KVG COLLEGE OF ENGINEERING

V SEMESTER CS & E

DBMS LABORATORY WITH MINI PROJECT

Subject Code: 18CSL58

(Choice Based Credit System (CBCS) & Outcome Based Education (OBE))

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KVG COLLEGE OF ENGINEERING

Kurunjibag-574327, Sullia D.K

VISION

To be a notable engineering college recognized for Academic, innovation and the societal relevance and impact of its pursuits.

MISSION

M1: Educate our students committed to the service and ethical application of science and technology.

M2: Provide resource to our Faculty and Student to enhance Engineering Knowledge through Industry-Institute Interactions.

M3: Practice Diversity and Inclusion amongst Our stakeholders through rural and social Outreach.

Vision

To produce qualified Computer Science and Engineering graduates with human values and ethical Skills.

Mission

M1: Impart students with strong fundamental concepts, analytical capability, programming and problem-solving skills.

M2: Encourage an ambience of education through faculty training, self-learning, sound academic practices and Industry related endeavors.

M3: Imbibe environment Conciseness, social awareness and responsibility in students to serve the society.

PEO's	Program Educational Objectives Statements
PEO1	**Apply Engineering Basics:** Analyze Engineering Challenges through application of mathematical and Algorithmic principles for real life technology projects.
PEO2	**Engineering Skills and techniques:** Apply skills like Analyzing, Designing, Implementing and Testing of Major and Minor Projects.
PEO3	**Individual and Team Work: Exhibit** collaborative abilities in the engineering projects like Communication Skill, work as individual or in a team with a sense of Social Responsibility.
PEO3	**Life Long Learning**: Initiate technological and skills required for comprehensive contribution as experts in the Chosen Profession.

PSO's	Program Specific Outcome Statements
PSO1	**Problem Solving Skills:** Specify, design, build and test analog, digital and embedded systems for signal processing
PSO2	**Professional Skills**: Understand and architect wired and wireless analog and digital communication systems as per specifications, and determine their performance
PSO3	**Ethics And Career Development:** Exhibit skills required for a successful career in the industry based on principles of software project management, teamwork, ethical practices, develop the spirit of free enterprise and provide innovative ideas towards analysis.

Program Outcomes

PO1: Engineering Knowledge: To apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, review research literature, and analyse complex Engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/Development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate considerations for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct Investigations of Complex Problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The Engineer and Society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and Sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and Team Work: Function effectively as an individual, and as a member of leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project Management and Finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long Learning: Recognize the need for, and have the preparation and ability to engage in independent and life- long learning in the broadest context of technological change.

DBMS LABORATORY WITH MINI PROJECT
(Effective from the academic year 2018 -2019)
SEMESTER – V

Course Code	**18CSL58**	**CIE Marks**	40
Number of Contact Hours/Week	0:2:2	**SEE Marks**	60
Total Number of Lab Contact Hours	36	**Exam Hours**	03

Credits – 2

Course Learning Objectives: This course (18CSL58) will enable students to:

- Foundation knowledge in database concepts, technology and practice to groom students into well-informed database application developers.
- Strong practice in SQL programming through a variety of database problems.
- Develop database applications using front-end tools and back-end DBMS.

Descriptions (if any):

PART-A: SQL Programming (Max. Exam Mks. 50)

- Design, develop, and implement the specified queries for the following problems using Oracle, MySQL, MS SQL Server, or any other DBMS under LINUX/Windows environment.
- Create Schema and insert at least 5 records for each table. Add appropriate database constraints.

PART-B: Mini Project (Max. Exam Mks. 30)

- Use Java, C#, PHP, Python, or any other similar front-end tool. All applications must be demonstrated on desktop/laptop as a stand-alone or web based application (Mobile apps on Android/IOS are not permitted.)

Installation procedure of the required software must be demonstrated, carried out in groups and documented in the journal.

Programs List:

PART A

1.	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, Programme_id, No-of_Copies) BOOK_LENDING(Book_id, Programme_id, Card_No, Date_Out, Due_Date) LIBRARY_PROGRAMME(Programme_id, Programme_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 Programme, 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.
2.	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 salesman who had more than one customer. 3. List all the salesman 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.
3.	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.
4.	Consider the schema for College Database: STUDENT(USN, SName, Address, Phone, Gender) SEMSEC(SSID, Sem, Sec) CLASS(USN, SSID) COURSE(Subcode, Title, Sem, Credits) IAMARKS(USN, Subcode, SSID, Test1, Test2, Test3, FinalIA) 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 Courses. 4. Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students. 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 8th semester A, B, and C section students.
5.	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 controlledby department number 5 (use NOT EXISTS operator). 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.
PART B: Mini Project	
•	For any problem selected
•	Make sure that the application should have five or more tables
•	Indicative areas include; health care

Laboratory Outcomes: The student should be able to:

- Create, Update and query on the database.
- Demonstrate the working of different concepts of DBMS
- Implement, analyze and evaluate the project developed for an application.

Conduct of Practical Examination:

- Experiment distribution
  - For laboratories having only one part: Students are allowed to pick one experiment from the lot with equal opportunity.
  - For laboratories having PART A and PART B: Students are allowed to pick one experiment from PART A and one experiment from PART B, with equal opportunity.
- Change of experiment is allowed only once and marks allotted for procedure to be made zero of the changed part only.
- Marks Distribution *(Courseed to change in accoradance with university regulations)*
  - k) For laboratories having only one part – Procedure + Execution + Viva-Voce: 15+70+15 = 100 Marks
  - l) For laboratories having PART A and PART B
    - i. Part A – Procedure + Execution + Viva = 6 + 28 + 6 = 40 Marks
    - ii. Part B – Procedure + Execution + Viva = 9 + 42 + 9 = 60 Marks

INDEX

Subject*: DBMS Laboratory with Mini Project* **Subject Code: *18CSL58***

BASIC CONCEPTS OF SQL

INTRODUCTION TO SQL

SQL stands for "Structured Query Language" and can be pronounced as "SQL" or "sequel – (Structured English Query Language)". It is a query language used for accessing and modifying information in the database. IBM first developed SQL in 1970s. Also, it is an ANSI/ISO standard. It has become a Standard Universal Language used by most of the relational database management systems (RDBMS). Some of the RDBMS systems are: Oracle, Microsoft SQL server, Sybase etc. Most of these have provided their own implementation thus enhancing its feature and making it a powerful tool. Supports all the three sublanguages of DBMS **DDL, DML, DCL.** Few of the SQL commands used in SQL programming are SELECT Statement, UPDATE Statement, INSERT INTO Statement, DELETE Statement, WHERE Clause, ORDER BY Clause, GROUP BY Clause, ORDER Clause, Joins, Views, GROUP Functions, Indexes etc.

SQL COMMANDS

SQL commands are instructions used to communicate with the database to perform specific task that work with data. SQL commands can be used not only for searching the database but also to perform various other functions like, for example, you can create tables, add data to tables, or modify data, drop the table, set permissions for users.

- **Data Definition Language (DDL)** - These SQL commands are used for creating, modifying, and dropping the structure of database objects. The commands are CREATE, ALTER, DROP, RENAME, and TRUNCATE.
- **Data Manipulation Language (DML)** - These SQL commands are used for storing, retrieving, modifying and deleting data. These commands are SELECT, INSERT, UPDATE, and DELETE.
- **Transaction Control Language (TCL)** - These SQL commands are used for managing changes affecting the data. These commands are COMMIT, ROLLBACK, and SAVEPOINT.
- **Data Control Language (DCL)** - These SQL commands are used for providing security to database objects. These commands are GRANT and REVOKE Data Definition Language (DDL).

CREATE TABLE STATEMENT

The CREATE TABLE Statement is used to create tables to store data. Integrity Constraints like primary key, unique key and foreign key can be defined for the columns while creating the table. The integrity constraints can be defined at column level or table level. The implementation and the syntax of the CREATE Statements differs for different RDBMS.

CREATE TABLE <*table_name*>

(<*column name*> <column type> [<*attribute constraint*>]

{, <*column name*> <column type> [<*attribute constraint*>]}

[<*table constraint*> {, <*table constraint*>}])

[<*table constraint*> {, <*table constraint*>}])

- *table_name* - is the name of the table.
- *column name*- is the name of the columns.
- *column type* - is the datatype for the column like *char, varchar date, number* etc.
- *attribute constraint* -is the constraint specified on the column like *NOT NULL, PRIMARY KEY, UNIQUE*
- *table constraint*- is specified through additional clauses at the end of a CREATE TABLE statement using *CHECK*

DATA TYPES

Numeric	NUMBER, NUMBER (S, P), INTEGER, INT, FLOAT, DECIMAL	Number value with a max number of column digits specified in parenthesis
Character	CHAR(N), VARCHAR(N)	Fixed-length character string, Variable-length character string, n is size
Boolean	TRUE, FALSE, AND NULL	
Date and Time	DATE (DD-MMM-YY) TIME (HH:MM: SS)	Eg: “12-jan-18”
Timestamp	Timestamp: DATE + TIME	30-MAR-20 10.10.34.569000 AM +05:30

SQL INTEGRITY CONSTRAINTS

Integrity Constraints are used to apply business rules for the database tables.The constraints available in SQL are **Foreign Key, Primary key, Not Null, Unique, Check.**
Constraints can be defined in two ways:
1. The constraints can be specified immediately after the column definition. This is called column-level definition.
2. The constraints can be specified after all the columns are defined. This is called table-level definition.

1. **Primary key**

   This constraint defines a column or combination of columns which uniquely identifies each row in the table.

   Syntax to define a Primary key at column level:

   $$Column\_name\ datatype\ [CONSTRAINT\ constraint\_name]\ PRIMARY\ KEY$$

   Syntax to define a Primary key at table level:

   $$[CONSTRAINT\ constraint\_name]\ PRIMARY\ KEY(column_{name1}, Column_{name2}, .)$$

2. **Foreign key or Referential Integrity**

   This constraint identifies any column referencing the PRIMARY KEY in another table. It establishes a relationship between two columns in the same table or between different tables. For a column to be defined as a Foreign Key, it should be defined as a Primary Key in the table which it is referring to. One or more columns can be defined as foreign keys. We can specify **RESTRICT, CASCADE, SET NULL** or **SET DEFAULT** on referential integrity constraints (foreign keys).

   Syntax to define a foreign key at column level:

   $$[\boldsymbol{CONSTRAINT\ constraintname}]\boldsymbol{REFERENCES\ referenced\ table\ name}(\boldsymbol{column\ name})$$

3. **Not Null Constraint**

   This constraint ensures all rows in the table contain a definite value for the column which is specified as not null. Which means a null value is not allowed.

   Syntax to define a Not Null constraint:

   $$[CONSTRAINT\ constraint\ name]\ NOT\ NULL$$

4. **Unique Key**

This constraint ensures that a column or a group of columns in each row have a distinct value. A column(s) can have a null value but the values cannot be duplicated.

Syntax to define a Unique key at column level:

$$[CONSTRAINT\ constraint\_name]\ UNIQUE$$

Syntax to define a Unique key at table level:

$$[CONSTRAINT\ constraint\_name]\ UNIQUE(column\_name)$$

5. **Check Constraint**

This constraint defines a business rule on a column. All the rows must satisfy this rule. The constraint can be applied for a single column or a group of columns.

Syntax to define a Check constraint:

$$[CONSTRAINT\ constraint\_name]\ CHECK\ (condition)$$

ALTER TABLE STATEMENT

The SQL ALTER TABLE command is used to modify the definition structure) of a table by modifying the definition of its columns. The ALTER command is used to perform the following functions.

- Add, drop, modify table columns
- Add and drop constraints
- Enable and Disable constraints

1. **Add a column to TABLE**

Used to **add an attribute** to/from one of the base relations. 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.

Syntax to Add a column to existing table:

$$ALTER\ TABLE\ table\_name\ ADD\ column\_name\ data\_type\ column\_constraint;$$

2. **Drop a column (an attribute) from TABLE**

Used to drop a column. All constraints and views that reference the column are dropped automatically, along with the column.

$$ALTER\ TABLE\ table\_name\ DROP\ COLUMN\ column\_name;$$

3. **Modify a column (an attribute) in TABLE**

Used to change the data type of a column in a table

$$ALTER\ TABLE\ table\ name\ \ MODIFY\ column\ name\ datatype;$$

4. **Add a constraint to TABLE**

Used to adds a table-level constraint to an existing table.

$$ALTER\ TABLE\ table\ name\ ADD\ CONSTRAINT\ constraint\ name\ CONSTRAINT\ TYPE\ (COLUMN\ NAME);$$

5. **Drop a constraint from a TABLE**

Used to drop primary key constraint, unique key constraint, foreign key constraint, check constraint and not null constraint using the same command

$$ALTER\ TABLE\ <table\ name>\ drop\ constraint\ <constraint\ name>;$$

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

$$DROP\ TABLE\ [\ schema.\ ]\ table\_name\ [\ CASCADE\ CONSTRAINTS];$$

THE SELECT-FROM-WHERE Structure of Basic SQL Queries

SQL has one basic statement for retrieving information from a database; the SELECT statement. The basic form of the SELECT statement, sometimes called a **mapping** or a **select-from-where block**, is formed of the three clauses SELECT, FROM, and WHERE and has the following form

$$SELECT\ <attribute\ list>$$
$$FROM\ <table\ list>$$
$$WHERE\ <condition>;$$

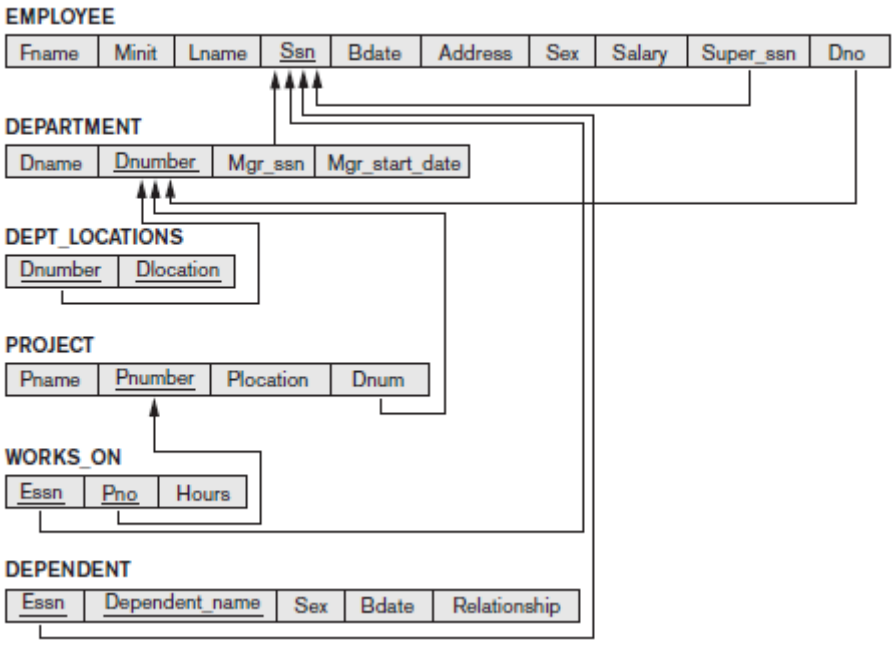
<attribute list> is a list of attribute names whose values are to be retrieved by the query.

<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.

All subsequent examples use *COMPANY* database as shown below:

Referential integrity constraints displayed on the COMPANY relational database schema.



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	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	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	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	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	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	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	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	1958-05-03	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 one relation

Q0: 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';
```

Example of a simple query on two relations

Q1: 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;
```

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 in WHERE is TRUE

Example:

Q4: 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:

Q5: Retrieve all the attribute values of EMPLOYEES who work in department 5.

```
Q5:  SELECT *
     FROM EMPLOYEE
     WHERE DNO=5;
```

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.

```
Q7:  SELECT SALARY
     FROM EMPLOYEE;

Q8:  SELECT DISTINCT SALARY
     FROM EMPLOYEE;
```

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.

Q9: Retrieve all employees whose address is in Houston, Texas. Here, the value of the ADDRESS attribute must contain the substring 'Houston,TX' in it.

```
Q9:  SELECT FNAME, LNAME
     FROM EMPLOYEE
     WHERE ADDRESS LIKE '%Houston,TX%' ;
```

Q10: 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.

```
Q10: SELECT FNAME, LNAME
     FROM EMPLOYEE
     WHERE BDATE LIKE '_____5_';
```

ORDER BY

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

Q11: 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.

```
Q11: 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.

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.

Q12: Retrieve the name and address of all employees who work for the 'Research' department.

```
Q12:  SELECT FNAME, LNAME, ADDRESS
      FROM EMPLOYEE
      WHERE DNO IN (  SELECT DNUMBER
                      FROM DEPARTMENT
                      WHERE DNAME='Research');
```

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

```
Q11:  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);
```

THE EXISTS FUNCTION

The EXISTS function in SQL is used to check whether the result of a nested query is *empty* (contains no tuples) or not. The result of EXISTS is a Boolean value **TRUE** if the nested query result contains at least one tuple, or **FALSE** if the nested query result contains no tuples.

```
Q11a: SELECT FNAME, LNAME
      FROM EMPLOYEE
```

```
WHERE EXISTS (SELECT *
              FROM DEPENDENT
              WHERE SSN=ESSN AND
        FNAME=DEPENDENT_NAME);
```

Query 12: Retrieve the names of employees who have no dependents.

```
Q12: SELECT FNAME, LNAME
     FROM EMPLOYEE
     WHERE NOT EXISTS ( SELECT *
                        FROM DEPENDENT
                        WHERE SSN=ESSN);
```

EXPLICIT SETS

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

Q 13: Retrieve the social security numbers of all employees who work on project number 1, 2, or 3.

```
Q13: 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.

Q14: Retrieve the names of all employees who do not have supervisors.

```
Q14: 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 15: Find the maximum salary, the minimum salary, and the average salary among employees who work for the 'Research' department.

```
Q15:  SELECT MAX (SALARY), MIN(SALARY), AVG(SALARY)
      FROM EMPLOYEE, DEPARTMENT
      WHERE DNO=DNUMBER AND DNAME='Research' ;
```

Q16. Retrieve the total number of employees in the company.

```
Q16:  SELECT COUNT (*)
      FROM EMPLOYEE
```

Q17.Retrieve the total number of number of employees in the 'Research' department.

```
Q17:  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**.

Q 18: For each department, retrieve the department number, the number of employees in the department, and their average salary.

```
Q18:  SELECT DNO, COUNT (*), AVG (SALARY)
      FROM EMPLOYEE
      GROUP BY DNO;
```

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)

Q 19: 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.

```
Q19:  SELECT PNUMBER, PNAME, COUNT (*)
      FROM PROJECT, WORKS_ON
      WHERE PNUMBER=PNO
      GROUP BY PNUMBER, PNAME
      HAVING COUNT (*) > 2;
```

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.

Q 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';
```

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 value 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');

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');
```

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.

1.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, Programme_id, No_of_Copies*)

BOOK_LENDING (*Book_id, Programme_id, Card_No, Date_Out, Due_Date*)

LIBRARY_PROGRAMME (*Programme_id, Programme_Name, Address*)

Write SQL queries to

1. Retrieve details of all books in the library _id, title, name of publisher, authors, number ofcopies in each Programme, etc.

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

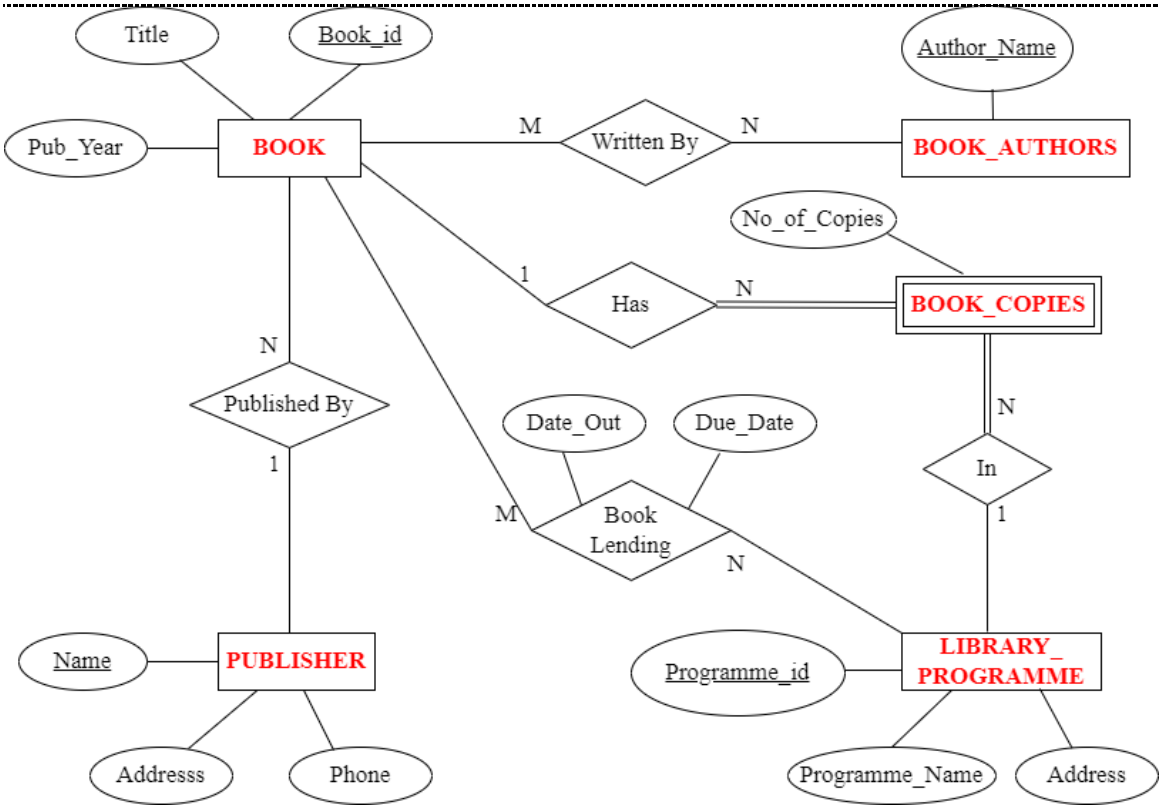
3. Delete a book in BOOK table. Update the contents of other tables to reflect this datamanipulation operation.

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:

ER DIAGRAM



SCHEMA DIAGRAM

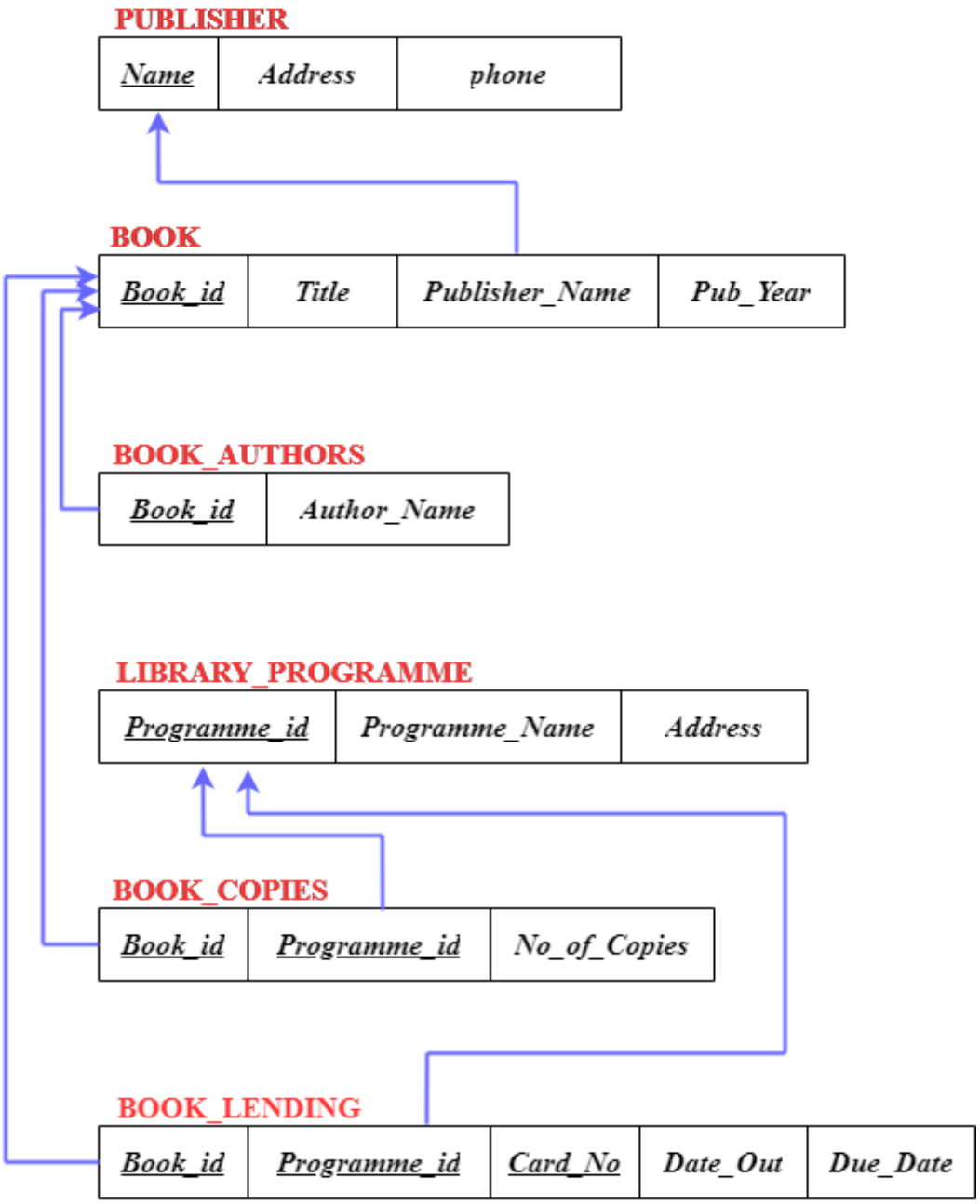


TABLE CREATION

CREATE TABLE BOOK (

BOOK_ID INTEGER PRIMARY KEY,

TITLE VARCHAR2 (20),

PUB_YEAR NUMBER (4),

PUBLISHER_NAME REFERENCES PUBLISHER (NAME) ON DELETE CASCADE);

CREATE TABLE BOOK_AUTHORS

```
(BOOK_ID INTEGER PRIMARY KEY,
AUTHOR_NAME VARCHAR (20),
FOREIGN KEY(BOOK_ID) REFERENCES BOOK (BOOK_ID) ON DELETE
CASCADE);

CREATE TABLE LIBRARY_PROGRAMME
(PROGRAMME_ID VARCHAR (10) PRIMARY KEY,
PROGRAMME_NAME VARCHAR (10),
ADDRESS VARCHAR (50));

CREATE TABLE BOOK_COPIES (
BOOK_ID INTEGER,
PROGRAMME_ID VARCHAR (10),
NO_OF_COPIES INTEGER,
FOREIGN KEY (BOOK_ID) REFERENCES BOOK (BOOK_ID) ON DELETE
CASCADE,
FOREIGN KEY (PROGRAMME_ID) REFERENCES
LIBRARY_PROGRAMME(PROGRAMME_ID) ON DELETE CASCADE,
PRIMARY KEY (BOOK_ID, PROGRAMME_ID));

CREATE TABLE BOOK_LENDING (
BOOK_ID INTEGER,
PROGRAMME_ID VARCHAR (10),
CARD_NO INTEGER,
DATE_OUT DATE,
DUE_DATE DATE,
FOREIGN KEY(BOOK_ID) REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE,
FOREIGN KEY(PROGRAMME_ID) REFERENCES
LIBRARY_PROGRAMME(PROGRAMME_ID) ON DELETE CASCADE,
PRIMARY KEY (BOOK_ID, PROGRAMME_ID, CARD_NO));
```

TABLE DESCRIPTIONS

DESC PUBLISHER;

```
SQL> DESC PUBLISHER;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 NAME                                      NOT NULL VARCHAR2(20)
 PHONE                                              NUMBER(10)
 ADDRESS                                            VARCHAR2(20)
```

DESC BOOK;

```
SQL> DESC BOOK;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BOOK_ID                                   NOT NULL NUMBER(38)
 TITLE                                              VARCHAR2(20)
 PUB_YEAR                                           NUMBER(4)
 PUBLISHER_NAME                                     VARCHAR2(20)
```

DESC BOOK_AUTHORS;

```
SQL> DESC BOOK_AUTHORS;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BOOK_ID                                   NOT NULL NUMBER(38)
 AUTHOR_NAME                                        VARCHAR2(20)
```

DESC LIBRARY_PROGRAMME;

```
SQL> DESC LIBRARY_PROGRAMME;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PROGRAMME_ID                              NOT NULL VARCHAR2(10)
 PROGRAMME_NAME                                     VARCHAR2(10)
 ADDRESS                                            VARCHAR2(30)
```

DESC BOOK_COPIES;

```
SQL> DESC BOOK_COPIES;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BOOK_ID                                   NOT NULL NUMBER(38)
 PROGRAMME_ID                              NOT NULL VARCHAR2(10)
 NO_OF_COPIES                                       NUMBER(38)
```

DESC BOOK_LENDING;

```
SQL> DESC BOOK_LENDING;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BOOK_ID                                   NOT NULL NUMBER(38)
 PROGRAMME_ID                              NOT NULL VARCHAR2(10)
 CARD_NO                                   NOT NULL NUMBER(38)
 DATE_OUT                                           DATE
 DUE_DATE                                           DATE
```

INSERTION OF VALUES TO TABLES

```
INSERT INTO PUBLISHER VALUES ('MCGRAWHILL',9989076587,'BANGALORE');
INSERT INTO PUBLISHER VALUES ('PEARSON', 9889076565,'NEWDELHI');
INSERT INTO PUBLISHER VALUES ('GREEN TEA PRESS',8970862340,'DELHI');
INSERT INTO PUBLISHER VALUES ('WILEY',9970862241,'PUNE');

INSERT INTO BOOK VALUES (1,'DBMS',2018,'PEARSON');
INSERT INTO BOOK VALUES (2,'ADBMS',2018,'PEARSON');
INSERT INTO BOOK VALUES (3, 'NETWORK',2016,'PEARSON');
INSERT INTO BOOK VALUES (4,'IMAGE PROCESSING',2019,'WILEY');
INSERT INTO BOOK VALUES (5,'PRINCIPLES OF MGT',2010,'MCGRAW HILL');
INSERT INTO BOOK VALUES (6,'ENVIRONMENTALSTUDIES',2017,'MCGRAW HILL');

INSERT INTO BOOK_AUTHORS VALUES (1,'NAVATHE');
INSERT INTO BOOK_AUTHORS VALUES (2,'NAVATHE');
INSERT INTO BOOK_AUTHORS VALUES (3,'TANENBAUM');
INSERT INTO BOOK_AUTHORS VALUES (4,'GONZALEZ');
INSERT INTO BOOK_AUTHORS VALUES (5,'TRIPATHY');
INSERT INTO BOOK_AUTHORS VALUES (6,'BENNY JOSEPH');

INSERT INTO LIBRARY_PROGRAMME VALUES ('P1','CS','SULLIA');
INSERT INTO LIBRARY_PROGRAMME VALUES ('P2','EC','SULLIA');
INSERT INTO LIBRARY_PROGRAMME VALUES ('P3','MECH','SULLIA');
INSERT INTO LIBRARY_PROGRAMME VALUES ('P4','CIVIL','SULLIA');

INSERT INTO BOOK_COPIES VALUES (1,'P1',20);
INSERT INTO BOOK_COPIES VALUES (2,'P1',5);
INSERT INTO BOOK_COPIES VALUES (3,'P1',15);
INSERT INTO BOOK_COPIES VALUES (3,'P2',10);
INSERT INTO BOOK_COPIES VALUES (4,'P2',18);
INSERT INTO BOOK_COPIES VALUES (4,'P1',5);
INSERT INTO BOOK_COPIES VALUES (5,'P3',15);
INSERT INTO BOOK_COPIES VALUES (6,'P4',20);
INSERT INTO BOOK_COPIES VALUES (6,'P1',2);
```

INSERT INTO BOOK_LENDING VALUES (1,'P1',101,'01-JAN-17','01-JUN-17');

INSERT INTO BOOK_LENDING VALUES (3,'P2',101,'11-JAN-17','11-MAR-17');

INSERT INTO BOOK_LENDING VALUES (4,'P2',101,'21-FEB-17','11-APR-17');

INSERT INTO BOOK_LENDING VALUES (6,'P4', 101,'12-APR-17','12-MAY-17');

INSERT INTO BOOK_LENDING VALUES (5,'P3',103,'15-MAR-17','15-MAY-17');

INSERT INTO BOOK_LENDING VALUES (2,'P1',102,'19-MAR-18','12-MAY-18');

RETRIEVAL OF INSERTED VALUES

SELECT * FROM PUBLISHER;

```
SQL> SELECT * FROM PUBLISHER;

NAME                      PHONE ADDRESS
-------------------- ---------- --------------------
MCGRAW HILL           9989076587 BANGALORE
PEARSON               9889076565 NEWDELHI
GREEN TEA PRESS       8970862340 DELHI
WILEY                 9970862241 PUNE
```

SELECT * FROM BOOK;

```
SQL> SELECT * FROM BOOK;

   BOOK_ID TITLE                    PUB_YEAR PUBLISHER_NAME
---------- -------------------- ---------- --------------------
         1 DBMS                       2018 PEARSON
         2 ADBMS                      2018 PEARSON
         3 NETWORK                    2016 PEARSON
         4 IMAGE PROCESSING           2019 WILEY
         5 PRINCIPLES OF MGT          2010 MCGRAW HILL
         6 ENVIRONMENTALSTUDIES       2017 MCGRAW HILL

6 rows selected.
```

SELECT * FROM BOOK_AUTHORS;

```
SQL> SELECT * FROM BOOK_AUTHORS;

   BOOK_ID AUTHOR_NAME
---------- --------------------
         1 NAVATHE
         2 NAVATHE
         3 TANENBAUM
         4 GONZALEZ
         5 TRIPATHY
         6 BENNY JOSEPH

6 rows selected.
```

SELECT * FROM LIBRARY_PROGRAMME;

```
SQL> SELECT * FROM LIBRARY_PROGRAMME;

PROGRAMME_ PROGRAMME_ ADDRESS
---------- ---------- ------------------------------
P1         CS         SULLIA
P2         EC         SULLIA
P3         MECH       SULLIA
P4         CIVIL      SULLIA
```

SELECT * FROM BOOK_COPIES;

```
SQL> SELECT * FROM BOOK_COPIES;

   BOOK_ID PROGRAMME_ NO_OF_COPIES
---------- ---------- ------------
         1 P1                   20
         2 P1                    5
         3 P1                   15
         3 P2                   10
         4 P2                   18
         4 P1                    5
         5 P3                   15
         6 P4                   20
         6 P1                    2

9 rows selected.
```

SELECT * FROM BOOK_LENDING;

```
SQL> SELECT * FROM BOOK_LENDING;

   BOOK_ID PROGRAMME_    CARD_NO DATE_OUT  DUE_DATE
---------- ---------- ---------- --------- ---------
         1 P1                101 01-JAN-17 01-JUN-17
         3 P2                101 11-JAN-17 11-MAR-17
         4 P2                101 21-FEB-17 11-APR-17
         6 P4                101 12-APR-17 12-MAY-17
         5 P3                103 15-MAR-17 15-MAY-17
         2 P1                102 19-MAR-18 12-MAY-18

6 rows selected.
```

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. PUBLISHER_NAME, A. AUTHOR_NAME, P.
PROGRAMME_ID, C.NO_OF_COPIES
FROM BOOK B, BOOK_AUTHORS A, BOOK_COPIES C,
LIBRARY_PROGRAMME P
WHERE B. BOOK_ID=A.BOOK_ID
AND B. BOOK_ID=C.BOOK_ID
AND P. PROGRAMME_ID =C. PROGRAMME_ID;
```

```
   BOOK_ID PUBLISHER_NAME       AUTHOR_NAME          PROGRAMME_ NO_OF_COPIES
---------- -------------------- -------------------- ---------- ------------
         1 PEARSON              NAVATHE              P1                   20
         2 PEARSON              NAVATHE              P1                    5
         3 PEARSON              TANENBAUM            P1                   15
         3 PEARSON              TANENBAUM            P2                   10
         4 WILEY                GONZALEZ             P2                   18
         4 WILEY                GONZALEZ             P1                    5
         5 MCGRAW HILL          TRIPATHY             P3                   15
         6 MCGRAW HILL          BENNY JOSEPH         P4                   20
         6 MCGRAW HILL          BENNY JOSEPH         P1                    2

9 rows selected.
```

2. **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
```

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

```
DELETE FROM BOOK
WHERE BOOK_ID=6;
```

```
1 row deleted.

Commit complete.
SQL> SELECT * FROM BOOK;

   BOOK_ID TITLE                PUB_YEAR PUBLISHER_NAME
---------- -------------------- ---------- --------------------
         1 DBMS                       2018 PEARSON
         2 ADBMS                      2018 PEARSON
         3 NETWORK                    2016 PEARSON
         4 IMAGE PROCESSING           2019 WILEY
         5 PRINCIPLES OF MGT          2010 MCGRAW HILL
```

4. **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;
```

```
View created.

SQL> SELECT * FROM V_PUBLICATION;

  PUB_YEAR
----------
      2018
      2018
      2016
      2019
      2010
```

```
SELECT *
FROM V_PUBLICATION
WHERE PUB_YEAR<2015;
```

```
  PUB_YEAR
----------
      2010
```

5. **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_PROGRAMME P
WHERE B. BOOK_ID=C.BOOK_ID AND
P. PROGRAMME_ID =C. PROGRAMME_ID;
```

```
View created.

SQL> SELECT * FROM V_BOOKS;

   BOOK_ID TITLE                NO_OF_COPIES
---------- -------------------- ------------
         1 DBMS                           20
         2 ADBMS                           5
         3 NETWORK                        15
         3 NETWORK                        10
         4 IMAGE PROCESSING               18
         4 IMAGE PROCESSING                5
         5 PRINCIPLES OF MGT              15

7 rows selected.
```

2.Consider the following schema for Order Database:

SALESMAN (*Salesman_id, Name, City, Commission*)

CUSTOMER (*Customer_id, Cust_Name, City, Grade, Salesman_id*)

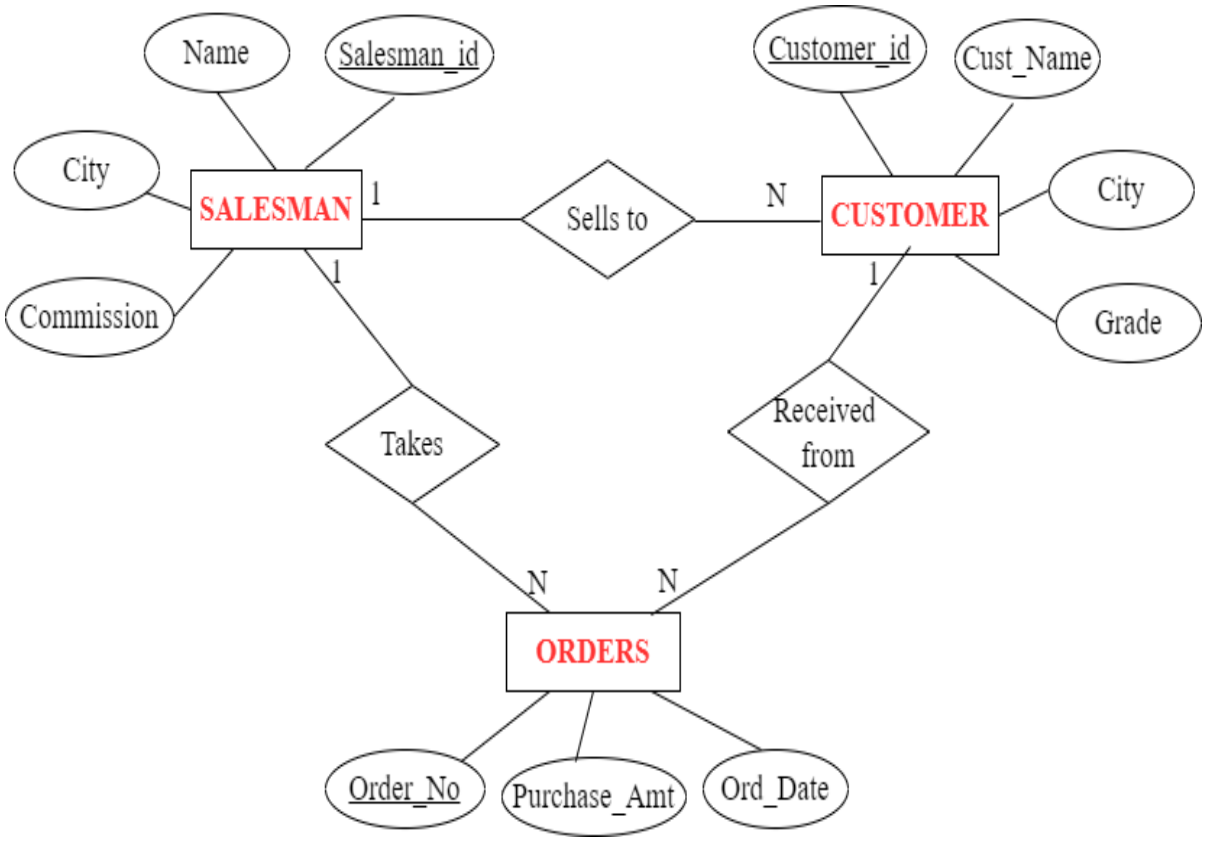
ORDERS (*Ord_No, Purchase_Amt, Ord_Date, Customer_id, Salesman_id*)

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:

ER DIAGRAM



SCHEMA DIAGRAM

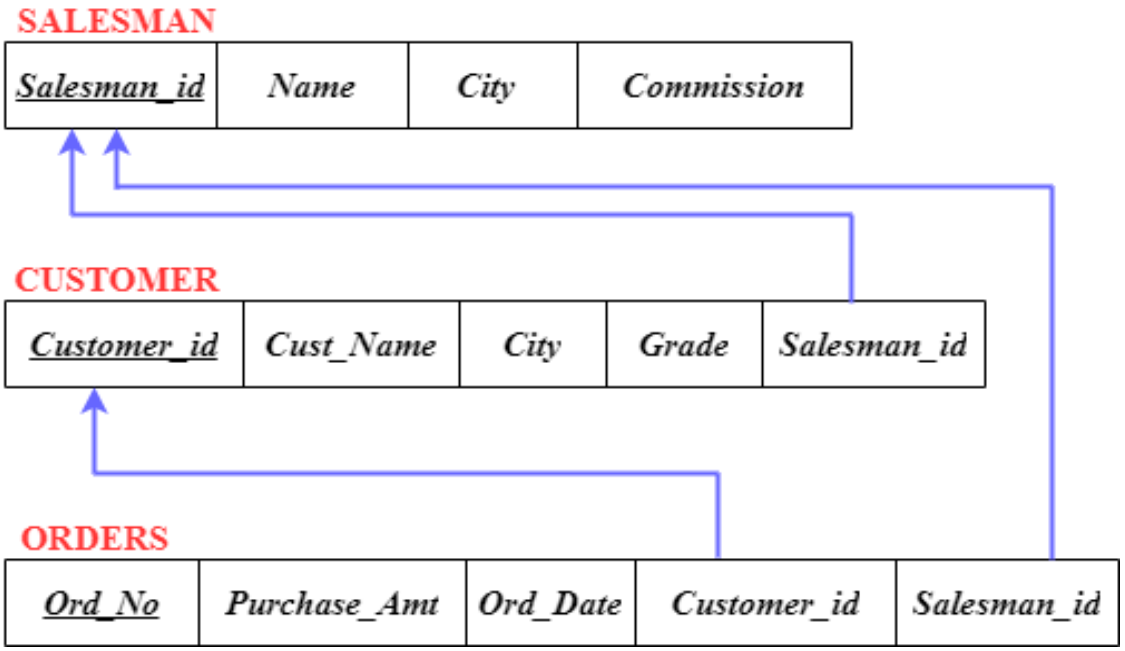


TABLE CREATION

CREATE TABLE SALESMAN (
SALESMAN_ID NUMBER (4),
NAME VARCHAR (20),
CITY VARCHAR (20),
COMMISSION NUMBER (4,2),
PRIMARY KEY (SALESMAN_ID));

CREATE TABLE CUSTOMER (
CUSTOMER_ID NUMBER (3),
CUST_NAME VARCHAR (20),
CITY VARCHAR (20),
GRADE NUMBER (3),
PRIMARY KEY (CUSTOMER_ID),
SALESMAN_ID REFERENCES SALESMAN (SALESMAN_ID) ON DELETE SET NULL);

CREATE TABLE ORDERS (
ORD_NO NUMBER (2),

PURCHASE_AMT NUMBER (10, 2),

ORD_DATE DATE,

PRIMARY KEY (ORD_NO),

CUSTOMER_ID REFERENCES CUSTOMER(CUSTOMER_ID) ON DELETE CASCADE,

SALESMAN_ID REFERENCES SALESMAN (SALESMAN_ID) ON DELETE CASCADE);

TABLE DESCRIPTIONS

DESC SALESMAN;

```
SQL> DESC SALESMAN;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SALESMAN_ID                               NOT NULL NUMBER(4)
 NAME                                               VARCHAR2(20)
 CITY                                               VARCHAR2(20)
 COMMISSION                                         NUMBER(4,2)
```

DESC CUSTOMER;

```
SQL> DESC CUSTOMER;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_ID                               NOT NULL NUMBER(3)
 CUST_NAME                                          VARCHAR2(20)
 CITY                                               VARCHAR2(20)
 GRADE                                              NUMBER(3)
 SALESMAN_ID                                        NUMBER(4)
```

DESC ORDERS;

```
SQL> DESC ORDERS;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORD_NO                                    NOT NULL NUMBER(2)
 PURCHASE_AMT                                       NUMBER(10,2)
 ORD_DATE                                           DATE
 CUSTOMER_ID                                        NUMBER(3)
 SALESMAN_ID                                        NUMBER(4)
```

INSERTION OF VALUES TO TABLES

INSERT INTO SALESMAN VALUES (1000, 'RAVI','BANGALORE',25.5);

INSERT INTO SALESMAN VALUES (2000, 'RAM','MANGALORE',20.5);

```
INSERT INTO SALESMAN VALUES (3000, 'KUMAR','MYSORE',15);
INSERT INTO SALESMAN VALUES (4000, 'JOHN','SULLIA',10.15);
INSERT INTO SALESMAN VALUES (5000,'HARSHA','PUTTUR',25);

INSERT INTO CUSTOMER VALUES (101,'VIVEK','MANGALORE',300, 1000);
INSERT INTO CUSTOMER VALUES (102,'BHASKAR','UDUPI',400,2000);
INSERT INTO CUSTOMER VALUES (103, 'CHETHAN', 'MYSORE', 200, 2000);
INSERT INTO CUSTOMER VALUES (104,'MAMATHA','BANGALORE',400,3000);
INSERT INTO CUSTOMER VALUES (105,'PREETHI','SULLIA',100,4000);

INSERT INTO ORDERS VALUES (11,5000,'20-JAN-21',101,1000);
INSERT INTO ORDERS VALUES (12,3000,'20-JAN-21',102,2000);
INSERT INTO ORDERS VALUES (13,4000,'17-MAY-21',103,1000);
INSERT INTO ORDERS VALUES (14,6000,'17-MAY-21',103,3000);
INSERT INTO ORDERS VALUES (15,4000,'17-JUL-21',105,5000);
```

RETRIEVAL OF INSERTED VALUES

SELECT * FROM SALESMAN;

```
SQL> SELECT * FROM SALESMAN;

SALESMAN_ID NAME                 CITY                 COMMISSION
----------- -------------------- -------------------- ----------
       1000 RAVI                 BANGALORE                  25.5
       2000 RAM                  MANGALORE                  20.5
       3000 KUMAR                MYSORE                       15
       4000 JOHN                 SULLIA                    10.15
       5000 HARSHA               PUTTUR                       25
```

SELECT * FROM CUSTOMER;

```
SQL> SELECT * FROM CUSTOMER;

CUSTOMER_ID CUST_NAME            CITY                 GRADE SALESMAN_ID
----------- -------------------- -------------------- ---------- -----------
        101 VIVEK                MANGALORE                 300        1000
        102 BHASKAR              UDUPI                     400        2000
        103 CHETHAN              BANGALORE                 200        2000
        104 MAMATHA              BANGALORE                 400        3000
        105 PREETHI              SULLIA                    100        4000
        106 RAJU                 SULLIA                    500        1000

6 rows selected.
```

SELECT * FROM ORDERS;

```
SQL> SELECT * FROM ORDERS;

    ORD_NO PURCHASE_AMT ORD_DATE  CUSTOMER_ID SALESMAN_ID
---------- ------------ --------- ----------- -----------
        11         5000 20-JAN-21         101        1000
        12         3000 20-JAN-21         102        2000
        13         4000 17-MAY-21         103        1000
        14         6000 17-MAY-21         103        3000
        15         4000 17-JUL-21         105        5000
```

QUERIES

---

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

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

```
    GRADE NO_OF_CUSTOMERS
---------- ---------------
       400               2
       500               1
```

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 CUSTOMER
           WHERE SALESMAN_ID=A.SALESMAN_ID);
```

```
SALESMAN_ID NAME
----------- --------------------
       1000 RAVI
       2000 RAM
```

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, CUSTOMER
WHERE SALESMAN.CITY = CUSTOMER.CITY
UNION
SELECT SALESMAN_ID, NAME, 'NO CUSTOMER', COMMISSION
FROM SALESMAN
WHERE NOT CITY = ANY
(SELECT CITY
FROM CUSTOMER)
ORDER BY  1 DESC;
```

```
SALESMAN_ID NAME                 CUST_NAME            COMMISSION
----------- -------------------- -------------------- ----------
       5000 HARSHA               NO CUSTOMER                  25
       4000 JOHN                 PREETHI                   10.15
       4000 JOHN                 RAJU                      10.15
       3000 KUMAR                NO CUSTOMER                  15
       2000 RAM                  VIVEK                      20.5
       1000 RAVI                 CHETHAN                    25.5
       1000 RAVI                 MAMATHA                    25.5

7 rows selected.
```

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

```
CREATE VIEW V_SALESMAN_LIST AS
SELECT ORD_NO, 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);
```

```
View created.

SQL> SELECT * FROM V_SALESMAN_LIST;

    ORD_NO ORD_DATE  SALESMAN_ID NAME
---------- --------- ----------- --------------------
        11 20-JAN-21        1000 RAVI
        14 17-MAY-21        3000 KUMAR
        15 17-JUL-21        5000 HARSHA
```

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

DELETE FROM SALESMAN
WHERE SALESMAN_ID=1000;

```
1 row deleted.

Commit complete.
SQL> SELECT * FROM SALESMAN;

SALESMAN_ID NAME                 CITY                 COMMISSION
----------- -------------------- -------------------- ----------
       2000 RAM                  MANGALORE                  20.5
       3000 KUMAR                MYSORE                       15
       4000 JOHN                 SULLIA                    10.15
       5000 HARSHA               PUTTUR                       25
```

3.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.

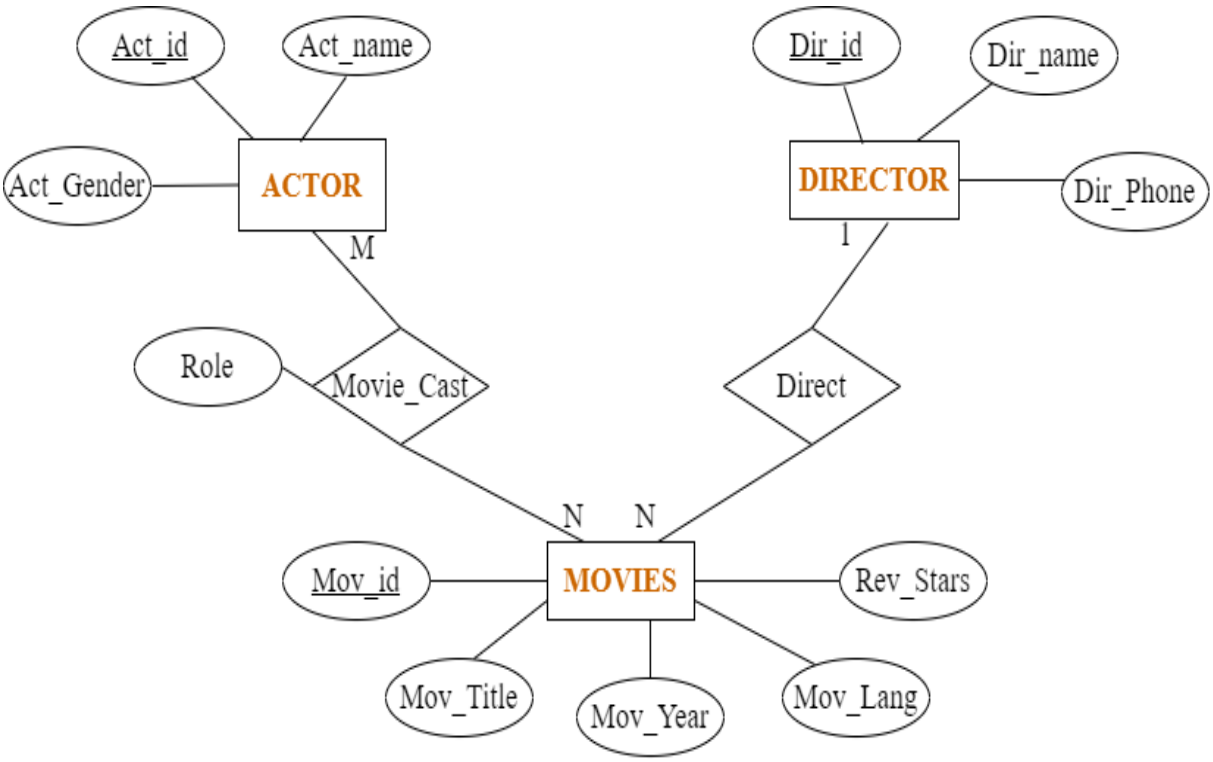
3. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).

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:

ER DIAGRAM



SCHEMA DIAGRAM

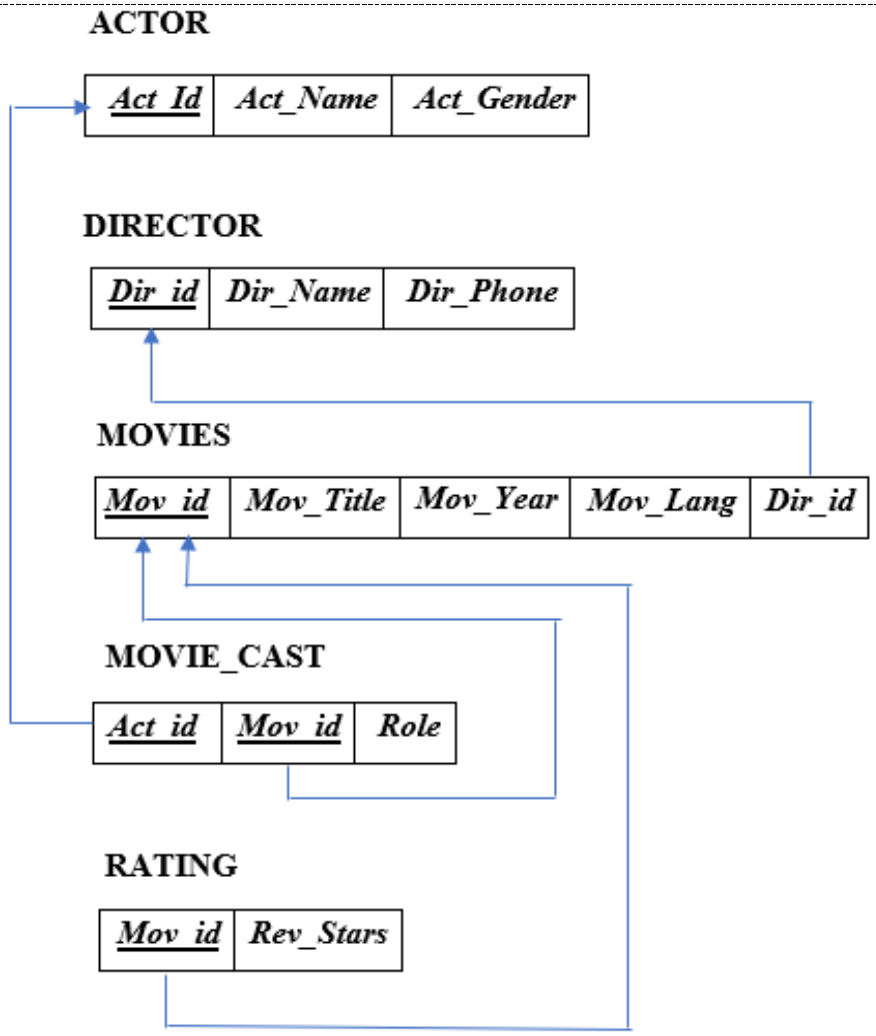


TABLE CREATION

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

CREATE TABLE DIRECTOR (
DIR_ID CHAR (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),
PRIMARY KEY (MOV_ID),
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 NUMBER (2),
 PRIMARY KEY (MOV_ID),
 FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID));
```

TABLE DESCRIPTION

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 CHAR(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                                             CHAR(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                                          NUMBER(2)
```

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 ACTOR VALUES (305,'TAYLOR','M');

INSERT INTO DIRECTOR VALUES ('D1','RAJAMOULI',8751611001);

INSERT INTO DIRECTOR VALUES ('D2','HITCHCOCK',7766138911);

INSERT INTO DIRECTOR VALUES ('D3','CHETHAN',9986776531);

INSERT INTO DIRECTOR VALUES ('D4','STEVEN SPIELBERG',8989776530);

INSERT INTO MOVIES VALUES (1001,'BAHUBALI-1', 2015,'TELAGU','D1');

INSERT INTO MOVIES VALUES (1002,'BAHUBALI-2', 2018,'TAMIL','D1');

INSERT INTO MOVIES VALUES (1003,’RAJAKUMARA’, 2020, ‘KANNADA’, ‘D3’);

```
INSERT INTO MOVIES VALUES (1004,'WAR HORSE', 2011,'ENGLISH','D4');
INSERT INTO MOVIES VALUES (1005,'THE BIRDS', 1963, 'ENGLISH', 'D2');

INSERT INTO MOVIE_CAST VALUES (301, 1001,'GUEST');
INSERT INTO MOVIE_CAST VALUES (301, 1002,'HEROINE');
INSERT INTO MOVIE_CAST VALUES (302, 1001,'HERO');
INSERT INTO MOVIE_CAST VALUES (302, 1005,'GUEST');
INSERT INTO MOVIE_CAST VALUES (304, 1004,'HERO');
INSERT INTO MOVIE_CAST VALUES (303, 1003,'HERO');

INSERT INTO RATING VALUES (1001, 4);
INSERT INTO RATING VALUES (1002, 3);
INSERT INTO RATING VALUES (1003, 4);
INSERT INTO RATING VALUES (1004, 3);
```

RETRIEVAL OF INSERTED VALUES

SELECT * FROM ACTOR;

```
SQL> SELECT * FROM ACTOR;

    ACT_ID ACT_NAME             A
---------- -------------------- -
       301 ANUSHKA              F
       302 PRABHAS              M
       303 PUNITH               M
       304 JERMY                M
       305 TAYLOR               M
```

SELECT * FROM DIRECTOR;

```
SQL> SELECT * FROM DIRECTOR;

DIR DIR_NAME              DIR_PHONE
--- -------------------- ----------
D1  RAJAMOULI            8751611001
D2  HITCHCOCK            7766138911
D3  CHETHAN              9986776531
D4  STEVEN SPIELBERG     8989776530
```

SELECT * FROM MOVIES;

```
SQL> SELECT * FROM MOVIES;

    MOV_ID MOV_TITLE                    MOV_YEAR MOV_LANG     DIR
---------- ------------------------- ---------- ------------ ---
      1001 BAHUBALI-1                      2015 TELAGU       D1
      1002 BAHUBALI-2                      2018 TAMIL        D1
      1004 WAR HORSE                       2011 ENGLISH      D4
      1005 THE BIRDS                       1963 ENGLISH      D2
      1003 RAJAKUMARA                      2020 KANNADA      D3
```

SELECT * FROM MOVIE_CAST;

```
SQL>  SELECT * FROM MOVIE_CAST;

    ACT_ID     MOV_ID ROLE
---------- ---------- ----------
       301       1001 GUEST
       301       1002 HEROINE
       302       1001 HERO
       302       1005 GUEST
       304       1004 HERO
       303       1003 HERO

6 rows selected.
```

SELECT * FROM RATING;

```
SQL> SELECT * FROM RATING;

    MOV_ID  REV_STARS
---------- ----------
      1001          4
      1002          3
      1003          4
      1004          3
```

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
-------------------------
THE BIRDS
```

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

```
SELECT MOV_TITLE
FROM MOVIES M, MOVIE_CAST MC
WHERE M.MOV_ID=MC.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 MC ON A.ACT_ID=MC.ACT_ID)
            JOIN MOVIES   M ON MC.MOV_ID=M.MOV_ID)
WHERE M.MOV_YEAR NOT BETWEEN 2000 AND 2015;
```

OR

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

```
ACT_NAME            MOV_TITLE                 MOV_YEAR
------------------- ------------------------- ----------
ANUSHKA             BAHUBALI-2                     2018
PRABHAS             THE BIRDS                      1963
PUNITH              RAJAKUMARA                     2020
```

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)
------------------------- --------------
BAHUBALI-1                             4
BAHUBALI-2                             3
RAJAKUMARA                             4
WAR HORSE                              3
```

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

```
SQL> SELECT * FROM RATING;

    MOV_ID  REV_STARS
---------- ----------
      1001          4
      1002          3
      1003          4
      1004          3
```

```
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 SPIELBERG'));
```

```
1 row updated.

Commit complete.
SQL> SELECT * FROM RATING;

    MOV_ID  REV_STARS
---------- ----------
      1001          4
      1002          3
      1003          4
      1004          5
```

4. Consider the schema for College Database:

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

SEMSEC (*SSID, Sem, Sec*)

CLASS (*USN, SSID*)

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

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

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 Courses.

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

5. Categorize students based on the following criterion:

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

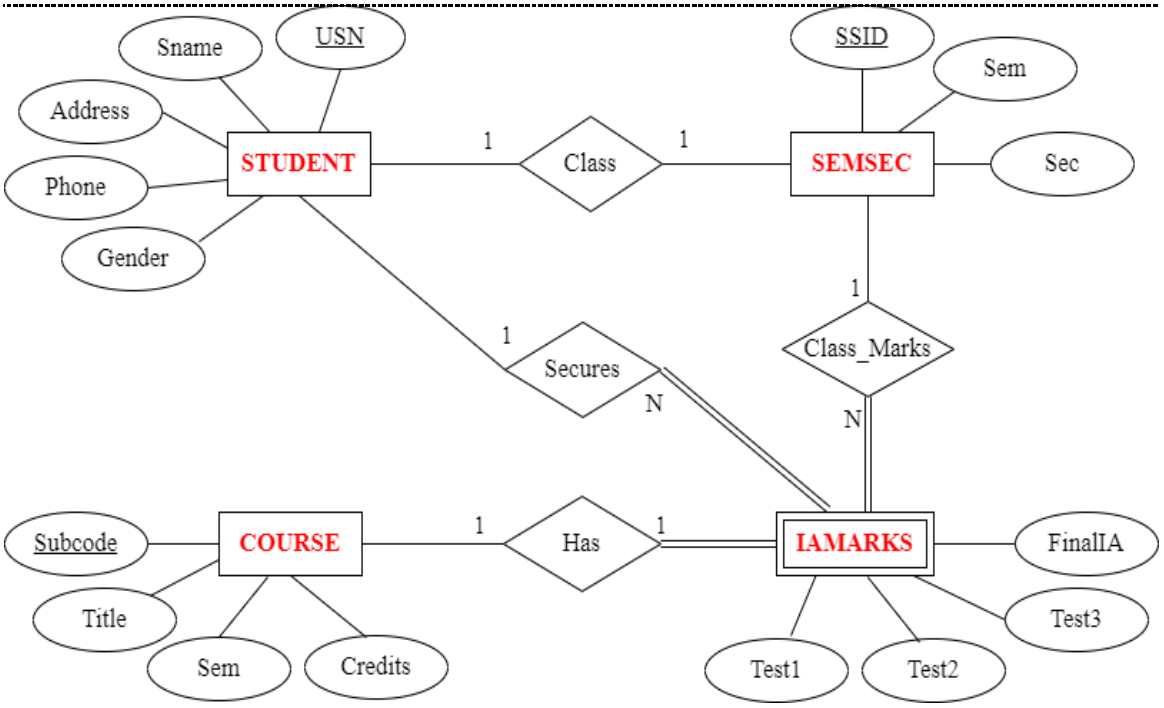
If FinalIA = 12 to 16 then CAT = "Average"

If FinalIA< 12 then CAT = "Weak"

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

Solution:

ER DIAGRAM



SCHEMA DIAGRAM

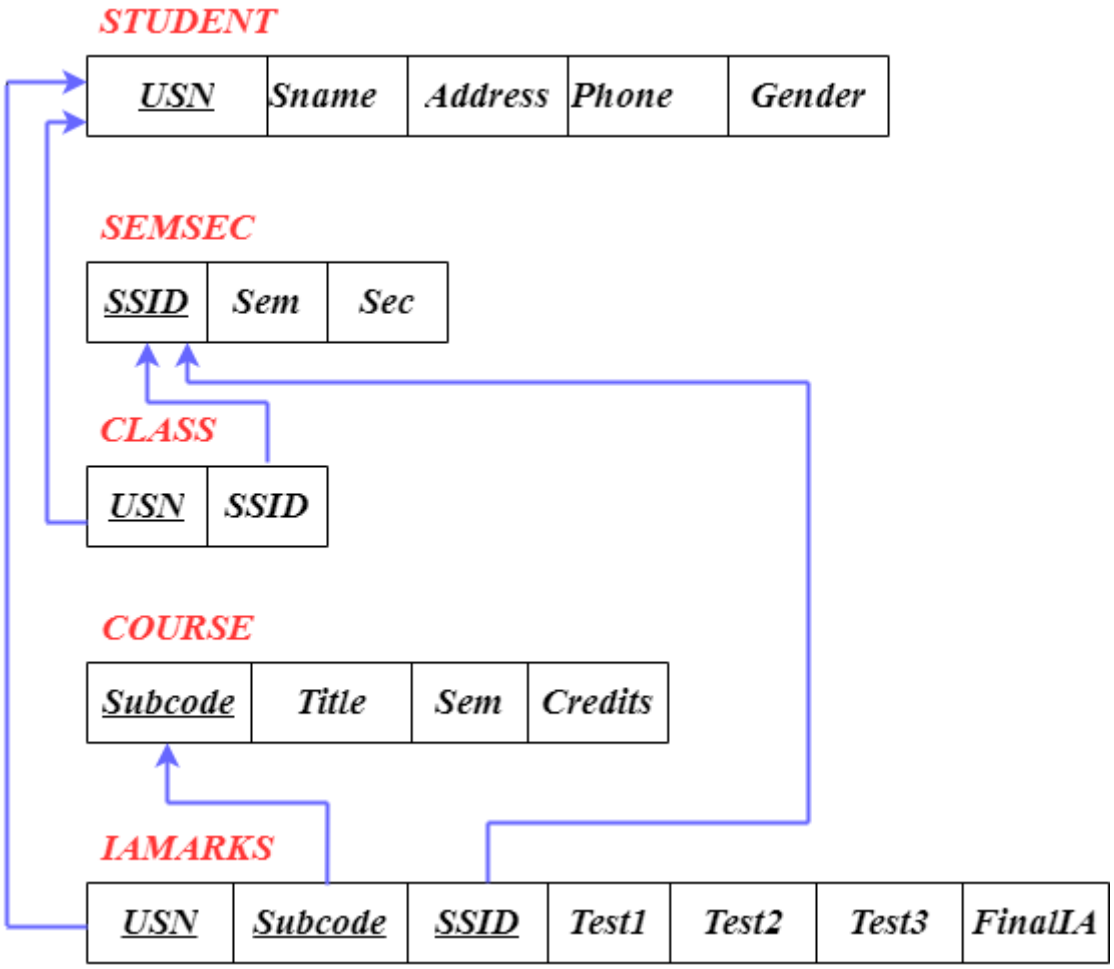


TABLE CREATION

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

CREATE TABLE SEMSEC (
SSID VARCHAR (2) PRIMARY KEY,
SEM NUMBER (1),
SEC CHAR (1),
CONSTRAINT CON_SEM CHECK (SEM BETWEEN 1 AND 8));
```

```
CREATE TABLE CLASS (
USN REFERENCES STUDENT (USN) ON DELETE CASCADE,
SSID REFERENCES SEMSEC (SSID) ON DELETE CASCADE,
PRIMARY KEY(USN));

CREATE TABLE COURSE (
SUBCODE VARCHAR (8) PRIMARY KEY,
TITLE VARCHAR (15),
SEM NUMBER (1),
CREDITS NUMBER (1));

CREATE TABLE IAMARKS (
USN CHAR (10),
SUBCODE VARCHAR (8),
SSID VARCHAR (2),
TEST1 NUMBER (2),
TEST2 NUMBER (2),
TEST3 NUMBER (2),
FINALIA NUMBER (2),
PRIMARY KEY (USN, SUBCODE, SSID),
FOREIGN KEY (USN) REFERENCES STUDENT (USN) ON DELETE CASCADE,
FOREIGN KEY (SUBCODE) REFERENCES COURSE (SUBCODE) ON DELETE
CASCADE,
FOREIGN KEY (SSID) REFERENCES SEMSEC (SSID) ON DELETE CASCADE);
```

TABLE DESCRIPTIONS

DESC STUDENT

```
SQL> DESC STUDENT
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USN                                       NOT NULL CHAR(10)
 SNAME                                              VARCHAR2(25)
 ADDRESS                                            VARCHAR2(25)
 PHONE                                              NUMBER(10)
 GENDER                                             CHAR(1)
```

DESC SEMSEC

```
SQL> DESC SEMSEC
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SSID                                      NOT NULL VARCHAR2(2)
 SEM                                                NUMBER(1)
 SEC                                                CHAR(1)
```

DESC CLASS

```
SQL> DESC CLASS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USN                                       NOT NULL CHAR(10)
 SSID                                               VARCHAR2(2)
```

DESC COURSE

```
SQL> DESC COURSE
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SUBCODE                                   NOT NULL VARCHAR2(8)
 TITLE                                              VARCHAR2(15)
 SEM                                                NUMBER(1)
 CREDITS                                            NUMBER(1)
```

DESC IAMARKS

```
SQL> DESC IAMARKS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USN                                       NOT NULL CHAR(10)
 SUBCODE                                   NOT NULL VARCHAR2(8)
 SSID                                      NOT NULL VARCHAR2(2)
 TEST1                                              NUMBER(2)
 TEST2                                              NUMBER(2)
 TEST3                                              NUMBER(2)
 FINALIA                                            NUMBER(2)
```

INSERTION OF VALUES TO TABLES

Insert Values into STUDENT

INSERT INTO **STUDENT** VALUES ('&USN','&SNAME','&ADDRESS', &PHONE,' &GENDER');

```
  1* INSERT INTO STUDENT VALUES ('&USN','&SNAME','&ADDRESS', &PHONE,'&GENDER')
SQL> /
Enter value for usn: 4KV15CS101
Enter value for sname: AMRUTHA
Enter value for address: MANGALORE
Enter value for phone: 6789000123
Enter value for gender: F
old   1: INSERT INTO STUDENT VALUES ('&USN','&SNAME','&ADDRESS', &PHONE,'&GENDER')
new   1: INSERT INTO STUDENT VALUES ('4KV15CS101','AMRUTHA','MANGALORE', 6789000123,'F')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for usn: 4KV15CS102
Enter value for sname: AASHISH
Enter value for address: SULLIA
Enter value for phone: 9087564123
Enter value for gender: M
old   1: INSERT INTO STUDENT VALUES ('&USN','&SNAME','&ADDRESS', &PHONE,'&GENDER')
new   1: INSERT INTO STUDENT VALUES ('4KV15CS102','AASHISH','SULLIA', 9087564123,'M')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for usn: 4KV15CS103
Enter value for sname: POOJA
Enter value for address: PUTTUR
Enter value for phone: 3428097661
Enter value for gender: F
old   1: INSERT INTO STUDENT VALUES ('&USN','&SNAME','&ADDRESS', &PHONE,'&GENDER')
new   1: INSERT INTO STUDENT VALUES ('4KV15CS103','POOJA','PUTTUR', 3428097661,'F')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for usn: 4KV16CS001
Enter value for sname: DEVAYANI
Enter value for address: SULLIA
Enter value for phone: 9087766543
Enter value for gender: F
old   1: INSERT INTO STUDENT VALUES ('&USN','&SNAME','&ADDRESS', &PHONE,'&GENDER')
new   1: INSERT INTO STUDENT VALUES ('4KV16CS001','DEVAYANI','SULLIA', 9087766543,'F')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for usn: 4KV16CS002
Enter value for sname: BHUVANESH
Enter value for address: BANGALORE
Enter value for phone: 9077112267
Enter value for gender: M
old   1: INSERT INTO STUDENT VALUES ('&USN','&SNAME','&ADDRESS', &PHONE,'&GENDER')
new   1: INSERT INTO STUDENT VALUES ('4KV16CS002','BHUVANESH','BANGALORE', 9077112267,'M')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for usn: 4KV17CS001
Enter value for sname: ANU
Enter value for address: SULLIA
Enter value for phone: 8097744121
Enter value for gender: F
old   1: INSERT INTO STUDENT VALUES ('&USN','&SNAME','&ADDRESS', &PHONE,'&GENDER')
new   1: INSERT INTO STUDENT VALUES ('4KV17CS001','ANU','SULLIA', 8097744121,'F')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for usn: 4KV17CS002
Enter value for sname: RAMESH
Enter value for address: SULLIA
Enter value for phone: 6780231453
Enter value for gender: M
old   1: INSERT INTO STUDENT VALUES ('&USN','&SNAME','&ADDRESS', &PHONE,'&GENDER')
new   1: INSERT INTO STUDENT VALUES ('4KV17CS002','RAMESH','SULLIA', 6780231453,'M')

1 row created.
```

Enter '/ 'to continue to insert the values

```
SQL> /
Enter value for usn: 4KV17CS003
Enter value for sname: NIKHIL
Enter value for address: PUTTUR
Enter value for phone: 2346780121
Enter value for gender: M
old   1: INSERT INTO STUDENT VALUES ('&USN','&SNAME','&ADDRESS', &PHONE,'&GENDER')
new   1: INSERT INTO STUDENT VALUES ('4KV17CS003','NIKHIL','PUTTUR', 2346780121,'M')

1 row created.
```

Enter '/ 'to continue to insert the values

```
SQL> /
Enter value for usn: 4KV18CS001
Enter value for sname: ANUSHA
Enter value for address: SULLIA
Enter value for phone: 9087654321
Enter value for gender: F
old   1: INSERT INTO STUDENT VALUES ('&USN','&SNAME','&ADDRESS', &PHONE,'&GENDER')
new   1: INSERT INTO STUDENT VALUES ('4KV18CS001','ANUSHA','SULLIA', 9087654321,'F')

1 row created.
```

Enter '/ 'to continue to insert the values

```
SQL> /
Enter value for usn: 4KV18CS002
Enter value for sname: ANWITHA
Enter value for address: SULLIA
Enter value for phone: 8907123456
Enter value for gender: F
old   1: INSERT INTO STUDENT VALUES ('&USN','&SNAME','&ADDRESS', &PHONE,'&GENDER')
new   1: INSERT INTO STUDENT VALUES ('4KV18CS002','ANWITHA','SULLIA', 8907123456,'F')

1 row created.
```

Insert Values into SEMSEC

INSERT INTO SEMSEC VALUES ('&SSID', &SEM,'&SEC')

```
SQL> /
Enter value for ssid: 2A
Enter value for sem: 2
Enter value for sec: A
old   1: INSERT INTO SEMSEC VALUES ('&SSID',&SEM,'&SEC')
new   1: INSERT INTO SEMSEC VALUES ('2A',2,'A')

1 row created.
```

Enter ' / ' to continue to insert the values

```
SQL> /
Enter value for ssid: 2B
Enter value for sem: 2
Enter value for sec: B
old   1: INSERT INTO SEMSEC VALUES ('&SSID',&SEM,'&SEC')
new   1: INSERT INTO SEMSEC VALUES ('2B',2,'B')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for ssid: 4A
Enter value for sem: 4
Enter value for sec: A
old   1: INSERT INTO SEMSEC VALUES ('&SSID',&SEM,'&SEC')
new   1: INSERT INTO SEMSEC VALUES ('4A',4,'A')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for ssid: 4B
Enter value for sem: 4
Enter value for sec: B
old   1: INSERT INTO SEMSEC VALUES ('&SSID',&SEM,'&SEC')
new   1: INSERT INTO SEMSEC VALUES ('4B',4,'B')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for ssid: 4C
Enter value for sem: 4
Enter value for sec: C
old   1: INSERT INTO SEMSEC VALUES ('&SSID',&SEM,'&SEC')
new   1: INSERT INTO SEMSEC VALUES ('4C',4,'C')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for ssid: 6A
Enter value for sem: 6
Enter value for sec: A
old   1: INSERT INTO SEMSEC VALUES ('&SSID',&SEM,'&SEC')
new   1: INSERT INTO SEMSEC VALUES ('6A',6,'A')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for ssid: 6B
Enter value for sem: 6
Enter value for sec: B
old   1: INSERT INTO SEMSEC VALUES ('&SSID',&SEM,'&SEC')
new   1: INSERT INTO SEMSEC VALUES ('6B',6,'B')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for ssid: 8A
Enter value for sem: 8
Enter value for sec: A
old   1: INSERT INTO SEMSEC VALUES ('&SSID',&SEM,'&SEC')
new   1: INSERT INTO SEMSEC VALUES ('8A',8,'A')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for ssid: 8B
Enter value for sem: 8
Enter value for sec: B
old   1: INSERT INTO SEMSEC VALUES ('&SSID',&SEM,'&SEC')
new   1: INSERT INTO SEMSEC VALUES ('8B',8,'B')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for ssid: 8C
Enter value for sem: 8
Enter value for sec: C
old   1: INSERT INTO SEMSEC VALUES ('&SSID',&SEM,'&SEC')
new   1: INSERT INTO SEMSEC VALUES ('8C',8,'C')

1 row created.
```

Insert Values into CLASS

INSERT INTO **CLASS** VALUES('&USN','&SSID');

```
SQL> INSERT INTO CLASS VALUES('&USN','&SSID');
Enter value for usn: 4KV15CS101
Enter value for ssid: 8A
old   1: INSERT INTO CLASS VALUES('&USN','&SSID')
new   1: INSERT INTO CLASS VALUES('4KV15CS101','8A')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for usn: 4KV15CS102
Enter value for ssid: 8B
old   1: INSERT INTO CLASS VALUES('&USN','&SSID')
new   1: INSERT INTO CLASS VALUES('4KV15CS102','8B')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for usn: 4KV15CS103
Enter value for ssid: 8C
old   1: INSERT INTO CLASS VALUES('&USN','&SSID')
new   1: INSERT INTO CLASS VALUES('4KV15CS103','8C')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for usn: 4KV16CS001
Enter value for ssid: 6A
old   1: INSERT INTO CLASS VALUES('&USN','&SSID')
new   1: INSERT INTO CLASS VALUES('4KV16CS001','6A')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for usn: 4KV16CS002
Enter value for ssid: 6B
old   1: INSERT INTO CLASS VALUES('&USN','&SSID')
new   1: INSERT INTO CLASS VALUES('4KV16CS002','6B')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for usn: 4KV17CS001
Enter value for ssid: 4A
old   1: INSERT INTO CLASS VALUES('&USN','&SSID')
new   1: INSERT INTO CLASS VALUES('4KV17CS001','4A')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for usn: 4KV17CS002
Enter value for ssid: 4B
old   1: INSERT INTO CLASS VALUES('&USN','&SSID')
new   1: INSERT INTO CLASS VALUES('4KV17CS002','4B')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for usn: 4KV17CS003
Enter value for ssid: 4C
old   1: INSERT INTO CLASS VALUES('&USN','&SSID')
new   1: INSERT INTO CLASS VALUES('4KV17CS003','4C')

1 row created.
```

Enter '/ ' to continue to insert the values

```
SQL> /
Enter value for usn: 4KV18CS001
Enter value for ssid: 2A
old   1: INSERT INTO CLASS VALUES('&USN','&SSID')
new   1: INSERT INTO CLASS VALUES('4KV18CS001','2A')

1 row created.
```

Enter '/ 'to continue to insert the values

```
SQL> /
Enter value for usn: 4KV18CS002
Enter value for ssid: 2B
old   1: INSERT INTO CLASS VALUES('&USN','&SSID')
new   1: INSERT INTO CLASS VALUES('4KV18CS002','2B')

1 row created.
```

Insert Values into COURSE

INSERT INTO COURSE VALUES ('&SUBCODE','&TITLE', &SEM,'&CREDITS');

```
SQL> INSERT INTO COURSE VALUES ('&SUBCODE','&TITLE', &SEM,'&CREDITS');
Enter value for subcode: 15CS81
Enter value for title: IOT
Enter value for sem: 8
Enter value for credits: 4
old   1: INSERT INTO COURSE VALUES ('&SUBCODE','&TITLE', &SEM,'&CREDITS')
new   1: INSERT INTO COURSE VALUES ('15CS81','IOT', 8,'4')

1 row created.
```

Enter ' / ' to continue to insert the values

```
SQL> /
Enter value for subcode: 15CS82
Enter value for title: BIG DATA
Enter value for sem: 8
Enter value for credits: 4
old   1: INSERT INTO COURSE VALUES ('&SUBCODE','&TITLE', &SEM,'&CREDITS')
new   1: INSERT INTO COURSE VALUES ('15CS82','BIG DATA', 8,'4')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for subcode: 15CS61
Enter value for title: CRYPTOGRAPHY
Enter value for sem: 6
Enter value for credits: 4
old   1: INSERT INTO COURSE VALUES ('&SUBCODE','&TITLE', &SEM,'&CREDITS')
new   1: INSERT INTO COURSE VALUES ('15CS61','CRYPTOGRAPHY', 6,'4')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for subcode: 15CS62
Enter value for title: GRAPHICS
Enter value for sem: 6
Enter value for credits: 4
old   1: INSERT INTO COURSE VALUES ('&SUBCODE','&TITLE', &SEM,'&CREDITS')
new   1: INSERT INTO COURSE VALUES ('15CS62','GRAPHICS', 6,'4')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for subcode: 15CS42
Enter value for title: SOFTWARE ENGG
Enter value for sem: 4
Enter value for credits: 3
old   1: INSERT INTO COURSE VALUES ('&SUBCODE','&TITLE', &SEM,'&CREDITS')
new   1: INSERT INTO COURSE VALUES ('15CS42','SOFTWARE ENGG', 4,'3')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for subcode: 15CS43
Enter value for title: ALGORITHMS
Enter value for sem: 4
Enter value for credits: 3
old   1: INSERT INTO COURSE VALUES ('&SUBCODE','&TITLE', &SEM,'&CREDITS')
new   1: INSERT INTO COURSE VALUES ('15CS43','ALGORITHMS', 4,'3')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for subcode: 18MAT21
Enter value for title: MATHS
Enter value for sem: 2
Enter value for credits: 4
old   1: INSERT INTO COURSE VALUES ('&SUBCODE','&TITLE', &SEM,'&CREDITS')
new   1: INSERT INTO COURSE VALUES ('18MAT21','MATHS', 2,'4')

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for subcode: 18CPS23
Enter value for title: C PROGRAM
Enter value for sem: 2
Enter value for credits: 3
old   1: INSERT INTO COURSE VALUES ('&SUBCODE','&TITLE', &SEM,'&CREDITS')
new   1: INSERT INTO COURSE VALUES ('18CPS23','C PROGRAM', 2,'3')

1 row created.
```

Insert Values into IAMARKS

INSERT INTO **IAMARKS** VALUES ('&USN','&SUBCODE','&SSID’, &TEST1, &TEST2, &TEST3, &FINALIA);

```
SQL> INSERT INTO IAMARKS VALUES ('&USN','&SUBCODE','&SSID’, &TEST1, &TEST2, &TEST3, &FINALIA);
Enter value for usn: 4KV15CS101
Enter value for subcode: 15CS81
Enter value for ssid: 8A
Enter value for test1: 15
Enter value for test2: 18
Enter value for test3: 14
Enter value for finalia: NULL
old   1: INSERT INTO IAMARKS VALUES ('&USN','&SUBCODE','&SSID', &TEST1, &TEST2, &TEST3, &FINALIA)
new   1: INSERT INTO IAMARKS VALUES ('4KV15CS101','15CS81','8A', 15, 18, 14, NULL)

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for usn: 4KV15CS101
Enter value for subcode: 15CS82
Enter value for ssid: 8A
Enter value for test1: 17
Enter value for test2: 20
Enter value for test3: 18
Enter value for finalia: NULL
old   1: INSERT INTO IAMARKS VALUES ('&USN','&SUBCODE','&SSID', &TEST1, &TEST2, &TEST3, &FINALIA)
new   1: INSERT INTO IAMARKS VALUES ('4KV15CS101','15CS82','8A', 17, 20, 18, NULL)

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for usn: 4KV15CS102
Enter value for subcode: 15CS81
Enter value for ssid: 8B
Enter value for test1: 16
Enter value for test2: 18
Enter value for test3: 19
Enter value for finalia: NULL
old   1: INSERT INTO IAMARKS VALUES ('&USN','&SUBCODE','&SSID', &TEST1, &TEST2, &TEST3, &FINALIA)
new   1: INSERT INTO IAMARKS VALUES ('4KV15CS102','15CS81','8B', 16, 18, 19, NULL)

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for usn: 4KV15CS102
Enter value for subcode: 15CS82
Enter value for ssid: 8B
Enter value for test1: 20
Enter value for test2: 18
Enter value for test3: 15
Enter value for finalia: NULL
old   1: INSERT INTO IAMARKS VALUES ('&USN','&SUBCODE','&SSID', &TEST1, &TEST2, &TEST3, &FINALIA)
new   1: INSERT INTO IAMARKS VALUES ('4KV15CS102','15CS82','8B', 20, 18, 15, NULL)

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for usn: 4KV15CS103
Enter value for subcode: 15CS81
Enter value for ssid: 8C
Enter value for test1: 18
Enter value for test2: 14
Enter value for test3: 17
Enter value for finalia: NULL
old   1: INSERT INTO IAMARKS VALUES ('&USN','&SUBCODE','&SSID', &TEST1, &TEST2, &TEST3, &FINALIA)
new   1: INSERT INTO IAMARKS VALUES ('4KV15CS103','15CS81','8C', 18, 14, 17, NULL)

1 row created.
```

Enter ‘ / ‘ to continue to insert the values

```
SQL> /
Enter value for usn: 4KV15CS103
Enter value for subcode: 15CS82
Enter value for ssid: 8C
Enter value for test1: 19
Enter value for test2: 14
Enter value for test3: 20
Enter value for finalia: NULL
old   1: INSERT INTO IAMARKS VALUES ('&USN','&SUBCODE','&SSID', &TEST1, &TEST2, &TEST3, &FINALIA)
new   1: INSERT INTO IAMARKS VALUES ('4KV15CS103','15CS82','8C', 19, 14, 20, NULL)

1 row created.
```

RETRIEVAL OF INSERTED VALUES

SELECT * FROM STUDENT;

```
SQL> SELECT * FROM STUDENT;

USN         SNAME                     ADDRESS                        PHONE G
----------  ------------------------- ------------------------- ---------- -
4KV15CS101 AMRUTHA                    MANGALORE                  6789000123 F
4KV15CS102 AASHISH                    SULLIA                     9087564123 M
4KV15CS103 POOJA                      PUTTUR                     3428097661 F
4KV16CS001 DEVAYANI                   SULLIA                     9087766543 F
4KV16CS002 BHUVANESH                  BANGALORE                  9077112267 M
4KV17CS001 ANU                        SULLIA                     8097744121 F
4KV17CS002 RAMESH                     SULLIA                     6780231453 M
4KV17CS003 NIKHIL                     PUTTUR                     2346780121 M
4KV18CS001 ANUSHA                     SULLIA                     9087654321 F
4KV18CS002 ANWITHA                    SULLIA                     8907123456 F

10 rows selected.
```

SELECT * FROM SEMSEC;

```
SQL> SELECT * FROM SEMSEC;

SS         SEM S
-- ---------- -
2A           2 A
2B           2 B
4A           4 A
4B           4 B
4C           4 C
6A           6 A
6B           6 B
8A           8 A
8B           8 B
8C           8 C

10 rows selected.
```

SELECT * FROM CLASS;

```
SQL> SELECT * FROM CLASS;

USN         SS
----------- --
4KV15CS101 8A
4KV15CS102 8B
4KV15CS103 8C
4KV16CS001 6A
4KV16CS002 6B
4KV17CS001 4A
4KV17CS002 4B
4KV17CS003 4C
4KV18CS001 2A
4KV18CS002 2B

10 rows selected.
```

SELECT * FROM COURSE;

```
SQL> SELECT * FROM COURSE;

SUBCODE  TITLE                    SEM    CREDITS
-------- --------------- ---------- ----------
15CS81   IOT                         8          4
15CS82   BIG DATA                    8          4
15CS61   CRYPTOGRAPHY                6          4
15CS62   GRAPHICS                    6          4
15CS42   SOFTWARE ENGG               4          3
15CS43   ALGORITHMS                 4          3
18MAT21  MATHS                       2          4
18CPS23  C PROGRAM                   2          3

8 rows selected.
```

SELECT * FROM IAMARKS;

```
SQL> SELECT * FROM IAMARKS;

USN         SUBCODE  SS      TEST1      TEST2      TEST3    FINALIA
----------- -------- -- ---------- ---------- ---------- ----------
4KV15CS101 15CS81   8A         15         18         14
4KV15CS101 15CS82   8A         17         20         18
4KV15CS102 15CS81   8B         16         18         19
4KV15CS102 15CS82   8B         20         18         15
4KV15CS103 15CS81   8C         18         14         17
4KV15CS103 15CS82   8C         19         14         20

6 rows selected.
```

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
4KV17CS003	NIKHIL	PUTTUR	2346780121	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 STUDENT 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
2	A	F	1
2	B	F	1
4	A	F	1
4	B	M	1
4	C	M	1
6	A	F	1
6	B	M	1
8	A	F	1
8	B	M	1
8	C	F	1

10 rows selected.

3. **Create a view of Test1 marks of student USN "1BI15CS101" in all Courses**.

```
CREATE VIEW TEST1_MARKS AS
SELECT USN, SUBCODE, TEST1
FROM IAMARKS
WHERE USN = '4KV15CS101';
```

```
View created.

SQL> SELECT * FROM TEST1_MARKS;

USN         SUBCODE        TEST1
---------- -------- ----------
4KV15CS101 15CS81            15
4KV15CS101 15CS82            17
```

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

```
SQL> SELECT * FROM IAMARKS;

USN         SUBCODE  SS      TEST1      TEST2      TEST3    FINALIA
---------- -------- -- ---------- ---------- ---------- ----------
4KV15CS101 15CS81   8A         15         18         14
4KV15CS101 15CS82   8A         17         20         18
4KV15CS102 15CS81   8B         16         18         19
4KV15CS102 15CS82   8B         20         18         15
4KV15CS103 15CS81   8C         18         14         17
4KV15CS103 15CS82   8C         19         14         20
```

UPDATE IAMARKS

SET FINALIA=((TEST1+TEST2+TEST3)-LEAST (TEST1, TEST2, TEST3))/2;

OR

UPDATE IAMARKS
SET
FINALIA=GREATEST((TEST1+TEST2),(TEST2+TEST3),(TEST3+TEST1))/2;

```
6 rows updated.

SQL> SELECT * FROM IAMARKS;

USN         SUBCODE  SS      TEST1      TEST2      TEST3    FINALIA
---------- -------- -- ---------- ---------- ---------- ----------
4KV15CS101 15CS81   8A         15         18         14         17
4KV15CS101 15CS82   8A         17         20         18         19
4KV15CS102 15CS81   8B         16         18         19         19
4KV15CS102 15CS82   8B         20         18         15         19
4KV15CS103 15CS81   8C         18         14         17         18
4KV15CS103 15CS82   8C         19         14         20         20

6 rows selected.
```

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 8th semester A, B, and C section students.

```
SELECT S.USN, S. SNAME, S. ADDRESS, S. PHONE, S. GENDER, IA. FINALIA,
  (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
 WHERE S.USN=IA.USN AND SS. SSID=IA.SSID AND SS.SEM=8;
```

```
USN        SNAME                    ADDRESS                       PHONE G    FINALIA CAT
---------- ------------------------ ------------------------ ---------- - ---------- -----------
4KV15CS101 AMRUTHA                  MANGALORE                 6789000123 F         17 OUTSTANDING
4KV15CS101 AMRUTHA                  MANGALORE                 6789000123 F         19 OUTSTANDING
4KV15CS102 AASHISH                  SULLIA                    9087564123 M         19 OUTSTANDING
4KV15CS102 AASHISH                  SULLIA                    9087564123 M         19 OUTSTANDING
4KV15CS103 POOJA                    PUTTUR                    3428097661 F         18 OUTSTANDING
4KV15CS103 POOJA                    PUTTUR                    3428097661 F         20 OUTSTANDING

6 rows selected.
```

5. 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.

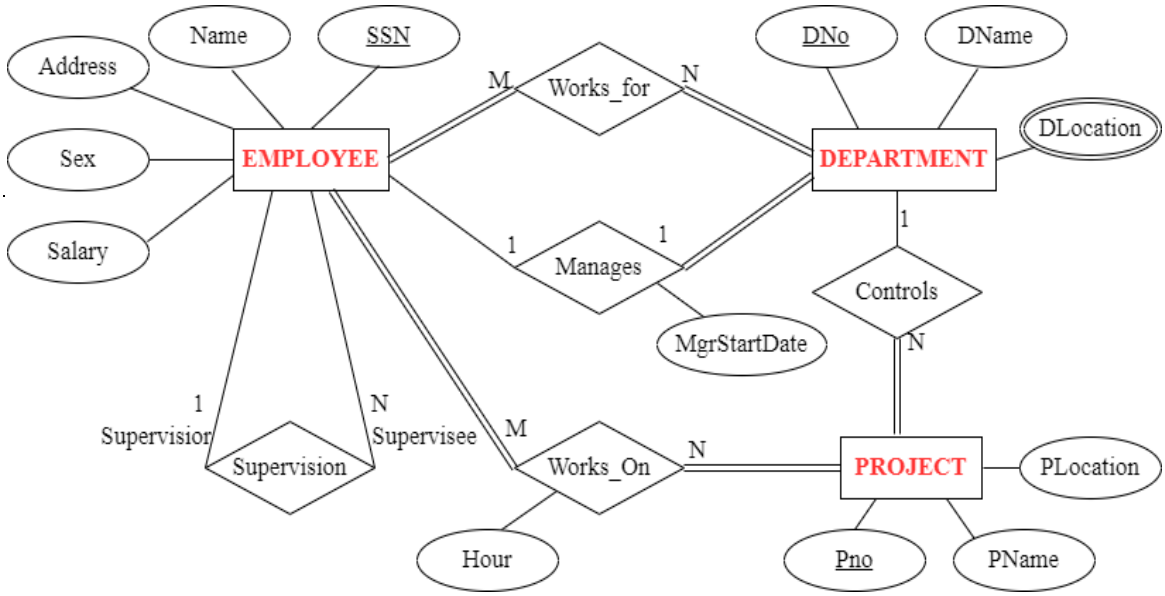
2. Show the resulting salaries if every employee working on the 'IoT' project is given a 10 percent raise.

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.

SOLUTION:

ER DIAGRAM



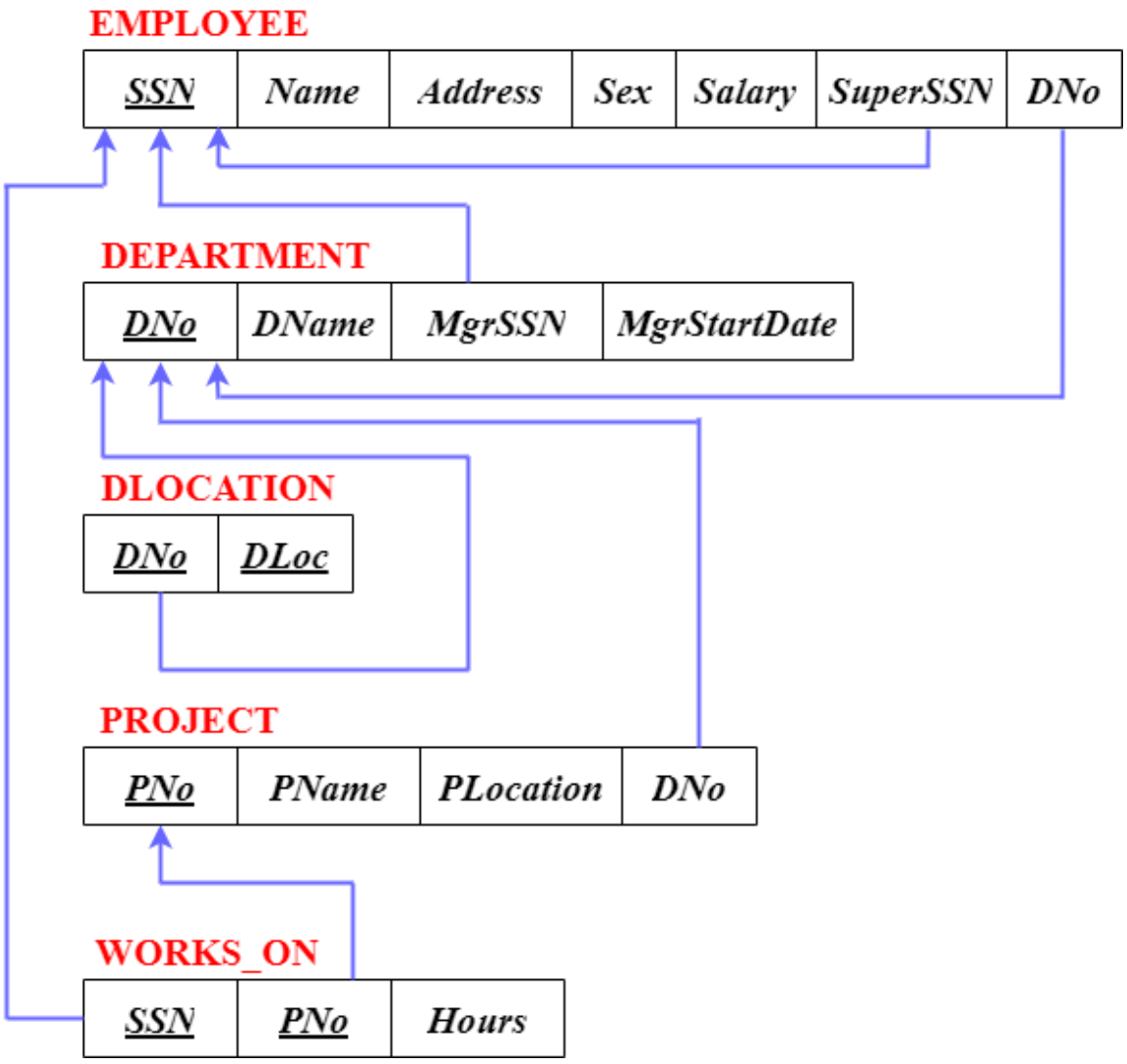


TABLE CREATION

CREATE TABLE EMPLOYEE

(SSN VARCHAR (10) PRIMARY KEY,

NAME VARCHAR (20),

ADDRESS VARCHAR (20),

SEX CHAR,

SALARY NUMBER (10,2),

SUPERSSN VARCHAR (10),

DNO NUMBER (2),

FOREIGN KEY(SUPERSSN) REFERENCES EMPLOYEE(SSN) ON DELETE CASCADE);

```
CREATE TABLE DEPARTMENT (
DNO NUMBER (2),
DNAME VARCHAR (15),
MGRSSN VARCHAR (10),
MGRSTARTDATE DATE,
PRIMARY KEY(DNO),
FOREIGN KEY(MGRSSN) REFERENCES EMPLOYEE(SSN) ON DELETE
CASCADE);
```

NOTE: Once DEPARTMENT and EMPLOYEE tables are created, we must alter EMPLOYEE table to add foreign constraint DNO using sql command.

```
ALTER TABLE EMPLOYEE
ADD FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNO) ON DELETE
CASCADE;
```

```
CREATE TABLE DLOCATION (
DNO NUMBER (2),
DLOC VARCHAR2(20),
PRIMARY KEY (DNO, DLOC),
FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNO));
```

```
CREATE TABLE PROJECT (
PNO NUMBER (4) PRIMARY KEY,
PNAME VARCHAR (20),
PLOCATION VARCHAR (20),
DNO REFERENCES DEPARTMENT (DNO));
```

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

TABLE DESCRIPTIONS

DESC EMPLOYEE;

```
SQL> DESC EMPLOYEE;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SSN                                       NOT NULL VARCHAR2(10)
 NAME                                               VARCHAR2(20)
 ADDRESS                                            VARCHAR2(20)
 SEX                                                CHAR(1)
 SALARY                                             NUMBER(10,2)
 SUPERSSN                                           VARCHAR2(10)
 DNO                                                NUMBER(2)
```

DESC DEPARTMENT;

```
SQL> DESC DEPARTMENT;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DNO                                       NOT NULL NUMBER(2)
 DNAME                                              VARCHAR2(15)
 MGRSSN                                             VARCHAR2(10)
 MGRSTARTDATE                                       DATE
```

DESC DLOCATION;

```
SQL> DESC DLOCATION;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DNO                                       NOT NULL NUMBER(2)
 DLOC                                      NOT NULL VARCHAR2(20)
```

DESC PROJECT;

```
SQL> DESC PROJECT;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PNO                                       NOT NULL NUMBER(4)
 PNAME                                              VARCHAR2(20)
 PLOCATION                                          VARCHAR2(20)
 DNO                                                NUMBER(2)
```

DESC WORKS_ON;

```
SQL> DESC WORKS_ON;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SSN                                       NOT NULL VARCHAR2(10)
 PNO                                       NOT NULL NUMBER(4)
 HOURS                                              NUMBER(2)
```

INSERTION OF VALUES TO TABLES

```
INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
('KVCS01',' RAVI ','SULLIA','M', 750000);

INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
(' KVCS02','MEENA','MANGLORE','F', 850000);

INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
(' KVCS03','SCOTT',' MANGLORE ','M', 650000);

INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
(' KVCS04','RANJINI','SULLIA','F',450000);

INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
('KVCS05','HEMA','SULLIA','F',910000);

INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
('KVCS06','ANJANA','SULLIA','F',500000);

INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
('KVEC01','JAMES','SULLIA','M',900000);

INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
('KVMBA01','KRISHNA','SULLIA','M',400000);

INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
('KVACC01','SCOTT','PUTTUR','M',910000);

INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
('KVACC02','RAJA','SULLIA','M',710000);

INSERT INTO EMPLOYEE (SSN, NAME, ADDRESS, SEX, SALARY) VALUES
('KVIT01','RAGHU’,'MANGALORE','M',400000);

INSERT INTO DEPARTMENT VALUES (1,'ACCOUNTS','KVACC01','01-JAN-15');

INSERT INTO DEPARTMENT VALUES (2,'IT','KVIT01','01-AUG-19');

INSERT INTO DEPARTMENT VALUES (3,'EC',' KVEC01','01-JUN-19');

INSERT INTO DEPARTMENT VALUES (4,'MBA','KVMBA01','01-JUN-20');

INSERT INTO DEPARTMENT VALUES (2,'CSE','KVCS01','01-JAN-18');
```

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

```
UPDATE EMPLOYEE
SET SUPERSSN='KVCS02', DNO=5
WHERE SSN=' KVCS01';
```

```
UPDATE EMPLOYEE
SET SUPERSSN='KVCS01', DNO=5
WHERE SSN=' KVCS02';

UPDATE EMPLOYEE
SET SUPERSSN='KVCS01', DNO=5
WHERE SSN=' KVCS03';

UPDATE EMPLOYEE
SET SUPERSSN='KVCS01', DNO=5
WHERE SSN=' KVCS04';

UPDATE EMPLOYEE
SET SUPERSSN='KVCS02', DNO=5
WHERE SSN=' KVCS05';

UPDATE EMPLOYEE
SET SUPERSSN='KVCS02', DNO=5
WHERE SSN=' KVCS06';

UPDATE EMPLOYEE
SET SUPERSSN=NULL, DNO=3
WHERE SSN='KVEC01';

UPDATE EMPLOYEE
SET SUPERSSN=NULL, DNO=4
WHERE SSN='KVMBA01';

UPDATE EMPLOYEE
SET SUPERSSN=NULL, DNO=1
WHERE SSN='KVACC01';

UPDATE EMPLOYEE
SET SUPERSSN='KVACC01', DNO=1
WHERE SSN='KVACC02';

UPDATE EMPLOYEE
SET SUPERSSN=NULL, DNO=2
WHERE SSN='KVIT01';

INSERT INTO DLOCATION VALUES (1,'SULLIA');

INSERT INTO DLOCATION VALUES (2,'PUTTUR');

INSERT INTO DLOCATION VALUES (3,'MANGALORE');

INSERT INTO DLOCATION VALUES (4,'MANGALORE');
```

```
INSERT INTO DLOCATION VALUES (5,'SULLIA');

INSERT INTO PROJECT VALUES (100,'IOT','BANGALORE',5);
INSERT INTO PROJECT VALUES (101,'WEB','MANGALORE',5);
INSERT INTO PROJECT VALUES (102,'BIGDATA','SULLIA',5);
INSERT INTO PROJECT VALUES (103,'BANK MANAGEMENT','MANGALORE',4);
INSERT INTO PROJECT VALUES (104,'SENSORS','MANGALORE',3);
INSERT INTO PROJECT VALUES (105,'SALARY MANAGEMENT','PUTTUR',1);

INSERT INTO WORKS_ON VALUES('KVCS01',100,8);
INSERT INTO WORKS_ON VALUES('KVCS02',100,4);
INSERT INTO WORKS_ON VALUES('KVCS02',101,7);
INSERT INTO WORKS_ON VALUES('KVCS02',102,12);
INSERT INTO WORKS_ON VALUES('KVCS03',100,10);
INSERT INTO WORKS_ON VALUES('KVCS03',101,6);
INSERT INTO WORKS_ON VALUES('KVCS03',102,4);
INSERT INTO WORKS_ON VALUES('KVMBA01',103,10);
INSERT INTO WORKS_ON VALUES('KVEC01',104,15);
INSERT INTO WORKS_ON VALUES('KVACC02',105,11);
```

RETRIEVAL OF INSERTED VALUES

SELECT * FROM EMPLOYEE;

```
SQL>  SELECT * FROM EMPLOYEE;

SSN        NAME                ADDRESS             S     SALARY SUPERSSN          DNO
---------- ------------------- ------------------- - ---------- ---------- ----------
KVCS01     RAVI                SULLIA              M     750000 KVCS02              5
KVCS02     MEENA               MANGLORE            F     850000 KVCS01              5
KVCS03     SCOTT               MANGLORE            M     650000 KVCS01              5
KVCS04     RANJINI             SULLIA              F     450000 KVCS01              5
KVCS05     HEMA                SULLIA              F     910000 KVCS02              5
KVCS06     ANJANA              SULLIA              F     500000 KVCS02              5
KVEC01     JAMES               SULLIA                    900000                     3
KVMBA01    KRISHNA             SULLIA              M     400000                     4
KVACC01    SCOTT               PUTTUR                    910000                     1
KVACC02    RAJA                SULLIA              M     710000 KVACC01             1
KVIT01     RAGHU               MANGALORE           M     400000                     2

11 rows selected.
```

SELECT * FROM DEPARTMENT;

```
SQL> SELECT * FROM DEPARTMENT;

       DNO DNAME           MGRSSN     MGRSTARTD
---------- --------------- ---------- ---------
         1 ACCOUNTS        KVACC01    01-JAN-15
         2 IT              KVIT01     01-AUG-19
         3 EC              KVEC01     01-JUN-19
         4 MBA             KVMBA01    01-JUN-20
         5 CSE             KVCS01     01-JAN-18
```

SELECT * FROM DLOCATION;

```
SQL> SELECT * FROM DLOCATION;

       DNO DLOC
---------- --------------------
         1 SULLIA
         2 PUTTUR
         3 MANGALORE
         4 MANGALORE
         5 SULLIA
```

SELECT * FROM PROJECT;

```
SQL> SELECT * FROM PROJECT;

       PNO PNAME                PLOCATION                 DNO
---------- -------------------- -------------------- ----------
       100 IOT                  BANGALORE                    5
       101 WEB                  MANGALORE                    5
       102 BIG DATA             SULLIA                       5
       103 BANK MANAGEMENT      MANGALORE                    4
       104 SENSORS              MANGALORE                    3
       105 SALARY MANAGEMENT    PUTTUR                       1

6 rows selected.
```

SELECT * FROM WORKS_ON;

```
SSN                 PNO      HOURS
---------- ---------- ----------
KVCS01             100          8
KVCS02             100          4
KVCS02             101          7
KVCS02             102         12
KVCS03             100         10
KVCS03             101          6
KVCS03             102          4
KVMBA01            103         10
KVEC01             104         15
KVACC02            105         11

10 rows selected.
```

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

```
       PNO
----------
       100
       101
       102
       105
```

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

```
SELECT E.SSN, E.NAME, E. SALARY,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';
```

```
SSN        NAME                 SALARY   INCR_SAL
---------- -------------------- ---------- ----------
KVCS01     RAVI                 750000     825000
KVCS02     MEENA                850000     935000
KVCS03     SCOTT                650000     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)
------------- ------------- ------------- -------------
      1620000        910000        710000        810000
```

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

```
SELECT E.SSN, E.NAME
FROM EMPLOYEE E
WHERE NOT EXISTS ((SELECT PNO
                   FROM PROJECT
                   WHERE DNO='5')
                      MINUS
                  (SELECT PNO
                   FROM WORKS_ON
                   WHERE E.SSN=SSN));
```

```
SSN        NAME
---------- --------------------
KVCS02     MEENA
KVCS03     SCOTT
```

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;
```

```
       DNO   COUNT(*)
---------- ----------
         5          4
```

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.

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 direct represents the view instead a definition of view is stored in data dictionary.

Growth and restructuring of base tables are 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 (A1, A2, …, An) is made up of the relation name R and the list of attributes Ai that it contains. A relation is defined as a set of tuples. Let r be the relation which contains set tuples (t1, t2, t3, ...,tn). Each tuple is an ordered list of values t=(v1,v2, ..., vn).

21. What is degree of a Relation?

It is the number of attributes 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 enables 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 normalization?

It is a process of analysing 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.

34. What is Functional Dependency?

A Functional dependency is denoted by X 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.

35. 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 A in F with a dependency Y 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.

36. What is Lossless join property?

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

37. What is 1 NF (Normal Form)?

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

38. What is Fully Functional dependency?

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

39. 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.

40. What is 3NF?

A relation schema R is in 3NF if it is in 2NF and for every FD X 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.

41. 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 A, X must be a candidate key.

42. What is 4NF?

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

X is subset or equal to (or) XY = R.

X is a super key.

43. What is 5NF?

A Relation schema R is said to be 5NF if for every join dependency {R1, R2, ..., Rn} that holds R, one the following is true

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

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. **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.

13. **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.

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

    CREATE TABLE AS SELECT command

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

    MONTHS_BETWEEN

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

    It is used to drop constraints specified on the table.

17. **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.

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 Modelling: *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)
SALARY	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