

## INTRODUCTION TO SQL

Pronounced as SEQUEL: Structured English QUERY Language

- Pure non-procedural query language
- Designed and developed by IBM, Implemented by Oracle
- 1978 System/R IBM- 1st Relational DBMS
- 1979 Oracle and Ingres
- 1982 SQL/DS and DB2 IBM
- Accepted by both ANSI + ISO as **Standard Query Language** for any RDBMS
- SQL86 (SQL1) : first by ANSI and ratified by ISO (SQL-87), minor revision on 89 (SQL-89)
- SQL92 (SQL2) : major revision
- SQL99 (SQL3) : add recursive query, trigger, some OO features, and non-scholar type
- SQL2003 : XML, Window functions, and sequences (Not free)
- Supports all the three sublanguages of DBMS: **DDL, DML, DCL**
- Supports Aggregate functions, String Manipulation functions, Set theory operations, Date Manipulation functions, rich set of operators ( IN, BETWEEN, LIKE, IS NULL, EXISTS)
- Supports REPORT writing features and Forms for designing GUI based applications

### DATA DEFINITION, CONSTRAINTS, AND SCHEMA CHANGES

Used to CREATE, ALTER, and DROP the descriptions of the database tables (relations)

#### **Data Definition in SQL**

#### CREATE, ALTER and DROP

table.....relation  
row.....tuple  
column.....attribute

#### DATA TYPES

- Numeric: NUMBER, NUMBER(s,p), INTEGER, INT, FLOAT, DECIMAL
- Character: CHAR(n), VARCHAR(n), VARCHAR2(n), CHAR VARYING(n)
- Bit String: BLOB, CLOB
- Boolean: true, false, and null

- Date and Time: DATE (YYYY-MM-DD) TIME( HH:MM:SS)
- Timestamp: DATE + TIME
- USER Defined types

### **CREATE SCHEMA**

Specifies a new database schema by giving it a name

Ex: CREATE SCHEMA COMPANY AUTHORIZATION Jsmith;

### **CREATE TABLE**

- Specifies a new base relation by giving it a name, and specifying each of its attributes and their data types

Syntax of CREATE Command:

**CREATE TABLE** <table name> ( <Attribute A1> <Data Type D1> [< Constarints>],  
    <Attribute A2> <Data Type D2> [< Constarints>],  
    .....  
    <Attribute An> <Data Type Dn> [< Constarints>],  
    [<integrity-constraint1>, <integrity-constraint k> ] );

- A constraint NOT NULL may be specified on an attribute

A constraint NOT NULL may be specified on an attribute

Ex: CREATE TABLE DEPARTMENT (  
    DNAME VARCHAR(10) NOT NULL,  
    DNUMBER INTEGER NOT NULL,  
    MGRSSN CHAR(9), MGRSTARTDATE CHAR(9) );

- Specifying the unique, primary key attributes, secondary keys, and referential integrity constraints (foreign keys).

Ex: CREATE TABLE DEPT (  
    DNAME VARCHAR(10) NOT NULL,  
    DNUMBER INTEGER NOT NULL,  
    MGRSSN CHAR(9),  
    MGRSTARTDATE CHAR(9),  
    PRIMARY KEY (DNUMBER),  
    UNIQUE (DNAME),

FOREIGN KEY (MGRSSN) REFERENCES EMP(SSN));

- We can specify RESTRICT, CASCADE, SET NULL or SET DEFAULT on referential integrity constraints (foreign keys)

Ex: CREATE TABLE DEPT (  
DNAME VARCHAR(10) NOT NULL,  
DNUMBER INTEGER NOT NULL,  
MGRSSN CHAR(9), MGRSTARTDATE CHAR(9),  
PRIMARY KEY (DNUMBER),  
UNIQUE (DNAME),  
FOREIGN KEY (MGRSSN) REFERENCES EMP  
ON DELETE SET DEFAULT ON UPDATE CASCADE);

### **DROP TABLE**

- Used to remove a relation (base table) and its definition.
- The relation can no longer be used in queries, updates, or any other commands since its description no longer exists

**Example:** DROP TABLE DEPENDENT;

### **ALTER TABLE:**

- Used to add an attribute to/from one of the base relations drop constraint -- The new attribute will have NULLs in all the tuples of the relation right after the command is executed; hence, the NOT NULL constraint is *not allowed* for such an attribute.

**Example:** ALTER TABLE EMPLOYEE ADD JOB VARCHAR2 (12);

- The database users must still enter a value for the new attribute JOB for each EMPLOYEE tuple. This can be done using the UPDATE command.

### **DROP A COLUMN (AN ATTRIBUTE)**

- ALTER TABLE COMPANY.EMPLOYEE DROP ADDRESS CASCADE; All constraints and views that reference the column are dropped automatically, along with the column. ALTER TABLE COMPANY.EMPLOYEE DROP ADDRESS RESTRICT; Successful if no views or constraints reference the column. ALTER TABLE COMPANY.DEPARTMENT ALTER MGRSSN DROP DEFAULT;

- ALTER TABLE COMPANY.DEPARTMENT ALTER MGRSSN SET DEFAULT “333445555”;

### **BASIC QUERIES IN SQL**

- SQL has one basic statement for retrieving information from a database; the SLELECT statement
- This is *not the same as* the SELECT operation of the relational algebra
- Important distinction between SQL and the formal relational model;
- SQL allows a table (relation) to have two or more tuples that are identical in all their attribute values
- Hence, an SQL relation (table) is a *multi-set* (sometimes called a bag) of tuples; it is *not* a set of tuples
- SQL relations can be constrained to be sets by using the CREATE UNIQUE INDEX command, or by using the DISTINCT option
- Basic form of the SQL SELECT statement is called a *mapping* of a *SELECT-FROM-WHERE block*

SELECT <attribute list> FROM <table list> WHERE <condition>

- <attribute list> is a list of attribute names whose values are to be retrieved by the query
- <table list > is a list of the relation names required to process the query
- <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query

### **SIMPLE SQL QUERIES**

Basic SQL queries correspond to using the following operations of the relational algebra:

SELECT

PROJECT

JOIN

All subsequent examples uses COMPANY database as shown below:

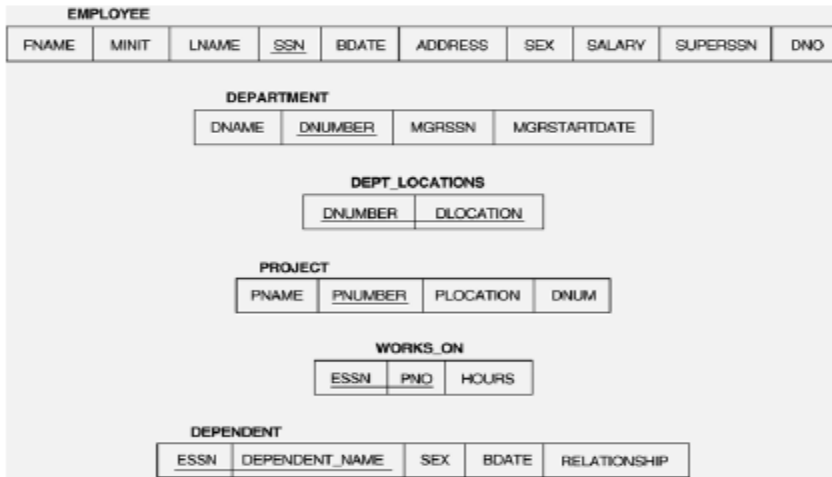
#### **Example of a simple query on one relation**

**Query 0: Retrieve the birth date and address of the employee whose name is 'John B. Smith'.**

Q0: SELECT BDATE, ADDRESS FROM EMPLOYEE

WHERE FNAME='John' AND MINIT='B' AND LNAME='Smith'

Similar to a SELECT-PROJECT pair of relational algebra operations: The SELECT-clause specifies the projection attributes and the WHERE-clause specifies the selection condition. However, the result of the query may contain duplicate tuples.



EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	1955-01-09	731 Fondren, Houston, TX	M	30000	333445555	5	
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5	
Alicia	J	Zelaya	999987777	1969-07-19	3321 Castle, Spring, TX	F	25000	987654321	4	
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Belaire, TX	F	43000	888665555	4	
Ramoth	K	Narsyan	888884444	1982-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5	
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5	
Ahmad	V	Jabbar	987987987	1959-03-29	960 Dallas, Houston, TX	M	25000	987654321	4	
James	E	Borg	888665555	1937-11-10	400 Stone, Houston, TX	M	55000	null	1	

DEPT_LOCATIONS	DNUMBER	DLOCATION
	1	Houston
	4	Storford
	5	Belaire
	5	Sugarland
	5	Houston

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-05-19

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	888884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Belaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Storford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Storford	4

DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	1986-04-05	DAUGHTER
	333445555	Theodore	M	1983-10-25	SON
	333445555	Joy	F	1959-05-03	SPOUSE
	987654321	Abner	M	1942-02-28	SPOUSE
	123456789	Michael	M	1988-01-04	SON
	123456789	Alice	F	1988-12-30	DAUGHTER
	123456789	Elizabeth	F	1967-05-05	SPOUSE

### **Example of a simple query on two relations**

**Query 1: Retrieve the name and address of all employees who work for the 'Research' department.**

```
Q1: SELECT FNAME, LNAME, ADDRESS FROM EMPLOYEE, DEPARTMENT
WHERE DNAME='Research' AND DNUMBER=DNO
```

Similar to a SELECT-PROJECT-JOIN sequence of relational algebra operations (DNAME='Research') is a selection condition (corresponds to a SELECT operation in relational algebra) (DNUMBER=DNO) is a join condition (corresponds to a JOIN operation in relational algebra)

### **Example of a simple query on three relations**

**Query 2: For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.**

```
Q2: SELECT PNUMBER, DNUM, LNAME, BDATE, ADDRESS FROM PROJECT,
DEPARTMENT, EMPLOYEE WHERE DNUM=DNUMBER AND MGRSSN=SSN
AND PLOCATION='Stafford'
```

In Q2, there are two join conditions The join condition DNUM=DNUMBER relates a project to its controlling department The join condition MGRSSN=SSN relates the controlling department to the employee who manages that department

### **ALIASES, \* AND DISTINCT, EMPTY WHERE-CLAUSE**

- In SQL, we can use the same name for two (or more) attributes as long as the attributes are in different relations
- A query that refers to two or more attributes with the same name must qualify the attribute name with the relation name by prefixing the relation name to the attribute name  
**Example:** EMPLOYEE.LNAME, DEPARTMENT.DNAME
- Some queries need to refer to the same relation twice. In this case, aliases are given to the relation name

#### **Example**

**Query 3: For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.**

```
Q3: SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME FROM EMPLOYEE E S
WHERE E.SUPERSSN=S.SSN
```

In Q3, the alternate relation names E and S are called aliases or tuple variables for the EMPLOYEE relation. We can think of E and S as two different copies of EMPLOYEE; E represents employees in role of supervisees and S represents employees in role of supervisors.

Aliasing can also be used in any SQL query for convenience. Can also use the AS keyword to specify aliases.

```
Q3: SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME FROM EMPLOYEE AS
E, EMPLOYEE AS S WHERE E.SUPERSSN=S.SSN
```

### **UNSPECIFIED WHERE-clause**

A missing WHERE-clause indicates no condition; hence, all tuples of the relations in the FROM-clause are selected. This is equivalent to the condition WHERE TRUE.

Example:

**Query 4: Retrieve the SSN values for all employees.**

```
Q4: SELECT SSN FROM EMPLOYEE
```

If more than one relation is specified in the FROM-clause and there is no join condition, then the CARTESIAN PRODUCT of tuples is selected.

Example:

```
Q5: SELECT SSN, DNAME FROM EMPLOYEE, DEPARTMENT
```

**Note:** It is extremely important not to overlook specifying any selection and join conditions in the WHERE-clause; otherwise, incorrect and very large relations may result.

### **USE OF \***

To retrieve all the attribute values of the selected tuples, a \* is used, which stands for all the attributes.

Examples:

**Retrieve all the attribute values of EMPLOYEES who work in department 5.**

```
Q1a: SELECT * FROM EMPLOYEE WHERE DNO=5
```

**Retrieve all the attributes of an employee and attributes of DEPARTMENT he works in for every employee of 'Research' department.**

Q1b: SELECT \* FROM EMPLOYEE, DEPARTMENT WHERE DNAME='Research'  
AND DNO=DNUMBER

### USE OF DISTINCT

SQL does not treat a relation as a set; duplicate tuples can appear. To eliminate duplicate tuples in a query result, the keyword DISTINCT is used

Example: the result of **Q1c** may have duplicate SALARY values whereas **Q1d** does not have any duplicate values

Q1c: SELECT SALARY FROM EMPLOYEE Q1d: SELECT **DISTINCT**  
SALARY FROM EMPLOYEE

### SET OPERATIONS

SQL has directly incorporated some set operations such as union operation (UNION), set difference (MINUS) and intersection (INTERSECT) operations. The resulting relations of these set operations are sets of tuples; duplicate tuples are eliminated from the result. The set operations apply only to union compatible relations; the two relations must have the same attributes and the attributes must appear in the same order

**Query 5: Make a list of all project numbers for projects that involve an employee whose last name is 'Smith' as a worker or as a manager of the department that controls the project.**

Q5: (SELECT PNAME FROM PROJECT, DEPARTMENT, EMPLOYEE WHERE  
DNUM=DNUMBER AND MGRSSN=SSN AND LNAME='Smith')

**UNION**

(SELECT PNAME FROM PROJECT, WORKS\_ON, EMPLOYEE WHERE  
PNUMBER=PNO AND ESSN=SSN AND NAME='Smith')

### NESTING OF QUERIES

A complete SELECT query, called a nested query, can be specified within the WHERE-clause of another query, called the outer query. Many of the previous queries can be specified in an alternative form using nesting

**Query 6: Retrieve the name and address of all employees who work for the 'Research' department.**



```
Q6: SELECT FNAME, LNAME, ADDRESS FROM EMPLOYEE WHERE DNO IN
(SELECT DNUMBER FROM DEPARTMENT WHERE DNAME='Research' )
```

**Note:** The nested query selects the number of the 'Research' department. The outer query selects an EMPLOYEE tuple if its DNO value is in the result of either nested query. The comparison operator IN compares a value v with a set (or multi-set) of values V, and evaluates to TRUE if v is one of the elements in V

In general, we can have several levels of nested queries. A reference to an unqualified attribute refers to the relation declared in the innermost nested query. In this example, the nested query is not correlated with the outer query

### **CORRELATED NESTED QUERIES**

If a condition in the WHERE-clause of a nested query references an attribute of a relation declared in the outer query, the two queries are said to be correlated. The result of a correlated nested query is different for each tuple (or combination of tuples) of the relation(s) the outer query

**Query 7: Retrieve the name of each employee who has a dependent with the same first name as the employee.**

```
Q7: SELECT E.FNAME, E.LNAME FROM EMPLOYEE AS E WHERE E.SSN IN
(SELECT ESSN FROM DEPENDENT WHERE ESSN=E.SSN AND
E.FNAME=DEPENDENT_NAME)
```

In Q7, the nested query has a different result in the outer query. A query written with nested SELECT... FROM... WHERE... blocks and using the = or IN comparison operators can *always* be expressed as a single block query. For example, Q7 may be written as in Q7a

```
Q7a: SELECT E.FNAME, E.LNAME FROM EMPLOYEE E, DEPENDENT D
WHERE E.SSN=D.ESSN AND E.FNAME=D.DEPENDENT_NAME
```

### **THE EXISTS FUNCTION**

EXISTS is used to check whether the result of a correlated nested query is empty (contains no tuples) or not. We can formulate Query 7 in an alternative form that uses EXIST.

```
Q7b: SELECT FNAME, LNAME FROM EMPLOYEE
WHERE EXISTS (SELECT * FROM DEPENDENT WHERE SSN=ESSN
AND FNAME=DEPENDENT_NAME)
```

**Query 8: Retrieve the names of employees who have no dependents.**

```
Q8: SELECT FNAME, LNAME FROM EMPLOYEE
WHERE NOT EXISTS
(SELECT * FROM DEPENDENT WHERE SSN=ESSN)
```

**Note:** In Q8, the correlated nested query retrieves all DEPENDENT tuples related to an EMPLOYEE tuple. If none exist, the EMPLOYEE tuple is selected

### **EXPLICIT SETS**

It is also possible to use an explicit (enumerated) set of values in the WHERE-clause rather than a nested query

**Query 9: Retrieve the social security numbers of all employees who work on project number 1, 2, or 3.**

```
Q9: SELECT DISTINCT ESSN FROM WORKS_ON WHERE PNO IN (1, 2, 3)
```

### **NULLS IN SQL QUERIES**

SQL allows queries that check if a value is NULL (missing or undefined or not applicable). SQL uses IS or IS NOT to compare NULLs because it considers each NULL value distinct from other NULL values, so equality comparison is not appropriate.

**Query 10: Retrieve the names of all employees who do not have supervisors.**

```
Q10: SELECT FNAME, LNAME FROM EMPLOYEE
WHERE SUPERSSN IS NULL
```

**Note:** If a join condition is specified, tuples with NULL values for the join attributes are not included in the result

### **AGGREGATE FUNCTIONS**

Include COUNT, SUM, MAX, MIN, and AVG

**Query 11: Find the maximum salary, the minimum salary, and the average salary among all employees.**

```
Q11: SELECT MAX (SALARY), MIN(SALARY), AVG(SALARY)
FROM EMPLOYEE
```

**Note:** Some SQL implementations may not allow more than one function in the SELECT-clause

**Query 12: Find the maximum salary, the minimum salary, and the average salary among employees who work for the 'Research' department.**

```
Q12: SELECT MAX (SALARY), MIN(SALARY), AVG(SALARY) FROM
EMPLOYEE, DEPARTMENT WHERE DNO=DNUMBER AND DNAME='Research'
```

**Queries 13 and 14: Retrieve the total number of employees in the company (Q13), and the number of employees in the 'Research' department (Q14).**

```
Q13: SELECT COUNT (*) FROM EMPLOYEE
```

```
Q14: SELECT COUNT (*) FROM EMPLOYEE, DEPARTMENT
WHERE DNO=DNUMBER AND DNAME='Research'
```

### **GROUPING**

- In many cases, we want to apply the aggregate functions to subgroups of tuples in a relation
- Each subgroup of tuples consists of the set of tuples that have the same value for the grouping attribute(s)
- The function is applied to each subgroup independently
- SQL has a GROUP BY-clause for specifying the grouping attributes, which must also appear in the SELECT-clause

**Query 15: For each department, retrieve the department number, the number of employees in the department, and their average salary.**

```
Q15: SELECT DNO, COUNT (*), AVG (SALARY)
FROM EMPLOYEE GROUP BY DNO
```

- In Q15, the EMPLOYEE tuples are divided into groups. Each group having the same value for the grouping attribute DNO
- The COUNT and AVG functions are applied to each such group of tuples separately
- The SELECT-clause includes only the grouping attribute and the functions to be applied on each group of tuples
- A join condition can be used in conjunction with grouping

**Query 16: For each project, retrieve the project number, project name, and the number of employees who work on that project.**

```
Q16: SELECT PNUMBER, PNAME, COUNT (*)
```

```
FROM PROJECT, WORKS_ON
WHERE PNUMBER=PNO

GROUP BY PNUMBER, PNAME
```

### **THE HAVING-CLAUSE**

Sometimes we want to retrieve the values of these functions for only those groups that satisfy certain conditions. The HAVING-clause is used for specifying a selection condition on groups (rather than on individual tuples)

**Query 17: For each project on which more than two employees work, retrieve the project number, project name, and the number of employees who work on that project.**

```
Q17: SELECT PNUMBER, PNAME, COUNT (*)
FROM PROJECT, WORKS_ON
WHERE PNUMBER=PNO
GROUP BY PNUMBER, PNAME

HAVING COUNT (*) > 2
```

### **SUBSTRING COMPARISON**

The LIKE comparison operator is used to compare partial strings. Two reserved characters are used: '%' (or '\*' in some implementations) replaces an arbitrary number of characters, and '\_' replaces a single arbitrary character.

**Query 18: Retrieve all employees whose address is in Houston, Texas. Here, the value of the ADDRESS attribute must contain the substring 'Houston,TX' in it.**

```
Q18: SELECT FNAME, LNAME
FROM EMPLOYEE WHERE ADDRESS LIKE '%Houston,TX%'
```

**Query 19: Retrieve all employees who were born during the 1950s.**

Here, '5' must be the 8th character of the string (according to our format for date), so the BDATE value is '\_\_\_\_\_5\_', with each underscore as a place holder for a single arbitrary character.

```
Q19: SELECT FNAME, LNAME
FROM EMPLOYEE WHERE BDATE LIKE '_____5_'
```

**Note:** The LIKE operator allows us to get around the fact that each value is considered atomic and indivisible. Hence, in SQL, character string attribute values are not atomic

## **ARITHMETIC OPERATIONS**

The standard arithmetic operators '+', '-', '\*', and '/' (for addition, subtraction, multiplication, and division, respectively) can be applied to numeric values in an SQL query result

**Query 20: Show the effect of giving all employees who work on the 'ProductX' project a 10% raise.**

```
Q20: SELECT FNAME, LNAME, 1.1*SALARY
FROM EMPLOYEE, WORKS_ON, PROJECT
WHERE SSN=ESSN
AND PNO=PNUMBER AND PNAME='ProductX'
```

## **ORDER BY**

The ORDER BY clause is used to sort the tuples in a query result based on the values of some attribute(s)

**Query 21: Retrieve a list of employees and the projects each works in, ordered by the employee's department, and within each department ordered alphabetically by employee last name.**

```
Q21: SELECT DNAME, LNAME, FNAME, PNAME
FROM DEPARTMENT, EMPLOYEE, WORKS_ON, PROJECT
WHERE DNUMBER=DNO
AND SSN=ESSN
AND PNO=PNUMBER
ORDER BY DNAME, LNAME
```

The default order is in ascending order of values. We can specify the keyword DESC if we want a descending order; the keyword ASC can be used to explicitly specify ascending order, even though it is the default

```
Ex: ORDER BY DNAME DESC, LNAME ASC, FNAME ASC
```

## **MORE EXAMPLE QUERIES:**

**Query 22: Retrieve the names of all employees who have two or more dependents.**

```
Q22: SELECT LNAME, FNAME FROM
EMPLOYEE
```

```
WHERE (SELECT COUNT (*) FROM DEPENDENT
      WHERE SSN=ESSN) ≥ 2);
```

**Query 23: List the names of managers who have least one dependent.**

```
Q23: SELECT FNAME, LNAME
      FROM EMPLOYEE
      WHERE EXISTS (SELECT * FROM DEPENDENT WHERE SSN=ESSN)
      AND EXISTS ( SELECT * FROM DEPARTMENT WHERE SSN=MGRSSN );
```

### **SPECIFYING UPDATES IN SQL**

There are three SQL commands to modify the database: **INSERT**, **DELETE**, and **UPDATE**.

#### **INSERT**

- In its simplest form, it is used to add one or more tuples to a relation
- Attribute values should be listed in the same order as the attributes were specified in the **CREATE TABLE** command

#### **Example:**

```
INSERT INTO EMPLOYEE VALUES ('Richard','K','Marini', '653298653', '30-DEC-52',
                              '98 Oak Forest,Katy,TX', 'M', 37000,'987654321', 4 )
```

- An alternate form of INSERT specifies explicitly the attribute names that correspond to the values in the new tuple. Attributes with NULL values can be left out

**Example:** Insert a tuple for a new EMPLOYEE for whom we only know the FNAME, LNAME, and SSN attributes.

```
INSERT INTO EMPLOYEE (FNAME, LNAME, SSN)VALUES ('Richard', 'Marini',
                                                '653298653')
```

**Important Note:** Only the constraints specified in the DDL commands are automatically enforced by the DBMS when updates are applied to the database. Another variation of INSERT allows insertion of multiple tuples resulting from a **query** into a relation

**Example:** Suppose we want to create a temporary table that has the name, number of employees, and total salaries for each department. A table DEPTS\_INFO is created first, and is loaded with the summary information retrieved from the database by the query.

```
CREATE TABLE DEPTS_INFO
      (DEPT_NAME VARCHAR (10),
```

```
NO_OF_EMPS INTEGER, TOTAL_SAL INTEGER);
```

```
INSERT INTO DEPTS_INFO (DEPT_NAME, NO_OF_EMPS, TOTAL_SAL)
SELECT DNAME, COUNT (*), SUM (SALARY) FROM DEPARTMENT,
EMPLOYEE WHERE DNUMBER=DNO GROUP BY DNAME ;
```

**Note:** The DEPTS\_INFO table may not be up-to-date if we change the tuples in either the DEPARTMENT or the EMPLOYEE relations *after* issuing the above. We have to create a view (see later) to keep such a table up to date.

### **DELETE**

- Removes tuples from a relation. Includes a WHERE-clause to select the tuples to be deleted
- Referential integrity should be enforced
- Tuples are deleted from only *one table* at a time (unless CASCADE is specified on a referential integrity constraint)
- A missing WHERE-clause specifies that *all tuples* in the relation are to be deleted; the table then becomes an empty table
- The number of tuples deleted depends on the number of tuples in the relation that satisfy the WHERE-clause

Examples:

```
1: DELETE FROM EMPLOYEE WHERE LNAME='Brown';
```

```
2: DELETE FROM EMPLOYEE WHERE SSN='123456789';
```

```
3: DELETE FROM EMPLOYEE WHERE DNO IN (SELECT DNUMBER
FROM DEPARTMENT WHERE DNAME='Research');
```

```
4: DELETE FROM EMPLOYEE;
```

### **UPDATE**

- Used to modify attribute values of one or more selected tuples
- A WHERE-clause selects the tuples to be modified
- An additional SET-clause specifies the attributes to be modified and their new values
- Each command modifies tuples *in the same relation*
- Referential integrity should be enforced

**Example1:** Change the location and controlling department number of project number 10 to 'Bellaire' and 5, respectively.

```
UPDATE PROJECT
SET PLOCATION = 'Bellaire', DNUM = 5 WHERE PNUMBER=10;
```

**Example2:** Give all employees in the 'Research' department a 10% raise in salary.

```
UPDATE EMPLOYEE
SET SALARY = SALARY *1.1
WHERE DNO IN (SELECT DNUMBER FROM DEPARTMENT
WHERE DNAME='Research');
```

### **SQL TRIGGERS**

- Objective: to monitor a database and take initiate action when a condition occurs
- Triggers are nothing but the procedures/functions that involve actions and fired/executed automatically whenever an event occurs such as an insert, delete, or update operation or pressing a button or when mouse button is clicked

### **VIEWS IN SQL**

- A view is a single *virtual table* that is derived from other tables. The other tables could be base tables or previously defined view.
  - Allows for limited update operations Since the table may not physically be stored
  - Allows full query operations
  - A convenience for expressing certain operations
  - A view does not necessarily exist in physical form, which limits the possible update operations that can be applied to views.
-



## LAB EXPERIMENTS

### PART A: SQL PROGRAMMING

A. Consider the following schema for a Library Database:

**BOOK** (*Book\_id*, Title, Publisher\_Name, Pub\_Year)

**BOOK\_AUTHORS** (*Book\_id*, Author\_Name)

**PUBLISHER** (*Name*, Address, Phone)

**BOOK\_COPIES** (*Book\_id*, Branch\_id, No-of\_Copies)

**BOOK\_LENDING** (*Book\_id*, Branch\_id, Card\_No, Date\_Out, Due\_Date)

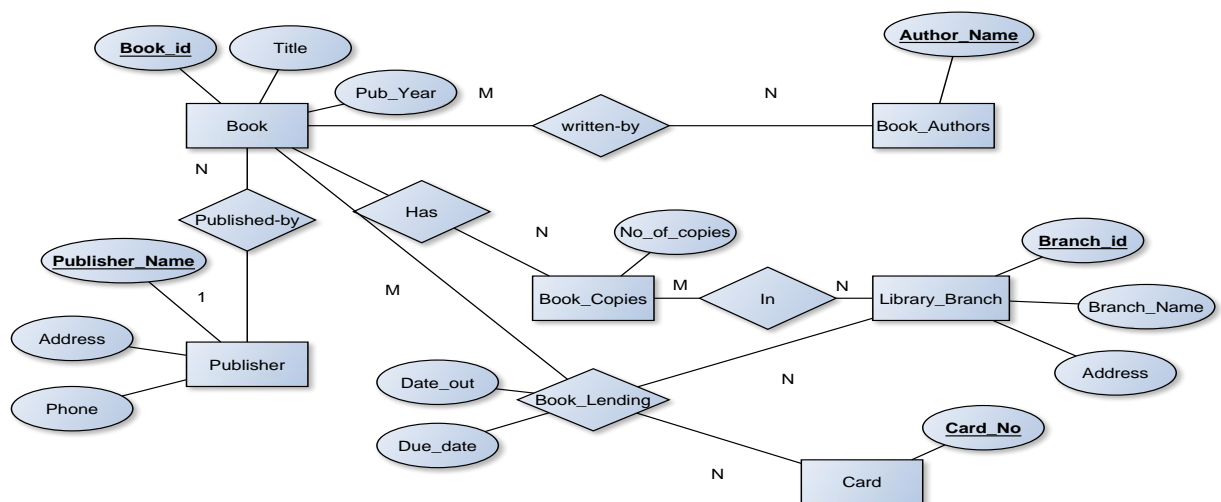
**LIBRARY\_BRANCH** (*Branch\_id*, Branch\_Name, Address)

Write SQL queries to

1. Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc.
2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017
3. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.
4. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.
5. Create a view of all books and its number of copies that are currently available in the Library.

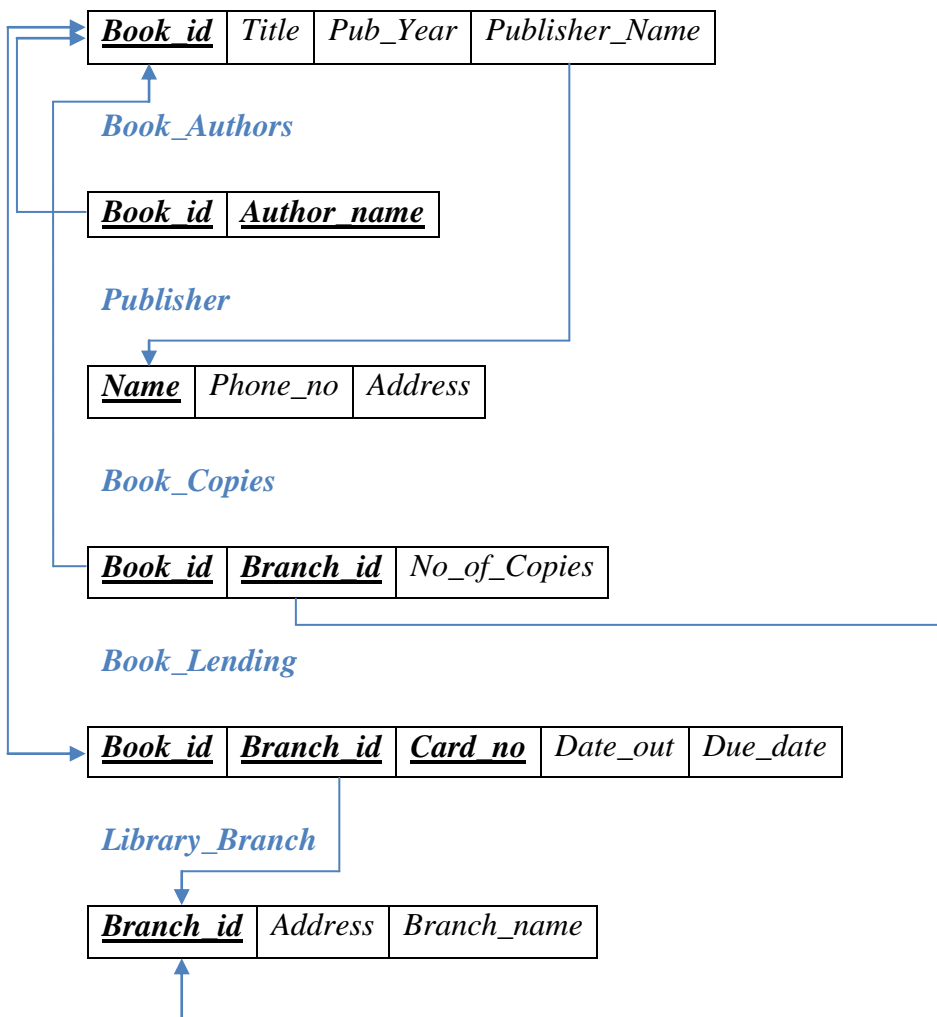
**Solution:**

### **Entity-Relationship Diagram**



## Schema Diagram

### *Book*



## Table Creation

```
CREATE TABLE PUBLISHER
(NAME VARCHAR2 (20) PRIMARY KEY,
PHONE INTEGER,
ADDRESS VARCHAR2 (20));
```

```
CREATE TABLE BOOK
(BOOK_ID INTEGER PRIMARY KEY,
TITLE VARCHAR2 (20),
PUB_YEAR VARCHAR2 (20),
PUBLISHER_NAME REFERENCES PUBLISHER (NAME) ON DELETE CASCADE);
```

```
CREATE TABLE BOOK_AUTHORS
(AUTHOR_NAME VARCHAR2 (20),
BOOK_ID REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE,
PRIMARY KEY (BOOK_ID, AUTHOR_NAME));
```

```
CREATE TABLE LIBRARY_BRANCH
(BRANCH_ID INTEGER PRIMARY KEY,
BRANCH_NAME VARCHAR2 (50),
ADDRESS VARCHAR2 (50));
```

```
CREATE TABLE BOOK_COPIES
(NO_OF_COPIES INTEGER,
BOOK_ID REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE,
BRANCH_ID REFERENCES LIBRARY_BRANCH (BRANCH_ID) ON DELETE
CASCADE,
PRIMARY KEY (BOOK_ID, BRANCH_ID));
```

```
CREATE TABLE CARD
(CARD_NO INTEGER PRIMARY KEY);
```

```
CREATE TABLE BOOK_LENDING
(DATE_OUT DATE,
DUE_DATE DATE,
BOOK_ID REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE,
BRANCH_ID REFERENCES LIBRARY_BRANCH (BRANCH_ID) ON DELETE
CASCADE,
CARD_NO REFERENCES CARD (CARD_NO) ON DELETE CASCADE,
PRIMARY KEY (BOOK_ID, BRANCH_ID, CARD_NO));
```

### Table Descriptions

DESC PUBLISHER;

```
SQL> desc publisher;
```

Name	Null?	Type
NAME	NOT NULL	VARCHAR2(20)
PHONE		NUMBER(38)
ADDRESS		VARCHAR2(20)

DESC BOOK;

SQL> DESC BOOK;

Name	Null?	Type
BOOK_ID	NOT NULL	NUMBER(38)
TITLE		VARCHAR2(20)
PUB_YEAR		VARCHAR2(20)
PUBLISHER_NAME		VARCHAR2(20)

DESC BOOK\_AUTHORS;

SQL> DESC BOOK\_AUTHORS;

Name	Null?	Type
AUTHOR_NAME	NOT NULL	VARCHAR2(20)
BOOK_ID	NOT NULL	NUMBER(38)

DESC LIBRARY\_BRANCH;

SQL> DESC LIBRARY\_BRANCH;

Name	Null?	Type
BRANCH_ID	NOT NULL	NUMBER(38)
BRANCH_NAME		VARCHAR2(50)
ADDRESS		VARCHAR2(50)

DESC BOOK\_COPIES;

SQL> DESC BOOK\_COPIES;

Name	Null?	Type
NO_OF_COPIES		NUMBER(38)
BOOK_ID	NOT NULL	NUMBER(38)
BRANCH_ID	NOT NULL	NUMBER(38)

DESC CARD;

SQL> DESC CARD;

Name	Null?	Type
CARD_NO	NOT NULL	NUMBER(38)

DESC BOOK\_LENDING;

SQL> desc book\_lending;

Name	Null?	Type
DATE_OUT		
DUE_DATE		
BOOK_ID		
BRANCH_ID		
CARD_NO		

### **Insertion of Values to Tables**

```
INSERT INTO PUBLISHER VALUES ('MCGRAW-HILL', 9989076587, 'BANGALORE');
INSERT INTO PUBLISHER VALUES ('PEARSON', 9889076565, 'NEWDELHI');
INSERT INTO PUBLISHER VALUES ('RANDOM HOUSE', 7455679345, 'HYDRABAD');
INSERT INTO PUBLISHER VALUES ('HACHETTE LIVRE', 8970862340, 'CHENAI');
INSERT INTO PUBLISHER VALUES ('GRUPO PLANETA', 7756120238, 'BANGALORE');
```

```
INSERT INTO BOOK VALUES (1,'DBMS','JAN-2017', 'MCGRAW-HILL');
INSERT INTO BOOK VALUES (2,'ADBMS','JUN-2016', 'MCGRAW-HILL');
INSERT INTO BOOK VALUES (3,'CN','SEP-2016', 'PEARSON');
INSERT INTO BOOK VALUES (4,'CG','SEP-2015', 'GRUPO PLANETA');
INSERT INTO BOOK VALUES (5,'OS','MAY-2016', 'PEARSON');
```

```
INSERT INTO BOOK_AUTHORS VALUES ('NAVATHE', 1);
INSERT INTO BOOK_AUTHORS VALUES ('NAVATHE', 2);
INSERT INTO BOOK_AUTHORS VALUES ('TANENBAUM', 3);
INSERT INTO BOOK_AUTHORS VALUES ('EDWARD ANGEL', 4);
INSERT INTO BOOK_AUTHORS VALUES ('GALVIN', 5);
```

```
INSERT INTO LIBRARY_BRANCH VALUES (10,'RR NAGAR', 'BANGALORE');
INSERT INTO LIBRARY_BRANCH VALUES (11,'RNSIT', 'BANGALORE');
INSERT INTO LIBRARY_BRANCH VALUES (12,'RAJAJI NAGAR', 'BANGALORE');
INSERT INTO LIBRARY_BRANCH VALUES (13,'NITTE', 'MANGALORE');
INSERT INTO LIBRARY_BRANCH VALUES (14,'MANIPAL', 'UDUPI');
```

```
INSERT INTO BOOK_COPIES VALUES (10, 1, 10);
INSERT INTO BOOK_COPIES VALUES (5, 1, 11);
INSERT INTO BOOK_COPIES VALUES (2, 2, 12);
INSERT INTO BOOK_COPIES VALUES (5, 2, 13);
INSERT INTO BOOK_COPIES VALUES (7, 3, 14);
INSERT INTO BOOK_COPIES VALUES (1, 5, 10);
INSERT INTO BOOK_COPIES VALUES (3, 4, 11);
```

```
INSERT INTO CARD VALUES (100);
INSERT INTO CARD VALUES (101);
INSERT INTO CARD VALUES (102);
INSERT INTO CARD VALUES (103);
INSERT INTO CARD VALUES (104);
```

```

INSERT INTO BOOK_LENDING VALUES ('01-JAN-17','01-JUN-17', 1, 10, 101);
INSERT INTO BOOK_LENDING VALUES ('11-JAN-17','11-MAR-17', 3, 14, 101);
INSERT INTO BOOK_LENDING VALUES ('21-FEB-17','21-APR-17', 2, 13, 101);
INSERT INTO BOOK_LENDING VALUES ('15-MAR-17','15-JUL-17', 4, 11, 101);
INSERT INTO BOOK_LENDING VALUES ('12-APR-17','12-MAY-17', 1, 11, 104);
SELECT * FROM PUBLISHER;

```

```
SQL> select * from publisher;
```

NAME	PHONE	ADDRESS
MCGRAW-HILL	9989076587	BANGALORE
PEARSON	9889076565	NEWDELHI
RANDOM HOUSE	7455679345	HYDRABAD
HACHETTE LIVRE	8970862340	CHENAI
GRUPO PLANETA	7756120238	BANGALORE

```
SELECT * FROM BOOK;
```

```
SQL> SELECT * FROM BOOK;
```

BOOK_ID	TITLE	PUB_YEAR	PUBLISHER_NAME
1	DBMS	JAN-2017	MCGRAW-HILL
2	ADBMS	JUN-2016	MCGRAW-HILL
3	CN	SEP-2016	PEARSON
4	CG	SEP-2015	GRUPO PLANETA
5	OS	MAY-2016	PEARSON

```
SELECT * FROM BOOK_AUTHORS;
```

```
SQL> SELECT * FROM BOOK_AUTHORS;
```

AUTHOR_NAME	BOOK_ID
NAUATHE	1
NAUATHE	2
TANENBAUM	3
EDWARD ANGEL	4
GALVIN	5

```
SELECT * FROM LIBRARY_BRANCH;
```

```
SQL> SELECT * FROM LIBRARY_BRANCH;
```

BRANCH_ID	BRANCH_NAME	ADDRESS
10	RR NAGAR	BANGALORE
11	RNSIT	BANGALORE
12	RAJAJI NAGAR	BANGALORE
13	NITTE	MANGALORE
14	MANIPAL	UDUPI

```
SELECT * FROM BOOK_COPIES;
```

```
SQL> SELECT * FROM BOOK_COPIES;
```

NO_OF_COPIES	BOOK_ID	BRANCH_ID
10	1	10
5	1	11
2	2	12
5	2	13
7	3	14
1	5	10
3	4	11

```
SELECT * FROM CARD;
```

```
SQL> SELECT * FROM CARD;
```

CARD_NO
100
101
102
103
104

```
SELECT * FROM BOOK_LENDING;
```

```
SQL> select * from book_lending;
```

DATE_OUT	DUE_DATE	BOOK_ID	BRANCH_ID	CARD_NO
01-JAN-17	01-JUN-17	1	10	101
11-JAN-17	11-MAR-17	3	14	101
21-FEB-17	21-APR-17	2	13	101
15-MAR-17	15-JUL-17	4	11	101
12-APR-17	12-MAY-17	1	11	104

### Queries:

1. Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc.

```
SELECT B.BOOK_ID, B.TITLE, B.PUBLISHER_NAME, A.AUTHOR_NAME,  
C.NO_OF_COPIES, L.BRANCH_ID  
FROM BOOK B, BOOK_AUTHORS A, BOOK_COPIES C, LIBRARY_BRANCH L  
WHERE B.BOOK_ID=A.BOOK_ID  
AND B.BOOK_ID=C.BOOK_ID  
AND L.BRANCH_ID=C.BRANCH_ID;
```

BOOK_ID	TITLE	PUBLISHER_NAME	AUTHOR_NAME	NO_OF_COPIES	BRANCH_ID
1	DBMS	MCGRAW-HILL	NAVATHE	10	10
1	DBMS	MCGRAW-HILL	NAVATHE	5	11
2	ADBMS	MCGRAW-HILL	NAVATHE	2	12
2	ADBMS	MCGRAW-HILL	NAVATHE	5	13
3	CN	PEARSON	TANENBAUM	7	14
5	OS	PEARSON	GALVIN	1	10
4	CG	GRUPO PLANETA	EDWARD ANGEL	3	11

1. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017.

```
SELECT CARD_NO
FROM BOOK_LENDING
WHERE DATE_OUT BETWEEN '01-JAN-2017' AND '01-JUL-2017'
GROUP BY CARD_NO
HAVING COUNT (*)>3;
```

```

CARD_NO
-----
      101
```

2. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.

```
DELETE FROM BOOK
WHERE BOOK_ID=3;
SQL> DELETE FROM BOOK
      2  WHERE BOOK_ID=3;

1 row deleted.

SQL> SELECT * FROM BOOK;
```

BOOK_ID	TITLE	PUB_YEAR	PUBLISHER_NAME
1	DBMS	JAN-2017	MCGRAW-HILL
2	ADBMS	JUN-2016	MCGRAW-HILL
4	CG	SEP-2015	GRUPO PLANETA
5	OS	MAY-2016	PEARSON

3. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.

```
CREATE VIEW V_PUBLICATION AS
SELECT PUB_YEAR
FROM BOOK;
```



**PUB\_YEAR**

-----  
**JAN-2017**  
**JUN-2016**  
**SEP-2016**  
**SEP-2015**  
**MAY-2016**

4. Create a view of all books and its number of copies that are currently available in the Library.

```
CREATE VIEW V_BOOKS AS
SELECT B.BOOK_ID, B.TITLE, C.NO_OF_COPIES
FROM BOOK B, BOOK_COPIES C, LIBRARY_BRANCH L
WHERE B.BOOK_ID=C.BOOK_ID
AND C.BRANCH_ID=L.BRANCH_ID;
```

<b>BOOK_ID</b>	<b>TITLE</b>	<b>NO_OF_COPIES</b>
1	DBMS	10
1	DBMS	5
2	ADBMS	2
2	ADBMS	5
3	CN	7
5	OS	1
4	CG	3

---

**B. Consider the following schema for Order Database:**

**SALESMAN** (*Salesman\_id, Name, City, Commission*)

**CUSTOMER** (*Customer\_id, Cust\_Name, City, Grade, Salesman\_id*)

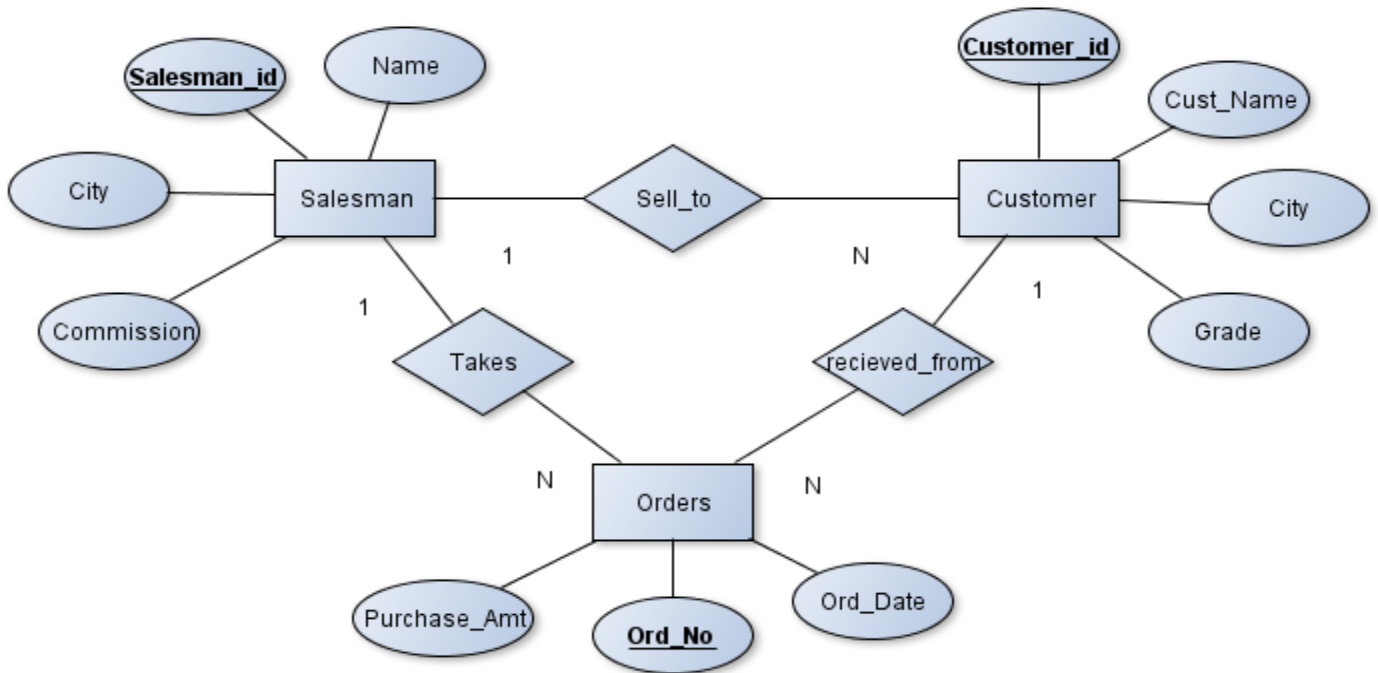
**ORDERS** (*Ord\_No, Purchase\_Amt, Ord\_Date, Customer\_id, Salesman\_id*)

Write SQL queries to

1. Count the customers with grades above Bangalore's average.
2. Find the name and numbers of all salesmen who had more than one customer.
3. List all salesmen and indicate those who have and don't have customers in their cities (Use UNION operation.)
4. Create a view that finds the salesman who has the customer with the highest order of a day.
5. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.

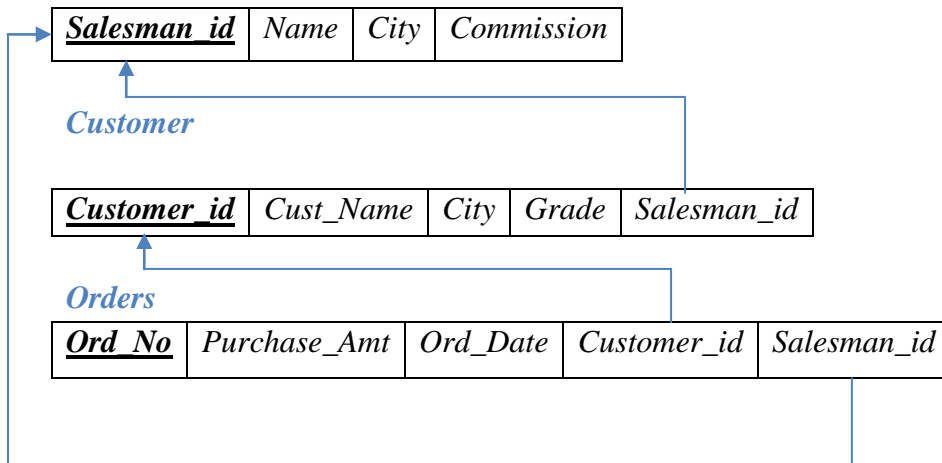
**Solution:**

**Entity-Relationship Diagram**



## Schema Diagram

### *Salesman*



## Table Creation

```
CREATE TABLE SALESMAN
(SALESMAN_ID NUMBER (4),
NAME VARCHAR2 (20),
CITY VARCHAR2 (20),
COMMISSION VARCHAR2 (20),
PRIMARY KEY (SALESMAN_ID));
```

```
CREATE TABLE CUSTOMER1
(CUSTOMER_ID NUMBER (4),
CUST_NAME VARCHAR2 (20),
CITY VARCHAR2 (20),
GRADE NUMBER (3),
PRIMARY KEY (CUSTOMER_ID),
SALESMAN_ID REFERENCES SALESMAN (SALESMAN_ID) ON DELETE SET NULL);
```

```
CREATE TABLE ORDERS
(ORD_NO NUMBER (5),
PURCHASE_AMT NUMBER (10, 2),
ORD_DATE DATE,
PRIMARY KEY (ORD_NO),
CUSTOMER_ID REFERENCES CUSTOMER1 (CUSTOMER_ID) ON DELETE CASCADE,
SALESMAN_ID REFERENCES SALESMAN (SALESMAN_ID) ON DELETE CASCADE);
```

**Table Descriptions**

DESC SALESMAN;

SQL&gt; DESC SALESMAN;

Name	Null?	Type
SALESMAN_ID	NOT NULL	NUMBER(4)
NAME		VARCHAR2(15)
CITY		VARCHAR2(15)
COMMISSION		NUMBER(3,2)

DESC CUSTOMER;

SQL&gt; DESC CUSTOMER1;

Name	Null?	Type
CUSTOMER_ID	NOT NULL	NUMBER(4)
CUST_NAME		VARCHAR2(15)
CITY		VARCHAR2(15)
GRADE		NUMBER(3)
SALESMAN_ID		NUMBER(4)

DESC ORDERS;

SQL&gt; DESC ORDERS;

Name	Null?	Type
ORD_NO	NOT NULL	NUMBER(5)
PURCHASE_AMT		NUMBER(10,2)
ORD_DATE		DATE
CUSTOMER_ID		NUMBER(4)
SALESMAN_ID		NUMBER(4)

**Insertion of Values to Tables**

```
INSERT INTO SALESMAN VALUES (1000, 'JOHN', 'BANGALORE', '25 %');
INSERT INTO SALESMAN VALUES (2000, 'RAVI', 'BANGALORE', '20 %');
INSERT INTO SALESMAN VALUES (3000, 'KUMAR', 'MYSORE', '15 %');
INSERT INTO SALESMAN VALUES (4000, 'SMITH', 'DELHI', '30 %');
INSERT INTO SALESMAN VALUES (5000, 'HARSHA', 'HYDRABAD', '15 %');
```

```
INSERT INTO CUSTOMER1 VALUES (10, 'PREETHI', 'BANGALORE', 100, 1000);
INSERT INTO CUSTOMER1 VALUES (11, 'VIVEK', 'MANGALORE', 300, 1000);
INSERT INTO CUSTOMER1 VALUES (12, 'BHASKAR', 'CHENNAI', 400, 2000);
INSERT INTO CUSTOMER1 VALUES (13, 'CHETHAN', 'BANGALORE', 200, 2000);
INSERT INTO CUSTOMER1 VALUES (14, 'MAMATHA', 'BANGALORE', 400, 3000);
```

```
INSERT INTO ORDERS VALUES (50, 5000, '04-MAY-17', 10, 1000);
INSERT INTO ORDERS VALUES (51, 450, '20-JAN-17', 10, 2000);
```

```
INSERT INTO ORDERS VALUES (52, 1000, '24-FEB-17', 13, 2000);
INSERT INTO ORDERS VALUES (53, 3500, '13-APR-17', 14, 3000);
INSERT INTO ORDERS VALUES (54, 550, '09-MAR-17', 12, 2000);
```

```
SELECT * FROM SALESMAN;
```

SALESMAN_ID	NAME	CITY	COMMISSION
1000	JOHN	BANGALORE	25 %
2000	RAVI	BANGALORE	20 %
3000	KUMAR	MYSORE	15 %
4000	SMITH	DELHI	30 %
5000	HARSHA	HYDRABAD	15 %

```
SELECT * FROM CUSTOMER1;
```

CUSTOMER_ID	CUST_NAME	CITY	GRADE	SALESMAN_ID
10	PREETHI	BANGALORE	100	1000
11	UIVEK	MANGALORE	300	1000
12	BHASKAR	CHENNAI	400	2000
13	CHETHAN	BANGALORE	200	2000
14	MAMATHA	BANGALORE	400	3000

```
SELECT * FROM ORDERS;
```

ORD_NO	PURCHASE_AMT	ORD_DATE	CUSTOMER_ID	SALESMAN_ID
50	5000	04-MAY-17	10	1000
51	450	20-JAN-17	10	2000
52	1000	24-FEB-17	13	2000
53	3500	13-APR-17	14	3000
54	550	09-MAR-17	12	2000

### Queries:

- Count the customers with grades above Bangalore's average.

```
SELECT GRADE, COUNT (DISTINCT CUSTOMER_ID)
FROM CUSTOMER1
GROUP BY GRADE
HAVING GRADE > (SELECT AVG(GRADE)
FROM CUSTOMER1
WHERE CITY='BANGALORE');
```

GRADE	COUNT(DISTINCTCUSTOMER_ID)
300	1
400	2

2. Find the name and numbers of all salesmen who had more than one customer.

```
SELECT SALESMAN_ID, NAME
FROM SALESMAN A
WHERE 1 < (SELECT COUNT (*)
          FROM CUSTOMER1
          WHERE SALESMAN_ID=A.SALESMAN_ID);
```

```
SALESMAN_ID NAME
-----
1000 JOHN
2000 RAVI
```

3. List all salesmen and indicate those who have and don't have customers in their cities (Use UNION operation.)

```
SELECT SALESMAN.SALESMAN_ID, NAME, CUST_NAME, COMMISSION
FROM SALESMAN, CUSTOMER1
WHERE SALESMAN.CITY = CUSTOMER1.CITY
UNION
SELECT SALESMAN_ID, NAME, 'NO MATCH', COMMISSION
FROM SALESMAN
WHERE NOT CITY = ANY
(SELECT CITY
 FROM CUSTOMER1)
ORDER BY 2 DESC;
```

SALESMAN_ID	NAME	CUST_NAME	COMMISSION
4000	SMITH	NO MATCH	30 %
2000	RAVI	CHETHAN	20 %
2000	RAVI	MAMATHA	20 %
2000	RAVI	PREETHI	20 %
3000	KUMAR	NO MATCH	15 %
1000	JOHN	CHETHAN	25 %
1000	JOHN	MAMATHA	25 %
1000	JOHN	PREETHI	25 %
5000	HARSHA	NO MATCH	15 %

4. Create a view that finds the salesman who has the customer with the highest order of a day.

```
CREATE VIEW ELITSALESMANAS
SELECT B.ORD_DATE, A.SALESMAN_ID, A.NAME
FROM SALESMAN A, ORDERS B
```

```

WHERE A.SALESMAN_ID = B.SALESMAN_ID
AND B.PURCHASE_AMT=(SELECT MAX (PURCHASE_AMT)
                     FROM ORDERS C
                     WHERE C.ORD_DATE = B.ORD_DATE);

```

ORD_DATE	SALESMAN_ID	NAME
04-MAY-17	1000	JOHN
20-JAN-17	2000	RAVI
24-FEB-17	2000	RAVI
13-APR-17	3000	KUMAR
09-MAR-17	2000	RAVI

5. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.

Use ON DELETE CASCADE at the end of foreign key definitions while creating child table orders and then execute the following:

Use ON DELETE SET NULL at the end of foreign key definitions while creating child table customers and then executes the following:

```

DELETE FROM SALESMAN
WHERE SALESMAN_ID=1000;
OR

```

We can also write the trigger, if not ON DELETE CASCADE;

```

CREATE TRIGGER 'DEL_SALESMAN' AFTER DELETE ON 'SALESMAN'
FOR EACH ROW
BEGIN
  DELETE FROM ORDERS WHERE SALESMAN_ID = OLD.SALESMAN_ID;
  UPDATE CUSTOMER1 SET SALESMAN_ID=NULL WHERE SALESMAN_ID =
  OLD.SALESMAN_ID;
END

```

```

SQL> DELETE FROM SALESMAN
      2  WHERE SALESMAN_ID=1000;

```

```

1 row deleted.

```

```

SQL> SELECT * FROM SALESMAN;

```

SALESMAN_ID	NAME	CITY	COMMISSION
2000	RAVI	BANGALORE	20 %
3000	KUMAR	MYSORE	15 %
4000	SMITH	DELHI	30 %
5000	HARSHA	HYDRABAD	15 %

**C. Consider the schema for Movie Database:**

**ACTOR** (Act\_id, Act\_Name, Act\_Gender)

**DIRECTOR** (Dir\_id, Dir\_Name, Dir\_Phone)

**MOVIES** (Mov\_id, Mov\_Title, Mov\_Year, Mov\_Lang, Dir\_id)

**MOVIE\_CAST** (Act\_id, Mov\_id, Role)

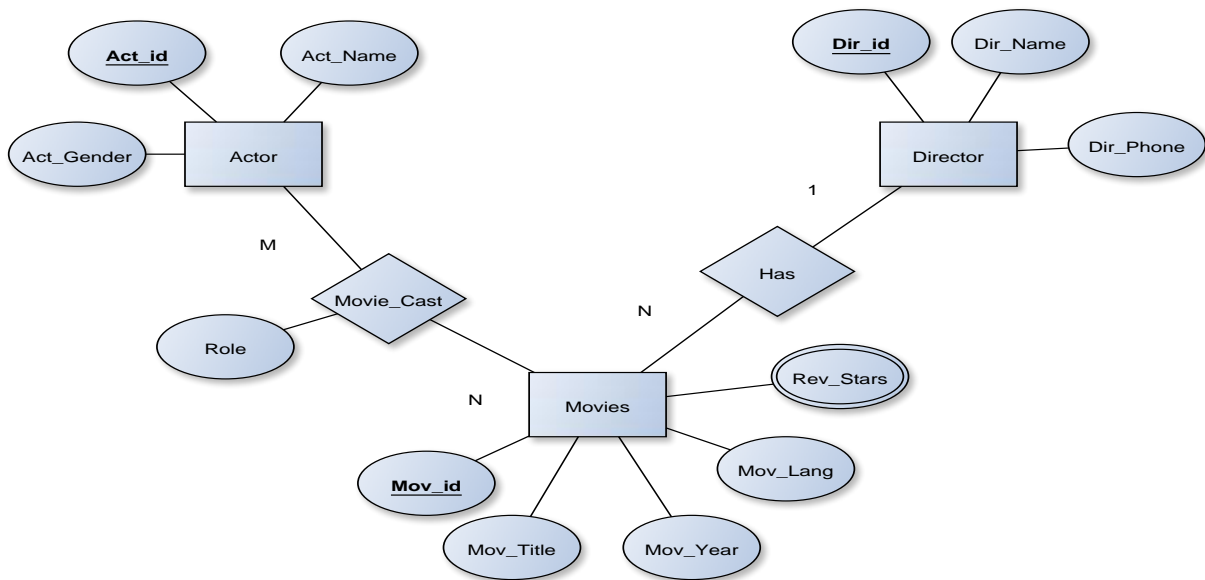
**RATING** (Mov\_id, Rev\_Stars)

Write SQL queries to

1. List the titles of all movies directed by 'Hitchcock'.
2. Find the movie names where one or more actors acted in two or more movies.
3. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).
4. Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.
5. Update rating of all movies directed by 'Steven Spielberg' to 5.

**Solution:**

**Entity-Relationship Diagram**





## Schema Diagram

### *Actor*

<u>Act_id</u>	Act_Name	Act_Gender
---------------	----------	------------

### *Director*

<u>Dir_id</u>	Dir_Name	Dir_Phone
---------------	----------	-----------

### *Movies*

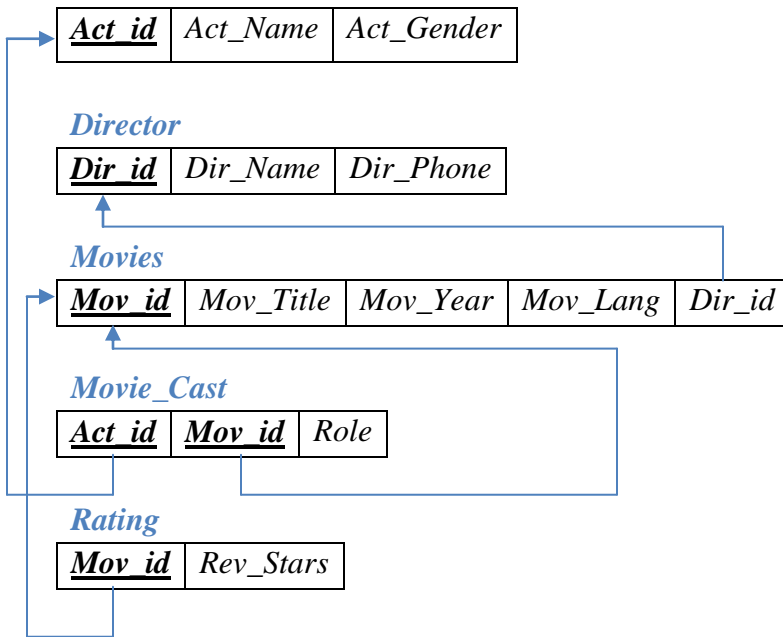
<u>Mov_id</u>	Mov_Title	Mov_Year	Mov_Lang	Dir_id
---------------	-----------	----------	----------	--------

### *Movie\_Cast*

<u>Act_id</u>	<u>Mov_id</u>	Role
---------------	---------------	------

### *Rating*

<u>Mov_id</u>	Rev_Stars
---------------	-----------



## Table Creation

```
CREATE TABLE ACTOR (  
  ACT_ID NUMBER (3),  
  ACT_NAME VARCHAR (20),  
  ACT_GENDER CHAR (1),  
  PRIMARY KEY (ACT_ID));
```

```
CREATE TABLE DIRECTOR (  
  DIR_ID NUMBER (3),  
  DIR_NAME VARCHAR (20),  
  DIR_PHONE NUMBER (10),  
  PRIMARY KEY (DIR_ID));
```

```
CREATE TABLE MOVIES (  
  MOV_ID NUMBER (4),  
  MOV_TITLE VARCHAR (25),  
  MOV_YEAR NUMBER (4),  
  MOV_LANG VARCHAR (12),  
  DIR_ID NUMBER (3),  
  PRIMARY KEY (MOV_ID),  
  FOREIGN KEY (DIR_ID) REFERENCES DIRECTOR (DIR_ID));
```

```
CREATE TABLE MOVIE_CAST (  

```

```
ACT_ID NUMBER (3),
MOV_ID NUMBER (4),
ROLE VARCHAR (10),
PRIMARY KEY (ACT_ID, MOV_ID),
FOREIGN KEY (ACT_ID) REFERENCES ACTOR (ACT_ID),
FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID));
```

```
CREATE TABLE RATING (
MOV_ID NUMBER (4),
REV_STARS VARCHAR (25),
PRIMARY KEY (MOV_ID),
FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID));
```

### Table Descriptions

DESC ACTOR;

```
SQL> DESC ACTOR;
```

Name	Null?	Type
ACT_ID	NOT NULL	NUMBER(3)
ACT_NAME		VARCHAR2(20)
ACT_GENDER		CHAR(1)

DESC DIRECTOR;

```
SQL> DESC DIRECTOR;
```

Name	Null?	Type
DIR_ID	NOT NULL	NUMBER(3)
DIR_NAME		VARCHAR2(20)
DIR_PHONE		NUMBER(10)

DESC MOVIES;

```
SQL> DESC MOVIES;
```

Name	Null?	Type
MOV_ID	NOT NULL	NUMBER(4)
MOV_TITLE		VARCHAR2(25)
MOV_YEAR		NUMBER(4)
MOV_LANG		VARCHAR2(12)
DIR_ID		NUMBER(3)

DESC MOVIE\_CAST;

```
SQL> DESC MOVIE_CAST;
Name                                     Null?    Type
-----
ACT_ID                                  NOT NULL NUMBER(3)
MOV_ID                                  NOT NULL NUMBER(4)
ROLE                                              VARCHAR2(10)
```

DESC RATING;

```
SQL> DESC RATING;
Name                                     Null?    Type
-----
MOV_ID                                  NOT NULL NUMBER(4)
REV_STARS                                        VARCHAR2(25)
```

### Insertion of Values to Tables

```
INSERT INTO ACTOR VALUES (301,'ANUSHKA','F');
INSERT INTO ACTOR VALUES (302,'PRABHAS','M');
INSERT INTO ACTOR VALUES (303,'PUNITH','M');
INSERT INTO ACTOR VALUES (304,'JERMY','M');
```

```
INSERT INTO DIRECTOR VALUES (60,'RAJAMOULI', 8751611001);
INSERT INTO DIRECTOR VALUES (61,'HITCHCOCK', 7766138911);
INSERT INTO DIRECTOR VALUES (62,'FARAN', 9986776531);
INSERT INTO DIRECTOR VALUES (63,'STEVEN SPIELBERG', 8989776530);
```

```
INSERT INTO MOVIES VALUES (1001,'BAHUBALI-2', 2017, 'TELAGU', 60);
INSERT INTO MOVIES VALUES (1002,'BAHUBALI-1', 2015, 'TELAGU', 60);
INSERT INTO MOVIES VALUES (1003,'AKASH', 2008, 'KANNADA', 61);
INSERT INTO MOVIES VALUES (1004,'WAR HORSE', 2011, 'ENGLISH', 63);
```

```
INSERT INTO MOVIE_CAST VALUES (301, 1002, 'HEROINE');
INSERT INTO MOVIE_CAST VALUES (301, 1001, 'HEROINE');
INSERT INTO MOVIE_CAST VALUES (303, 1003, 'HERO');
INSERT INTO MOVIE_CAST VALUES (303, 1002, 'GUEST');
INSERT INTO MOVIE_CAST VALUES (304, 1004, 'HERO');
```

```
INSERT INTO RATING VALUES (1001, 4);
INSERT INTO RATING VALUES (1002, 2);
INSERT INTO RATING VALUES (1003, 5);
```

INSERT INTO RATING VALUES (1004, 4);

SELECT \* FROM ACTOR;

SQL> SELECT \* FROM ACTOR;

ACT_ID	ACT_NAME	A
301	ANUSHKA	F
302	PRABHAS	M
303	PUNITH	M
304	JERMY	M

SELECT \* FROM DIRECTOR;

SQL> SELECT \* FROM DIRECTOR;

DIR_ID	DIR_NAME	DIR_PHONE
60	RAJAMOULI	8751611001
61	HITCHCOCK	7766138911
62	FARAN	9986776531
63	STEVEN SPIELBERG	8989776530

SELECT \* FROM MOVIES;

SQL> SELECT \* FROM MOVIES;

MOV_ID	MOV_TITLE	MOV_YEAR	MOV_LANG	DIR_ID
1001	BAHUBALI-2	2017	TELAGU	60
1002	BAHUBALI-1	2015	TELAGU	60
1003	AKASH	2008	KANNADA	61
1004	WAR HORSE	2011	ENGLISH	63

SELECT \* FROM MOVIE\_CAST;

SQL> SELECT \* FROM MOVIE\_CAST;

ACT_ID	MOV_ID	ROLE
301	1002	HEROINE
301	1001	HEROINE
303	1003	HERO
303	1002	GUEST
304	1004	HERO

```
SELECT * FROM RATING;
```

```
SQL> SELECT * FROM RATING;
```

```
   MOV_ID REV_STARS
-----
   1001    4
   1002    2
   1003    5
   1004    4
```

### Queries:

1. List the titles of all movies directed by 'Hitchcock'.

```
SELECT MOV_TITLE
FROM MOVIES
WHERE DIR_ID IN (SELECT DIR_ID
                 FROM DIRECTOR
                 WHERE DIR_NAME = 'HITCHCOCK');
```

```
   MOV_TITLE
-----
   AKASH
```

2. Find the movie names where one or more actors acted in two or more movies.

```
SELECT MOV_TITLE
FROM MOVIES M, MOVIE_CAST MV
WHERE M.MOV_ID=MV.MOV_ID AND ACT_ID IN (SELECT ACT_ID
                                         FROM MOVIE_CAST GROUP BY ACT_ID
                                         HAVING COUNT (ACT_ID)>1)
GROUP BY MOV_TITLE
HAVING COUNT (*)>1;
```

```
   MOV_TITLE
-----
   BAHUBALI-1
```

3. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).

```
SELECT ACT_NAME, MOV_TITLE, MOV_YEAR
FROM ACTOR A
```

```
JOIN MOVIE_CAST C
  ON A.ACT_ID=C.ACT_ID
JOIN MOVIES M
  ON C.MOV_ID=M.MOV_ID
WHERE M.MOV_YEAR NOT BETWEEN 2000 AND 2015;
```

OR

```
SELECT A.ACT_NAME, A.ACT_NAME, C.MOV_TITLE, C.MOV_YEAR
FROM ACTOR A, MOVIE_CAST B, MOVIES C
WHERE A.ACT_ID=B.ACT_ID
AND B.MOV_ID=C.MOV_ID
AND C.MOV_YEAR NOT BETWEEN 2000 AND 2015;
```

ACT_NAME	MOV_TITLE	MOV_YEAR
ANUSHKA	BAHUBALI-2	2017

4. Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.

```
SELECT MOV_TITLE, MAX (REV_STARS)
FROM MOVIES
INNER JOIN RATING USING (MOV_ID)
GROUP BY MOV_TITLE
HAVING MAX (REV_STARS)>0
ORDER BY MOV_TITLE;
```

MOV_TITLE	MAX(REV_STARS)
AKASH	5
BAHUBALI-1	2
BAHUBALI-2	4
WAR HORSE	4

5. Update rating of all movies directed by 'Steven Spielberg' to 5  
KL

```
UPDATE RATING
SET REV_STARS=5
```

```
WHERE MOV_ID IN (SELECT MOV_ID FROM MOVIES
                  WHERE DIR_ID IN (SELECT DIR_ID
                                   FROM DIRECTOR
                                   WHERE DIR_NAME = 'STEVEN
SPELBERG')));
```

```
SQL> SELECT * FROM RATING;
```

```
MOV_ID REV_STARS
-----
1001 4
1002 2
1003 5
1004 5
```

---

**D. Consider the schema for College Database:**

**STUDENT** (USN, SName, Address, Phone, Gender)

**SEMSEC** (SSID, Sem, Sec)

**CLASS** (USN, SSID)

**SUBJECT** (Subcode, Title, Sem, Credits)

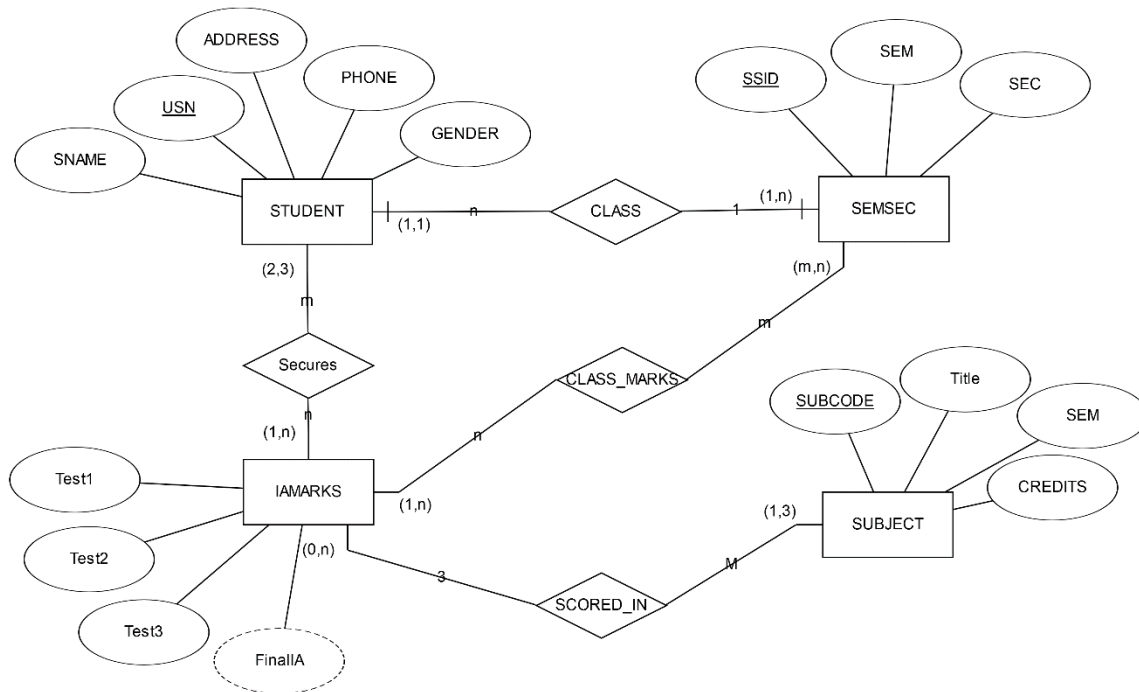
**IAMARKS** (USN, Subcode, SSID, Test1, Test2, Test3, FinallA)

Write SQL queries to

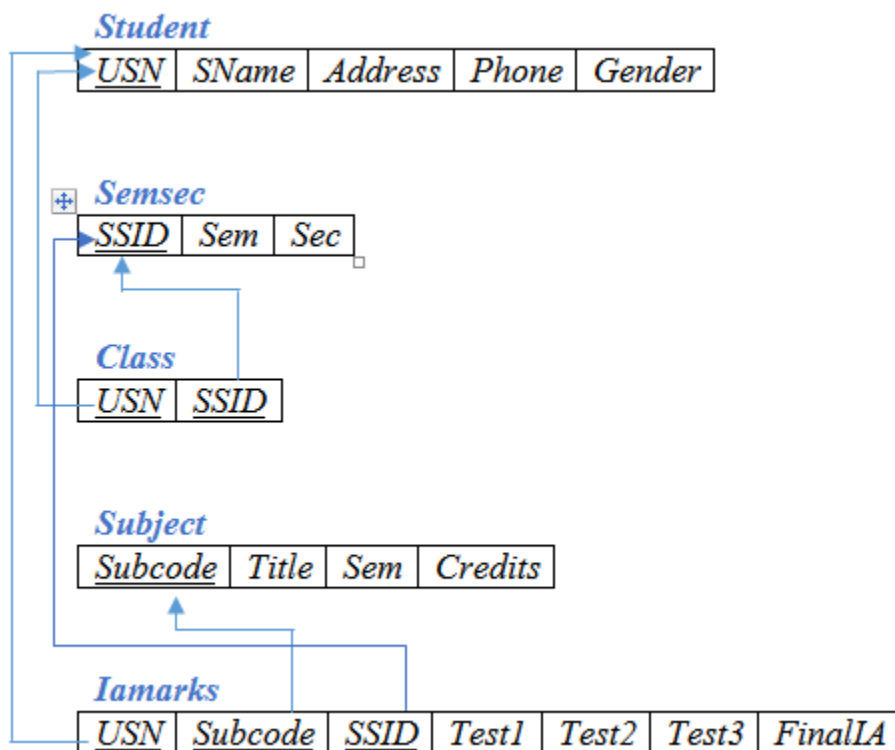
1. List all the student details studying in fourth semester 'C' section.
2. Compute the total number of male and female students in each semester and in each section.
3. Create a view of Test1 marks of student USN '1BI15CS101' in all subjects.
4. Calculate the FinallA (average of best two test marks) and update the corresponding table for all students.
5. Categorize students based on the following criterion:  
 If FinallA = 17 to 20 then CAT = 'Outstanding'  
 If FinallA = 12 to 16 then CAT = 'Average'  
 If FinallA < 12 then CAT = 'Weak'  
 Give these details only for 8th semester A, B, and C section students.

**Solution:**

**Entity - Relationship Diagram**





**Schema Diagram****Table Creation**

```
CREATE TABLE STUDENT (
  USN VARCHAR (10) PRIMARY KEY,
  SNAME VARCHAR (25),
  ADDRESS VARCHAR (25),
  PHONE NUMBER (10),
  GENDER CHAR (1));
```

```
CREATE TABLE SEMSEC (
  SSID VARCHAR (5) PRIMARY KEY,
  SEM NUMBER (2),
  SEC CHAR (1));
```

```
CREATE TABLE CLASS (
  USN VARCHAR (10),
  SSID VARCHAR (5),
  PRIMARY KEY (USN, SSID),
  FOREIGN KEY (USN) REFERENCES STUDENT (USN),
  FOREIGN KEY (SSID) REFERENCES SEMSEC (SSID));
```

```
CREATE TABLE SUBJECT (  
SUBCODE VARCHAR (8),  
TITLE VARCHAR (20),  
SEM NUMBER (2),  
CREDITS NUMBER (2),  
PRIMARY KEY (SUBCODE));
```

```
CREATE TABLE IAMARKS (  
USN VARCHAR (10),  
SUBCODE VARCHAR (8),  
SSID VARCHAR (5),  
TEST1 NUMBER (2),  
TEST2 NUMBER (2),  
TEST3 NUMBER (2),  
FINALIA NUMBER (2),  
PRIMARY KEY (USN, SUBCODE, SSID),  
FOREIGN KEY (USN) REFERENCES STUDENT (USN),  
FOREIGN KEY (SUBCODE) REFERENCES SUBJECT (SUBCODE),  
FOREIGN KEY (SSID) REFERENCES SEMSEC (SSID));
```

### **Table Descriptions**

DESC STUDENT;

Name  
-----

USN  
SNAME  
ADDRESS  
PHONE  
GENDER

DESC SEMSEC;

```
SQL> DESC SEMSEC;
```

Name  
-----

SSID  
SEM  
SEC

DESC CLASS;

```
SQL> DESC CLASS;
Name
-----
USN
SSID
```

DESC SUBJECT;

```
SQL> DESC SUBJECT1;
Name
-----
SUBCODE
TITLE
SEM
CREDITS
```

DESC IAMARKS;

```
SQL> DESC IAMARKS;
Name
-----
USN
SUBCODE
SSID
TEST1
TEST2
TEST3
FINALIA
```

### Insertion of values to tables

```
INSERT INTO STUDENT VALUES ('1RN13CS020','AKSHAY','BELAGAVI',
8877881122,'M');
INSERT INTO STUDENT VALUES ('1RN13CS062','SANDHYA','BENGALURU',
7722829912,'F');
INSERT INTO STUDENT VALUES ('1RN13CS091','TEESHA','BENGALURU',
7712312312,'F');
INSERT INTO STUDENT VALUES ('1RN13CS066','SUPRIYA','MANGALURU',
8877881122,'F');
INSERT INTO STUDENTVALUES ('1RN14CS010','ABHAY','BENGALURU',
9900211201,'M');
INSERT INTO STUDENT VALUES ('1RN14CS032','BHASKAR','BENGALURU',
9923211099,'M');
INSERT INTO STUDENTVALUES ('1RN14CS025','ASMI','BENGALURU', 7894737377,'F');
INSERT INTO STUDENT VALUES ('1RN15CS011','AJAY','TUMKUR', 9845091341,'M');
```

```
INSERT INTO STUDENT VALUES ('1RN15CS029','CHITRA','DAVANGERE',
7696772121,'F');
INSERT INTO STUDENT VALUES ('1RN15CS045','JEEVA','BELLARY', 9944850121,'M');
INSERT INTO STUDENT VALUES ('1RN15CS091','SANTOSH','MANGALURU',
8812332201,'M');
INSERT INTO STUDENT VALUES ('1RN16CS045','ISMAIL','KALBURGI',
9900232201,'M');
INSERT INTO STUDENT VALUES ('1RN16CS088','SAMEERA','SHIMOGA',
9905542212,'F');
INSERT INTO STUDENT VALUES ('1RN16CS122','VINAYAKA','CHIKAMAGALUR',
8800880011,'M');
```

```
INSERT INTO SEMSEC VALUES ('CSE8A', 8,'A');
INSERT INTO SEMSEC VALUES ('CSE8B', 8,'B');
INSERT INTO SEMSEC VALUES ('CSE8C', 8,'C');
```

```
INSERT INTO SEMSEC VALUES ('CSE7A', 7,'A');
INSERT INTO SEMSEC VALUES ('CSE7B', 7,'B');
INSERT INTO SEMSEC VALUES ('CSE7C', 7,'C');
```

```
INSERT INTO SEMSEC VALUES ('CSE6A', 6,'A');
INSERT INTO SEMSEC VALUES ('CSE6B', 6,'B');
INSERT INTO SEMSEC VALUES ('CSE6C', 6,'C');
```

```
INSERT INTO SEMSEC VALUES ('CSE5A', 5,'A');
INSERT INTO SEMSEC VALUES ('CSE5B', 5,'B');
INSERT INTO SEMSEC VALUES ('CSE5C', 5,'C');
```

```
INSERT INTO SEMSEC VALUES ('CSE4A', 4,'A');
INSERT INTO SEMSEC VALUES ('CSE4B', 4,'B');
INSERT INTO SEMSEC VALUES ('CSE4C', 4,'C');
```

```
INSERT INTO SEMSEC VALUES ('CSE3A', 3,'A');
INSERT INTO SEMSEC VALUES ('CSE3B', 3,'B');
INSERT INTO SEMSEC VALUES ('CSE3C', 3,'C');
```

```
INSERT INTO SEMSEC VALUES ('CSE2A', 2,'A');
INSERT INTO SEMSEC VALUES ('CSE2B', 2,'B');
INSERT INTO SEMSEC VALUES ('CSE2C', 2,'C');
INSERT INTO SEMSEC VALUES ('CSE1A', 1,'A');
```

```
INSERT INTO SEMSEC VALUES ('CSE1B', 1, 'B');
INSERT INTO SEMSEC VALUES ('CSE1C', 1, 'C');
```

```
INSERT INTO CLASS VALUES ('1RN13CS020', 'CSE8A');
INSERT INTO CLASS VALUES ('1RN13CS062', 'CSE8A');
INSERT INTO CLASS VALUES ('1RN13CS066', 'CSE8B');
INSERT INTO CLASS VALUES ('1RN13CS091', 'CSE8C');
```

```
INSERT INTO CLASS VALUES ('1RN14CS010', 'CSE7A');
INSERT INTO CLASS VALUES ('1RN14CS025', 'CSE7A');
INSERT INTO CLASS VALUES ('1RN14CS032', 'CSE7A');
```

```
INSERT INTO CLASS VALUES ('1RN15CS011', 'CSE4A');
INSERT INTO CLASS VALUES ('1RN15CS029', 'CSE4A');
INSERT INTO CLASS VALUES ('1RN15CS045', 'CSE4B');
INSERT INTO CLASS VALUES ('1RN15CS091', 'CSE4C');
```

```
INSERT INTO CLASS VALUES ('1RN16CS045', 'CSE3A');
INSERT INTO CLASS VALUES ('1RN16CS088', 'CSE3B');
INSERT INTO CLASS VALUES ('1RN16CS122', 'CSE3C');
```

```
INSERT INTO SUBJECT VALUES ('10CS81', 'ACA', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS82', 'SSM', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS83', 'NM', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS84', 'CC', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS85', 'PW', 8, 4);
```

```
INSERT INTO SUBJECT VALUES ('10CS71', 'OOAD', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS72', 'ECS', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS73', 'PTW', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS74', 'DWDM', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS75', 'JAVA', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS76', 'SAN', 7, 4);
```

```
INSERT INTO SUBJECT VALUES ('15CS51', 'ME', 5, 4);
INSERT INTO SUBJECT VALUES ('15CS52', 'CN', 5, 4);
INSERT INTO SUBJECT VALUES ('15CS53', 'DBMS', 5, 4);
INSERT INTO SUBJECT VALUES ('15CS54', 'ATC', 5, 4);
INSERT INTO SUBJECT VALUES ('15CS55', 'JAVA', 5, 3);
INSERT INTO SUBJECT VALUES ('15CS56', 'AI', 5, 3);
```

```

INSERT INTO SUBJECT VALUES ('15CS41','M4', 4, 4);
INSERT INTO SUBJECT VALUES ('15CS42','SE', 4, 4);
INSERT INTO SUBJECT VALUES ('15CS43','DAA', 4, 4);
INSERT INTO SUBJECT VALUES ('15CS44','MPMC', 4, 4);
INSERT INTO SUBJECT VALUES ('15CS45','OOC', 4, 3);
INSERT INTO SUBJECT VALUES ('15CS46','DC', 4, 3);

```

```

INSERT INTO SUBJECT VALUES ('15CS31','M3', 3, 4);
INSERT INTO SUBJECT VALUES ('15CS32','ADE', 3, 4);
INSERT INTO SUBJECT VALUES ('15CS33','DSA', 3, 4);
INSERT INTO SUBJECT VALUES ('15CS34','CO', 3, 4);
INSERT INTO SUBJECT VALUES ('15CS35','USP', 3, 3);
INSERT INTO SUBJECT VALUES ('15CS36','DMS', 3, 3);

```

```

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1RN13CS091','10CS81','CSE8C', 15, 16, 18);
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1RN13CS091','10CS82','CSE8C', 12, 19, 14);
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1RN13CS091','10CS83','CSE8C', 19, 15, 20);
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1RN13CS091','10CS84','CSE8C', 20, 16, 19);
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1RN13CS091','10CS85','CSE8C', 15, 15, 12);

```

```
SELECT * FROM STUDENT;
```

```
SQL> SELECT * FROM STUDENT1;
```

USN	SNAME	ADDRESS	PHONE	G
1RN13CS020	AKSHAY	BELAGAVI	8877881122	M
1RN13CS062	SANDHYA	BENGALURU	7722829912	F
1RN13CS091	TEESHA	BENGALURU	7712312312	F
1RN13CS066	SUPRIYA	MANGALURU	8877881122	F
1RN14CS010	ABHAY	BENGALURU	9900211201	M
1RN14CS032	BHASKAR	BENGALURU	9923211099	M
1RN15CS011	AJAY	TUMKUR	9845091341	M
1RN15CS029	CHITRA	DAVANGERE	7696772121	F
1RN15CS045	JEEVA	BELLARY	9944850121	M
1RN15CS091	SANTOSH	MANGALURU	8812332201	M
1RN16CS045	ISMAIL	KALBURGI	9900232201	M
1RN16CS088	SAMEERA	SHIMOGA	9905542212	F
1RN16CS122	VINAYAKA	CHIKAMAGALUR	8800880011	M
1RN14CS025	ASMI	BENGALURU	7894737377	F

```
SELECT * FROM SEMSEC;
```

```
SQL> SELECT * FROM SEMSEC;
```

SSID	SEM	S
CSE8A	8	A
CSE8B	8	B
CSE8C	8	C
CSE7A	7	A
CSE7B	7	B
CSE7C	7	C
CSE6A	6	A
CSE6B	6	B
CSE6C	6	C
CSE5A	5	A
CSE5B	5	B
CSE5C	5	C
CSE4A	4	A
CSE4B	4	B
CSE4C	4	C
CSE3A	3	A
CSE3B	3	B
CSE3C	3	C
CSE2A	2	A
CSE2C	2	C
CSE2B	2	B
CSE1A	1	A
CSE1B	1	B
CSE1C	1	C

```
SELECT * FROM CLASS;
```

```
SQL> SELECT * FROM CLASS;
```

USN	SSID
1RN13CS020	CSE8A
1RN13CS062	CSE8A
1RN13CS066	CSE8B
1RN13CS091	CSE8C
1RN14CS010	CSE7A
1RN14CS025	CSE7A
1RN14CS032	CSE7A
1RN15CS011	CSE4A
1RN15CS029	CSE4A
1RN15CS045	CSE4B
1RN15CS091	CSE4C
1RN16CS045	CSE3A
1RN16CS088	CSE3B
1RN16CS122	CSE3C

```
14 rows selected.
```

SELECT \* FROM SUBJECT;

SUBCODE	TITLE	SEM	CREDITS
10CS81	ACA	8	4
10CS82	SSM	8	4
10CS83	NM	8	4
10CS84	CC	8	4
10CS85	PW	8	4
10CS71	OOAD	7	4
10CS72	ECS	7	4
10CS73	PTW	7	4
10CS74	DWDM	7	4
10CS75	JAVA	7	4
10CS76	SAN	7	4
15CS51	ME	5	4
15CS52	CN	5	4
15CS53	DBMS	5	4
15CS54	ATC	5	4
15CS55	JAVA	5	3
15CS56	AI	5	3
15CS41	M4	4	4
15CS42	SE	4	4
15CS43	DAA	4	4
15CS44	MPMC	4	4
15CS45	OOO	4	3
15CS46	DC	4	3
15CS31	M3	3	4
15CS32	ADE	3	4
15CS33	DSA	3	4
15CS34	CO	3	4
15CS35	USP	3	3
15CS36	DMS	3	3

SELECT \* FROM IAMARKS;

SQL> SELECT \* FROM IAMARKS;

USN	SUBCODE	SSID	TEST1	TEST2	TEST3	FINALIA
1RN13CS091	10CS81	CSE8C	15	16	18	
1RN13CS091	10CS82	CSE8C	12	19	14	
1RN13CS091	10CS83	CSE8C	19	15	20	
1RN13CS091	10CS84	CSE8C	20	16	19	
1RN13CS091	10CS85	CSE8C	15	15	12	

### Queries:

1. List all the student details studying in fourth semester 'C' section.

```
SELECT S.*, SS.SEM, SS.SEC
FROM STUDENT S, SEMSEC SS, CLASS C
WHERE S.USN = C.USN AND
SS.SSID = C.SSID AND
SS.SEM = 4 AND
```



SS.Sec='C');

USN	SNAME	ADDRESS	PHONE G	SEM S
1RN15CS091	SANTOSH	HANGALURU	8812332201 M	4 C

2. Compute the total number of male and female students in each semester and in each section.

```
SELECT SS.SEM, SS.SEC, S.GENDER, COUNT (S.GENDER) AS COUNT
FROM STUDENT1 S, SEMSEC SS, CLASS C
WHERE S.USN = C.USN AND
SS.SSID = C.SSID
GROUP BY SS.SEM, SS.SEC, S.GENDER
ORDER BY SEM;
```

SEM	S	G	COUNT
3	A	M	1
3	B	F	1
3	C	M	1
4	A	F	1
4	A	M	1
4	B	M	1
4	C	M	1
7	A	F	1
7	A	M	2
8	A	F	1
8	A	M	1
8	B	F	1
8	C	F	1

3. Create a view of Test1 marks of student USN '1BI15CS101' in all subjects.

```
CREATE VIEW STU_TEST1_MARKS_VIEW
AS
SELECT TEST1, SUBCODE
FROM IAMARKS
WHERE USN = '1RN13CS091';
```

TEST1	SUBCODE
15	10CS81
12	10CS82
19	10CS83
20	10CS84
15	10CS85

**4. Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.**

```
CREATE OR REPLACE PROCEDURE AVGMARKS
IS
CURSOR C_IAMARKS IS
SELECT GREATEST(TEST1,TEST2) AS A, GREATEST(TEST1,TEST3) AS B,
GREATEST(TEST3,TEST2) AS C
FROM IAMARKS
WHERE FINALIA IS NULL
FOR UPDATE;

C_A NUMBER;
C_B NUMBER;
C_C NUMBER;
C_SM NUMBER;
C_AV NUMBER;

BEGIN
OPEN C_IAMARKS;
LOOP
FETCH C_IAMARKS INTO C_A, C_B, C_C;
EXIT WHEN C_IAMARKS%NOTFOUND;
--DBMS_OUTPUT.PUT_LINE(C_A || ' ' || C_B || ' ' || C_C);
IF (C_A != C_B) THEN
C_SM:=C_A+C_B;
ELSE
C_SM:=C_A+C_C;
END IF;

C_AV:=C_SM/2;
--DBMS_OUTPUT.PUT_LINE('SUM = '||C_SM);
--DBMS_OUTPUT.PUT_LINE('AVERAGE = '||C_AV);
UPDATE IAMARKS SET FINALIA=C_AV WHERE CURRENT OF C_IAMARKS;

END LOOP;
CLOSE C_IAMARKS;
END;
/
```

**Note:** Before execution of PL/SQL procedure, IAMARKS table contents are:

SELECT \* FROM IAMARKS;

SQL> SELECT \* FROM IAMARKS;

USN	SUBCODE	SSID	TEST1	TEST2	TEST3	FINALIA
1RN13CS091	10CS81	CSE8C	15	16	18	
1RN13CS091	10CS82	CSE8C	12	19	14	
1RN13CS091	10CS83	CSE8C	19	15	20	
1RN13CS091	10CS84	CSE8C	20	16	19	
1RN13CS091	10CS85	CSE8C	15	15	12	

Below SQL code is to invoke the PL/SQL stored procedure from the command line:

```
BEGIN
  AVGMARKS; --IT INVOKES THE STORED PROCEDURE
END;
/
```

SQL> select \* from IAMARKS;

USN	SUBCODE	SSID	TEST1	TEST2	TEST3	FINALIA
1RN13CS091	10CS81	CSE8C	15	16	18	17
1RN13CS091	10CS82	CSE8C	12	19	14	17
1RN13CS091	10CS83	CSE8C	19	15	20	20
1RN13CS091	10CS84	CSE8C	20	16	19	20
1RN13CS091	10CS85	CSE8C	15	15	12	15

5. Categorize students based on the following criterion:

If FinalIA = 17 to 20 then CAT = 'Outstanding'

If FinalIA = 12 to 16 then CAT = 'Average'

If FinalIA < 12 then CAT = 'Weak'

Give these details only for 8<sup>th</sup> semester A, B, and C section students.

```
SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER,
(CASE
  WHEN IA.FINALIA BETWEEN 17 AND 20 THEN 'OUTSTANDING'
  WHEN IA.FINALIA BETWEEN 12 AND 16 THEN 'AVERAGE'
  ELSE 'WEAK'
END) AS CAT
FROM STUDENT S, SEMSEC SS, IAMARKS IA, SUBJECT SUB
WHERE S.USN = IA.USN AND
SS.SSID = IA.SSID AND
SUB.SUBCODE = IA.SUBCODE AND
SUB.SEM = 8;
```

USN	SNAME	ADDRESS	PHONE	G	CAT
1RN13CS091	TEESHA	BENGALURU	7712312312	F	OutStanding
1RN13CS091	TEESHA	BENGALURU	7712312312	F	OutStanding
1RN13CS091	TEESHA	BENGALURU	7712312312	F	OutStanding
1RN13CS091	TEESHA	BENGALURU	7712312312	F	OutStanding
1RN13CS091	TEESHA	BENGALURU	7712312312	F	Average

---

**E. Consider the schema for Company Database:**

**EMPLOYEE** (SSN, Name, Address, Sex, Salary, SuperSSN, DNo)

**DEPARTMENT** (DNo, DName, MgrSSN, MgrStartDate)

**DLOCATION** (DNo, DLoc)

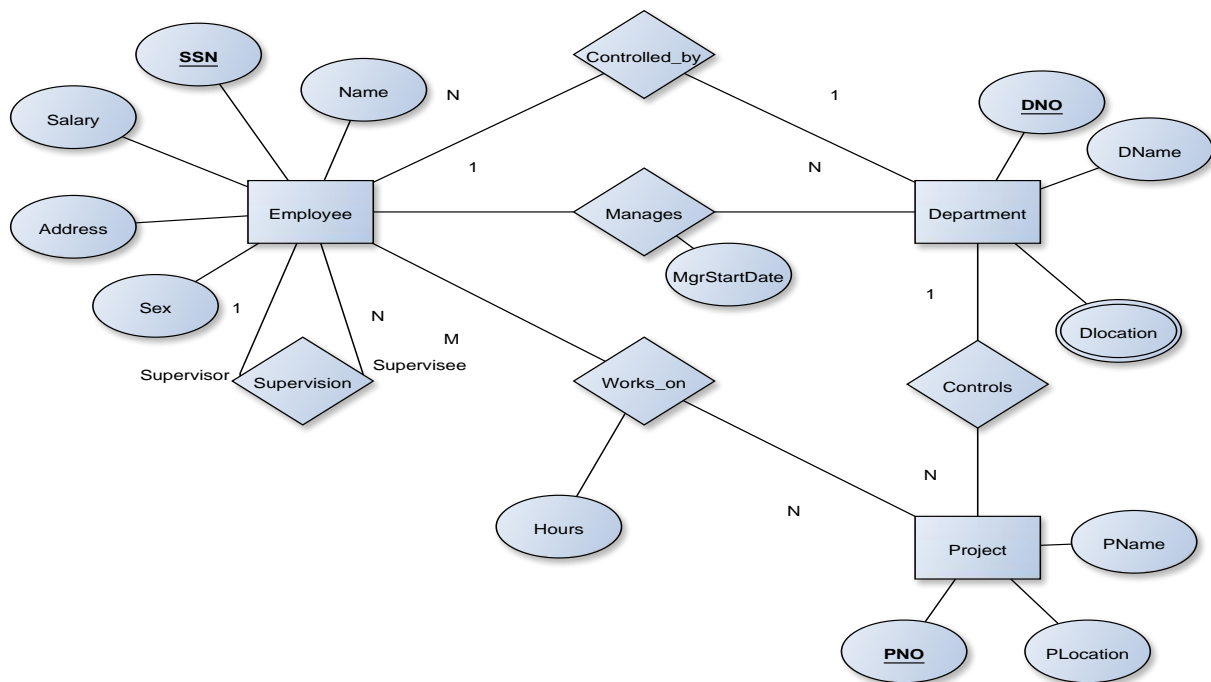
**PROJECT** (PNo, PName, PLocation, DNo)

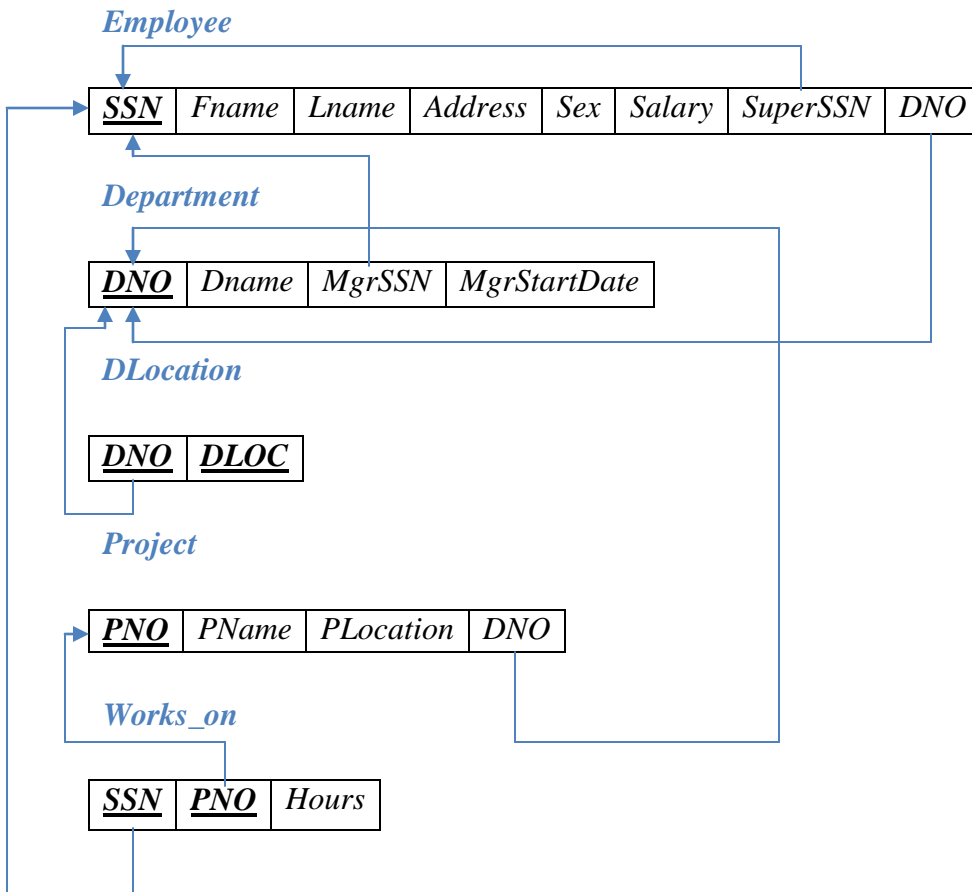
**WORKS\_ON** (SSN, PNo, Hours)

Write SQL queries to

1. Make a list of all project numbers for projects that involve an employee whose last name is 'Scott', either as a worker or as a manager of the department that controls the project.
2. Show the resulting salaries if every employee working on the 'IoT' project is given a 10 percent raise.
3. Find the sum of the salaries of all employees of the 'Accounts' department, as well as the maximum salary, the minimum salary, and the average salary in this department
4. Retrieve the name of each employee who works on all the projects controlled by department number 5 (use NOT EXISTS operator). For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than Rs. 6,00,000.

**Entity-Relationship Diagram**



**Schema Diagram****Table Creation**

```
CREATE TABLE DEPARTMENT
(DNO VARCHAR2 (20) PRIMARY KEY,
DNAME VARCHAR2 (20),
MGRSTARTDATE DATE);
```

```
CREATE TABLE EMPLOYEE
(SSN VARCHAR2 (20) PRIMARY KEY,
FNAME VARCHAR2 (20),
LNAME VARCHAR2 (20),
ADDRESS VARCHAR2 (20),
SEX CHAR (1),
SALARY INTEGER,
SUPERSSN REFERENCES EMPLOYEE (SSN),
DNO REFERENCES DEPARTMENT (DNO));
```

**NOTE:** Once DEPARTMENT and EMPLOYEE tables are created we must alter department table to add foreign constraint MGRSSN using sql command

```
ALTER TABLE DEPARTMENT  
ADD MGRSSN REFERENCES EMPLOYEE (SSN);
```

```
CREATE TABLE DLOCATION  
(DLOC VARCHAR2 (20),  
DNO REFERENCES DEPARTMENT (DNO),  
PRIMARY KEY (DNO, DLOC));
```

```
CREATE TABLE PROJECT  
(PNO INTEGER PRIMARY KEY,  
PNAME VARCHAR2 (20),  
PLOCATION VARCHAR2 (20),  
DNO REFERENCES DEPARTMENT (DNO));
```

```
CREATE TABLE WORKS_ON  
(HOURS NUMBER (2),  
SSN REFERENCES EMPLOYEE (SSN),  
PNO REFERENCES PROJECT(PNO),  
PRIMARY KEY (SSN, PNO));
```

### **Table Descriptions**

DESC EMPLOYEE;

```
SQL> DESC EMPLOYEE;
```

```
Name
```

```
-----  
SSN  
FNAME  
LNAME  
ADDRESS  
SEX  
SALARY  
SUPERSSN  
DNO
```

DESC DEPARTMENT;

```
SQL> DESC DEPARTMENT;
```

```
Name
```

```
-----  
DNO  
DNAME  
MGRSTARTDATE  
MGRSSN
```

DESC DLOCATION;

```
SQL> DESC DLOCATION;
```

```
Name
```

```
-----  
DLOC  
DNO
```

DESC PROJECT;

```
SQL> DESC PROJECT;
```

```
Name
```

```
-----  
PNO  
PNAME  
PLOCATION  
DNO
```

DESC WORKS\_ON;

```
SQL> DESC WORKS_ON;
```

```
Name
```

```
-----  
HOURS  
SSN  
PNO
```

### **Insertion of values to tables**

```
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES  
(‘RNSECE01’, ‘JOHN’, ‘SCOTT’, ‘BANGALORE’, ‘M’, 450000);
```

```
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES  
(‘RNSCSE01’, ‘JAMES’, ‘SMITH’, ‘BANGALORE’, ‘M’, 500000);
```

```
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES  
(‘RNSCSE02’, ‘HEARN’, ‘BAKER’, ‘BANGALORE’, ‘M’, 700000);
```

```
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES  
(‘RNSCSE03’, ‘EDWARD’, ‘SCOTT’, ‘MYSORE’, ‘M’, 500000);
```

```
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES  
(‘RNSCSE04’, ‘PAVAN’, ‘HEGDE’, ‘MANGALORE’, ‘M’, 650000);
```

```
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES  
(‘RNSCSE05’, ‘GIRISH’, ‘MALYA’, ‘MYSORE’, ‘M’, 450000);
```



```
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES
('RNSCSE06','NEHA','SN','BANGALORE','F', 800000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES
('RNSACC01','AHANA','K','MANGALORE','F', 350000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES
('RNSACC02','SANTHOSH','KUMAR','MANGALORE','M', 300000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES
('RNSISE01','VEENA','M','MYSORE','M', 600000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES
('RNSIT01','NAGESH','HR','BANGALORE','M', 500000);
```

```
INSERT INTO DEPARTMENT VALUES ('1','ACCOUNTS','01-JAN-01','RNSACC02');
INSERT INTO DEPARTMENT VALUES ('2','IT','01-AUG-16','RNSIT01');
INSERT INTO DEPARTMENT VALUES ('3','ECE','01-JUN-08','RNSECE01');
INSERT INTO DEPARTMENT VALUES ('4','ISE','01-AUG-15','RNSISE01');
INSERT INTO DEPARTMENT VALUES ('5','CSE','01-JUN-02','RNSCSE05');
```

**Note: update entries of employee table to fill missing fields SUPERSSN and DNO**

```
UPDATE EMPLOYEE SET
SUPERSSN=NULL, DNO='3'
WHERE SSN='RNSECE01';
```

```
UPDATE EMPLOYEE SET
SUPERSSN='RNSCSE02', DNO='5'
WHERE SSN='RNSCSE01';
```

```
UPDATE EMPLOYEE SET
SUPERSSN='RNSCSE03', DNO='5'
WHERE SSN='RNSCSE02';
```

```
UPDATE EMPLOYEE SET
SUPERSSN='RNSCSE04', DNO='5'
WHERE SSN='RNSCSE03';
```

```
UPDATE EMPLOYEE SET
DNO='5', SUPERSSN='RNSCSE05'
WHERE SSN='RNSCSE04';
```

```
UPDATE EMPLOYEE SET
DNO='5', SUPERSSN='RNSCSE06'
WHERE SSN='RNSCSE05';
```

```
UPDATE EMPLOYEE SET
DNO='5', SUPERSSN=NULL
WHERE SSN='RNSCSE06';
```

```
UPDATE EMPLOYEE SET
DNO='1', SUPERSSN='RNSACC02'
WHERE SSN='RNSACC01';
```

```
UPDATE EMPLOYEE SET
DNO='1', SUPERSSN=NULL
WHERE SSN='RNSACC02';
```

```
UPDATE EMPLOYEE SET
DNO='4', SUPERSSN=NULL
WHERE SSN='RNSISE01';
```

```
UPDATE EMPLOYEE SET
DNO='2', SUPERSSN=NULL
WHERE SSN='RNSIT01';
```

```
INSERT INTO DLOCATION VALUES ('BANGALORE', '1');
INSERT INTO DLOCATION VALUES ('BANGALORE', '2');
INSERT INTO DLOCATION VALUES ('BANGALORE', '3');
INSERT INTO DLOCATION VALUES ('MANGALORE', '4');
INSERT INTO DLOCATION VALUES ('MANGALORE', '5');
```

```
INSERT INTO PROJECT VALUES (100,'IOT','BANGALORE','5');
INSERT INTO PROJECT VALUES (101,'CLOUD','BANGALORE','5');
INSERT INTO PROJECT VALUES (102,'BIGDATA','BANGALORE','5');
INSERT INTO PROJECT VALUES (103,'SENSORS','BANGALORE','3');
INSERT INTO PROJECT VALUES (104,'BANK MANAGEMENT','BANGALORE','1');
INSERT INTO PROJECT VALUES (105,'SALARY MANAGEMENT','BANGALORE','1');
INSERT INTO PROJECT VALUES (106,'OPENSTACK','BANGALORE','4');
INSERT INTO PROJECT VALUES (107,'SMART CITY','BANGALORE','2');
```

```

INSERT INTO WORKS_ON VALUES (4, 'RNSCSE01', 100);
INSERT INTO WORKS_ON VALUES (6, 'RNSCSE01', 101);
INSERT INTO WORKS_ON VALUES (8, 'RNSCSE01', 102);
INSERT INTO WORKS_ON VALUES (10, 'RNSCSE02', 100);
INSERT INTO WORKS_ON VALUES (3, 'RNSCSE04', 100);
INSERT INTO WORKS_ON VALUES (4, 'RNSCSE05', 101);
INSERT INTO WORKS_ON VALUES (5, 'RNSCSE06', 102);
INSERT INTO WORKS_ON VALUES (6, 'RNSCSE03', 102);
INSERT INTO WORKS_ON VALUES (7, 'RNSECE01', 103);
INSERT INTO WORKS_ON VALUES (5, 'RNSACC01', 104);
INSERT INTO WORKS_ON VALUES (6, 'RNSACC02', 105);
INSERT INTO WORKS_ON VALUES (4, 'RNSISE01', 106);
INSERT INTO WORKS_ON VALUES (10, 'RNSIT01', 107);

```

```
SELECT * FROM EMPLOYEE;
```

SSN	FNAME	LNAME	ADDRESS	S	SALARY	SUPERSSN	DNO
RNSECE01	JOHN	SCOTT	BANGALORE	M	450000		3
RNSCSE01	JAMES	SMITH	BANGALORE	M	500000	RNSCSE02	5
RNSCSE02	HEARN	BAKER	BANGALORE	M	700000	RNSCSE03	5
RNSCSE03	EDWARD	SCOTT	MYSORE	M	500000	RNSCSE04	5
RNSCSE04	PAVAN	HEGDE	MANGALORE	M	650000	RNSCSE05	5
RNSCSE05	GIRISH	MALYA	MYSORE	M	450000	RNSCSE06	5
RNSCSE06	NEHA	SN	BANGALORE	F	800000		5
RNSACC01	AHANA	K	MANGALORE	F	350000	RNSACC02	1
RNSACC02	SANTHOSH	KUMAR	MANGALORE	M	300000		1
RNSISE01	VEENA	M	MYSORE	M	600000		4
RNSIT01	NAGESH	HR	BANGALORE	M	500000		2

```
SELECT * FROM DEPARTMENT;
```

```
SQL> SELECT * FROM DEPARTMENT;
```

DNO	DNAME	MGRSTARTD	MGRSSN
1	ACCOUNTS	01-JAN-01	RNSACC02
2	IT	01-AUG-16	RNSIT01
3	ECE	01-JUN-08	RNSECE01
4	ISE	01-AUG-15	RNSISE01
5	CSE	01-JUN-02	RNSCSE05

```
SELECT * FROM DLOCATION;
```

DLOC	DNO
BANGALORE	1
BANGALORE	2
BANGALORE	3
MANGALORE	4
MANGALORE	5

SELECT \* FROM PROJECT;

PNO	PNAME	PLOCATION	DNO
100	IOT	BANGALORE	5
101	CLOUD	BANGALORE	5
102	BIGDATA	BANGALORE	5
103	SENSORS	BANGALORE	3
104	BANK MANAGEMENT	BANGALORE	1
105	SALARY MANAGEMENT	BANGALORE	1
106	OPENSTACK	BANGALORE	4
107	SMART CITY	BANGALORE	2

SELECT \* FROM WORKS\_ON;

HOURS	SSN	PNO
4	RNSCSE01	100
6	RNSCSE01	101
8	RNSCSE01	102
10	RNSCSE02	100
3	RNSCSE04	100
4	RNSCSE05	101
5	RNSCSE06	102
6	RNSCSE03	102
7	RNSECE01	103
5	RNSACC01	104
6	RNSACC02	105
4	RNSISE01	106
10	RNSIT01	107

### Queries:

1. Make a list of all project numbers for projects that involve an employee whose last name is 'Scott', either as a worker or as a manager of the department that controls the project.

```
(SELECT DISTINCT P.PNO
FROM PROJECT P, DEPARTMENT D, EMPLOYEE E
WHERE E.DNO=D.DNO
AND D.MGRSSN=E.SSN
AND E.LNAME='SCOTT')
UNION
(SELECT DISTINCT P1.PNO
FROM PROJECT P1, WORKS_ON W, EMPLOYEE E1
WHERE P1.PNO=W.PNO
AND E1.SSN=W.SSN
AND E1.LNAME='SCOTT');
```

```
      PNO
-----
      100
      101
      102
      103
      104
      105
      106
      107
```

2. Show the resulting salaries if every employee working on the 'IoT' project is given a 10 percent raise.

```
SELECT E.FNAME, E.LNAME, 1.1*E.SALARY AS INCR_SAL
FROM EMPLOYEE E, WORKS_ON W, PROJECT P
WHERE E.SSN=W.SSN
AND W.PNO=P.PNO
AND P.PNAME='IOT';
```

FNAME	LNAME	INCR_SAL
JAMES	SMITH	550000
HEARN	BAKER	770000
PAVAN	HEGDE	715000

3. Find the sum of the salaries of all employees of the 'Accounts' department, as well as the maximum salary, the minimum salary, and the average salary in this department

```
SELECT SUM (E.SALARY), MAX (E.SALARY), MIN (E.SALARY), AVG
(E.SALARY)
FROM EMPLOYEE E, DEPARTMENT D
WHERE E.DNO=D.DNO
AND D.DNAME='ACCOUNTS';
```

SUM(E.SALARY)	MAX(E.SALARY)	MIN(E.SALARY)	AVG(E.SALARY)
650000	350000	300000	325000

4. Retrieve the name of each employee who works on all the projects Controlled by department number 5 (use NOT EXISTS operator).

```
SELECT E.FNAME, E.LNAME
FROM EMPLOYEE E
WHERE NOT EXISTS((SELECT PNO
                  FROM PROJECT
```

```
WHERE DNO='5')
EXCEPT(SELECT PNO
FROM WORKS_ON
WHERE E.SSN=SSN));
```

Or

```
SELECT LNAME, FNAME, DNO
FROM EMPLOYEE
WHERE NOT EXISTS (SELECT *
FROM WORKS_ON B
WHERE (B.PNO IN(SELECT P.PNO
FROM PROJECT P
WHERE P.DNO='5')
AND
NOT EXISTS (SELECT *
FROM WORKS_ON C
WHERE C.SSN=SSN
AND C.PNO=B.PNO)));
```

LNAME	FNAME
SCOTT	JOHN
SMITH	JAMES
BAKER	HEARN
SCOTT	EDWARD
HEGDE	PAVAN
MALYA	GIRISH
SN	NEHA
K	AHANA
KUMAR	SANTHOSH
M	VEENA
HR	NAGESH

5. For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than Rs. 6, 00,000.

```
SELECT D.DNO, COUNT (*)
FROM DEPARTMENT D, EMPLOYEE E
WHERE D.DNO=E.DNO
AND E.SALARY>600000
AND D.DNO IN (SELECT E1.DNO
FROM EMPLOYEE E1
GROUP BY E1.DNO
HAVING COUNT (*)>5)
```

GROUP BY D.DNO;

<b>DNO</b>	<b>COUNT (*)</b>
5	3

---

## Viva Questions

### 1. What is SQL?

Structured Query Language

### 2. What is database?

A database is a logically coherent collection of data with some inherent meaning, representing some aspect of real world and which is designed, built and populated with data for a specific purpose.

### 3. What is DBMS?

It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of defining, constructing and manipulating the database for various applications.

### 4. What is a Database system?

The database and DBMS software together is called as Database system.

### 5. Advantages of DBMS?

- Redundancy is controlled.
- Unauthorized access is restricted.
- Providing multiple user interfaces.
- Enforcing integrity constraints.
- Providing backup and recovery.

### 6. Disadvantage in File Processing System?

- Data redundancy & inconsistency.
- Difficult in accessing data.
- Data isolation.
- Data integrity.
- Concurrent access is not possible.
- Security Problems.

### 7. Describe the three levels of data abstraction?

There are three levels of abstraction:

- Physical level: The lowest level of abstraction describes how data are stored.
- Logical level: The next higher level of abstraction, describes what data are stored in database and what relationship among those data.



- View level: The highest level of abstraction describes only part of entire database.

## 8. Define the "integrity rules"

There are two Integrity rules.

- Entity Integrity: States that "Primary key cannot have NULL value"
- Referential Integrity: States that "Foreign Key can be either a NULL value or should be Primary Key value of other relation."

## 9. What is extension and intension?

Extension - It is the number of tuples present in a table at any instance. This is time dependent.

Intension - It is a constant value that gives the name, structure of table and the constraints laid on it.

## 10. What is Data Independence?

Data independence means that "the application is independent of the storage structure and access strategy of data". In other words, The ability to modify the schema definition in one level should not affect the schema definition in the next higher level.

Two types of Data Independence:

- Physical Data Independence: Modification in physical level should not affect the logical level.
- Logical Data Independence: Modification in logical level should affect the view level.

NOTE: Logical Data Independence is more difficult to achieve

## 11. What is a view? How it is related to data independence?

A view may be thought of as a virtual table, that is, a table that does not really exist in its own right but is instead derived from one or more underlying base table. In other words, there is no stored file that directly represents the view instead a definition of view is stored in data dictionary.

Growth and restructuring of base tables is not reflected in views. Thus the view can insulate users from the effects of restructuring and growth in the database. Hence accounts for logical data independence.

## 12. What is Data Model?

A collection of conceptual tools for describing data, data relationships data semantics and constraints.

### **13. What is E-R model?**

This data model is based on real world that consists of basic objects called entities and of relationship among these objects. Entities are described in a database by a set of attributes.

### **14. What is Object Oriented model?**

This model is based on collection of objects. An object contains values stored in instance variables within the object. An object also contains bodies of code that operate on the object. These bodies of code are called methods. Objects that contain same types of values and the same methods are grouped together into classes.

### **15. What is an Entity?**

It is an 'object' in the real world with an independent existence.

### **16. What is an Entity type?**

It is a collection (set) of entities that have same attributes.

### **17. What is an Entity set?**

It is a collection of all entities of particular entity type in the database.

### **18. What is an Extension of entity type?**

The collections of entities of a particular entity type are grouped together into an entity set.

### **19. What is an attribute?**

It is a particular property, which describes the entity.

### **20. What is a Relation Schema and a Relation?**

A relation Schema denoted by  $R(A_1, A_2, \dots, A_n)$  is made up of the relation name  $R$  and the list of attributes  $A_i$  that it contains. A relation is defined as a set of tuples. Let  $r$  be the relation which contains set tuples  $(t_1, t_2, t_3, \dots, t_n)$ . Each tuple is an ordered list of  $n$ -values  $t=(v_1, v_2, \dots, v_n)$ .

### **21. What is degree of a Relation?**

It is the number of attribute of its relation schema.

### **22. What is Relationship?**

It is an association among two or more entities.

### **23. What is Relationship set?**

The collection (or set) of similar relationships.

**24. What is Relationship type?**

Relationship type defines a set of associations or a relationship set among a given set of entity types.

**25. What is degree of Relationship type?**

It is the number of entity type participating.

**26. What is DDL (Data Definition Language)?**

A data base schema is specified by a set of definitions expressed by a special language called DDL.

**27. What is VDL (View Definition Language)?**

It specifies user views and their mappings to the conceptual schema.

**28. What is SDL (Storage Definition Language)?**

This language is to specify the internal schema. This language may specify the mapping between two schemas.

**29. What is Data Storage - Definition Language?**

The storage structures and access methods used by database system are specified by a set of definition in a special type of DDL called data storage- definition language.

**30. What is DML (Data Manipulation Language)?**

This language that enable user to access or manipulate data as organized by appropriate data model.

- Procedural DML or Low level: DML requires a user to specify what data are needed and how to get those data.
- Non-Procedural DML or High level: DML requires a user to specify what data are needed without specifying how to get those data.

**31. What is DML Compiler?**

It translates DML statements in a query language into low-level instruction that the query evaluation engine can understand.

**32. What is Relational Algebra?**

It is a procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation.

**33. What is Relational Calculus?**

It is an applied predicate calculus specifically tailored for relational databases proposed by E.F. Codd. E.g. of languages based on it are DSL, ALPHA, QUEL.

### 34. What is normalization?

It is a process of analyzing the given relation schemas based on their Functional Dependencies (FDs) and primary key to achieve the properties

- Minimizing redundancy
- Minimizing insertion, deletion and update anomalies.

### 35. What is Functional Dependency?

A Functional dependency is denoted by  $X \rightarrow Y$  between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuple that can form a relation state r of R. The constraint is for any two tuples t1 and t2 in r if  $t1[X] = t2[X]$  then they have  $t1[Y] = t2[Y]$ . This means the value of X component of a tuple uniquely determines the value of component Y.

### 36. When is a functional dependency F said to be minimal?

- Every dependency in F has a single attribute for its right hand side.
- We cannot replace any dependency  $X \rightarrow A$  in F with a dependency  $Y \rightarrow A$  where Y is a proper subset of X and still have a set of dependency that is equivalent to F.
- We cannot remove any dependency from F and still have set of dependency that is equivalent to F.

### 37. What is Multivalued dependency?

Multivalued dependency denoted by  $X \twoheadrightarrow Y$  specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation r of R: if two tuples t1 and t2 exist in r such that  $t1[X] = t2[X]$  then t3 and t4 should also exist in r with the following properties

- $t3[X] = t4[X] = t1[X] = t2[X]$
- $t3[Y] = t1[Y]$  and  $t4[Y] = t2[Y]$
- $t3[Z] = t2[Z]$  and  $t4[Z] = t1[Z]$

where  $[Z = (R - (X \cup Y))]$

### 38. What is Lossless join property?

It guarantees that the spurious tuple generation does not occur with respect to relation schemas after decomposition.

### 39. What is 1 NF (Normal Form)?

The domain of attribute must include only atomic (simple, indivisible) values.

### 40. What is Fully Functional dependency?

It is based on concept of full functional dependency. A functional dependency  $X \rightarrow Y$  is fully functional dependency if removal of any attribute A from X means that the dependency does not hold any more.

### 41. What is 2NF?

A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on primary key.

### 42. What is 3NF?

A relation schema R is in 3NF if it is in 2NF and for every FD  $X \rightarrow A$  either of the following is true

- X is a Super-key of R.
- A is a prime attribute of R.

In other words, if every non prime attribute is non-transitively dependent on primary key.

### 43. What is BCNF (Boyce-Codd Normal Form)?

A relation schema R is in BCNF if it is in 3NF and satisfies additional constraints that for every FD  $X \rightarrow A$ , X must be a candidate key.

### 44. What is 4NF?

A relation schema R is said to be in 4NF if for every Multivalued dependency  $X \twoheadrightarrow Y$  that holds over R, one of following is true

- X is subset or equal to (or)  $XY = R$ .
- X is a super key.

### 45. What is 5NF?

A Relation schema R is said to be 5NF if for every join dependency  $\{R_1, R_2, \dots, R_n\}$  that holds R, one the following is true

- $R_i = R$  for some i.
- The join dependency is implied by the set of FD, over R in which the left side is key of R.

#### **46. What is Domain-Key Normal Form?**

A relation is said to be in DKNF if all constraints and dependencies that should hold on the constraint can be enforced by simply enforcing the domain constraint and key constraint on the relation.

#### **47. What are partial, alternate,, artificial, compound and natural key?**

Partial Key:

It is a set of attributes that can uniquely identify weak entities and that are related to same owner entity. It is sometime called as Discriminator.

Alternate Key:

All Candidate Keys excluding the Primary Key are known as Alternate Keys.

ArtificialKey:

If no obvious key, either stand alone or compound is available, then the last resort is to simply create a key, by assigning a unique number to each record or occurrence. Then this is known as developing an artificial key.

CompoundKey:

If no single data element uniquely identifies occurrences within a construct, then combining multiple elements to create a unique identifier for the construct is known as creating a compound key.

NaturalKey:

When one of the data elements stored within a construct is utilized as the primary key, then it is called the natural key.

#### **48. What is indexing and what are the different kinds of indexing?**

Indexing is a technique for determining how quickly specific data can be found.

- Binary search style indexing
- B-Tree indexing
- Inverted list indexing
- Memory resident table
- Table indexing

**49. What is system catalog or catalog relation? How is better known as?**

A RDBMS maintains a description of all the data that it contains, information about every relation and index that it contains. This information is stored in a collection of relations maintained by the system called metadata. It is also called data dictionary.

**50. What is meant by query optimization?**

The phase that identifies an efficient execution plan for evaluating a query that has the least estimated cost is referred to as query optimization.

**51. What is join dependency and inclusion dependency?**

JoinDependency:

A Join dependency is generalization of Multivalued dependency. A JD  $\{R_1, R_2, \dots, R_n\}$  is said to hold over a relation R if  $R_1, R_2, R_3, \dots, R_n$  is a lossless-join decomposition of R. There is no set of sound and complete inference rules for JD.

InclusionDependency:

An Inclusion Dependency is a statement of the form that some columns of a relation are contained in other columns. A foreign key constraint is an example of inclusion dependency.

**52. What is durability in DBMS?**

Once the DBMS informs the user that a transaction has successfully completed, its effects should persist even if the system crashes before all its changes are reflected on disk. This property is called durability.

**53. What do you mean by atomicity and aggregation?**

Atomicity:

Either all actions are carried out or none are. Users should not have to worry about the effect of incomplete transactions. DBMS ensures this by undoing the actions of incomplete transactions.

Aggregation:

A concept which is used to model a relationship between a collection of entities and relationships. It is used when we need to express a relationship among relationships.

#### **54. What is a Phantom Deadlock?**

In distributed deadlock detection, the delay in propagating local information might cause the deadlock detection algorithms to identify deadlocks that do not really exist. Such situations are called phantom deadlocks and they lead to unnecessary aborts.

#### **55. What is a checkpoint and when does it occur?**

A Checkpoint is like a snapshot of the DBMS state. By taking checkpoints, the DBMS can reduce the amount of work to be done during restart in the event of subsequent crashes.

#### **56. What are the different phases of transaction?**

Different phases are

- Analysis phase
- Redo Phase
- Undo phase

#### **57. What do you mean by flat file database?**

It is a database in which there are no programs or user access languages. It has no cross-file capabilities but is user-friendly and provides user-interface management.

#### **58. What is "transparent DBMS"?**

It is one, which keeps its Physical Structure hidden from user.

#### **59. Brief theory of Network, Hierarchical schemas and their properties**

Network schema uses a graph data structure to organize records example for such a database management system is CTCG while a hierarchical schema uses a tree data structure example for such a system is IMS.

#### **60. What is a query?**

A query with respect to DBMS relates to user commands that are used to interact with a data base. The query language can be classified into data definition language and data manipulation language.

#### **61. What do you mean by Correlated subquery?**

Subqueries, or nested queries, are used to bring back a set of rows to be used by the parent query. Depending on how the subquery is written, it can be executed once for the parent



query or it can be executed once for each row returned by the parent query. If the subquery is executed for each row of the parent, this is called a *correlated subquery*.

A correlated subquery can be easily identified if it contains any references to the parent subquery columns in its WHERE clause. Columns from the subquery cannot be referenced anywhere else in the parent query. The following example demonstrates a non-correlated subquery.

E.g. Select \* From CUST Where '10/03/1990' IN (Select ODATE From ORDER Where CUST.CNUM = ORDER.CNUM)

**62. What are the primitive operations common to all record management systems?**

Addition, deletion and modification.

**63. Name the buffer in which all the commands that are typed in are stored**

'Edit' Buffer

**64. What are the unary operations in Relational Algebra?**

PROJECTION and SELECTION.

**65. Are the resulting relations of PRODUCT and JOIN operation the same?**

No.

*PRODUCT*: Concatenation of every row in one relation with every row in another.

*JOIN*: Concatenation of rows from one relation and related rows from another.

**66. What is RDBMS KERNEL?**

Two important pieces of RDBMS architecture are the kernel, which is the software, and the data dictionary, which consists of the system-level data structures used by the kernel to manage the database

You might think of an RDBMS as an operating system (or set of subsystems), designed specifically for controlling data access; its primary functions are storing, retrieving, and securing data. An RDBMS maintains its own list of authorized users and their associated privileges; manages memory caches and paging; controls locking for concurrent resource usage; dispatches and schedules user requests; and manages space usage within its table-space structures.

**67. Name the sub-systems of a RDBMS**

I/O, Security, Language Processing, Process Control, Storage Management, Logging and Recovery, Distribution Control, Transaction Control, Memory Management, Lock Management

**68. Which part of the RDBMS takes care of the data dictionary? How**

Data dictionary is a set of tables and database objects that is stored in a special area of the database and maintained exclusively by the kernel.

**69. What is the job of the information stored in data-dictionary?**

The information in the data dictionary validates the existence of the objects, provides access to them, and maps the actual physical storage location.

**70. Not only RDBMS takes care of locating data it also—————**

Determines an optimal access path to store or retrieve the data

**71. How do you communicate with an RDBMS?**

You communicate with an RDBMS using Structured Query Language (SQL)

**72. Define SQL and state the differences between SQL and other conventional programming Languages**

SQL is a nonprocedural language that is designed specifically for data access operations on normalized relational database structures. The primary difference between SQL and other conventional programming languages is that SQL statements specify what data operations should be performed rather than how to perform them.

**73. Name the three major set of files on disk that compose a database in Oracle**

There are three major sets of files on disk that compose a database. All the files are binary. These are

- Database files
- Control files
- Redo logs

The most important of these are the database files where the actual data resides. The control files and the redo logs support the functioning of the architecture itself.

All three sets of files must be present, open, and available to Oracle for any data on the database to be useable. Without these files, you cannot access the database, and the database administrator might have to recover some or all of the database using a backup, if there is one.

**74. What is an Oracle Instance?**

The Oracle system processes, also known as Oracle background processes, provide functions for the user processes—functions that would otherwise be done by the user processes themselves

Oracle database-wide system memory is known as the SGA, the system global area or shared global area. The data and control structures in the SGA are shareable, and all the Oracle background processes and user processes can use them.

The combination of the SGA and the Oracle background processes is known as an Oracle instance

**75. What are the four Oracle system processes that must always be up and running for the database to be useable**

The four Oracle system processes that must always be up and running for the database to be useable include DBWR (Database Writer), LGWR (Log Writer), SMON (System Monitor), and PMON (Process Monitor).

**76. What are database files, control files and log files. How many of these files should a database have at least? Why?**

**Database Files**

The database files hold the actual data and are typically the largest in size. Depending on their sizes, the tables (and other objects) for all the user accounts can go in one database file—but that's not an ideal situation because it does not make the database structure very flexible for controlling access to storage for different users, putting the database on different disk drives, or backing up and restoring just part of the database.

You must have at least one database file but usually, more than one files are used. In terms of accessing and using the data in the tables and other objects, the number (or location) of the files is immaterial.

The database files are fixed in size and never grow bigger than the size at which they were created

**ControlFiles**

The control files and redo logs support the rest of the architecture. Any database must have at least one control file, although you typically have more than one to guard against loss. The control file records the name of the database, the date and time it was created, the location of the database and redo logs, and the synchronization information to ensure that all three sets of files are always in step. Every time you add a new database or redo log file to the database, the information is recorded in the control files.

## Redo Logs

Any database must have at least two redo logs. These are the journals for the database; the redo logs record all changes to the user objects or system objects. If any type of failure occurs, the changes recorded in the redo logs can be used to bring the database to a consistent state without losing any committed transactions. In the case of non-data loss failure, Oracle can apply the information in the redo logs automatically without intervention from the DBA.

The redo log files are fixed in size and never grow dynamically from the size at which they were created.

### **77. What is ROWID?**

The ROWID is a unique database-wide physical address for every row on every table. Once assigned (when the row is first inserted into the database), it never changes until the row is deleted or the table is dropped.

The ROWID consists of the following three components, the combination of which uniquely identifies the physical storage location of the row.

- Oracle database file number, which contains the block with the rows
- Oracle block address, which contains the row
- The row within the block (because each block can hold many rows)

The ROWID is used internally in indexes as a quick means of retrieving rows with a particular key value. Application developers also use it in SQL statements as a quick way to access a row once they know the ROWID

### **78. What is Oracle Block? Can two Oracle Blocks have the same address?**

Oracle "formats" the database files into a number of Oracle blocks when they are first created—making it easier for the RDBMS software to manage the files and easier to read data into the memory areas.

The block size should be a multiple of the operating system block size. Regardless of the block size, the entire block is not available for holding data; Oracle takes up some space to manage the contents of the block. This block header has a minimum size, but it can grow.

These Oracle blocks are the smallest unit of storage. Increasing the Oracle block size can improve performance, but it should be done only when the database is first created.

Each Oracle block is numbered sequentially for each database file starting at 1. Two blocks can have the same block address if they are in different database files.

**79. What is database Trigger?**

A database trigger is a PL/SQL block that can be defined to automatically execute for insert, update, and delete statements against a table. The trigger can be defined to execute once for the entire statement or once for every row that is inserted, updated, or deleted. For any one table, there are twelve events for which you can define database triggers. A database trigger can call database procedures that are also written in PL/SQL.

**80. Name two utilities that Oracle provides, which are used for backup and recovery.**

Along with the RDBMS software, Oracle provides two utilities that you can use to back up and restore the database. These utilities are Export and Import.

The Export utility dumps the definitions and data for the specified part of the database to an operating system binary file. The Import utility reads the file produced by an export, recreates the definitions of objects, and inserts the data.

If Export and Import are used as a means of backing up and recovering the database, all the changes made to the database cannot be recovered since the export was performed. The best you can do is recovering the database to the time when the export was last performed.

**81. Name two utilities that Oracle provides, which are used for backup and recovery.**

Along with the RDBMS software, Oracle provides two utilities that you can use to back up and restore the database. These utilities are Export and Import.

The Export utility dumps the definitions and data for the specified part of the database to an operating system binary file. The Import utility reads the file produced by an export, recreates the definitions of objects, and inserts the data.

If Export and Import are used as a means of backing up and recovering the database, all the changes made to the database cannot be recovered since the export was performed. The best you can do is recovering the database to the time when the export was last performed.

**82. What are stored-procedures? And what are the advantages of using them.**

Stored procedures are database objects that perform a user defined operation. A stored procedure can have a set of compound SQL statements. A stored procedure executes the SQL commands and returns the result to the client. Stored procedures are used to reduce network traffic.

**83. Tables derived from the ERD**

- a) Are totally unnormalised
- b) Are always in 1NF
- c) Can be further denormalised
- d) May have multi-valued attributes
- e) Are always in 1NF

**84. Spurious tuples may occur due to**

- i. Bad normalization
- ii. Theta joins
- iii. Updating tables from join
  - a) i& ii
  - b) ii & iii
  - c) i& iii
  - d) ii & iii

(a) i& iii because theta joins are joins made on keys that are not primary keys.

**85. In mapping of ERD to DFD**

- a) entities in ERD should correspond to an existing entity/store in DFD
  - b) entity in DFD is converted to attributes of an entity in ERD
  - c) relations in ERD has 1 to 1 correspondence to processes in DFD
  - d) relationships in ERD has 1 to 1 correspondence to flows in DFD
- (a) entities in ERD should correspond to an existing entity/store in DFD

**86. A dominant entity is the entity**

- a) on the N side in a 1 : N relationship
  - b) on the 1 side in a 1 : N relationship
  - c) on either side in a 1 : 1 relationship
  - d) nothing to do with 1 : 1 or 1 : N relationship
- (b) on the 1 side in a 1 : N relationship

**87. Select 'NORTH', CUSTOMER From CUST\_DTLS Where REGION = 'N' Order By  
CUSTOMER Union Select 'EAST', CUSTOMER From CUST\_DTLS Where REGION  
= 'E' Order By CUSTOMER**

The above is

- a) Not an error
- b) Error - the string in single quotes 'NORTH' and 'SOUTH'

c) Error - the string should be in double quotes

d) Error - ORDER BY clause

(d) Error - the ORDER BY clause. Since ORDER BY clause cannot be used in UNIONS

### **88. What is Storage Manager?**

It is a program module that provides the interface between the low-level data stored in database, application programs and queries submitted to the system.

### **89. What is Buffer Manager?**

It is a program module, which is responsible for fetching data from disk storage into main memory and deciding what data to be cache in memory.

### **90. What is Transaction Manager?**

It is a program module, which ensures that database, remains in a consistent state despite system failures and concurrent transaction execution proceeds without conflicting.

### **91. What is File Manager?**

It is a program module, which manages the allocation of space on disk storage and data structure used to represent information stored on a disk.

### **92. What is Authorization and Integrity manager?**

It is the program module, which tests for the satisfaction of integrity constraint and checks the authority of user to access data.

### **93. What are stand-alone procedures?**

Procedures that are not part of a package are known as stand-alone because they independently defined. A good example of a stand-alone procedure is one written in a SQL\*Forms application. These types of procedures are not available for reference from other Oracle tools. Another limitation of stand-alone procedures is that they are compiled at run time, which slows execution.

### **94. What are cursors give different types of cursors.**

PL/SQL uses cursors for all database information accesses statements. The language supports the use two types of cursors

➤ *Implicit*

➤ *Explicit*

### **95. What is cold backup and hot backup (in case of Oracle)?**

➤ Cold Backup:

It is copying the three sets of files (database files, redo logs, and control file) when the instance is shut down. This is a straight file copy, usually from the disk directly to tape. You must shut down the instance to guarantee a consistent copy.

If a cold backup is performed, the only option available in the event of data file loss is restoring all the files from the latest backup. All work performed on the database since the last backup is lost.

➤ **Hot Backup:**

Some sites (such as worldwide airline reservations systems) cannot shut down the database while making a backup copy of the files. The cold backup is not an available option.

So different means of backing up database must be used — the hot backup. Issue a SQL command to indicate to Oracle, on a tablespace-by-tablespace basis, that the files of the tablespace are to be backed up. The users can continue to make full use of the files, including making changes to the data. Once the user has indicated that he/she wants to back up the tablespace files, he/she can use the operating system to copy those files to the desired backup destination.

The database must be running in ARCHIVELOG mode for the hot backup option. If a data loss failure does occur, the lost database files can be restored using the hot backup and the online and offline redo logs created since the backup was done. The database is restored to the most consistent state without any loss of committed transactions.

### **96. How can you find the minimal key of relational schema?**

Minimal key is one which can identify each tuple of the given relation schema uniquely. For finding the minimal key it is required to find the closure that is the set of all attributes that are dependent on any given set of attributes under the given set of functional dependency.

**Algo. I** Determining  $X^+$ , closure for X, given set of FDs F

1. Set  $X^+ = X$
2. Set Old  $X^+ = X^+$
3. For each FD  $Y \rightarrow Z$  in F and if Y belongs to  $X^+$  then add Z to  $X^+$
4. Repeat steps 2 and 3 until Old  $X^+ = X^+$

**Algo. II** Determining minimal K for relation schema R, given set of FDs F

1. Set K to R that is make K a set of all attributes in R



2. For each attribute A in K
  - a. Compute  $(K - A)^+$  with respect to F
  - b. If  $(K - A)^+ = R$  then set  $K = (K - A)^+$

**97. What do you understand by dependency preservation?**

Given a relation R and a set of FDs F, dependency preservation states that the closure of the union of the projection of F on each decomposed relation  $R_i$  is equal to the closure of F. i.e.,

$$((\Pi_{R_1}(F)) \cup \dots \cup (\Pi_{R_n}(F)))^+ = F^+$$

if decomposition is not dependency preserving, then some dependency is lost in the decomposition.

**98. What is meant by Proactive, Retroactive and Simultaneous Update.**

*Proactive Update:*

The updates that are applied to database before it becomes effective in real world.

*Retroactive Update:*

The updates that are applied to database after it becomes effective in real world.

*Simultaneous Update:*

The updates that are applied to database at the same time when it becomes effective in real world

**.What are the different types of JOIN operations?**

Equi Join: This is the most common type of join which involves only equality comparisons. The disadvantage in this type of join is that there

---

**SQL Questions:**

**1. Which is the subset of SQL commands used to manipulate Oracle Database structures, including tables?**

Data Definition Language (DDL)

**2. What operator performs pattern matching?**

LIKE operator

**3. What operator tests column for the absence of data?**

IS NULL operator

**4. Which command executes the contents of a specified file?**

START <filename> or @<filename>

**5. What is the parameter substitution symbol used with INSERT INTO command?**

&

**6. Which command displays the SQL command in the SQL buffer, and then executes it?**

RUN

**7. What are the wildcards used for pattern matching?**

For single character substitution and % for multi-character substitution

**8. State true or false. EXISTS, SOME, ANY are operators in SQL.**

True

**9. State true or false. !=, <>, ^= all denote the same operation.**

True

**10. What are the privileges that can be granted on a table by a user to others?**

Insert, update, delete, select, references, index, execute, alter, all

**11. What command is used to get back the privileges offered by the GRANT command?**

REVOKE

**12. Which system tables contain information on privileges granted and privileges obtained?**

USER\_TAB\_PRIVS\_MADE, USER\_TAB\_PRIVS\_RECD

**13. Which system table contains information on constraints on all the tables created?**

USER\_CONSTRAINTS

**14. TRUNCATE TABLE EMP;**

**DELETE FROM EMP;**

**Will the outputs of the above two commands differ?**

Both will result in deleting all the rows in the table EMP.

**15. What the difference is between TRUNCATE and DELETE commands?**

TRUNCATE is a DDL command whereas DELETE is a DML command. Hence DELETE operation can be rolled back, but TRUNCATE operation cannot be rolled back. WHERE clause can be used with DELETE and not with TRUNCATE.

**16. What command is used to create a table by copying the structure of another table?**

*Answer:*

CREATE TABLE AS SELECT command

**Explanation:**

To copy only the structure, the WHERE clause of the SELECT command should contain a FALSE statement as in the following.

```
CREATE TABLE NEWTABLE AS SELECT * FROM EXISTINGTABLE WHERE  
1=2;
```

If the WHERE condition is true, then all the rows or rows satisfying the condition will be copied to the new table.

**17. What will be the output of the following query?**

```
SELECT REPLACE (TRANSLATE(LTRIM(RTRIM('!! ATHEN !!,!'), '!'), 'AN',  
**),'*', 'TROUBLE') FROM DUAL;  
TROUBLETHETROUBLE
```

**18. What will be the output of the following query?**

```
SELECT DECODE(TRANSLATE('A','1234567890','1111111111'), '1','YES', 'NO');
```

*Answer :*

NO

**Explanation :**

The query checks whether a given string is a numerical digit.

**19. What does the following query do?**

```
SELECT SAL + NVL(COMM,0) FROM EMP;
```

This displays the total salary of all employees. The null values in the commission column will be replaced by 0 and added to salary.

**20. Which date function is used to find the difference between two dates?**

MONTHS\_BETWEEN

**21. Why does the following command give a compilation error?**

```
DROP TABLE &TABLE_NAME;
```

Variable names should start with an alphabet. Here the table name starts with an '&' symbol.

**22. What is the advantage of specifying WITH GRANT OPTION in the GRANT command?**

The privilege receiver can further grant the privileges he/she has obtained from the owner to any other user.

**23. What is the use of the DROP option in the ALTER TABLE command?**

It is used to drop constraints specified on the table.

**24. What is the value of 'comm' and 'sal' after executing the following query if the initial value of 'sal' is 10000?**

```
UPDATE EMP SET SAL = SAL + 1000, COMM = SAL*0.1;
```

sal = 11000, comm = 1000

**25. What is the use of DESC in SQL?**

DESC has two purposes. It is used to describe a schema as well as to retrieve rows from table in descending order.

The query `SELECT * FROM EMP ORDER BY ENAME DESC` will display the output sorted on ENAME in descending order.

**26. What is the use of CASCADE CONSTRAINTS?**

When this clause is used with the DROP command, a parent table can be dropped even when a child table exists.

**27. Which function is used to find the largest integer less than or equal to a specific value?**

FLOOR

**28. What is the output of the following query?**

```
SELECT TRUNC(1234.5678,-2) FROM DUAL;
```

1200

**SQL HANDSON****Questions Based On Data Definition Language Commands:****1. Create A Table EMP And DEPT Using The Following Information.****a. DEPT:**

COLUMN NAME	DATATYPE (SIZE)
-----	
DEPTNO	NUMBER (2)
DNAME	VARCHAR2 (14)
LOC	VARCHAR2 (14)

**b. EMP:**

COLUMN NAME	DATATYPE(SIZE)
-----	
EMPNO	NUMBER (4)
ENAME	VARCHAR2 (10)
JOB	VARCHAR2 (9)
MGR	NUMBER (4)
HIREDATE	DATE
SAL	NUMBER (7, 2)
COMM	NUMBER (7, 2)
DEPTNO	NUMBER (2)

**2. Check the Default Size of a Number, Char and Date Data types.****3. Describe the Structure of the Table and EMP Table.****4. Add two columns to the table EMP with the following information in one single****ALTER COMMAND.**

COLUMN NAME	DATATYPE (SIZE)
-----	
SEX	CHAR (1)
PLACE	CHAR (15)

5. Modify the column **job** present in the **EMP** table with the following information given below:

COLUMN NAME	DATATYPE (SIZE)
-----	
JOB	VARCHAR2 (15)

6. modify the column **ENAME** present in the **EMP** table with the following information given below:

COLUMN NAME	DATATYPE (SIZE)
-----	
ENAME	CHAR (15)

7. Decrease the size for the column **EMPNO** with the following information:-

COLUMN NAME	DATATYPE (SIZE)
-----	
EMPNO	NUMBER (2)

8. Modify the column name of **EMPNO** to **EMPLOYEE\_NUMBER** present in the **EMP** table verify the result.

9. Add a new column **nationality** placed between **JOB** and **MGR** columns and verify the result

10. Drop the table **DEPT** and **EMP**.

11. What is the data type of the column **HIREDATE** and how many bytes it occupies.

12. Create a table **EMP** and **DEPT** using the following information.

a. **DEPT:**

COLUMN NAME	DATATYPE(SIZE)
-----	
DEPTNO	NUMBER(2) CONSTRAINT PK_DEPTNO PRIMARY KEY
DNAME	VARCHAR2(14)
LOC	VARCHAR2(14)

b. **EMP:**

COLUMN NAME	DATATYPE(SIZE)
-------------	----------------

```
-----  
EMPNO                NUMBER(4)    CONSTRAINT    PK_EMPNO  
                    PRIMARY KEY  
ENAME                VARCHAR2(10)  CONSTRAINT    UQ_DEPTNO  
                    UNIQUE  
JOB                  VARCHAR2(9)  
MGR                  NUMBER(4)  
HIREDATE             DATE DEFAULT SYSDATE  
SAL                  NUMBER(7,2)    CONSTRAINT    CK_SAL  
                    CHECK(SAL)  
COMM                 NUMBER(7,2)  
DEPTNO               NUMBER(2)    CONSTRAINT    FK_DEPTNO  
                    REFERENCE
```

**13. Select all the constraints in the EMP table**

SOL: SELECT \* FROM USER\_CONSTRAINTS WHERE TABLE\_NAME='EMP';

**14. select the owner, constraints name, constraints type, table name, status for DEPT table**

SOL: SELECT OWNER, CONSTRAINTS\_NAME, CONSTRAINTS\_TYPE, TABLE\_NAME, STATUS FROM USER\_CONSTRAINTS WHERE TABLE\_NAME='DEPT';

**15. Drop the constraints UQ\_FNAME from EMP table**

SOL: ALTER TABLE EMP DROP CONSTRAINT UQ\_FNAME;

**16. Add a new column PINCODE with not null constraints to the existing table DEPT**

SOL: ALTER TABLE DEPT ADD(PINCODE NUMBER(6) NOT NULL);

**17. Disable the constraints PK\_DEPTNO present in the DEPT table**

SOL: ALTER TABLE DEPT DISABLE CONSTRAINTS PK\_DEPTNO;

**18. Enable the constraints PK\_DEPTNO which is defined in the DEPTNO column of DEPT table;**

SOL: ALTER TABLE DEPT ENABLE CONSTRAINTS PK\_DEPTNO;

**19. Insert the given values into the tables:**

**EMP:**

- (i) 7369, SMITH, CLERK, 7902, 17 – DEC – 80, 800, NULL, 20
- (ii) 7499, ALLEN SALEMAN, 7698, 20 – FEB – 81, 1600, 300, 30
- (iii) 7521, WARD, SALESMAN, 7698, 22 – FEB – 81, 1250, 500, 30
- (iv) 7566, JONES, MANAGER, 7839, 02 – APR – 81, 2975, NULL, 20
- (v) 7654, MARTIN, SALESMAN, 7698, 28 – SEP – 81, 1250, 1400, 30
- (vi) 7698, BLAKE, MANAGER, 7839, 01 – MAY – 81, 2850, NULL, 30
- (vii) 7782, CLERK, MANAGER, 7839, 09 – JUN – 81, 2450, NULL, 10
- (viii) 7788, SCOTT, ANALYST, 7566, 19 – NOV – 96, 3000, NULL, 20
- (ix) 7839, KING, PRESIDENT, NULL, 17 – NOV – 81, 5000, NULL, 10
- (x) 7844, TURNER, SALESMAN, 7698, 08 – SEP – 81, 1500, 0, 30
- (xi) 7876, ADAMS, CLERK , 7788, 23 – DEC – 96, 1100, NULL, 20
- (xii) 7900, JAMES, CLERK, 7698, 03 – DEC – 81, 950, NULL, 30
- (xiii) 7902, FORD, ANALYST, 7566, 03 – DEC – 81, 3000, NULL, 20
- (xiv) 7934, MILLER, CLERK, 7782, 23 – JAN – 82, 1300, NULL, 10
- (xv) 7943, JOHN, CLERK, 7943, 10 – DEC – 83, 2000, NULL, 50

**DEPT:**

- (i) 10, ACCOUNTING, NEW YORK
- (ii) 20, RESEARCH, DALLAS
- (iii)30, SALES, CHICAGO
- (iv)40, OPERATIONS, BOSTON
- (v) 50, COMPUTER, AMERICA

**20. Insert only the records of employee number, name, salary into EMP table**

SOL: INSERT INTO EMP (EMPNO, ENAME, SAL) VALUES ('7955', 'PAUL', 1200);

**21. insert two rows into EMP table using parameter substitution**

SOL: INSERT INTO EMP VALUES (&EMPNO, '&ENAME', '&JOB', &MGR, '&HIREDATE', &SAL, &COMM, &DEPTNO);

**22. insert the current transaction date temporary table**

SOL: INSERT INTO TEMP VALUES (SYSDATE);



**Problems on Select Command:**

**23. List the Information of all Employees**

SOL: SELECT \* FROM EMP;

**24. List the information of all the departments**

SOL: SELECT \* FROM DEPT;

**25. LIST THE DEPARTMENT NUMBERS, EMPLOYEE NUMBERS AND THEIR MANAGERS NUMBERS**

SOL: SELECT DEPT. DEPTNO, EMP. EMPNO, EMP. MGR FROM EMP, DEPT;

**26. List department name and locations from DEPT table**

SOL: SELECT DNAME, LOC, FROM DEPT;

**27. List the information of employees and their departments in a single DMI command**

SOL: SELECT \*. FROM EMP;

**28. Copy all the records of their columns EMPNO, ENAME, JOB from EMP table and insert the records into a temp table with column names same as EMPNO, ENAME, JOB**

SOL: INSERT INTO TEMP (EMPNO, ENAME, JOB) SELECT EMPNO, NAME, JOB

**29. List the details of both the tables**

SOL: SELECT \* FROM EMP, DEPT;

**30. List the information of all the employees present in the user named SCOTT**

SOL: SELECT \* FROM SCOTT. EMP;

**31. List the information of the departments from your BATCHMATES DEPT table**

SOL: SELECT \* FROM ORA252P. DEPT;

**32. List out the table names in your schema**

SOL: SELECT \* FROM TAB;

**33. List all the system tables**

SOL: SELECT \*FROM SYS. DICTIONARY;

**34. Get the information of the maximum available blocks allotted to a particular user for creating tables from the system tables**

SOL: SELECT \* FROM USER\_TS\_QUOTAS;

**35. List out all the privileges given to a particular user**

**36. List out all the tables which start with 'S'**

SOL: SELECT TABLE\_NAME FROM USER\_TABLES WHERE TABLE\_NAME LIKE 'S%';

**37. Copy the structure of dept table alone to a temporary table called TEMP1**

SOL: CREAT TABLE TEMP1, AS SELECT \* FROM DEPT WHERE 1=2;

**Problems on update command:**

**38. Update the salary by 10% hike to analysts working in department number 20 and 30**

SOL: UPDATE EMP SET SAL = SAL + 0. 1 WHERE DEPTNO IN (10,20) AND JOB = 'ANALYST';

**39. Give 5% raise in salary to all the salesman**

SOL: UPDATE EMP SET SAL=SAL+.5 WHERE JOB = 'SALESMAN';

**40. PROMOTE ALL THE EMPLOYEES DISCRIBED AS SALESMAN TO SALES OFFICER IF THEIR GROSS SALARY PER MOUNTH IS GREATER THEN 3000**

SOL: UPDATE EMP SET JOB = 'SALESOFFICER' WHERE JOB = 'SALESMAN' AND SAL>3000;

**41. Give all the employees of commission of rs500**

SOL: UPDATE EMP SET COMM = 500;

**42. Change the department of JAMES to 20**

SOL: UPDATE EMP SET DEPTNO = 20 WHERE ENAME = 'JAMES';

**43. Calculate all the employees total salary with commission**

SOL: SELECT SALES SAL +NVL (COMM) "TOTAL" FROM EMP;

**Problems on delete command:**

**44. Delete all the records of employees**

SOL: DELETE FROM EMP;

**45. Get back the original records of employees back**

SOL: ROLLBACK;

**46. Allen's record only**

SOL: DELETE FROM EMP WHERE ENAME = 'ALLEN';

**47. Delete records of ename column only and verify it**

SOL: NOT POSSIBLE

**48. Delete the records of employee number 7782**

SOL: DELETE FROM EMP WHERE EMPNO=7782;

**49. Delete the employee's records who doesn't have commission**

SOL: DELETE FROM EMP WHERE COMM IS NULL;

**50. Get back the original records back**

SOL: ROLLBACK;

**51. Delete the duplicate records of the employee table**

SOL: DELETE FROM EMP A WHERE ROWID<> (SELECT MIN (ROWID) FROM EMP B WHERE A. EMPNO=B. EMPNO);

**52. Delete the first five records of employee table**

SOL: DELETE FROM EMP X WHERE 5 > (SELECT COUNT (ROWID) FROM EMP Y WHERE Y. ROWID < X.ROWID);

**53. Delete the rows of the temp table permanently**

SOL: TRUNCATE TABLE TEMP;

**Problems on transactional commands:**

**54. Update a record of EMP table and save the changes permanently in the database**

SOL: UPDATE EMP SET SAL=SAL+100 WHERE EMPNO=100; COMMIT;

**55. Sql \* plus has the facility to automatically save all the records without issuing the TCL command which is that?**

SOL: SET AUTOCOMMIT ON

**56. Give all the privileges you have of a database object to another**

SOL: GRANT ALL ON EMP TO ORA253A;

**57. Give only select, insert privileges to another user**

SOL: GRANT SELECT, INSERT ON EMP TO ORA267A;

**58. List the user's id and which database object you have granted**

SOL: SELECT \* FROM USE\_TAB\_PRIVS;

---

## **ADDITIONAL EXERCISES ON DATABASE APPLICATION LABORATORY**

**Exercise 1:** To understand some simple Database Applications and build Conceptual Data Model.

- a) Select an enterprise that you are familiar with (for example, a school, a college, a company, a small business, a club or association). List all the information that this enterprise uses.
- b) Describe the steps involved in the database design process using E-R Modeling: *Requirements Analysis, Identify Entity Sets, Identify Relationship Sets, Value Sets and Attributes, Specifying Primary keys, Building E-R diagram, Implementation*
- c) For the following mini-world example database applications, Design and Develop Conceptual Data Model (E-R Diagram) with all the necessary entities, attributes, constraints and relationships.
  - i. **Medical Clinic Database** – The clinic has a number of regular patients and new patients come to the clinic regularly. Patients make appointments to see one of the doctors; several doctors attend the clinic and they each have their own hours. Some doctors are General Practitioners (GPs) while others are specialists(cardiologists, dermatologists etc.,). Patients have families and the family relationships are important. A medical record of each patient needs to be maintained. Information on prescriptions, insurance, allergies, etc needs to be maintained. Different doctors may charge different fees. Billing has to be done for Patients.
  - ii. **University Database** - The Visvesvaraya Technological University (VTU) is a large Institution with several campuses scattered across Karnataka. Academically, the university is divided into a number of faculties, such as Faculty of Engineering, Faculty of Architecture, Faculty of Management and Faculty of Science. Some of the Faculties operate on a number of campuses. Faculties, in turn, are divided into schools; for example, the School of Architecture, the School of Information Technology. Each school is headed by a Director and has a number of teaching and non-teaching staff. Each school offers many courses. Each course consists of a fixed core of subjects and a number of electives from other courses. Each student in the University is enrolled in a single course of study. A subject is taught to the students

who have registered for that subject by a teacher. A student is awarded a grade in each subject taken.

- iii. **Construction Company Database** - A construction company has many branches spread all over the country. The company has two types of constructions to offer: Housing and Commercial. The housing company provides low-income housing, medium-style housing, and high-end housing schemes, while commercial side, it offers multiplexes and shopping zones. The customers of the company may be individuals or corporate clients. Company stores the information about employees' works for it.
- iv. **Time Table Preparation** - An Engineering College has a number of Branches. Each Branch has number sections, a number of courses and a number of faculty members teaching the courses. Each branch has a number of class rooms and laboratories. Each course may be scheduled in a class room at a particular time.

**Note:** Similar applications may be explored and given as assignments to students in a group.

**Exercise 2:** Design and build Relational Data Model for each of the application scenarios of exercise 1 specifying all possible constraints. Extend the same for a database application of students' choice.

**Exercise 3** To understand and demonstrate DDL, DML and DCL Commands of SQL

- a. Create a table called EMP with the following structure and describe it.

Name	Type
EMPNO	NUMBER (6)
ENAME	VARCHAR2 (20)
DOB	DATE
JOB	VARCHAR2 (10)
DEPTNO	NUMBER (2)
SAL	NUMBER (7,2)

Allow NULL for all columns except ENAME and JOB. EMPNO is the Primary Key

- b. Add a column EXPERIENCE of type NUMERIC to the EMP table. Allow NULL to it.
- c. Modify the column width of the JOB field of EMP table.
- d. Create DEPT table with the following structure and describe it

Name	Type
DEPTNO	NUMBER (2)
DNAME	VARCHAR2 (15)
LOCN	VARCHAR2 (10)

DEPTNO is the Primary Key and DNAME cannot be NULL

- e. Add constraint to check the SAL value of EMP Table. SAL must be > 6000.
  - f. Drop a column EXPERIENCE from the EMP table.
  - g. Insert a single record into DEPT table. Repeat this for inserting at least 3 records
  - h. Insert more than a record into EMP table using a single insert command. Insert at least 10 records
  - i. Update the EMP table to set the salary of all employees to Rs. 30000/- for a given JOB type
  - j. Create a pseudo table EMPLOYEE with the same structure as the table EMP using SELECT clause.
  - k. Delete employees from EMP table for a given JOB type. Delete the first five records of EMP table
  - l. Grant all/some privileges of EMP table to DEPT table
  - m. Revoke some/all privileges of EMP table from DEPT table
  - n. Truncate the EMP table and drop the DEPT table
  - o. Demonstrate the use of COMMIT, SAVEPOINT and ROLLBACK commands
-