

RNS INSTITUTE OF TECHNOLOGY

Dr. Vishnuvardhan Road, Channasandra, Bengaluru - 560 098



Department of Information Science and Engineering

DBMS LABORATORY WITH MINI PROJECT MANUAL

V Semester

18CSL58

Faculty-in-charge

Dr. R Rajkumar

Associate Professor

RNS INSTITUTE OF TECHNOLOGY

Dr. Vishnuvardhan Road, Channasandra, Bengaluru -560 098

Department of Information Science and Engineering



VISION of the College

Building RNSIT into a World - Class Institution

MISSION of the College

To impart high quality education in Engineering, Technology and Management with a difference, enabling students to excel in their career by

1. Attracting quality Students and preparing them with a strong foundation in fundamentals so as to achieve distinctions in various walks of life leading to outstanding contributions.
2. Imparting value based, need based, and choice based, and skill based professional education to the aspiring youth and carving them into disciplined, World class Professionals with social responsibility.
3. Promoting excellence in Teaching, Research and Consultancy that galvanizes academic consciousness among Faculty and Students.
4. Exposing Students to emerging frontiers of knowledge in various domains and make them suitable for Industry, Entrepreneurship, Higher studies, and Research & Development.
5. Providing freedom of action and choice for all the Stake holders with better visibility.

VISION of the Department

Fostering winning professionals of Strong Informative Potentials

MISSION of the Department

Imparting high quality education in the area of Information Science so as to graduate the students with good fundamentals, "Information System Integration", "Software Creation" capability & suitably train them to thrive in Industries, higher schools of learning and R & D centers with a comprehensive perspective.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

ISE Graduates within three-four years of graduation will have

- **PEO1:** Acquired the fundamentals of computers and applied knowledge of Information Science & Engineering and continue to develop their technical competencies by problem solving using programming.
- **PEO2:** Ability to formulate problems attained the Proficiency to develop system/application software in a scalable and robust manner with various platforms, tools and frameworks to provide cost effective solutions.
- **PEO3:** Obtained the capacity to investigate the necessities of the software Product, adapt to technological advancement, promote collaboration and interdisciplinary activities, Protecting Environment and developing Comprehensive leadership.
- **PEO4:** Enabled to be employed and provide innovative solutions to real-world problems across different domains.
- **PEO5:** Possessed communication skills, ability to work in teams, professional ethics, social responsibility, entrepreneur and management, to achieve higher career goals, and pursue higher studies.

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

- **PO1: Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization for the solution of complex engineering problems
- **PO2: Problem analysis:** Identify, formulate, research literature, and analyze 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 consideration for public health and safety, and 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 modeling 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 teamwork:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- **PO10: Communication:** Communicate effectively on complex engineering activities with the engineering community and with the 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.

PROGRAM SPECIFIC OUTCOMES (PSOs)

ISE Graduates will have

- **PSO1 – Problem Solving Abilities:** Ability to demonstrate the fundamental and theoretical concepts, analyze the real time problems and develop customized software solutions by applying the knowledge of mathematics and algorithmic techniques.
- **PSO2 – Applied Engineering Skills:** Enable creative thinking, Ability to apply standard practices and strategies, technical skills in software design, development, integration of systems and management for improving the security, reliability and survivability of the infrastructure.
- **PSO3 – General Expertise and Higher Learning:** Ability to exchange knowledge effectively demonstrate the ability of teamwork, documentation skills, professional ethics, entrepreneurial skills and continuing higher education in the field of Information technology.

RNS INSTITUTE OF TECHNOLOGY

Dr. VISHNUVARDHAN ROAD, CHANNASANDRA, BENGALURU -560 098

Department of Information Science and Engineering

DBMS Laboratory with Mini project

Subject Code: 18CSL58

Hours/Week : 01L+2P

Total Hours : 40

I.A. Marks : 40

Exam Hours: 03

Exam Marks: 60

Course objectives

This course 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.

Course Outcomes

After studying this course, students will be able to:

| | |
|-----|--|
| CO1 | Design ER diagram, relational schema and develop database for a given problem and perform operations using simple SQL queries. |
| CO2 | Create ER diagram, relational schema and develop database for a given problem using SET and DELETE operation. |
| CO3 | Design ER diagram, relational schema and develop database for a given problem using JOIN operation and eliminate redundancies. |
| CO4 | Create ER diagram, relational schema and develop database for a given problem using nested and co-related queries |
| CO5 | Design and develop a mini-project for real world application using suitable DBMS. |
| CO6 | Apply the knowledge of Java/ PHP / C# / Python / any other front end tool to build a GUI to store, retrieve and manipulate data. |

CO mapping to PO/PSOs

| CO / PO & PSO | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 | PSO1 | PSO2 | PSO3 |
|------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|------|------|------|------|
| 18CSL58.1 | 3 | 3 | 3 | | | | | | 3 | | 2 | 3 | 1 | | |
| 18CSL58.2 | 3 | 3 | 3 | 2 | | | | | 3 | | 2 | 3 | 2 | | |
| 18CSL58.3 | 3 | 3 | 3 | 2 | | | | | 3 | | 2 | 3 | 2 | | |
| 18CSL58.4 | | 3 | 3 | 3 | | | | | 3 | | 2 | | 2 | | |
| 18CSL58.5 | | 3 | 3 | 3 | 2 | 1 | 2 | | 3 | 3 | | 3 | | 3 | 2 |
| 18CSL58.6 | | 3 | 3 | 3 | 2 | 1 | 2 | | 3 | 3 | 3 | 3 | | 3 | 2 |

DBMS Laboratory with Mini project

List of Programs

| Sl. No | Experiment | CO |
|-----------|--|------------|
| 1 | <p>Part A: SQL Programming</p> <p>Consider the following schema for a Library Database:</p> <p>BOOK (<u>Book_id</u>, Title, Publisher_Name, Pub_Year)</p> <p>BOOK_AUTHORS (<u>Book_id</u>, Author_Name)</p> <p>PUBLISHER (<u>Name</u>, Address, Phone)</p> <p>BOOK_COPIES (<u>Book_id</u>, <u>Branch_id</u>, No-of_Copies)</p> <p>BOOK_LENDING (<u>Book_id</u>, <u>Branch_id</u>, <u>Card_No</u>, Date_Out, Due_Date)</p> <p>LIBRARY_BRANCH (<u>Branch_id</u>, Branch_Name, Address)</p> <p>Write SQL queries to</p> <ol style="list-style-type: none"> 1. Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc. 2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017. 3. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation. 4. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query. 5. Create a view of all books and its number of copies that are currently available in the Library. | CO1 CO2 |
| 2 | <p>Consider the following schema for Order Database:</p> <p>SALESMAN (<u>Salesman_id</u>, Name, City, Commission)</p> <p>CUSTOMER (<u>Customer_id</u>, Cust_Name, City, Grade, Salesman_id)</p> <p>ORDERS (<u>Ord_No</u>, Purchase_Amt, Ord_Date, Customer_id, Salesman_id)</p> <p>Write SQL queries to</p> <ol style="list-style-type: none"> 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. | CO1 CO2 |
| 3 | <p>Consider the schema for Movie Database:</p> <p>ACTOR (<u>Act_id</u>, Act_Name, Act_Gender)</p> <p>DIRECTOR (<u>Dir_id</u>, Dir_Name, Dir_Phone)</p> <p>MOVIES (<u>Mov_id</u>, Mov_Title, Mov_Year, Mov_Lang, Dir_id)</p> <p>MOVIE_CAST (<u>Act_id</u>, <u>Mov_id</u>, Role)</p> <p>RATING (<u>Mov_id</u>, Rev_Stars)</p> <p>Write SQL queries to</p> <ol style="list-style-type: none"> 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 | CO1 CO2 |

| | | |
|---|--|------------|
| | <p>movie after 2015 (use JOIN operation).</p> <p>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.</p> <p>5. Update rating of all movies directed by ‘Steven Spielberg’ to 5</p> | |
| 4 | <p>Consider the schema for College Database:</p> <p>STUDENT (<u>USN</u>, SName, Address, Phone, Gender)</p> <p>SEMSEC (<u>SSID</u>, Sem, Sec)</p> <p>CLASS (<u>USN</u>, <u>SSID</u>)</p> <p>SUBJECT (Subcode, Title, Sem, Credits)</p> <p>IAMARKS (<u>USN</u>, <u>Subcode</u>, <u>SSID</u>, Test1, Test2, Test3, FinalIA)</p> <p>Write SQL queries to</p> <ol style="list-style-type: none"> 1. List all the student details studying in fourth semester ‘C’ section. 2. Compute the total number of male and female students in each semester and in each section. 3. Create a view of Test1 marks of student USN ‘1BI15CS101’ in all subjects. 4. Calculate the 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. | CO3 CO4 |
| 5 | <p>Consider the schema for Company Database:</p> <p>EMPLOYEE (<u>SSN</u>, Name, Address, Sex, Salary, SuperSSN, DNo)</p> <p>DEPARTMENT (<u>DNo</u>, DName, MgrSSN, MgrStartDate)</p> <p>DLOCATION (<u>DNo</u>, <u>DLoc</u>)</p> <p>PROJECT (<u>PNo</u>, PName, PLocation, DNo)</p> <p>WORKS_ON (<u>SSN</u>, <u>PNo</u>, Hours)</p> <p>Write SQL queries to</p> <ol style="list-style-type: none"> 1. Make a list of all project numbers for projects that involve an employee whose last name is ‘Scott’, either as a worker or 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). 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. | CO3 CO4 |
| 7 | Part A: SQL Programming-Mini-Project | CO5 CO6 |

DBMS Laboratory with Mini project

Subject Code: 18CSL58
Hours/Week : 01L+2P
Total Hours : 40

I.A. Marks : 40
Exam Hours: 03
Exam Marks: 60

Regular Lab Assessment based on Write-up and Execution Rubrics (Max: 15 marks)

| | | Above Average | Average | Below Average |
|----|---|--|---|--|
| a. | Understanding of problem and approach to solve (4 Marks) | Able to analyze the given problem and efficiently implement using suitable high-level language instructions. (4-3) | Able to analyze the problem and moderate understanding of high-level language instructions. (3-2) | Poor understanding of high-level language instructions or No program write-up. (2-1) |
| b. | Execution and Viva Voce (7 Marks) | Program executed for varied inputs with valid results and able to answer all five questions appropriately. (7-5) | Program is executed for some inputs and able to answer three-two questions. (5-3) | Program has compilation errors or no Execution and not answered any questions. (2-1) |
| c. | Results and Documentation (4 Marks) | Program and results obtained are legibly written / documented. (4-3) | Program and results obtained is acceptably documented. (3-2) | No Proper results and poor documentation. (2-1) |

Lab Internal Assessment Test Rubrics (Max: 10 marks)

| | | Above Average | Average | Below Average |
|----|----------------------------|---|---|---|
| a. | Write-up (3 Marks) | Able to write the complete code. (3) | Able to write the code with few errors. (2-1) | Unable to write. (1-0) |
| b. | Execution (4 Marks) | Executed successfully for all the inputs given. (4-3) | Obtained partially correct results. (3-2) | Program has compilation errors or No Execution. (2-1) |
| c. | Viva Voce (3 Marks) | Able to answer all questions correctly. (3-2) | Able to answer 1 - 2 questions. (2-1) | Not answered any. (1-0) |

Mini-Project Assessment Rubrics (Max: 15 marks)

| | | Above Average | Average | Below Average |
|----|---|---|--|--|
| a. | Demonstration / Presentation (5 Marks) | Able to develop a mini-project completely and given presentation along with the team members. (5 - 4) | Able to develop a mini-project partially and given presentation along with the team members. (3-2) | Not able to develop mini-project or absent for the presentation. (2-0) |
| b. | Report (5 Marks) | Proper document according to format specified by documenting all the phases of project. (5-4) | Able to document project phases impartially and not following the standard format (3-2) | Format is not as per the specified / standard template or not able to draft the Report (1 - 0) |
| c | Viva Voce (5 Marks) | Able to answer all questions correctly. (5) | Able to answer 2 - 3 questions. (3 - 2) | Answered partially or Not answered any. (2 - 0) |

Lesson Planning for Laboratory Experiments

| Week No. | Schedule / Plan |
|----------|---|
| 1 | Introduction to |
| 2 | Program 1: Design of ER Diagram and Schema Diagram. Create, Insert, Queries |
| 3 | Program 2: Design of ER Diagram and Schema Diagram. Create, Insert, Queries |
| 4 | Mini Project - Design of ER and Schema Diagram Presentation |
| 5 | Lab Test I: Program 1 and 2 |
| 6 | Program 3: Design of ER Diagram and Schema Diagram. Create, Insert, Queries |
| 7 | Program 4: Design of ER Diagram and Schema Diagram. Create, Insert, Queries |
| 8 | Program 5: Design of ER Diagram and Schema Diagram. Create, Insert, Queries |
| 9 | Mini Project Backend creation in Oracle / MySQL / MS SQL Server / any other RDBMS |
| 10 | Mini Project Frontend implementation using appropriate language to design GUI / Web – Based application |
| 11 | Mini Project Integration and Testing / Verification & Validation |
| 12 | Mini Project Report writing exercise and final demonstration / presentation and submission |
| 13 | Lab Test II: All 5 programs |

INTRODUCTION TO SQL

Pronounced as SEQUEL: Structured English QUERY Language

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

DATA DEFINITION, CONSTRAINTS, AND SCHEMA CHANGES

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

Data Definition in SQL

CREATE, ALTER and DROP

| | |
|-------------|-----------|
| table..... | relation |
| row..... | tuple |
| column..... | attribute |

DATA TYPES

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

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

CREATE SCHEMA

Specifies a new database schema by giving it a name

Ex: CREATE SCHEMA COMPANY AUTHORIZATION Jsmith;

CREATE TABLE

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

Syntax of CREATE Command:

CREATE TABLE <table name> (<Attribute A1> <Data Type D1> [<Constarints>],

<Attribute A2> <Data Type D2> [<Constarints>],

.....

<Attribute An> <Data Type Dn> [<Constarints>],

[<integrity-constraint1>, <integrity-constraint k>]);

- A constraint NOT NULL may be specified on an attribute

A constraint NOT NULL may be specified on an attribute

Ex: CREATE TABLE DEPARTMENT (

DNAME VARCHAR(10) NOT NULL,

DNUMBER INTEGER NOT NULL,

MGRSSN CHAR(9), MGRSTARTDATE CHAR(9));

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

Ex: CREATE TABLE DEPT (

DNAME VARCHAR(10) NOT NULL,

DNUMBER INTEGER NOT NULL,

MGRSSN CHAR(9),

MGRSTARTDATE CHAR(9),

PRIMARY KEY (DNUMBER),

UNIQUE (DNAME),

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

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

Ex: CREATE TABLE DEPT (

 DNAME VARCHAR(10) NOT NULL,

 DNUMBER INTEGER NOT NULL,

 MGRSSN CHAR(9), MGRSTARTDATE CHAR(9),

 PRIMARY KEY (DNUMBER),

 UNIQUE (DNAME),

 FOREIGN KEY (MGRSSN) REFERENCES EMP

 ON DELETE SET DEFAULT ON UPDATE CASCADE);

DROP TABLE

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

Example: DROP TABLE DEPENDENT;

ALTER TABLE:

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

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

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

DROP A COLUMN (AN ATTRIBUTE)

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

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

BASIC QUERIES IN SQL

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

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

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

SIMPLE SQL QUERIES

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

SELECT

PROJECT

JOIN

All subsequent examples uses COMPANY database as shown below:

Example of a simple query on one relation

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

Q0: SELECT BDATE, ADDRESS FROM EMPLOYEE

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

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

| EMPLOYEE | | | | | | | | | |
|-----------------------|---------|----------------|-----------|--------------|--------------|-----|--------|----------|-----|
| FNAME | MINIT | LNAME | SSN | BDATE | ADDRESS | SEX | SALARY | SUPERSSN | DNO |
| DEPARTMENT | | | | | | | | | |
| | DNAME | DNUMBER | MGRSSN | MGRSTARTDATE | | | | | |
| DEPT_LOCATIONS | | | | | | | | | |
| | DNUMBER | DLOCATION | | | | | | | |
| PROJECT | | | | | | | | | |
| | PNAME | PNUMBER | PLOCATION | DNUM | | | | | |
| WORKS_ON | | | | | | | | | |
| | ESSN | PNO | HOURS | | | | | | |
| DEPENDENT | | | | | | | | | |
| | ESSN | DEPENDENT_NAME | SEX | BDATE | RELATIONSHIP | | | | |

| EMPLOYEE | FNAME | MINIT | LNAME | SSN | BDATE | ADDRESS | SEX | SALARY | SUPERSSN | DNO |
|----------|-------|---------|-----------|------------|--------------------------|---------|-------|-----------|----------|-----|
| John | B | Smith | 123456789 | 1965-01-09 | 731 Fondren, Houston, TX | M | 30000 | 33345555 | 5 | |
| Franklin | T | Wong | 33345555 | 1955-12-08 | 638 Voss, Houston, TX | M | 40000 | 88866555 | 5 | |
| Alicia | J | Zelaya | 98988777 | 1968-07-19 | 3321 Castle, Spring, TX | F | 25000 | 987654321 | 4 | |
| Jennifer | S | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellaire, TX | F | 43000 | 888665525 | 4 | |
| Ramona | K | Norayan | 66688444 | 1962-09-15 | 975 Fire Oak, Humble, TX | M | 38000 | 33345555 | 5 | |
| Joyce | A | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX | F | 25000 | 33345555 | 5 | |
| Ahmad | V | Jabbar | 987987987 | 1969-03-29 | 960 Dallas, Houston, TX | M | 25000 | 987654321 | 4 | |
| James | E | Borg | 888665525 | 1937-11-10 | 460 Stone, Houston, TX | M | 55000 | null | 1 | |

| DEPARTMENT | DNAME | DNUMBER | MGRSSN | MGRSTARTDATE | DEPT_LOCATIONS | DNUMBER | DLOCATION |
|----------------|-------|---------|-----------|--------------|----------------|---------|-----------|
| Research | | 5 | 33345555 | 1988-05-22 | | 1 | Houston |
| Administration | | 4 | 987654321 | 1995-01-01 | | 4 | Stafford |
| Headquarters | | 1 | 888665555 | 1981-06-19 | | 5 | Bellaire |
| | | | | | | 5 | Sugarland |
| | | | | | | 5 | Houston |

| WORKS_ON | ESSN | PNO | HOURS | PROJECT | PNAME | PNUMBER | PLOCATION | DNUM |
|----------|-----------|-----|-------|-----------------|-------|-----------|-----------|------|
| | 123456789 | 1 | 32.5 | ProductX | 1 | ReSale | | 5 |
| | 123456789 | 2 | 7.5 | ProductY | 2 | Sugarland | | 5 |
| | 66688444 | 3 | 40.0 | ProductZ | 3 | Houston | | 5 |
| | 453453453 | 1 | 20.0 | Computerization | 10 | Stafford | | 4 |
| | 453453453 | 2 | 20.0 | Reorganization | 20 | Houston | | 1 |
| | 33345555 | 2 | 10.0 | Newbenefits | 30 | Stafford | | 4 |
| | 33345555 | 3 | 10.0 | | | | | |
| | 33345555 | 10 | 10.0 | | | | | |
| | 33345555 | 20 | 10.0 | | | | | |
| | 99988777 | 30 | 30.0 | | | | | |
| | 99988777 | 10 | 10.0 | | | | | |
| | 987987987 | 10 | 35.0 | | | | | |
| | 987987987 | 30 | 5.0 | | | | | |
| | 987654321 | 30 | 20.0 | | | | | |
| | 987654321 | 20 | 15.0 | | | | | |
| | 888665555 | 20 | null | | | | | |

| DEPENDENT | ESSN | DEPENDENT_NAME | SEX | BDATE | RELATIONSHIP |
|-----------|-----------|----------------|------------|-------|--------------|
| 93345555 | Alice | F | 1968-04-05 | | DAUGHTER |
| 93345555 | Theodore | M | 1983-10-25 | | SON |
| 93345555 | Jay | F | 1968-05-03 | | SPOUSE |
| 987654321 | Abner | M | 1942-02-28 | | SPOUSE |
| 123456789 | Michael | M | 1968-01-04 | | SON |
| 123456789 | Alice | F | 1988-12-30 | | DAUGHTER |
| 123456789 | Elizabeth | F | 1967-05-05 | | SPOUSE |

Example of a simple query on two relations

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

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

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

Example of a simple query on three relations

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

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

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

ALIASES, * AND DISTINCT, EMPTY WHERE-CLAUSE

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

Example

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

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

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

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

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

UNSPECIFIED WHERE-clause

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

Example:

Query 4: Retrieve the SSN values for all employees.

Q4: SELECT SSN FROM EMPLOYEE

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

Example:

Q5: SELECT SSN, DNAME FROM EMPLOYEE, DEPARTMENT

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

USE OF *

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

Examples:

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

Q1a: SELECT * FROM EMPLOYEE WHERE DNO=5

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

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

USE OF DISTINCT

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

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

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

SET OPERATIONS

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

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

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

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

NESTING OF QUERIES

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

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

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

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

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

CORRELATED NESTED QUERIES

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

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

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

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

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

THE EXISTS FUNCTION

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

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

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

```

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

```

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

EXPLICIT SETS

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

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

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

NULLS IN SQL QUERIES

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

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

```

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

```

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

AGGREGATE FUNCTIONS

Include COUNT, SUM, MAX, MIN, and AVG

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

```

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

```

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

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

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

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

Q13: `SELECT COUNT (*) FROM EMPLOYEE`

Q14: `SELECT COUNT (*) FROM EMPLOYEE, DEPARTMENT`

`WHERE DNO=DNUMBER AND DNAME='Research'`

GROUPING

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

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

Q15: `SELECT DNO, COUNT (*), AVG (SALARY)`

`FROM EMPLOYEE GROUP BY DNO`

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

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

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

```
FROM PROJECT, WORKS_ON
WHERE PNUMBER=PNO
GROUP BY PNUMBER, PNAME
```

THE HAVING-CLAUSE

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

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

```
Q17: SELECT PNUMBER, PNAME, COUNT (*)
      FROM PROJECT, WORKS_ON
      WHERE PNUMBER=PNO
      GROUP BY PNUMBER, PNAME
      HAVING COUNT (*) > 2
```

SUBSTRING COMPARISON

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

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

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

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

Here, '5' must be the 8th character of the string (according to our format for date), so the BDATE value is '_____5_', with each underscore as a place holder for a single arbitrary character.

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

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

ARITHMETIC OPERATIONS

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

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

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

ORDER BY

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

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

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

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

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

MORE EXAMPLE QUERIES:

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

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

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

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

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

SPECIFYING UPDATES IN SQL

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

INSERT

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

Example:

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

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

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

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

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

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

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

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

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

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

DELETE

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

Examples:

- 1: DELETE FROM EMPLOYEE WHERE LNAME='Brown';
- 2: DELETE FROM EMPLOYEE WHERE SSN='123456789';
- 3: DELETE FROM EMPLOYEE WHERE DNO IN (SELECT DNUMBER
FROM DEPARTMENT WHERE DNAME='Research');
- 4: DELETE FROM EMPLOYEE;

UPDATE

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

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

UPDATE PROJECT

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

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

UPDATE EMPLOYEE

SET SALARY = SALARY *1.1

WHERE DNO IN (SELECT DNUMBER FROM DEPARTMENT

WHERE DNAME='Research');

SQL TRIGGERS

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

VIEWS IN SQL

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

LAB EXPERIMENTS

PART A: SQL PROGRAMMING

A. Consider the following schema for a Library Database:

BOOK (Book_id, Title, Publisher_Name, Pub_Year)

BOOK_AUTHORS (Book_id, Author_Name)

PUBLISHER (Name, Address, Phone)

BOOK_COPIES (Book_id, Branch_id, No-of_Copies)

BOOK_LENDING (Book_id, Branch_id, Card_No, Date_Out, Due_Date)

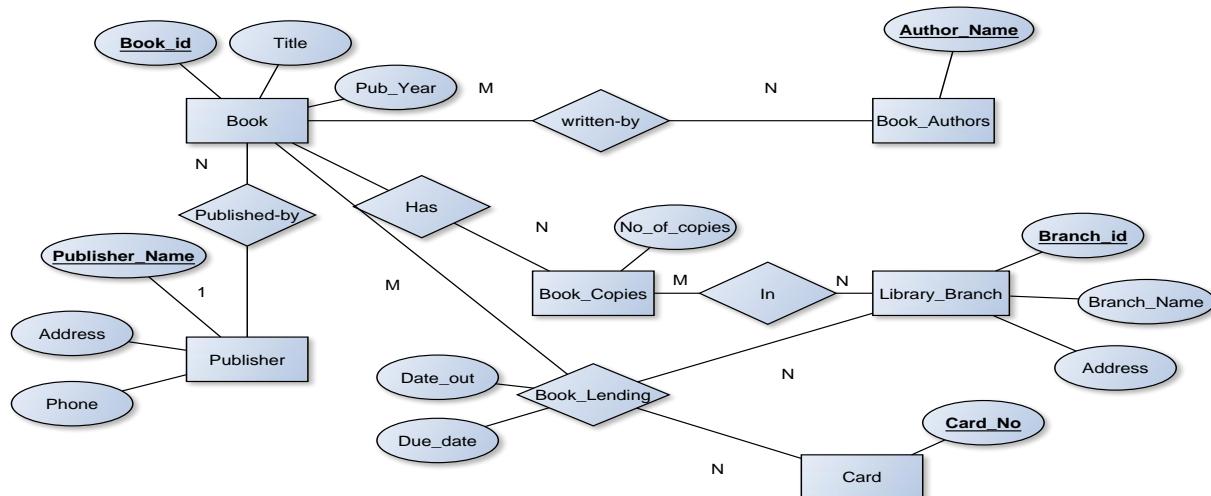
LIBRARY_BRANCH (Branch_id, Branch_Name, Address)

Write SQL queries to

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

Solution:

Entity-Relationship Diagram



Schema Diagram

Book

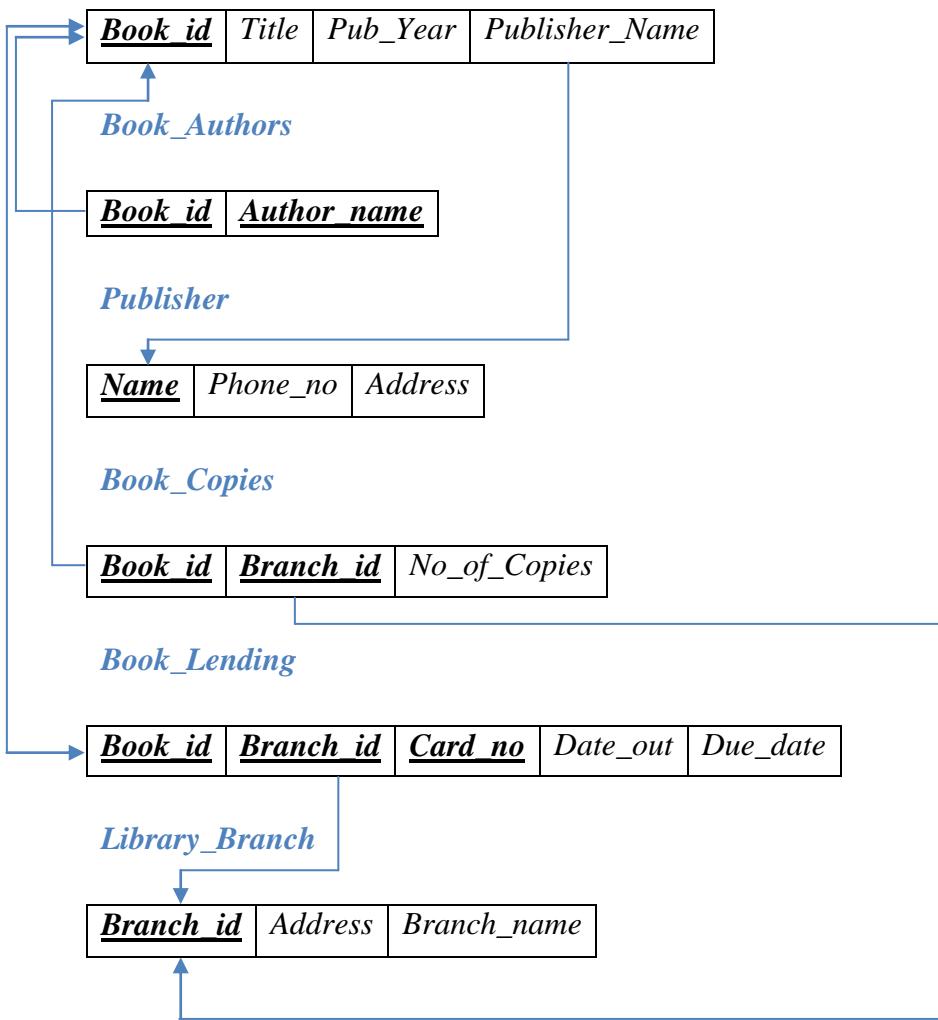


Table Creation

```

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

```

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

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

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

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

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

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

Table Descriptions

```
DESC PUBLISHER;
```

```
SQL> desc publisher;
```

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

```
DESC BOOK;
```

SQL> DESC BOOK;

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

DESC BOOK_AUTHORS;

SQL> DESC BOOK_AUTHORS;

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

DESC LIBRARY_BRANCH;

SQL> DESC LIBRARY_BRANCH;

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

DESC BOOK_COPIES;

SQL> DESC BOOK_COPIES;

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

DESC CARD;

SQL> DESC CARD;

| Name | Null? | Type |
|---------|----------|------------|
| CARD_NO | NOT NULL | NUMBER(38) |

DESC BOOK_LENDING;

SQL> desc book_lending;

| Name | Null? | Type |
|-----------|-------|------|
| DATE_OUT | | |
| DUE_DATE | | |
| BOOK_ID | | |
| BRANCH_ID | | |
| CARD_NO | | |

Insertion of Values to Tables

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

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

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

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

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

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

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

SQL> select * from publisher;

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

SELECT * FROM BOOK;

SQL> SELECT * FROM BOOK;

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

SELECT * FROM BOOK_AUTHORS;

SQL> SELECT * FROM BOOK_AUTHORS;

| AUTHOR_NAME | BOOK_ID |
|--------------|---------|
| NAVATHE | 1 |
| NAVATHE | 2 |
| TANENBAUM | 3 |
| EDWARD ANGEL | 4 |
| GALVIN | 5 |

SELECT * FROM LIBRARY_BRANCH;

SQL> SELECT * FROM LIBRARY_BRANCH;

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

SELECT * FROM BOOK_COPIES;

SQL> SELECT * FROM BOOK_COPIES;

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

SELECT * FROM CARD;

SQL> SELECT * FROM CARD;

CARD_NO

| CARD_NO |
|---------|
| 100 |
| 101 |
| 102 |
| 103 |
| 104 |

SELECT * FROM BOOK_LENDING;

SQL> select * from book_lending;

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

Queries:

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

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

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

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

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

| CARD_NO |
|---------|
| 101 |

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

```
DELETE FROM BOOK
WHERE BOOK_ID=3;
SQL> DELETE FROM BOOK
  2 WHERE BOOK_ID=3;
1 row deleted.
```

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

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

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

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

PUB_YEAR

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

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

CREATE VIEW V_BOOKS AS

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

| BOOK_ID | TITLE | NO_OF_COPIES |
|----------------|--------------|---------------------|
| 1 | DBMS | 10 |
| 1 | DBMS | 5 |
| 2 | ADBMS | 2 |
| 2 | ADBMS | 5 |
| 3 | CN | 7 |
| 5 | OS | 1 |
| 4 | CG | 3 |

B. Consider the following schema for Order Database:

SALESMAN (Salesman_id, Name, City, Commission)

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

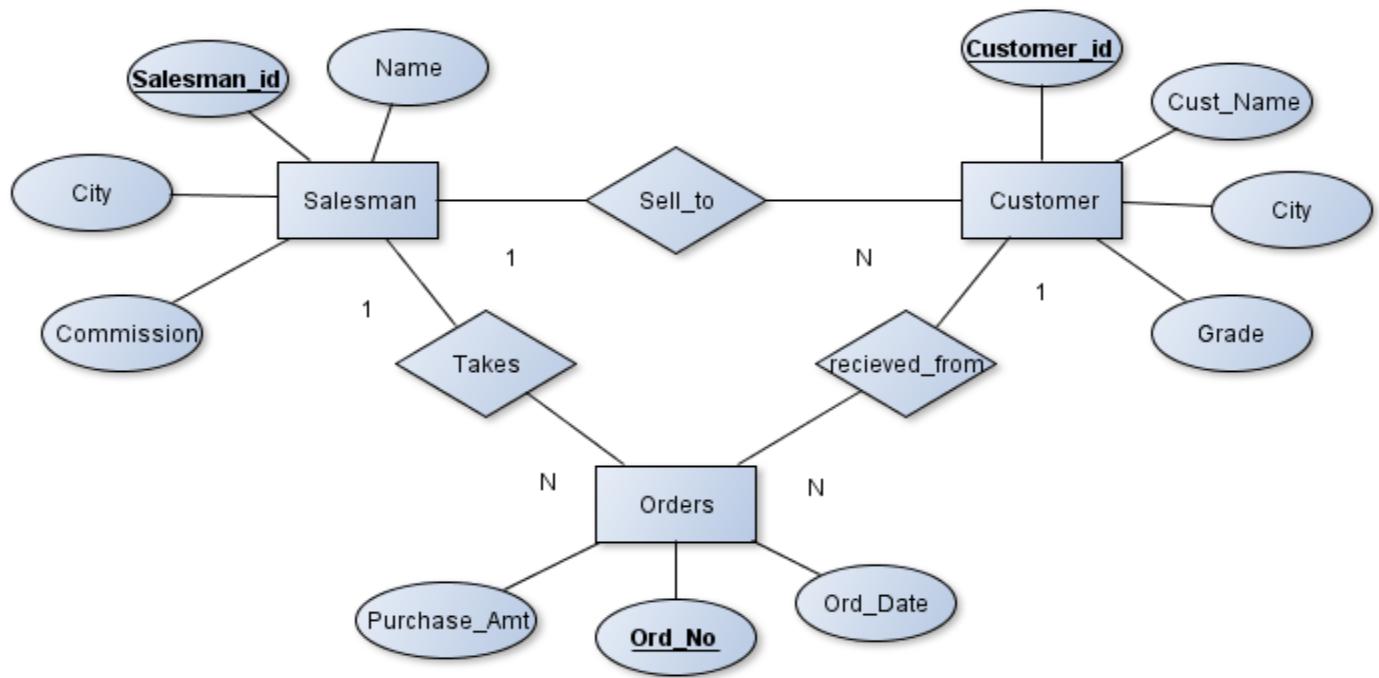
ORDERS (Ord_No, Purchase_Amt, Ord_Date, Customer_id, Salesman_id)

Write SQL queries to

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

Solution:

Entity-Relationship Diagram



Schema Diagram

Salesman

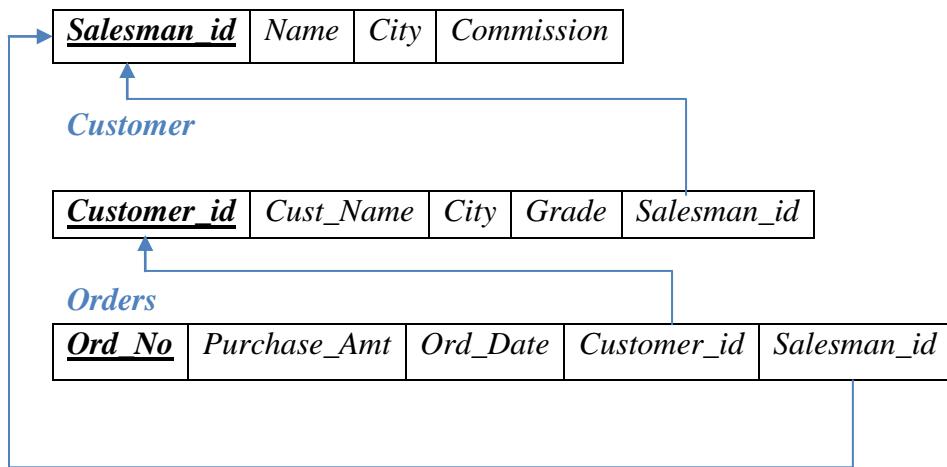


Table Creation

```

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

```

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

```

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

Table Descriptions

DESC SALESMAN;

```
SQL> DESC SALESMAN;
  Name          Null?    Type
-----  -----
SALESMAN_ID      NOT NULL NUMBER(4)
NAME            VARCHAR2(15)
CITY             VARCHAR2(15)
COMMISSION       NUMBER(3,2)
```

DESC CUSTOMER;

```
SQL> DESC CUSTOMER1;
  Name          Null?    Type
-----  -----
CUSTOMER_ID      NOT NULL NUMBER(4)
CUST_NAME        VARCHAR2(15)
CITY              VARCHAR2(15)
GRADE             NUMBER(3)
SALESMAN_ID       NUMBER(4)
```

DESC ORDERS;

```
SQL> DESC ORDERS;
  Name          Null?    Type
-----  -----
ORD_NO           NOT NULL NUMBER(5)
PURCHASE_AMT    NUMBER(10,2)
ORD_DATE         DATE
CUSTOMER_ID      NUMBER(4)
SALESMAN_ID      NUMBER(4)
```

Insertion of Values to Tables

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

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

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

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

SELECT * FROM SALESMAN;

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

SELECT * FROM CUSTOMER1;

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

SELECT * FROM ORDERS;

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

Queries:

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

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

| GRADE | COUNT(DISTINCT CUSTOMER_ID) |
|-------|-----------------------------|
| 300 | 1 |
| 400 | 2 |

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

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

| <u>SALESMAN_ID</u> | <u>NAME</u> |
|--------------------|-------------|
| 1000 | JOHN |
| 2000 | RAVI |

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

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

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

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

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

```

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

```

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

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

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

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

```

DELETE FROM SALESMAN
WHERE SALESMAN_ID=1000;
OR

```

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

```

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

```

```

SQL> DELETE FROM SALESMAN
  2  WHERE SALESMAN_ID=1000;

1 row deleted.

```

```

SQL> SELECT * FROM SALESMAN;

```

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

C. Consider the schema for Movie Database:

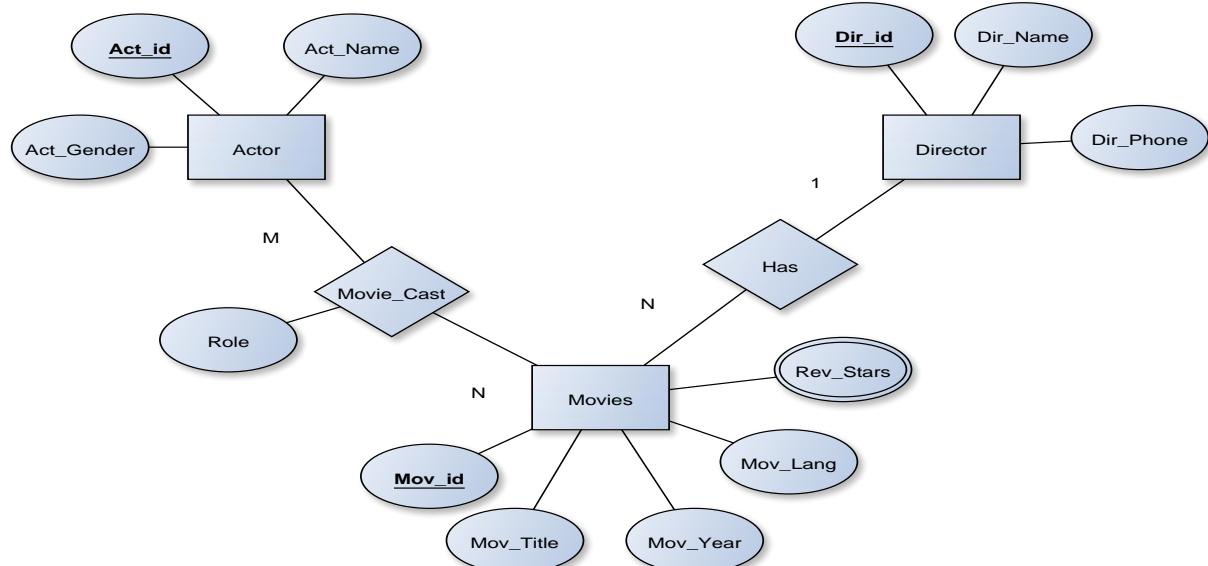
ACTOR (Act_id, Act_Name, Act_Gender)
DIRECTOR (Dir_id, Dir_Name, Dir_Phone)
MOVIES (Mov_id, Mov_Title, Mov_Year, Mov_Lang, Dir_id)
MOVIE_CAST (Act_id, Mov_id, Role)
RATING (Mov_id, Rev_Stars)

Write SQL queries to

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

Solution:

Entity-Relationship Diagram



Schema Diagram

Actor

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

Director

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

Movies

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

Movie_Cast

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

Rating

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

Table Creation

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

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

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

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

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

Table Descriptions

DESC ACTOR;

```
SQL> DESC ACTOR;
  Name          Null?    Type
  ---           ---     ---
ACT_ID        NOT NULL NUMBER(3)
ACT_NAME      VARCHAR2(20)
ACT_GENDER    CHAR(1)
```

DESC DIRECTOR;

```
SQL> DESC DIRECTOR;
  Name          Null?    Type
  ---           ---     ---
DIR_ID        NOT NULL NUMBER(3)
DIR_NAME      VARCHAR2(20)
DIR_PHONE    NUMBER(10)
```

DESC MOVIES;

```
SQL> DESC MOVIES;
  Name          Null?    Type
  ---           ---     ---
MOV_ID        NOT NULL NUMBER(4)
MOV_TITLE    VARCHAR2(25)
MOV_YEAR      NUMBER(4)
MOV_LANG      VARCHAR2(12)
DIR_ID        NUMBER(3)
```

DESC MOVIE_CAST;

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

DESC RATING;

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

Insertion of Values to Tables

INSERT INTO ACTOR VALUES (301,'ANUSHKA','F');

INSERT INTO ACTOR VALUES (302,'PRABHAS','M');

INSERT INTO ACTOR VALUES (303,'PUNITH','M');

INSERT INTO ACTOR VALUES (304,'JERMY','M');

INSERT INTO DIRECTOR VALUES (60,'RAJAMOULI', 8751611001);

INSERT INTO DIRECTOR VALUES (61,'HITCHCOCK', 7766138911);

INSERT INTO DIRECTOR VALUES (62,'FARAN', 9986776531);

INSERT INTO DIRECTOR VALUES (63,'STEVEN SPIELBERG', 8989776530);

INSERT INTO MOVIES VALUES (1001,'BAHUBALI-2', 2017, 'TELAGU', 60);

INSERT INTO MOVIES VALUES (1002,'BAHUBALI-1', 2015, 'TELAGU', 60);

INSERT INTO MOVIES VALUES (1003,'AKASH', 2008, 'KANNADA', 61);

INSERT INTO MOVIES VALUES (1004,'WAR HORSE', 2011, 'ENGLISH', 63);

INSERT INTO MOVIE_CAST VALUES (301, 1002, 'HEROINE');

INSERT INTO MOVIE_CAST VALUES (301, 1001, 'HEROINE');

INSERT INTO MOVIE_CAST VALUES (303, 1003, 'HERO');

INSERT INTO MOVIE_CAST VALUES (303, 1002, 'GUEST');

INSERT INTO MOVIE_CAST VALUES (304, 1004, 'HERO');

INSERT INTO RATING VALUES (1001, 4);

INSERT INTO RATING VALUES (1002, 2);

INSERT INTO RATING VALUES (1003, 5);

INSERT INTO RATING VALUES (1004, 4);

SELECT * FROM ACTOR;

SQL> SELECT * FROM ACTOR;

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

SELECT * FROM DIRECTOR;

SQL> SELECT * FROM DIRECTOR;

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

SELECT * FROM MOVIES;

SQL> SELECT * FROM MOVIES;

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

SELECT * FROM MOVIE_CAST;

SQL> SELECT * FROM MOVIE_CAST;

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

```
SELECT * FROM RATING;
```

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

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

Queries:

1. List the titles of all movies directed by ‘Hitchcock’.

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

| MOV_TITLE |
|-----------|
| AKASH |

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

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

| MOV_TITLE |
|------------|
| BAHUBALI-1 |

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

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

```

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

```

```

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

```

OR

```

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

```

| ACT_NAME | MOV_TITLE | MOV_YEAR |
|----------|------------|----------|
| ANUSHKA | BAHUBALI-2 | 2017 |

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

```

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

```

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

- Update rating of all movies directed by ‘Steven Spielberg’ to 5

KL

```

UPDATE RATING
SET REV_STARS=5

```

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

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

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

D. Consider the schema for College Database:

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

SEMSEC (SSID, Sem, Sec)

CLASS (USN, SSID)

SUBJECT (Subcode, Title, Sem, Credits)

IAMARKS (USN, Subcode, SSID, Test1, Test2, Test3, 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 subjects.
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’

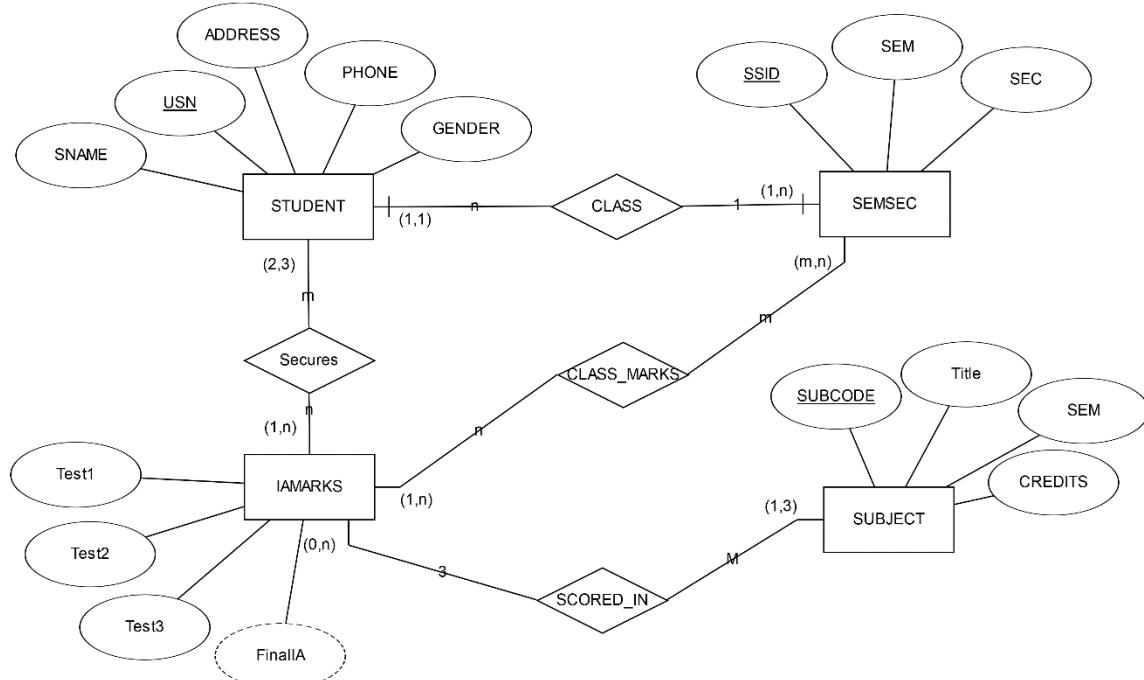
If FinalIA = 12 to 16 then CAT = ‘Average’

If FinalIA < 12 then CAT = ‘Weak’

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

Solution:

Entity - Relationship Diagram



Schema Diagram

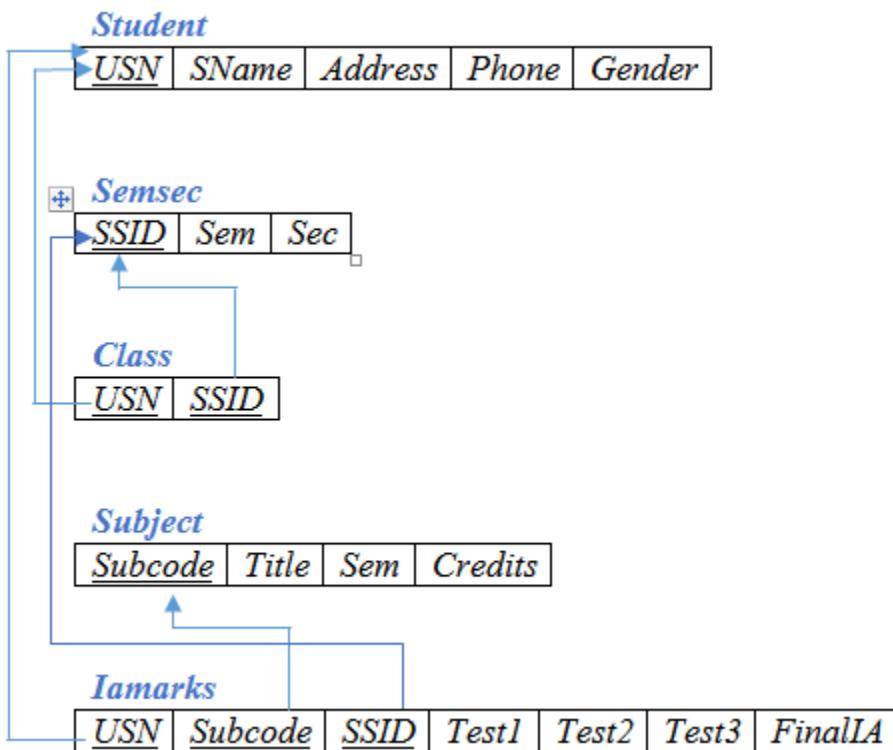


Table Creation

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

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

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

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

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

Table Descriptions

```
DESC STUDENT;
```

| Name |
|---------|
| USN |
| SNAME |
| ADDRESS |
| PHONE |
| GENDER |

```
DESC SEMSEC;
```

```
SQL> DESC SEMSEC;
Name
-----
SSID
SEM
SEC
```

DESC CLASS;

```
SQL> DESC CLASS;
  Name
```

```
-----  
USN  
SSID
```

DESC SUBJECT;

```
SQL> DESC SUBJECT1;
  Name
```

```
-----  
SUBCODE  
TITLE  
SEM  
CREDITS
```

DESC IAMARKS;

```
SQL> DESC IAMARKS;
  Name
```

```
-----  
USN  
SUBCODE  
SSID  
TEST1  
TEST2  
TEST3  
FINALIA
```

Insertion of values to tables

```
INSERT INTO STUDENT VALUES ('1RN13CS020','AKSHAY','BELAGAVI',  
8877881122,'M');
```

```
INSERT INTO STUDENT VALUES ('1RN13CS062','SANDHYA','BENGALURU',  
7722829912,'F');
```

```
INSERT INTO STUDENT VALUES ('1RN13CS091','TEESHA','BENGALURU',  
7712312312,'F');
```

```
INSERT INTO STUDENT VALUES ('1RN13CS066','SUPRIYA','MANGALURU',  
8877881122,'F');
```

```
INSERT INTO STUDENTVALUES ('1RN14CS010','ABHAY','BENGALURU',  
9900211201,'M');
```

```
INSERT INTO STUDENT VALUES ('1RN14CS032','BHASKAR','BENGALURU',  
9923211099,'M');
```

```
INSERT INTO STUDENTVALUES ('1RN14CS025','ASMI','BENGALURU', 7894737377,'F');
```

```
INSERT INTO STUDENT VALUES ('1RN15CS011','AJAY','TUMKUR', 9845091341,'M');
```

INSERT INTO STUDENT VALUES ('1RN15CS029','CHITRA','DAVANGERE',
7696772121,'F');

INSERT INTO STUDENT VALUES ('1RN15CS045','JEEVA','BELLARY', 9944850121,'M');

INSERT INTO STUDENT VALUES ('1RN15CS091','SANTOSH','MANGALURU',
8812332201,'M');

INSERT INTO STUDENT VALUES ('1RN16CS045','ISMAIL','KALBURGI',
9900232201,'M');

INSERT INTO STUDENT VALUES ('1RN16CS088','SAMEERA','SHIMOGA',
9905542212,'F');

INSERT INTO STUDENT VALUES ('1RN16CS122','VINAYAKA','CHIKAMAGALUR',
8800880011,'M');

INSERT INTO SEMSEC VALUES ('CSE8A', 8,'A');

INSERT INTO SEMSEC VALUES ('CSE8B', 8,'B');

INSERT INTO SEMSEC VALUES ('CSE8C', 8,'C');

INSERT INTO SEMSEC VALUES ('CSE7A', 7,'A');

INSERT INTO SEMSEC VALUES ('CSE7B', 7,'B');

INSERT INTO SEMSEC VALUES ('CSE7C', 7,'C');

INSERT INTO SEMSEC VALUES ('CSE6A', 6,'A');

INSERT INTO SEMSEC VALUES ('CSE6B', 6,'B');

INSERT INTO SEMSEC VALUES ('CSE6C', 6,'C');

INSERT INTO SEMSEC VALUES ('CSE5A', 5,'A');

INSERT INTO SEMSEC VALUES ('CSE5B', 5,'B');

INSERT INTO SEMSEC VALUES ('CSE5C', 5,'C');

INSERT INTO SEMSEC VALUES ('CSE4A', 4,'A');

INSERT INTO SEMSEC VALUES ('CSE4B', 4,'B');

INSERT INTO SEMSEC VALUES ('CSE4C', 4,'C');

INSERT INTO SEMSEC VALUES ('CSE3A', 3,'A');

INSERT INTO SEMSEC VALUES ('CSE3B', 3,'B');

INSERT INTO SEMSEC VALUES ('CSE3C', 3,'C');

INSERT INTO SEMSEC VALUES ('CSE2A', 2,'A');

INSERT INTO SEMSEC VALUES ('CSE2B', 2,'B');

INSERT INTO SEMSEC VALUES ('CSE2C', 2,'C');

INSERT INTO SEMSEC VALUES ('CSE1A', 1,'A');

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

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

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

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

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

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

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

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

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

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

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

SELECT * FROM STUDENT;

SQL> SELECT * FROM STUDENT1;

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

SELECT * FROM SEMSEC;

SQL> SELECT * FROM SEMSEC;

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

SELECT * FROM CLASS;

SQL> SELECT * FROM CLASS;

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

14 rows selected.

SELECT * FROM SUBJECT;

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

SELECT * FROM IAMARKS;

SQL> SELECT * FROM IAMARKS;

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

Queries:

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

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

```
SELECT S.*
FROM STUDENT S
JOIN CLASS C ON S.USN = C.USN
JOIN SEMSEC SS ON C.SSID = SS.SSID
WHERE SS.SEM = 4
AND SS.SEC = 'C';
```

SS.SEC='C';

| USN | SNAME | ADDRESS | PHONE | G | SEM | S |
|------------|---------|-----------|------------|---|-----|---|
| 1RN15CS091 | SANTOSH | MANGALURU | 8812332201 | M | 4 | C |

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

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

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

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

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

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

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

```

CREATE OR REPLACE PROCEDURE AVGMARKS
IS
CURSOR C_IAMARKS IS
SELECT GREATEST(TEST1,TEST2) AS A, GREATEST(TEST1,TEST3) AS B,
GREATEST(TEST3,TEST2) AS C
FROM IAMARKS
WHERE FINALIA IS NULL
FOR UPDATE;
C_A NUMBER;
C_B NUMBER;
C_C NUMBER;
C_SM NUMBER;
C_AV NUMBER;
UPDATE IAMARKS
SET FINALIA = (SELECT ROUND((TEST1 + TEST2 + TEST3 - LEAST(TEST1, TEST2, TEST3)) / 2, 2)
               FROM (SELECT TEST1, TEST2, TEST3
                     FROM IAMARKS
                     WHERE USN = IAMARKS.USN
                     ORDER BY TEST1 DESC, TEST2 DESC, TEST3 DESC
                     FETCH FIRST 2 ROWS ONLY) AS BestTwo)
BEGIN
OPEN C_IAMARKS;
LOOP
FETCH C_IAMARKS INTO C_A, C_B, C_C;
EXIT WHEN C_IAMARKS%NOTFOUND;
--DBMS_OUTPUT.PUT_LINE(C_A || ' ' || C_B || ' ' || C_C);
IF (C_A != C_B) THEN
C_SM:=C_A+C_B;
ELSE
C_SM:=C_A+C_C;
END IF;

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

END LOOP;
CLOSE C_IAMARKS;
END;
/

```

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

SELECT * FROM IAMARKS;

SQL> SELECT * FROM IAMARKS;

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

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

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

SQL> select * from IAMARKS;

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

5. Categorize students based on the following criterion:

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

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

If FinalIA < 12 then CAT = 'Weak'

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

```
SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER,
(CASE
    WHEN IA.FINALIA BETWEEN 17 AND 20 THEN 'OUTSTANDING'
    WHEN IA.FINALIA BETWEEN 12 AND 16 THEN 'AVERAGE'
    ELSE 'WEAK'
END) AS CAT
FROM STUDENT S, SEMSEC SS, IAMARKS IA, SUBJECT SUB
WHERE S.USN = IA.USN AND
SS.SSID = IA.SSID AND
SUB.SUBCODE = IA.SUBCODE AND
SUB.SEM = 8;
SELECT S.USN, S.SNAME, S.GENDER, SS.SEM , SS.SEC ,IAM.FINALIA,
CASE
    WHEN IAM.FINALIA BETWEEN 17 AND 20 THEN 'Outstanding'
    WHEN IAM.FINALIA BETWEEN 12 AND 16 THEN 'Average'
    ELSE 'Weak'
END AS CAT
FROM STUDENT S
JOIN CLASS C ON S.USN = C.USN
JOIN SEMSEC SS ON C.SSID = SS.SSID
JOIN IAMARKS IAM ON S.USN = IAM.USN
WHERE SS.SEM = 8
AND SS.SEC IN ('A', 'B', 'C');
```

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

E. Consider the schema for Company Database:

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

DEPARTMENT (DNo, DName, MgrSSN, MgrStartDate)

DLOCATION (DNo,DLoc)

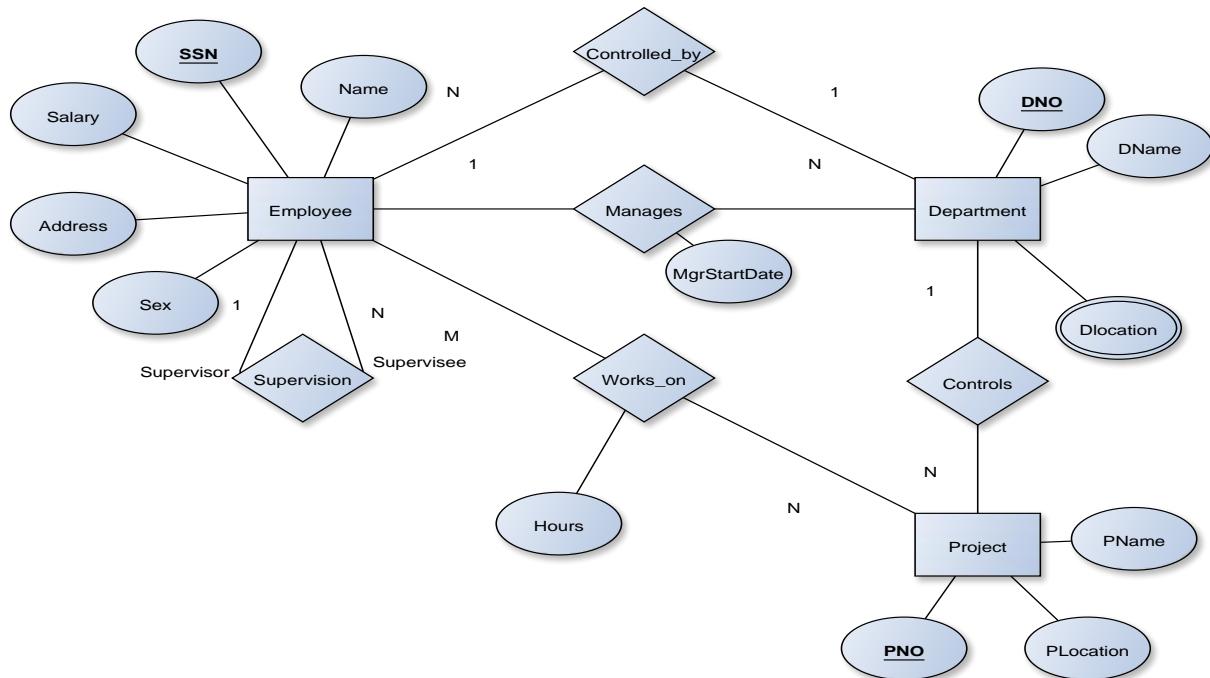
PROJECT (PNo, PName, PLocation, DNo)

WORKS_ON (SSN, PNo, Hours)

Write SQL queries to

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

Entity-Relationship Diagram



Schema Diagram

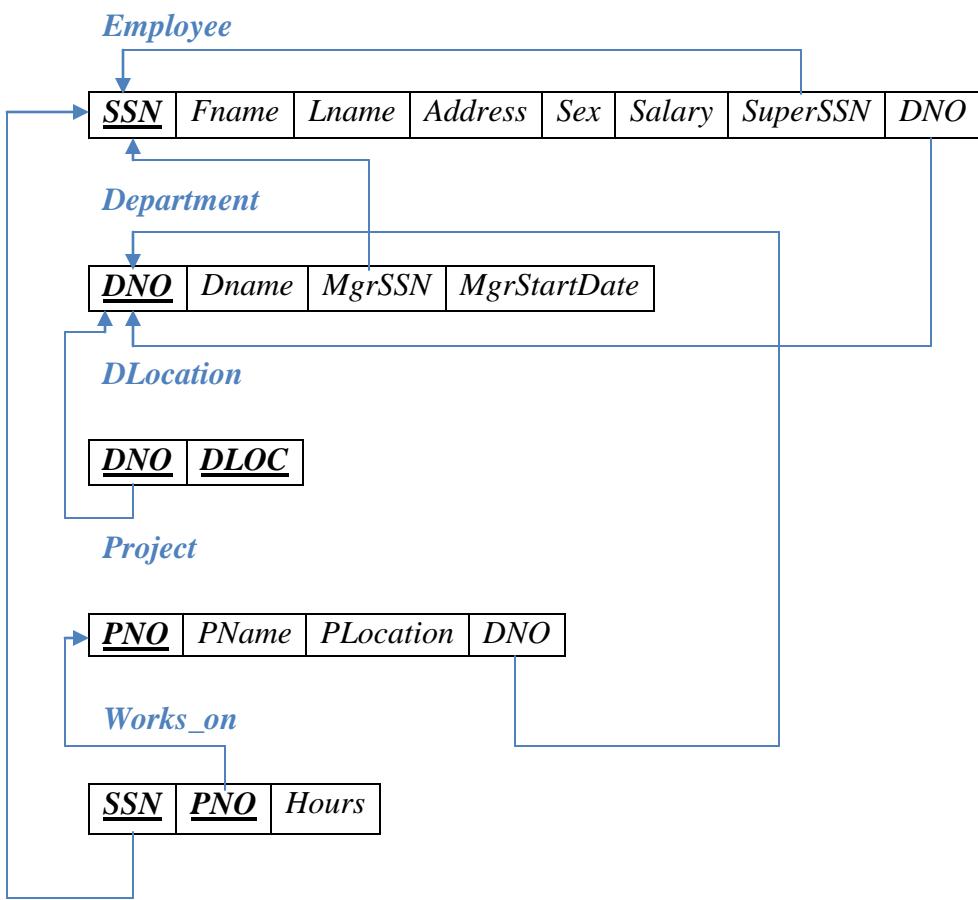


Table Creation

```

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

```

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

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

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

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

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

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

Table Descriptions

```
DESC EMPLOYEE;
```

```
SQL> DESC EMPLOYEE;
```

```
Name
```

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

DESC DEPARTMENT;

SQL> DESC DEPARTMENT;

Name

DNO
DNAME
MGRSTARTDATE
MGRSSN

DESC DLOCATION;

SQL> DESC DLOCATION;

Name

DLOC
DNO

DESC PROJECT;

SQL> DESC PROJECT;

Name

PNO
PNAME
PLOCATION
DNO

DESC WORKS_ON;

SQL> DESC WORKS_ON;

Name

HOURS
SSN
PNO

Insertion of values to tables

```
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES ('RNSECE01', 'JOHN', 'SCOTT', 'BANGALORE', 'M', 450000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES ('RNSCSE01', 'JAMES', 'SMITH', 'BANGALORE', 'M', 500000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES ('RNSCSE02', 'HEARN', 'BAKER', 'BANGALORE', 'M', 700000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES ('RNSCSE03', 'EDWARD', 'SCOTT', 'MYSORE', 'M', 500000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES ('RNSCSE04', 'PAVAN', 'HEGDE', 'MANGALORE', 'M', 650000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY) VALUES ('RNSCSE05', 'GIRISH', 'MALYA', 'MYSORE', 'M', 450000);
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

```

SELECT * FROM EMPLOYEE;

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

SELECT * FROM DEPARTMENT;

SQL> SELECT * FROM DEPARTMENT;

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

SELECT * FROM DLOCATION;

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

SELECT * FROM PROJECT;

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

SELECT * FROM WORKS_ON;

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

Queries:

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

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

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

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

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

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

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

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

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

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

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

```

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

```

Or

```

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

```

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

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

```

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

```

GROUP BY D.DNO;

| DNO | COUNT(*) |
|-----|----------|
| 5 | 3 |
