculty_name` column should be `FACULTY`.
2. Confirm that the view works. Display the contents of the `FACULTY_VU` view.

	FACULTY_ID	FACULTY	MENTOR_ID
1	800	JILL MILLER	(null)
2	810	JAMES BORG	(null)
3	820	LYNN BROWN	810
4	830	ARTHUR SMITH	800
5	840	SALLY JONES	800

3. Using your `FACULTY_VU` view, write a query to display all faculty names and their mentor IDs.

	FACULTY	MENTOR_ID
1	JILL MILLER	(null)
2	JAMES BORG	(null)
3	LYNN BROWN	810
4	ARTHUR SMITH	800
5	SALLY JONES	800

4. Department 10 needs access to its courses data. Create a view named `DEPT10` that contains `course_id`, `course_name`, `session_id`, and `department_id` for all the courses in department 10. Label the view columns `COURSENO`, `COURSE`, `SESSIONNO`, and `DEPTNO`. For security purposes, do not allow a course to be reassigned to another department through the view.

5. Display the structure and contents of the `DEPT10` view.

```
Name       Null     Type
---------- -------- ------------
COURSENO   NOT NULL NUMBER
COURSE              VARCHAR2(50)
SESSIONNO           NUMBER
DEPTNO              NUMBER
```

	COURSENO	COURSE	SESSIONNO	DEPTNO
1	190	PRINCIPLES OF ACCOUNTING	100	10
2	191	INTRODUCTION TO BUSINESS LAW	100	10
3	192	COST ACCOUNTING	100	10
4	193	STRATEGIC TAX PLANNING FOR BUSINESS	100	10

6. Test your view. Attempt to reassign the course, `COST ACCOUNTING`, to department `20`.

```
Error report -
SQL Error: ORA-01402: view WITH CHECK OPTION where-clause violation
01402. 00000 -  "view WITH CHECK OPTION where-clause violation"
*Cause:
*Action:
```

7. Create a view `COURSE_DET_VU` that contains detailed course information combined from two tables, `AD_COURSE_DETAILS` and `AD_DEPARTMENT`. The view should contain the `COURSE_ID`, `COURSE_NAME`, `SESSION_ID`, `DEPARTMENT_NAME`, and `HOD` columns.

8. Display the structure and contents of the `COURSE_DET_VU` view.

```
Name              Null     Type
----------------- -------- ------------
COURSEID          NOT NULL NUMBER
COURSENAME                 VARCHAR2(50)
SESSIONID                  NUMBER
DEPARTMENTNAME             VARCHAR2(50)
HEADOFDEPARTMENT           VARCHAR2(50)
```

	COURSEID	COURSENAME	SESSIONID	DEPARTMENTNAME	HEADOFDEPARTMENT
1	192	COST ACCOUNTING	100	ACCOUNTING	MARK SMITH
2	191	INTRODUCTION TO BUSINESS LAW	100	ACCOUNTING	MARK SMITH
3	190	PRINCIPLES OF ACCOUNTING	100	ACCOUNTING	MARK SMITH
4	193	STRATEGIC TAX PLANNING FOR BUSINESS	100	ACCOUNTING	MARK SMITH
5	196	INTRODUCTION TO PLANT PHYSIOLOGY	200	BIOLOGY	DAVE GOLD
6	195	CELL BIOLOGY	200	BIOLOGY	DAVE GOLD
7	194	GENERAL BIOLOGY	200	BIOLOGY	DAVE GOLD
8	197	MARINE BIOLOGY	200	BIOLOGY	DAVE GOLD
9	198	SIMULATION AND MODELING	300	COMPUTER SCIENCE	LINDA BROWN
10	199	WEB PROGRAMMING	300	COMPUTER SCIENCE	LINDA BROWN
11	187	DATA STRUCTURES	300	COMPUTER SCIENCE	LINDA BROWN
12	188	OOAD	300	COMPUTER SCIENCE	LINDA BROWN
13	189	COLLEGE READING	100	LITERATURE	ANITA TAYLOR
14	176	BUSINESS WRITING	200	LITERATURE	ANITA TAYLOR
15	175	AMERICAN LITERATURE	300	LITERATURE	ANITA TAYLOR

9. Modify `FACULTY_VU` to ensure that no DML operations can be performed through it.

10. Try to remove the details of `faculty_id` 800. Test if this DML operation is allowed.

```
Error report -
SQL Error: ORA-42399: cannot perform a DML operation on a read-only view
42399.0000 - "cannot perform a DML operation on a read-only view"
```

11. You need to determine the names and definitions of all the views in your schema. Create a report that retrieves the following view information: the view name and text from the `USER_VIEWS` data dictionary view.

    Note: You can see the complete definition of the view if you use Run Script (or press F5) in SQL Developer. If you use Execute Statement (or press F9) in SQL Developer, scroll horizontally in the results pane.

	VIEW_NAME	TEXT
1	EMP_DETAILS_VIEW	SELECT e.employee_id, e.job_id, e.manager_id, e.department_id, d.location_id, l.country_id, e.first_name, e.last_name, e.salary,
2	FACULTY_VU	SELECT faculty_id, faculty_name faculty, mentor_idFROM ad_faculty_detailsWITH READ ONLY
3	DEPT10	SELECT course_id courseno, course_name course, session_id sessionno, department_id deptno FROM ad_course_details WHERE
4	COURSE_DET_VU	SELECT c.course_id, c.course_name, c.session_id,d.department_name, d.hod FROM ad_course_details c JOIN ad_department d

12. Remove the views created in this practice.

Solution 18-1: Creating Views

1. The University wants to hide the salary of the faculty in the AD_FACULTY_DETAILS table. Create a view called FACULTY_VU based on faculty_id, faculty_name, and mentor_id from the AD_FACULTY_DETAILS table. The heading for the faculty name should be FACULTY.

```
CREATE OR REPLACE VIEW faculty_vu AS
    SELECT faculty_id, faculty_name faculty, mentor_id
    FROM ad_faculty_details;
```

2. Confirm that the view works. Display the contents of the FACULTY_VU view.

```
SELECT   *
FROM     faculty_vu;
```

3. Using your FACULTY_VU view, write a query to display all faculty names and their mentor IDs.

```
SELECT   faculty, mentor_id
FROM     faculty_vu;
```

   Note that you can use the column alias faculty in place of the actual column name, faculty_name.

4. Department 10 needs access to its courses data. Create a view named DEPT10 that contains course_id, course_name, session_id, and department_id for all the courses in department 10. Label the view columns COURSENO, COURSE, SESSIONNO, and DEPTNO. For security purposes, do not allow a course to be reassigned to another department through the view.

```
CREATE VIEW dept10 AS
   SELECT   course_id courseno, course_name course,
            session_id sessionno, department_id deptno
   FROM     ad_course_details
   WHERE    department_id = 10
   WITH CHECK OPTION CONSTRAINT course_dept_10;
```

5. Display the structure and contents of the DEPT10 view.

```
DESCRIBE dept10

SELECT   *
FROM     dept10;
```

6. Test your view. Attempt to reassign the course, COST ACCOUNTING, to department 20.

```
UPDATE   dept10
SET      deptno = 20
WHERE    course = 'COST ACCOUNTING';
```

The error is because the DEPT10 view has been created with the WITH CHECK OPTION constraint. This ensures that the DEPTNO column in the view is protected from being changed.

7. Create a view COURSE_DET_VU that contains detailed course information combined from two tables, AD_COURSE_DETAILS and AD_DEPARTMENT. The view should contain the COURSE_ID, COURSE_NAME, SESSION_ID, DEPARTMENT_NAME, and HOD columns.

```
CREATE OR REPLACE VIEW course_det_vu
  (CourseID, CourseName, SessionID, DepartmentName,HeadOfDepartment)
AS SELECT   c.course_id, c.course_name,
           c.session_id,d.department_name, d.hod
   FROM     ad_course_details c JOIN ad_department d
   USING    (department_id);
```

8. Display the structure and contents of the COURSE_DET_VU view.

```
DESCRIBE course_det_vu
select * from course_det_vu;
```

9. Modify FACULTY_VU to ensure that no DML operations can be performed through it.

```
CREATE OR REPLACE VIEW faculty_vu AS
SELECT faculty_id, faculty_name faculty, mentor_id
FROM ad_faculty_details
WITH READ ONLY;
```

10. Try to remove the details of faculty_id 800. Test if this DML operation is allowed.

```
DELETE FROM faculty_vu
WHERE  faculty_id = 800;
```

The error is because the faculty_vu view has been created with the WITH READ ONLY option. Any attempt to remove a row from a view with a read-only constraint results in an error.

11. You need to determine the names and definitions of all the views in your schema. Create a report that retrieves the following view information: the view name and text from the USER_VIEWS data dictionary view.

Note: The EMP_DETAILS_VIEW was created as part of your schema.

Note: You can see the complete definition of the view if you use Run Script (or press F5) in SQL Developer. If you use Execute Statement (or press F9) in SQL Developer, scroll horizontally in the results pane.

```
SELECT   view_name, text
FROM     user_views;
```

12. Remove the views created in this practice.

```
DROP VIEW faculty_vu;
DROP VIEW dept10;
DROP VIEW course_det_vu;
```

Practices for Lesson 19: Creating Sequences, Synonyms, and Indexes

Chapter 19

Practices for Lesson 19: Overview

Practice Overview

This practice covers the following topics:

- Creating sequences
- Using sequences
- Querying dictionary views for sequence information
- Creating synonyms
- Querying dictionary views for synonyms information
- Creating indexes
- Querying dictionary views for indexes information

Practice 19-1: Creating Sequences, Synonyms, and Indexes

Overview

This practice provides you with a variety of exercises in creating and using a sequence, an index, and a synonym.

Tasks

1. Create the `PARENT` table based on the following table instance chart. Confirm that the table is created.

Column Name	ID	NAME
Key Type	Primary key	
Data Type	NUMBER	VARCHAR2
Length	7	25

```
Name Null     Type
---- -------- ------------
ID   NOT NULL NUMBER(7)
NAME          VARCHAR2(25)
```

2. You need a sequence that can be used with the primary key column of the `PARENT` table. The sequence should start at 100 and have a maximum value of 1,000. Have your sequence increment by 10. Name the sequence `PARENT_ID_SEQ`.
3. To test your sequence, write queries to insert two rows in the `PARENT` table. Be sure to use the sequence that you created for the `ID` column. Add two parent names: `John Fleming` and `Mark Smith`. Confirm your additions.

	ID	NAME
1	100	John Fleming
2	110	Mark Smith

4. Find the names of your sequences. Write a query to display the following information about your sequences: sequence name, maximum value, increment size, and last number.

	SEQUENCE_NAME	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
1	DEPARTMENTS_SEQ	9990	10	280
2	EMPLOYEES_SEQ	999999999999999999999999999	1	207
3	LOCATIONS_SEQ	9900	100	3300
4	PARENT_ID_SEQ	1000	10	300

5. Create a synonym for your `AD_STUDENT_DETAILS` table. Call it `student`. Use the synonym to query the table to view all the rows.

```
CREATE SYNONYM student FOR ad_student_details;
select * from student;
```

Script Output | Query Result | SQL | All Rows Fetched: 8 in 0.312 seconds

	STUDENT_ID	FIRST_NAME	PARENT_ID	STUDENT_REG_YEAR	EMAIL_ADDR
1	720	JACK	600	01-JAN-14	jack@email.com
2	740	RHONDA	620	01-SEP-14	(null)
3	750	ROBERT	610	01-MAR-14	robert@email.com
4	760	JEANNE	610	01-MAR-14	(null)
5	770	MILLS	630	01-APR-15	mills@email.com
6	710	NINA	630	01-JAN-13	nina@email.com
7	780	NATHAN	640	01-JAN-16	(null)
8	730	NOAH	640	01-JUN-14	noah@email.com

6. Find the names of all the synonyms that are in your schema.

	SYNONYM_NAME	TABLE_OWNER	TABLE_NAME	DB_LINK	ORIGIN_CON_ID
1	STUDENT	ORA03	AD_STUDENT_DETAILS	(null)	3

7. Drop the `STUDENT` synonym.
8. Create a nonunique index on the `NAME` column in the `PARENT` table.
9. Create the `COURSE_DEPT` table based on the following table instance chart. Name the index for the `PRIMARY KEY` column `COURSE_PK_IDX`. Then query the data dictionary view to find the index name, table name, and whether the index is unique.

Column Name	COURSE_ID	COURSE_DEPARTMENT
Primary Key	Yes	
Data Type	NUMBER	VARCHAR2
Length	3	30

	INDEX_NAME	TABLE_NAME	UNIQUENESS
1	COURSE_PK_IDX	COURSE_DEPT	NONUNIQUE

10. Drop the tables and sequences created in this practice.

Solution 19-1: Creating Sequences, Synonyms, and Indexes

1. Create the PARENT table based on the following table instance chart. Confirm that the table is created.

Column Name	ID	NAME
Key Type	Primary key	
Data Type	NUMBER	VARCHAR2
Length	7	25

```
CREATE TABLE parent
 (id   NUMBER(7)CONSTRAINT parent_id_pk PRIMARY KEY,
  name VARCHAR2(25));
```

To confirm that the table was created and to view its structure, issue the following command:

```
DESCRIBE parent;
```

2. You need a sequence that can be used with the primary key column of the PARENT table. The sequence should start at 100 and have a maximum value of 1,000. Have your sequence increment by 10. Name the sequence PARENT_ID_SEQ.

```
CREATE SEQUENCE parent_id_seq
  START WITH 100
  INCREMENT BY 10
  MAXVALUE 1000;
```

3. To test your sequence, write queries to insert two rows in the PARENT table. Be sure to use the sequence that you created for the ID column. Add two parent names: John Fleming and Mark Smith. Confirm your additions.

```
INSERT INTO parent
VALUES (parent_id_seq.nextval, 'John Fleming');
INSERT INTO parent
VALUES (parent_id_seq.nextval, 'Mark Smith');

--View the inserted records to check the sequence values
SELECT * from parent;
```

4. Find the names of your sequences. Write a query to display the following information about your sequences: sequence name, maximum value, increment size, and last number.

```
SELECT   sequence_name, max_value, increment_by, last_number
FROM user_sequences;
```

5. Create a synonym for your AD_STUDENT_DETAILS table. Call it student. Use the synonym to query the table to view all the rows.

```
CREATE SYNONYM student FOR ad_student_details;
SELECT * FROM student;
```

6. Find the names of all the synonyms that are in your schema.

```
SELECT * FROM   user_synonyms;
```

7. Drop the STUDENT synonym.

```
DROP SYNONYM student;
```

8. Create a nonunique index on the NAME column in the PARENT table.

```
CREATE INDEX parent_name_idx ON parent(name);
```

9. Create the COURSE_DEPT table based on the following table instance chart. Name the index for the PRIMARY KEY column COURSE_PK_IDX. Then query the data dictionary view to find the index name, table name, and whether the index is unique.

Column Name	COURSE_ID	COURSE_DEPARTMENT
Primary Key	Yes	
Data Type	NUMBER	VARCHAR2
Length	3	30

```
CREATE TABLE course_dept
       (COURSE_id NUMBER(3)
       PRIMARY KEY USING INDEX
       (CREATE INDEX COURSE_pk_idx ON
        course_dept(course_id)),
        course_department VARCHAR2(30));
SELECT INDEX_NAME, TABLE_NAME, UNIQUENESS
FROM USER_INDEXES
WHERE TABLE_NAME = 'COURSE_DEPT';
```

10. Drop the tables and sequences created in this practice.

```
DROP TABLE parent;
DROP TABLE course_dept;
DROP SEQUENCE parent_id_seq;
```

Practices for Lesson 20: Managing Constraints, Temporary Tables, and External Tables

Chapter 20

Practices for Lesson 20: Overview

Practice Overview

This practice covers the following topics:

- Adding and dropping constraints
- Deferring constraints
- Creating and querying external tables

Practice 20-1: Managing Constraints, Temporary Tables, and External Tables

Overview

In this practice, you add, drop, and defer constraints. You create and query an external table.

Tasks

1. Create the COURSE_DEPT table based on the following table instance chart. Enter the syntax in the SQL Worksheet. Then, execute the statement to create the table. Confirm that the table is created.

Column Name	ID	NAME
Data Type	NUMBER	VARCHAR2
Length	7	25

```
Name Null Type
---- ---- ------------
ID        NUMBER(7)
NAME      VARCHAR2(25)
```

2. Populate the COURSE_DEPT table with data from the AD_DEPARTMENT table. Include only the columns that you need. Confirm that the rows are inserted.

	ID	NAME
1	10	ACCOUNTING
2	20	BIOLOGY
3	30	COMPUTER SCIENCE
4	40	LITERATURE
5	50	BUSINESS MANAGEMENT

3. Create the COURSE table based on the following table instance chart. Enter the syntax in the SQL Worksheet. Then execute the statement to create the table. Confirm that the table is created.

Column Name	COURSE_ID	COURSE_NAME	DEPT_ID
Data Type	NUMBER	VARCHAR2	NUMBER
Length	7	25	7

```
Name        Null Type
----------- ---- ------------
COURSE_ID       NUMBER(7)
COURSE_NAME     VARCHAR2(25)
DEPT_ID         NUMBER(7)
```

4. Add a table-level `PRIMARY KEY` constraint to the `COURSE` table on the `COURSE_ID` column. The constraint should be named at creation. Name the constraint `course_id_pk`.
5. Create a `PRIMARY KEY` constraint on the `COURSE_DEPT` table by using the `ID` column. The constraint should be named at creation. Name the constraint `course_dept_id_pk`.
6. Add a foreign key reference on the `COURSE` table that ensures that the course is not assigned to a nonexistent department. Name the constraint `course_dept_id_fk`.
7. Modify the `COURSE` table. Add a `FEES` column of the `NUMBER` data type with precision 2 and scale 9. Add a constraint to the `FEES` column that ensures that the value is greater than zero.
8. Drop the `COURSE` and `COURSE_DEPT` tables so that they cannot be restored.
9. Create an external table `library_items_ext`. Use the `ORACLE_LOADER` access driver.

   Note: The `library_items.dat` file is saved in the `/home/oracle/emp_dir` folder on your database file system. A directory object `emp_dir` is already created for this exercise and you have been granted `READ` and `WRITE` privileges on the same.

   `library_items.dat` has records in the following format:

   ```
   2354,    2264, 13.21, 150,
   2355,    2289, 46.23, 200,
   2355,    2264, 50.00, 100,
   ```

   a. Open the `lab_20_09.sql` file. Observe the code snippet to create the `library_items_ext` external table. Then replace **<***TODO1***>**, **<***TODO2***>**, **<***TODO3***>**, and <*TODO4*> as appropriate and save the file as `lab_20_09_soln.sql`. Run the script to create the external table.

   b. Query the `library_items_ext` table.

	CATEGORY_ID	BOOK_ID	BOOK_PRICE	QUANTITY
1	2354	2264	13.21	150
2	2355	2289	46.23	200
3	2355	2264	50	100

10. Create the `course_books` table based on the following table instance chart. Name the primary key constraint, `course_books_pk`. In the second step, populate it with data. Set the primary key as deferred and observe what happens at the end of the transaction.

Column Name	BOOK_ID	TITLE
Data Type	NUMBER	VARCHAR2
Length	7	20
Key	PRIMARY KEY	

   a. Observe that the `course_books_pk` primary key is not created as deferrable.

```
Table COURSE_BOOKS created.
```

   b. Populate data into the `course_books` table with the following two rows. What do you observe?

      - `300,'Organizations'`
      - `300,'Change Management'`

```
Error starting at line : 8 in command -
INSERT INTO course_books VALUES(300,'Change Management')
Error report -
SQL Error: ORA-00001: unique constraint (ORA02.COURSE_BOOKS_PK) violated
00001. 00000 -  "unique constraint (%s.%s) violated"
*Cause:    An UPDATE or INSERT statement attempted to insert a duplicate key.
           For Trusted Oracle configured in DBMS MAC mode, you may see
           this message if a duplicate entry exists at a different level.
*Action:   Either remove the unique restriction or do not insert the key.
```

c. Set the `course_books_pk` constraint as deferred. What do you observe?

```
Error starting at line : 9 in command -
SET CONSTRAINT course_books_pk DEFERRED
Error report -
SQL Error: ORA-02447: cannot defer a constraint that is not deferrable
02447. 00000 -  "cannot defer a constraint that is not deferrable"
*Cause:    An attempt was made to defer a nondeferrable constraint
*Action:   Drop the constraint and create a new one that is deferrable
```

d. Drop the `course_books_pk` constraint.

```
Table COURSE_BOOKS altered.
```

e. Modify the `course_books` table definition to add the `course_books_pk` constraint as deferrable this time.

```
Table COURSE_BOOKS altered.
```

f. Set the `course_books_pk` constraint as deferred.

```
Constraint COURSE_BOOKS_PK succeeded.
```

g. Populate data into the `course_books` table with the following rows. What do you observe?

- `300,'Change Management'`
- `300,'Personality'`
- `350,'Creativity'`

```
1 row inserted.

1 row inserted.

1 row inserted.
```

h. Commit the transaction. What do you observe?

```
Error starting at line : 21 in command -
commit
Error report -
SQL Error: ORA-02091: transaction rolled back
ORA-00001: unique constraint (ORA02.COURSE_BOOKS_PK) violated
02091. 00000 -  "transaction rolled back"
*Cause:    Also see error 2092. If the transaction is aborted at a remote
           site then you will only see 2091; if aborted at host then you will
           see 2092 and 2091.
*Action:   Add rollback segment and retry the transaction.
```

	BOOK_ID	TITLE
1	300	Organizations

Solution 20-1: Managing Constraints, Temporary Tables, and External Tables

Solution

1. Create the COURSE_DEPT table based on the following table instance chart. Enter the syntax in the SQL Worksheet. Then, execute the statement to create the table. Confirm that the table is created.

Column Name	ID	NAME
Data Type	NUMBER	VARCHAR2
Length	7	25

```
CREATE TABLE course_dept
(id NUMBER(7),name VARCHAR2(25));

DESCRIBE course_dept
```

2. Populate the COURSE_DEPT table with data from the AD_DEPARTMENT table. Include only the columns that you need.

```
INSERT INTO course_dept
      SELECT  department_id, department_name
      FROM    ad_department;

SELECT * FROM course_dept;
```

3. Create the COURSE table based on the following table instance chart. Enter the syntax in the SQL Worksheet. Then execute the statement to create the table. Confirm that the table is created.

Column Name	COURSE_ID	COURSE_NAME	DEPT_ID
Data Type	NUMBER	VARCHAR2	NUMBER
Length	7	25	7

```
CREATE TABLE  course
(course_id     NUMBER(7),
 course_name   VARCHAR2(25),
 dept_id       NUMBER(7));

DESCRIBE course
```

4. Add a table-level PRIMARY KEY constraint to the COURSE table on the COURSE_ID column. The constraint should be named at creation. Name the constraint course_id_pk.

```
ALTER TABLE    course
ADD CONSTRAINT course_id_pk PRIMARY KEY (course_id);
```

5. Create a PRIMARY KEY constraint on the COURSE_DEPT table by using the ID column. The constraint should be named at creation. Name the constraint course_dept_id_pk.

```
ALTER TABLE    course_dept
ADD CONSTRAINT course_dept_id_pk PRIMARY KEY(id);
```

6. Add a foreign key reference on the COURSE table that ensures that the course is not assigned to a nonexistent department. Name the constraint course_dept_id_fk.

```
ALTER TABLE course
ADD CONSTRAINT course_dept_id_fk
FOREIGN KEY (dept_id) REFERENCES course_dept(id);
```

7. Modify the COURSE table. Add a FEES column of the NUMBER data type with precision 2 and scale 9. Add a constraint to the FEES column that ensures that the value is greater than zero.

```
ALTER TABLE course
ADD fees NUMBER(9,2)
CONSTRAINT course_fess_ck CHECK (fees > 0);
```

8. Drop the COURSE and COURSE_DEPT tables so that they cannot be restored.

```
DROP TABLE course PURGE;
DROP TABLE course_dept PURGE;
```

9. Create an external table library_items_ext. Use the ORACLE_LOADER access driver.

   Note: The library_items.dat file is saved in the /home/oracle/emp_dir folder on your database file system. A directory object emp_dir is already created for this exercise and you have been granted READ and WRITE privileges on the same.

   library_items.dat has records in the following format:

```
2354,    2264, 13.21, 150,
2355,    2289, 46.23, 200,
2355,    2264, 50.00, 100,
```

a. Open the `lab_20_09.sql` file. Observe the code snippet to create the `library_items_ext` external table. Replace ***<TODO1>***, ***<TODO2>***, ***<TODO3>***, and ***<TODO4>*** as shown in the following code and save the file as `lab_20_09_soln.sql`. Run the script to create the external table.

```
CREATE TABLE library_items_ext ( category_id   number(12)
                               , book_id number(6)
                               , book_price number(8,2)
                               , quantity   number(8)
                               )
ORGANIZATION EXTERNAL
 (TYPE ORACLE_LOADER
  DEFAULT DIRECTORY emp_dir
  ACCESS PARAMETERS (RECORDS DELIMITED BY NEWLINE
                     FIELDS TERMINATED BY ',')
 LOCATION ('library_items.dat')
 )
REJECT LIMIT UNLIMITED;
```

b. Query the `library_items_ext` table.

```
SELECT * FROM library_items_ext;
```

10. Create the `course_books` table and populate it with data. Set the primary key as deferred and observe what happens at the end of the transaction.

a. Observe that the `course_books_pk` primary key is not created as deferrable.

Note: Ignore the error message, "table or view does not exist". `DROP TABLE` statement is given here just to make sure the table does not exist already.

```
DROP TABLE course_books CASCADE CONSTRAINTS;
CREATE TABLE course_books (book_id number(7),
                        title varchar2(20), CONSTRAINT
course_books_pk  PRIMARY KEY (book_id));
```

b. Run the following `INSERT` statements to populate data into the `course_books` table. What do you observe?

```
INSERT INTO course_books VALUES(300,'Organizations');
INSERT INTO course_books VALUES(300,'Change Management');
```

The first row is inserted. However, you see the `ora-00001` error with the insertion of the second row.

c. Set the `course_books_pk` constraint as deferred. What do you observe?

```
SET CONSTRAINT course_books_pk DEFERRED;
```

You see the following error: "ORA-02447: Cannot defer a constraint that is not deferrable."

d. Drop the `course_books_pk` constraint.

```
ALTER TABLE course_books DROP CONSTRAINT course_books_pk;
```

e. Modify the `course_books` table definition to add the `course_books_pk` constraint as deferrable this time.

```
ALTER TABLE course_books ADD (CONSTRAINT course_books_pk PRIMARY
KEY (book_id) DEFERRABLE);
```

f. Set the `course_books_pk` constraint as deferred.

```
SET CONSTRAINT course_books_pk DEFERRED;
```

g. Populate data into the `course_books` table by using `INSERT` statement. What do you observe?

```
INSERT INTO course_books VALUES (300,'Change Management');
INSERT INTO course_books VALUES (300,'Personality');
INSERT INTO course_books VALUES (350,'Creativity');
```

You see that all the rows are inserted.

h. Commit the transaction. What do you observe?

```
COMMIT;
SELECT * FROM course_books;
```

You see that the transaction is rolled back by the database at this point, because the `COMMIT` failed due to constraint violation.

Practices for Lesson 21: Using Advanced Subqueries

Chapter 21

Practices for Lesson 21: Overview

Practice Overview

This practice covers the following topics:

- Creating multiple-column subqueries
- Writing correlated subqueries
- Using the `EXISTS` operator
- Using scalar subqueries
- Using the `WITH` clause

Practice 21: Using Advanced Subqueries

Overview

In this practice, you write multiple-column subqueries, and correlated and scalar subqueries. You also solve problems by writing the `WITH` clause.

Tasks

1. Write a query to display the first name, parent ID, and registration date of any student whose parent ID and registration date match the parent ID and registration date of any student who does not have a valid email address.

	FIRST_NAME	PARENT_ID	STUDENT_REG_YEAR
1	RHONDA	620	01-SEP-14
2	JEANNE	610	01-MAR-14
3	ROBERT	610	01-MAR-14
4	NATHAN	640	01-JAN-16

2. Display the course name, department name, and session ID of any course whose department ID and `session_id` match the department ID and `session_id` of any course that comes under the department whose HOD is `MARK SMITH`.

	COURSE_NAME	DEPARTMENT_NAME	SESSION_ID
1	STRATEGIC TAX PLANNING FOR BUSINESS	ACCOUNTING	100
2	PRINCIPLES OF ACCOUNTING	ACCOUNTING	100
3	INTRODUCTION TO BUSINESS LAW	ACCOUNTING	100
4	COST ACCOUNTING	ACCOUNTING	100

3. Create a query to display the course ID and course name for all courses that have the same session ID and `department_ID` of Web Programming.

   Note: Do not display Web Programming in the result set.

	COURSE_ID	COURSE_NAME
1	188	OOAD
2	187	DATA STRUCTURES
3	198	SIMULATION AND MODELING

4. Create a query to display the faculty who earn a salary that is higher than the salary of all faculty with Mentor ID equal to 810 (`MENTOR_ID = 810`). Sort the results by salary from the highest to the lowest.

	FACULTY_ID	FACULTY_NAME	SALARY
1	800	JILL MILLER	24000
2	840	SALLY JONES	20850
3	810	JAMES BORG	17000
4	830	ARTHUR SMITH	11500

5. Display details such as the faculty ID, faculty name, and salary of faculty who teach courses with names beginning with “C.”

	FACULTY_ID	FACULTY_NAME	SALARY
1	800	JILL MILLER	24000
2	810	JAMES BORG	17000
3	830	ARTHUR SMITH	11500

6. Write a query to find all students who scored more than the average marks for a course. Display the student ID, marks, course ID, and the average marks for the course. Sort by average marks and round to two decimals. Use aliases for the columns retrieved by the query as shown in the sample output.

	REG_NO	MARKS	COURSE_ID	COURSE_AVG
1	760	70	192	62
2	770	91	188	78
3	720	97	193	79
4	770	99	187	79.5
5	730	87	199	81.5
6	710	91	176	84.5
7	720	91	190	85
8	750	97	175	87

7. Find all faculties who are not mentors.
   a. First, do this by using the `NOT EXISTS` operator.

	FACULTY_NAME
1	LYNN BROWN
2	ARTHUR SMITH
3	SALLY JONES

   b. Can this be done by using the `NOT IN` operator? How, or why not? If not, try out using another solution.

	FACULTY_NAME
1	LYNN BROWN
2	ARTHUR SMITH
3	SALLY JONES

8. Write a query to display the student ID and course ID of students who have scored less than the average marks in that course.

	STUDENT_ID	COURSE_ID
1	740	199
2	750	192
3	750	176
4	760	188
5	760	187
6	760	190
7	710	175
8	780	192
9	780	193

9. Write a query to display the exam type of exams that have the same name with later start dates but higher exam ID.

	EXAM_TYPE
1	MCE
2	SA
3	ESS
4	OB
5	FIB

10. Write a query to display the course ID, course names, and department names of all courses.

    Note: Use a scalar subquery to retrieve the department name in the `SELECT` statement.

	COURSE_ID	COURSE_NAME	DEPARTMENT
1	192	COST ACCOUNTING	ACCOUNTING
2	191	INTRODUCTION TO BUSINESS LAW	ACCOUNTING
3	190	PRINCIPLES OF ACCOUNTING	ACCOUNTING
4	193	STRATEGIC TAX PLANNING FOR BUSINESS	ACCOUNTING
5	196	INTRODUCTION TO PLANT PHYSIOLOGY	BIOLOGY
6	195	CELL BIOLOGY	BIOLOGY
7	194	GENERAL BIOLOGY	BIOLOGY
8	197	MARINE BIOLOGY	BIOLOGY
9	198	SIMULATION AND MODELING	COMPUTER SCIENCE
10	199	WEB PROGRAMMING	COMPUTER SCIENCE
11	187	DATA STRUCTURES	COMPUTER SCIENCE
12	188	OOAD	COMPUTER SCIENCE
13	189	COLLEGE READING	LITERATURE
14	176	BUSINESS WRITING	LITERATURE
15	175	AMERICAN LITERATURE	LITERATURE

11. Write a query to display the course names of courses where the total marks scored by a student for the course is above one-twelfth (1/12) of the total marks scored by students in all courses. Use the `WITH` clause to write this query. Name the query `SUMMARY`.

	COURSE_NAME	COURSE_TOTAL
1	COST ACCOUNTING	186
2	AMERICAN LITERATURE	174
3	PRINCIPLES OF ACCOUNTING	170
4	BUSINESS WRITING	169
5	WEB PROGRAMMING	163
6	DATA STRUCTURES	159
7	STRATEGIC TAX PLANNING FOR BUSINESS	158
8	OOAD	156

Solution 21: Using Advanced Subqueries

Solution

1. Write a query to display the first name, parent ID, and registration date of any student whose parent ID and registration date match the parent ID and registration date of any student who does not have a valid email address.

```
SELECT first_name, parent_id, student_reg_year
FROM   ad_student_details
WHERE (parent_id, student_reg_year) IN
          (SELECT parent_id, student_reg_year
           FROM    ad_student_details
           WHERE   email_addr IS NULL);
```

2. Display the course name, department name, and session ID of any course whose department ID and `session_id` match the department ID and `session_id` of any course that comes under the department whose HOD is `MARK SMITH`.

```
SELECT c.course_name, d.department_name, c.session_id
FROM   ad_course_details c JOIN ad_department d
ON  c.department_id = d.department_id
AND  (c.department_id, session_id) IN
                       (SELECT c.department_id, c.session_id
                        FROM ad_course_details c JOIN
                        ad_department d
                        ONc.department_id = d.department_id
                        AND     d.hod = 'MARK SMITH');
```

3. Create a query to display the course ID and course name for all courses that have the same session ID and `department_ID` of Web Programming.

   Note: Do not display Web Programming in the result set.

```
SELECT course_id, course_name
FROM   ad_course_details
WHERE (session_id, department_id) IN
             (SELECT session_id, department_id
              FROM     ad_course_details
              WHERE course_name = 'WEB PROGRAMMING')
              AND course_name != 'WEB PROGRAMMING';
```

4. Create a query to display the faculty who earn a salary that is higher than the salary of all faculty with Mentor ID equal to 810 (`MENTOR_ID = 810`). Sort the results by salary from the highest to the lowest.

```
SELECT faculty_id, faculty_name, salary
    FROM   ad_faculty_details
    WHERE  salary > ALL
                (SELECT salary
                 FROM   ad_faculty_details
    WHERE  mentor_id = 810)
    ORDER BY salary DESC;
```

5. Display details such as the faculty ID, faculty name, and salary of faculty who teach courses with names beginning with “C.”

```
     SELECT faculty_id, faculty_name, salary
     FROM   ad_faculty_details
     WHERE  faculty_id IN (SELECT faculty_id
                           FROM ad_faculty_course_details
                           WHERE course_id IN
                               (SELECT  course_id
                               FROM ad_course_details
                               WHERE course_name LIKE 'C%'));
```

6. Write a query to find all students who scored more than the average marks for a course. Display the student ID, marks, course ID, and the average marks for the course. Sort by average marks and round to two decimals. Use aliases for the columns retrieved by the query as shown in the sample output.

```
     SELECT e.student_id reg_no, e.marks marks, e.course_id,
            ROUND(AVG(a.marks),2) course_avg
       FROM   ad_exam_results e, ad_exam_results a
       WHERE e.course_id = a.course_id
       AND    e.marks > (SELECT AVG(marks)
                  FROM   ad_exam_results
                  WHERE course_id = e.course_id)
       GROUP BY e.student_id, e.marks, e.course_id
       ORDER BY AVG(a.marks);
```

7. Find all faculties who are not mentors.
   a. First, do this by using the NOT EXISTS operator.

```
SELECT outer.faculty_name
FROM    ad_faculty_details outer
WHERE  NOT EXISTS (SELECT 'X'
                   FROM ad_faculty_details inner
                    WHERE inner.mentor_id =
                          outer.faculty_id);
```

   b. Can this be done by using the NOT IN operator? How, or why not?

```
SELECT outer.faculty_name
FROM   ad_faculty_details outer
WHERE  outer.faculty_id
NOT IN (SELECT inner.mentor_id
        FROM   ad_faculty_details inner);
```

   This alternative solution is not a good one. The subquery picks up a NULL value, so the entire query returns no rows. The reason is that all conditions that compare a NULL value result in NULL. Whenever NULL values are likely to be part of the value set, *do not* use NOT IN as a substitute for EXISTS. A much better solution would be a subquery such as the following:

```
SELECT faculty_name
FROM ad_faculty_details
WHERE faculty_id NOT IN (SELECT mentor_id
                         FROM ad_faculty_details WHERE
                         mentor_id IS NOT NULL);
```

8. Write a query to display the student ID and course ID of students who have scored less than the average marks in a course.

```
SELECT student_id, course_id
FROM    ad_exam_results outer
WHERE outer.marks < (SELECT AVG(inner.marks)
                     FROM ad_exam_results inner
                     WHERE inner.course_id
                           = outer.course_id);
```

9. Write a query to display the exam type of exams that have the same name with later start dates but higher exam ID.

```
SELECT  exam_type
 FROM    ad_exam_details outer
 WHERE EXISTS (SELECT 'X'
               FROM ad_exam_details inner
               WHERE inner.name =
                     outer.name
               AND inner.start_date > outer.start_date
               AND inner.exam_id > outer.exam_id);
```

10. Write a query to display the course ID, course names, and department names of all courses.

    Note: Use a scalar subquery to retrieve the department name in the `SELECT` statement.

```
SELECT course_id, course_name,
      (SELECT department_name
        FROM ad_department d
       WHERE   c.department_id =
           d.department_id ) department
FROM ad_course_details c
ORDER BY department;
```

11. Write a query to display the course names of courses where the total marks scored by a student for the course is above one-twelfth (1/12) of the total marks scored by students in all courses. Use the `WITH` clause to write this query. Name the query `SUMMARY`.

```
WITH
summary AS (
  SELECT d.course_name, SUM(e.marks) AS course_total
  FROM ad_exam_results e JOIN ad_course_details d
  ON e.course_id = d.course_id
  GROUP BY d.course_name)
SELECT course_name, course_total
FROM summary
WHERE course_total > ( SELECT SUM(course_total) * 1/12
                    FROM summary  )
ORDER BY course_total DESC;
```

Practices for Lesson 22: Manipulating Data by Using Advanced Subqueries

Chapter 22

Practices for Lesson 22: Overview

Practice Overview

This practice covers the following topics:

- Using subqueries to manipulate data
- Inserting by using a subquery as a target
- Using the `WITH CHECK OPTION` keyword on DML statements
- Using correlated subqueries to update and delete rows

Practice 22: Manipulating Data by Using Advanced Subqueries

Overview

In this practice, you test your knowledge about using subqueries to manipulate data, using the `WITH CHECK OPTION` keyword on DML statements, and using correlated subqueries to update and delete rows.

Tasks

1. Which of the following statements are true?
   a. Subqueries are used to retrieve data by using an inline view.
   b. Subqueries cannot be used to copy data from one table to another.
   c. Subqueries update data in one table based on the values of another table.
   d. Subqueries delete rows from one table based on rows in another table.

2. Fill in the blanks:
   a. You can use a subquery in place of the table name in the _____ clause of the `INSERT` statement.

   Options:

   1) `FROM`
   2) `INTO`
   3) `FOR UPDATE`
   4) `VALUES`

3. The `WITH CHECK OPTION` keyword prohibits you from changing rows that are not in the subquery.
   a. `TRUE`
   b. `FALSE`

4. The `SELECT` list of this subquery must have the same number of columns as the column list of the `VALUES` clause.
   a. `TRUE`
   b. `FALSE`

5. You can use a correlated subquery to delete only those rows that also exist in another table.
   a. `TRUE`
   b. `FALSE`

6. Write a query by using `WITH CHECK OPTION` to insert a record into the `ad_exam_results` table for a student who has scored 40 marks in the `MCE` type of exam with course ID `191`.

```
1 row inserted.
```

Solution 22: Manipulating Data by Using Advanced Subqueries

1. Which of the following statements are true?
   a. Subqueries are used to retrieve data by using an inline view.
   b. Subqueries cannot be used to copy data from one table to another.
   c. Subqueries update data in one table based on the values of another table.
   d. Subqueries delete rows from one table based on rows in another table.

Answer: a, c, and d

2. Fill in the blanks:
   a. You can use a subquery in place of the table name in the _____ clause of the `INSERT` statement.

Options:

1) `FROM`
2) `INTO`
3) `FOR UPDATE`
4) `VALUES`

Answer: 2

3. The `WITH CHECK OPTION` keyword prohibits you from changing rows that are not in the subquery.
   a. `TRUE`
   b. `FALSE`

Answer: a

4. The `SELECT` list of this subquery must have the same number of columns as the column list of the `VALUES` clause.
   a. `TRUE`
   b. `FALSE`

Answer: a

5. You can use a correlated subquery to delete only those rows that also exist in another table.
   a. `TRUE`
   b. `FALSE`

Answer: a

6. Write a query by using `WITH CHECK OPTION` to insert a record into the `ad_exam_results` table for a student who has scored 40 marks in the `MCE` type of exam with course ID `191`.

```
INSERT INTO (SELECT student_id, exam_id, course_id, marks
             FROM   ad_exam_results
             WHERE  exam_id IN
             (SELECT exam_id
              FROM ad_exam_details
              NATURAL JOIN ad_exam_type
              WHERE exam_name = 'Multiple Choice Exams')
              WITH CHECK OPTION)
VALUES (740, 500, 191, 35);
```

Practices for Lesson 23: Controlling User Access

Chapter 23

Practices for Lesson 23: Overview

Practice Overview

This practice covers the following topics:

- Creating a new user
- Granting the user system privileges through a predefined role
- Granting the user privileges to your table
- Accessing data in the new user's SQL Developer session

Practice 23-1: Controlling User Access

Overview

You have been designated as the project lead. In the role of project lead, you need to ensure that your team has access to the pertinent database information. You grant query privilege on your table to another user.

Tasks

1. What privilege should a user be given to log on to the Oracle server? Is this a system privilege or an object privilege?

   _____

2. What privilege should a user be given to create tables?

   _____

3. If you create a table, who can pass along privileges to other users in your table?

   _____

4. You are the DBA. You create many users who require the same system privileges. What should you use to make your job easier?

   _____

5. `User21` is the owner of the `EMP` table and grants the `DELETE` privilege to `User22` by using the `WITH GRANT OPTION` clause. `User22` then grants the `DELETE` privilege on `EMP` to `User23`. `User21` now finds that `User23` has the privilege and revokes it from `User22`. Which user can now delete data from the `EMP` table?

   _____

6. You want to grant `SCOTT` the privilege to update data in the `DEPARTMENTS` table. You also want to enable `SCOTT` to grant this privilege to other users. What command do you use?

   _____

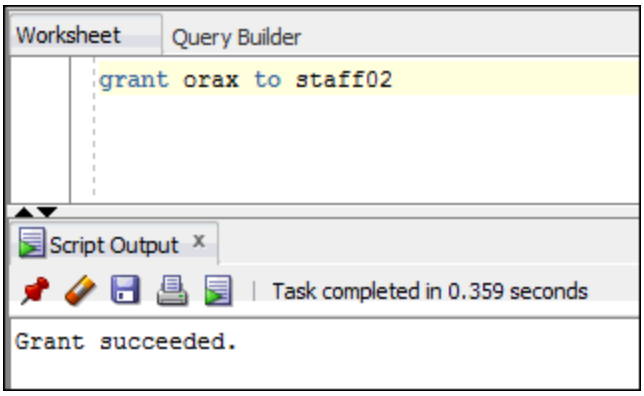
To complete question 7 and the subsequent questions, you need to connect to the database by using SQL Developer. If you are not already connected, do the following to connect:

1. Click the SQL Developer desktop icon.
2. In the Connections Navigator, use your *oraxx* account and the corresponding password provided by your instructor to log on to the database.

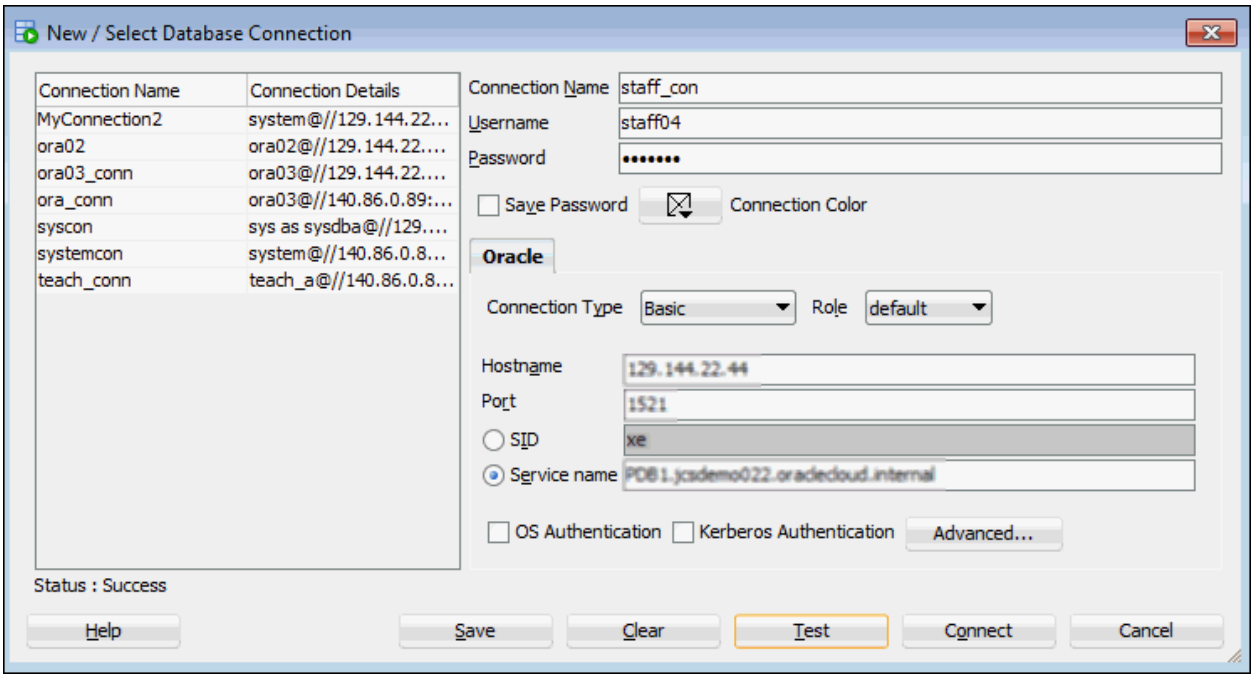
7. You want the University staff to be able to access the student details. First, you need to create a new user who will have access to the tables containing the student details. Create a new user, `staffxx` (append your `ORA` user number at the end of the name; for example, if you are using the `ORA02` account, create the new user as `staff02`).

```
User STAFF02 created.
```

8. The new user does not have any system privileges. To log on to Oracle Database, a user must have the `CREATE SESSION` system privilege. To make sure the new user has all the privileges required for this practice, a role `orax` (there is no need to replace the x with any number; the role name is `orax`) was already created for you. Grant this role to the new user.



9. Open a new SQL Developer session by clicking the SQL Developer desktop icon. Create a new connection, `staff_con`. Enter the connection details provided to you by your instructor. Test the connection. Click Connect.



10. Now, try to access the student records from the `AD_STUDENT_DETAILS` table using the new connection, `staff_con`. Run a simple SQL select statement to retrieve all the records from the `ad_student_details` table. Check the displayed error.

```
ORA-01031: insufficient privileges
01031. 00000 -  "insufficient privileges"
*Cause:    An attempt was made to perform a database operation without
           the necessary privileges.
*Action:   Ask your database administrator or designated security
           administrator to grant you the necessary privileges
Error at Line: 1 Column: 21
```

11. Go back to your previous SQL Developer connection. Grant `select` privileges on the tables you want the `staffxx` user to have access to. For now, grant `select` privileges on the `ad_student_details` table to the `staffxx` user. Commit the changes.
12. Switch back to the `staff_con` SQL Developer session. Now, try to access the students' records by running the same simple SQL select statement you executed before the user was granted the `select` privilege.

	STUDENT_ID	FIRST_NAME	PARENT_ID	STUDENT_REG_YEAR	EMAIL_ADDR
1	720	JACK	600	01-JAN-14	jack@email.com
2	740	RHONDA	620	01-SEP-14	(null)
3	750	ROBERT	610	01-MAR-14	robert@email.com
4	760	JEANNE	610	01-MAR-14	(null)
5	770	MILLS	630	01-APR-15	mills@email.com
6	710	NINA	630	01-JAN-13	nina@email.com
7	780	NATHAN	640	01-JAN-16	(null)
8	730	NOAH	640	01-JUN-14	noah@email.com

13. Switch back to original session and take back the privileges on the `AD_STUDENT_DETAILS` table granted to the `staffxx` user.

Solution 23-1: Controlling User Access

1. What privilege should a user be given to log on to the Oracle server? Is this a system or an object privilege?
   The `CREATE SESSION` system privilege

2. What privilege should a user be given to create tables?
   The `CREATE TABLE` privilege

3. If you create a table, who can pass along privileges to other users in your table?
   You or anyone you have given those privileges to by using `WITH GRANT OPTION`

4. You are the DBA. You create many users who require the same system privileges. What should you use to make your job easier?
   Create a role containing the system privileges and grant the role to the users.

5. `User21` is the owner of the `EMP` table and grants `DELETE` privileges to `User22` by using the `WITH GRANT OPTION` clause. `User22` then grants `DELETE` privileges on `EMP` to `User23`. `User21` now finds that `User23` has the privilege and revokes it from `User22`. Which user can now delete data from the `EMP` table?

   Only `User21`

6. You want to grant `SCOTT` the privilege to update data in the `DEPARTMENTS` table. You also want to enable `SCOTT` to grant this privilege to other users. What command do you use?

```
GRANT UPDATE ON departments TO scott WITH GRANT OPTION;
```

To complete question 7 and the subsequent questions, you need to connect to the database by using SQL Developer. If you are not already connected, do the following to connect:

a. Click the SQL Developer desktop icon.

b. In the Connections Navigator, use your *oraxx* account and the corresponding password provided by your instructor to log on to the database.

7. You want the University staff to be able to access the student details. First, you need to create a new user who will have access to the tables containing the student details. Create a new user, `staffxx` (append your `ORA` user number at the end of the name; for example, if you are using the `ORA02` account, create the new user as `staff02`).

```
create user staffxx identified by staffxx;
```

8. The new user does not have any system privileges. To log on to Oracle Database, a user must have the `CREATE SESSION` system privilege. To make sure the new user has all the privileges required for this practice, a role `orax` (there is no need to replace the x with any number; the role name is `orax`) was already created for you. Grant this role to the new user.

```
GRANT orax
to staffxx;
```

9. Open a new SQL Developer session by clicking the SQL Developer desktop icon. Create a new connection, `staff_con`. Enter the connection details provided to you by your instructor. Test the connection. Click Connect.

10. Now, try to access the student records from the `AD_STUDENT_DETAILS` table using the new connection, `staff_con`. Run a simple SQL select statement to retrieve all the records from the `ad_student_details` table.

```
select * from oraxx.ad_student_details;
```

Note that you get an “insufficient privileges” error. This is because the `oraxx` user has not granted `select` privileges on the `ad_student_details` table to the `staffxx` user.

11. Go back to your previous SQL Developer connection. Grant `select` privileges on the tables you want the `staffxx` user to have access to. For now, grant `select` privileges on the `ad_student_details` table to the `staffxx` user. Commit the changes.

```
grant select on ad_student_details to staffxx;
commit;
```

12. Switch back to the `staff_con` SQL Developer session. Now, try to access the students’ records by running a simple SQL select statement.

```
Select * from oraxx.ad_student_details;
```

Note that the `staffxx` user is now able to view the student records.

13. Switch back to original session and take back the privileges on the `AD_STUDENT_DETAILS` table granted to the `staffxx` user.

```
REVOKE select on ad_student_details from staffxx;
```