Vision of the Institute

"To be a leader in imparting value based Technical Education and Research for the benefit of society".

Mission of the Institute

M1	To Provide State-of-the-art Infrastructure Facilities
M2	To Implement modern Pedagogical Methods in delivering the Academic Programs with Experienced and Committed Faculty
M3	To Create a Vibrant Ambience that promotes Learning, Research, Invention and Innovation
M4	To Undertake Skill Development Programmes for Academic Institutions and Industries
M5	To Enhance Industry Institute Interaction through Collaborative Research and Consultancy
M6	To Relentlessly Pursue Professional Excellence with Ethical and Moral Values

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION

"Be a premier department in the field of Computer Science & Engineering to meet the technological challenges of the society"

Mission of the Department

MD 1	To provide state of the art infrastructure facilities
MD 2	To provide exposure to the latest tools in the area of computer hardware and software
MD 3	To strive for academic excellence through research in Computer Science and Engineering with creative teaching-learning pedagogy
MD 4	To establish Industry Institute Interaction and make students ready for the Industrial environment
MD 5	To transform students into entrepreneurial, technically competent, socially responsible and ethical computer science professional

PROGRAMME EDUCATIONAL OBJECTIVES

PEO 1	Graduates possess advanced knowledge of Computer Science & Engineering and excel in leadership roles to serve the society
PEO 2	Graduates of the program will apply Computer Engineering tools in core technologies for improving knowledge in the Interdisciplinary Research and/or Entrepreneurs
PEO 3	Graduates adapt Value-Based Proficiency in solving real time problems.

PROGRAMME SPECIFIC OUTCOMES

PSO 1	Professional Skills: Ability of using mathematical methodologies for analysis of computing concepts, data structure, computer hardware, layered technologies and suitable algorithm which in turn helps graduates to model, design and implement a system to meet specific requirement
PSO 2	Software Skills: Ability to build Software Engineering System for lifecycle development by using analytical knowledge in Computer Science & Engineering and applying modern methodologies

PROGRAM OUTCOMES (POS)

PO1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**Problem analysis**: Identify, formulate, review research literature, and 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 the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions
PO5	**Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and 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 team work**: 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 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.

COURSE DETAILS

Course Name: DBMS Lab with Mini Project

Subject Code: 17CSL58

COURSE OBJECTIVES

This course will enable students to

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

SYLLABUS

DBMS LABORATORY WITH MINI PROJECT
[As per Choice Based Credit System (CBCS) scheme]

Subject Code : 18CSL58 **I.A. Marks : 40**
Hours/Week : 01I + 02P **Exam Hours : 03**
Total Hours : 40 **Exam Marks : 100**

Description (If any):
PART-A: SQL Programming (Max. Exam Mks. 60)

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

PART-B: Mini Project (Max. Exam Mks. 40)

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

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.

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 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.

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.

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.

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

Part B: Mini project
- For any problem selected, write the ER Diagram, apply ER-mapping rules, normalize the relations, and follow the application development process.
- Make sure that the application should have five or more tables, at least one trigger and one stored procedure, using suitable frontend tool.
- Indicative areas include; health care, education, industry, transport, supply chain, etc.

COURSE OUTCOMES

At the end of the Course, the students will be able to:

C358.1	Design a database schema for a given problem-domain
C358.2	Create, populate , query and maintain tables of a database using PL/SQL
C358.3	Demonstrate the working of different concepts of DBMS
C358.4	Develop database applications using front-end tools and back-end DBMS.
C358.5	Work with other people in a team, communicating ideas effectively in speech and in writing

CO-PO Mapping

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
C5 3 1		3	3										2	

C5.3.2	1				2							1	
C5.3.3	3	2			2							1	
C5.3.4			3		3							2	3
C5.3.5									3	3	1		1
Avg.	**2**	**2.5**	**3**		**2.3**				**3**	**3**	**1**	**1.5**	**2**

Conduction of Practical Examination:

1. All laboratory experiments from part A are to be included for practical examination.
2. Mini project has to be evaluated for 40 Marks
3. Report should be prepared in a standard format prescribed for project work.
4. Students are allowed to pick one experiment from the lot.
5. Strictly follow the instructions as printed on the cover page of answer script.
6. Marks distribution:

a) Part A: Procedure + Conduction + Viva:09 + 42 +09 =60 Marks

b) Part B: Demonstration + Report + Viva voce = 20+14+06 = 40 Marks

7. Change of experiment is allowed only once and marks allotted to the procedure part to be made zero.

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 *A*1> <Data Type *D*1> [< Constarints>],
  <Attribute *A*2> <Data Type *D*2> [< Constarints>],
  ……

<Attribute *A*n> <Data Type *D*n> [< Constarints>],
[<integrity-constraint1>, <integrity-constraint k>]);
- A constraint NOT NULL may be specified on an attribute
A constraint NOT NULL may be specified on an attribute
Ex: CREATE TABLE DEPARTMENT (DNAME VARCHAR(10) NOT NULL, DNUMBER INTEGER NOT NULL, MGRSSN CHAR(9), MGRSTARTDATE CHAR(9));

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

```
Ex: CREATE TABLE DEPT (
DNAME VARCHAR(10) NOT NULL,
DNUMBER INTEGER NOT NULL,
MGRSSN CHAR(9),
MGRSTARTDATE CHAR(9),
PRIMARY KEY (DNUMBER),
UNIQUE (DNAME),
FOREIGN KEY (MGRSSN) REFERENCES EMP(SSN));
```

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

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

DROP TABLE

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

  Example: DROP TABLE DEPENDENT;

ALTER TABLE:

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

  Example: ALTER TABLE EMPLOYEE ADD JOB VARCHAR2 (12);
- The database users must still enter a value for the new attribute JOB for each EMPLOYEE tuple. This can be done using the UPDATE command.

DROP A COLUMN (AN ATTRIBUTE)

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

- ALTER TABLE COMPANY.DEPARTMENT ALTER MGRSSN SET DEFAULT "333445555";

BASIC QUERIES IN SQL

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

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

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:

Schema Diagram

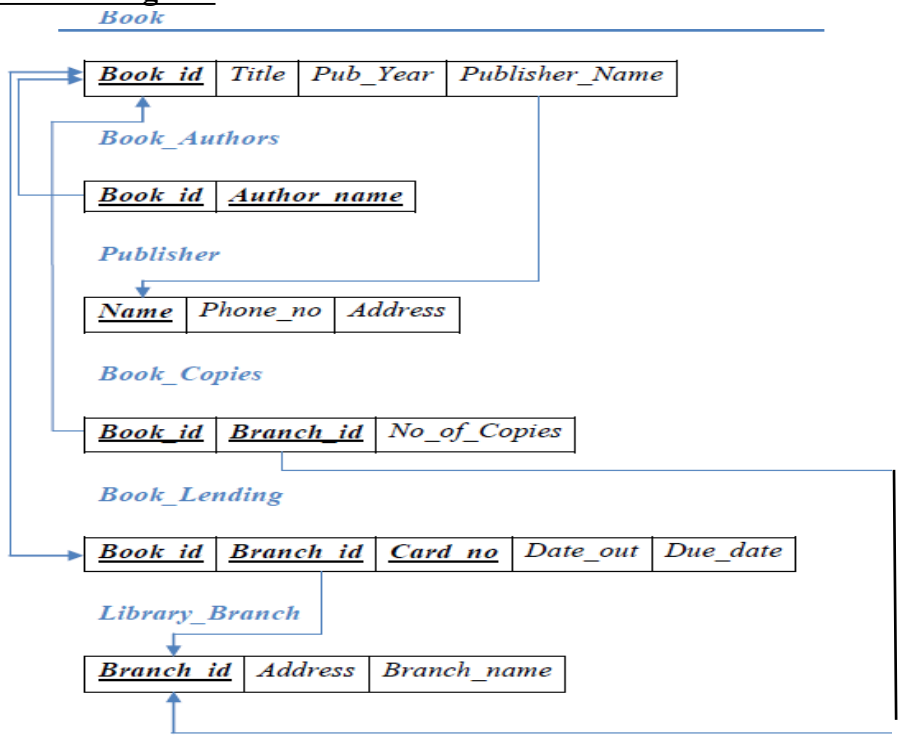

Table Creation

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 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 int,
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
 ----------------------------------------------------------------------------------------------------
 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,'AMCEC','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
----------
       100
       101
       102
       103
       104
```

SELECT * FROM BOOK_LENDING;

```
SQL> select * from book_lending;

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

Queries:

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

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

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

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

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

```
   CARD_NO
----------
       101
```

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

   DELETE FROM BOOK
   WHERE BOOK_ID=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
```

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

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

```
PUB_YEAR
-------------
JAN-2017
JUN-2016
SEP-2016
SEP-2015
MAY-2016
```

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

```
CREATE VIEW V_BOOKS AS
SELECT B.BOOK_ID, B.TITLE, C.NO_OF_COPIES
FROM BOOK B, BOOK_COPIES C, LIBRARY_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:
Schema Diagram

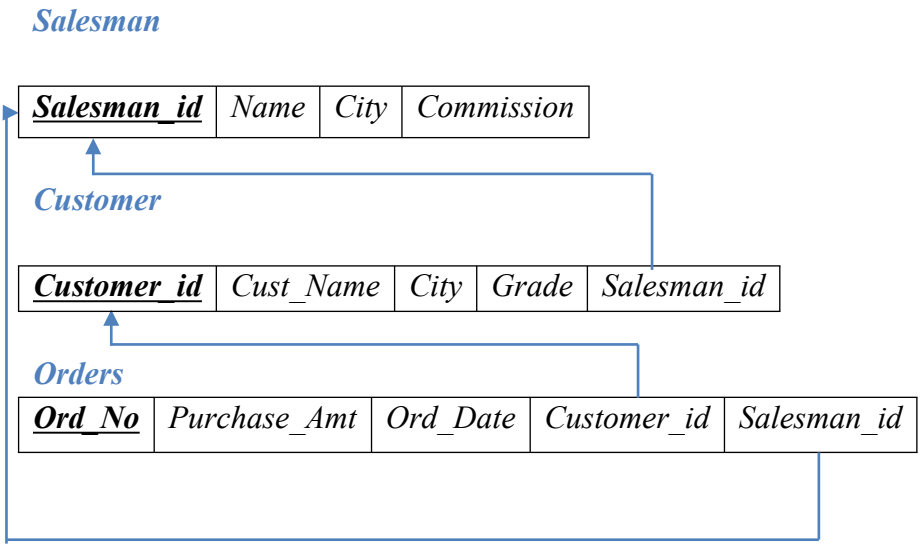

Table Creation

```
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 CUSTOMER1;

```
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:

1. **Count the customers with grades above Bangalore's average.**
   SELECT GRADE, COUNT (DISTINCT CUSTOMER_ID)
   FROM CUSTOMER1
   GROUP BY GRADE

```
HAVING GRADE > (SELECT AVG(GRADE)
FROM CUSTOMER1
WHERE CITY='BANGALORE');
```

```
   GRADE COUNT(DISTINCTCUSTOMER_ID)
-------- --------------------------
     300                          1
     400                          2
```

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

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

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

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

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

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

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

```
CREATE VIEW ELITSALESMAN AS
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;
```

```
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:

Schema Diagram

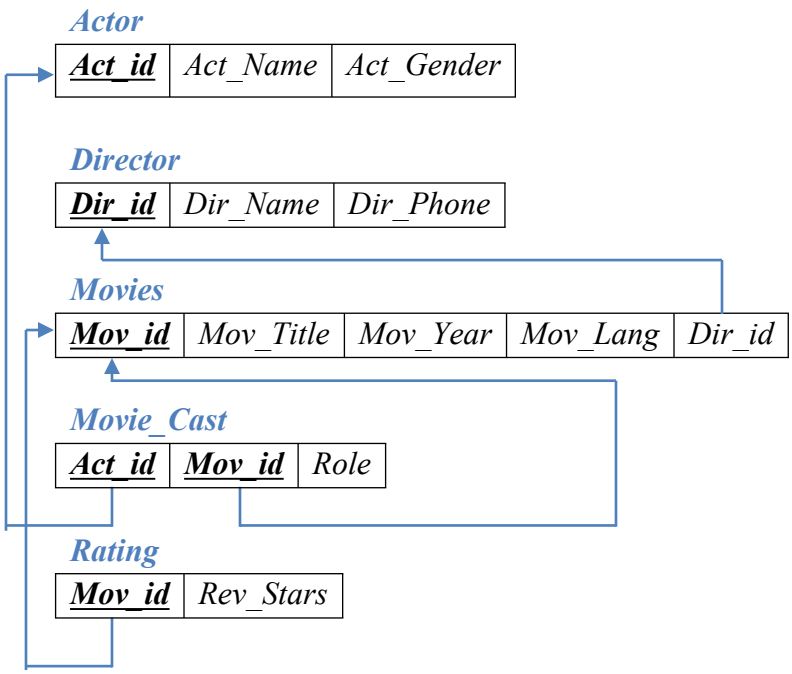

Table Creation

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

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

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

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

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

Table Descriptions

DESC ACTOR;

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

DESC DIRECTOR;

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

DESC MOVIES;

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

DESC MOVIE_CAST;

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

DESC RATING;

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

Insertion of Values to Tables

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

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

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

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

INSERT INTO RATING VALUES (1001, 4);

INSERT INTO RATING VALUES (1002, 2);
INSERT INTO RATING VALUES (1003, 5);
INSERT INTO RATING VALUES (1004, 4);

SELECT * FROM ACTOR;

```
SQL> SELECT * FROM ACTOR;

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

SELECT * FROM DIRECTOR;

```
SQL> SELECT * FROM DIRECTOR;

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

SELECT * FROM MOVIES;

```
SQL> SELECT * FROM MOVIES;

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

SELECT * FROM MOVIE_CAST;

```
SQL> SELECT * FROM MOVIE_CAST;

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

SELECT * FROM RATING;

```
SQL> SELECT * FROM RATING;

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

Queries:

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

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

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

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

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

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

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

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

OR

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

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

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

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

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

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

```
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:

Schema Diagram

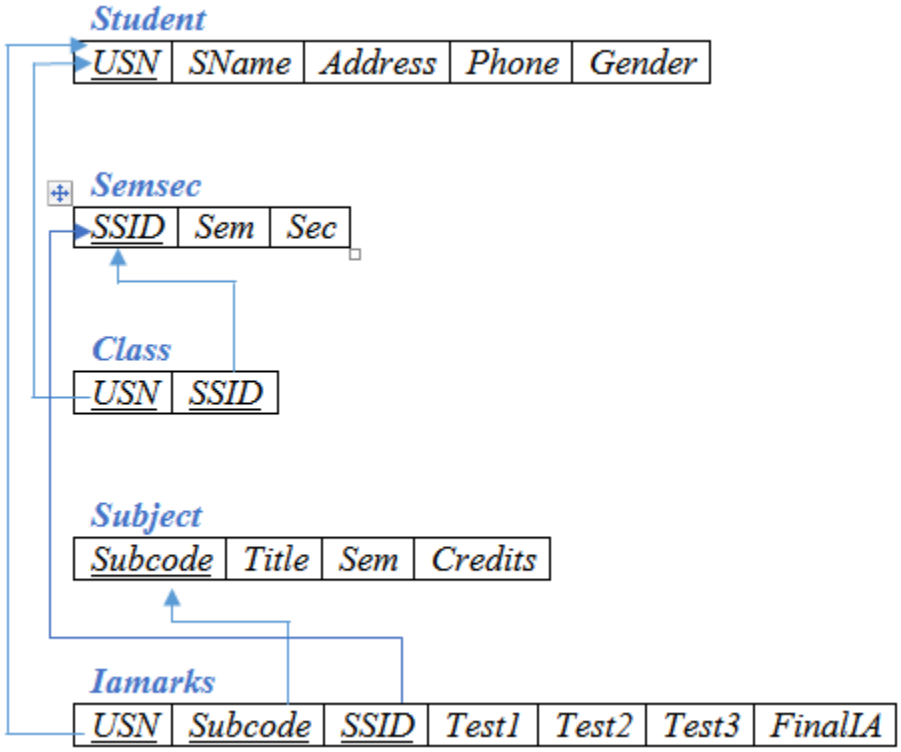

Table Creation

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

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

CREATE TABLE CLASS (
USN VARCHAR (10),
SSID VARCHAR (5),
PRIMARY KEY (USN, SSID),
FOREIGN KEY (USN) REFERENCES STUDENT (USN),
FOREIGN KEY (SSID) REFERENCES SEMSEC (SSID));
CREATE TABLE SUBJECT (
SUBCODE VARCHAR (8),
TITLE VARCHAR (20),
SEM NUMBER (2),
CREDITS NUMBER (2),
PRIMARY KEY (SUBCODE));

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

Table Descriptions

DESC STUDENT;

```
Name
----------------------------------------------------------------------------------------------
USN
SNAME
ADDRESS
PHONE
GENDER
```

DESC SEMSEC;

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

DESC CLASS;

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

DESC SUBJECT;

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

DESC IAMARKS;

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

Insertion of values to tables

INSERT INTO STUDENT VALUES ('1RN13CS020','AKSHAY','BELAGAVI', 8877881122,'M');
INSERT INTO STUDENT VALUES ('1RN13CS062','SANDHYA','BENGALURU', 7722829912,'F');

```
INSERT INTO STUDENT VALUES ('1RN13CS091','TEESHA','BENGALURU',
7712312312,'F');
INSERT INTO STUDENT VALUES ('1RN13CS066','SUPRIYA','MANGALURU',
8877881122,'F');
INSERT INTO STUDENTVALUES ('1RN14CS010','ABHAY','BENGALURU',
9900211201,'M');
INSERT INTO STUDENT VALUES ('1RN14CS032','BHASKAR','BENGALURU',
9923211099,'M');
INSERT INTO STUDENTVALUES ('1RN14CS025','ASMI','BENGALURU', 7894737377,'F');
INSERT INTO STUDENT VALUES ('1RN15CS011','AJAY','TUMKUR', 9845091341,'M');
INSERT INTO STUDENT VALUES ('1RN15CS029','CHITRA','DAVANGERE',
7696772121,'F');
INSERT INTO STUDENT VALUES ('1RN15CS045','JEEVA','BELLARY', 9944850121,'M');
INSERT INTO STUDENT VALUES ('1RN15CS091','SANTOSH','MANGALURU',
8812332201,'M');
INSERT INTO STUDENT VALUES ('1RN16CS045','ISMAIL','KALBURGI',
9900232201,'M');
INSERT INTO STUDENT VALUES ('1RN16CS088','SAMEERA','SHIMOGA',
9905542212,'F');
INSERT INTO STUDENT VALUES ('1RN16CS122','VINAYAKA','CHIKAMAGALUR',
8800880011,'M');

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

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

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

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

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

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

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

INSERT INTO SEMSEC VALUES ('CSE2A', 2,’A’);
INSERT INTO SEMSEC VALUES (‘CSE2B’, 2,'B’);
INSERT INTO SEMSEC VALUES ('CSE2C', 2,'C');
INSERT INTO SEMSEC VALUES (‘CSE1A', 1,'A');
INSERT INTO SEMSEC VALUES (‘CSE1B’, 1,’B’);
INSERT INTO SEMSEC VALUES ('CSE1C', 1,’C’);

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

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

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

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

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

INSERT INTO SUBJECT VALUES ('10CS71','OOAD', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS72','ECS', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS73','PTW', 7, 4);
```

```
INSERT INTO SUBJECT VALUES ('10CS74','DWDM', 7, 4);
INSERT INTO SUBJECT VALUES (‘10CS75','JAVA', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS76','SAN', 7, 4);

INSERT INTO SUBJECT VALUES ('15CS51', 'ME', 5, 4);
INSERT INTO SUBJECT VALUES ('15CS52','CN', 5, 4);
INSERT INTO SUBJECT VALUES ('15CS53','DBMS', 5, 4);
INSERT INTO SUBJECT VALUES ('15CS54','ATC', 5, 4);
INSERT INTO SUBJECT VALUES ('15CS55','JAVA', 5, 3);
INSERT INTO SUBJECT VALUES ('15CS56','AI', 5, 3);
INSERT INTO SUBJECT VALUES ('15CS41','M4', 4, 4);
INSERT INTO SUBJECT VALUES ('15CS42','SE', 4, 4);
INSERT INTO SUBJECT VALUES ('15CS43','DAA', 4, 4);
INSERT INTO SUBJECT VALUES ('15CS44','MPMC', 4, 4);
INSERT INTO SUBJECT VALUES ('15CS45','OOC', 4, 3);
INSERT INTO SUBJECT VALUES ('15CS46','DC', 4, 3);

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

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

SELECT * FROM STUDENT;
```

```
SQL> SELECT * FROM STUDENT1;

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

SELECT * FROM SEMSEC;

```
SQL> SELECT * FROM SEMSEC;

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

SELECT * FROM CLASS;

```
SQL> SELECT * FROM CLASS;

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

14 rows selected.
```

SELECT * FROM SUBJECT;

SUBCODE	TITLE	SEM	CREDITS
10CS81	ACA	8	4
10CS82	SSM	8	4
10CS83	NM	8	4
10CS84	CC	8	4
10CS85	PW	8	4
10CS71	OOAD	7	4
10CS72	ECS	7	4
10CS73	PTW	7	4
10CS74	DWDM	7	4
10CS75	JAVA	7	4
10CS76	SAN	7	4
15CS51	ME	5	4
15CS52	CN	5	4
15CS53	DBMS	5	4
15CS54	ATC	5	4
15CS55	JAVA	5	3
15CS56	AI	5	3
15CS41	M4	4	4
15CS42	SE	4	4
15CS43	DAA	4	4
15CS44	MPMC	4	4
15CS45	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
 SS.SEc='C';
```

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

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

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

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

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

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

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

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

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

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

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

```
        END IF;

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

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

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

SELECT * FROM IAMARKS;

```
SQL> SELECT * FROM IAMARKS;

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

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

```
BEGIN
AVGMARKS;
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;
```

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.**

Schema Diagram

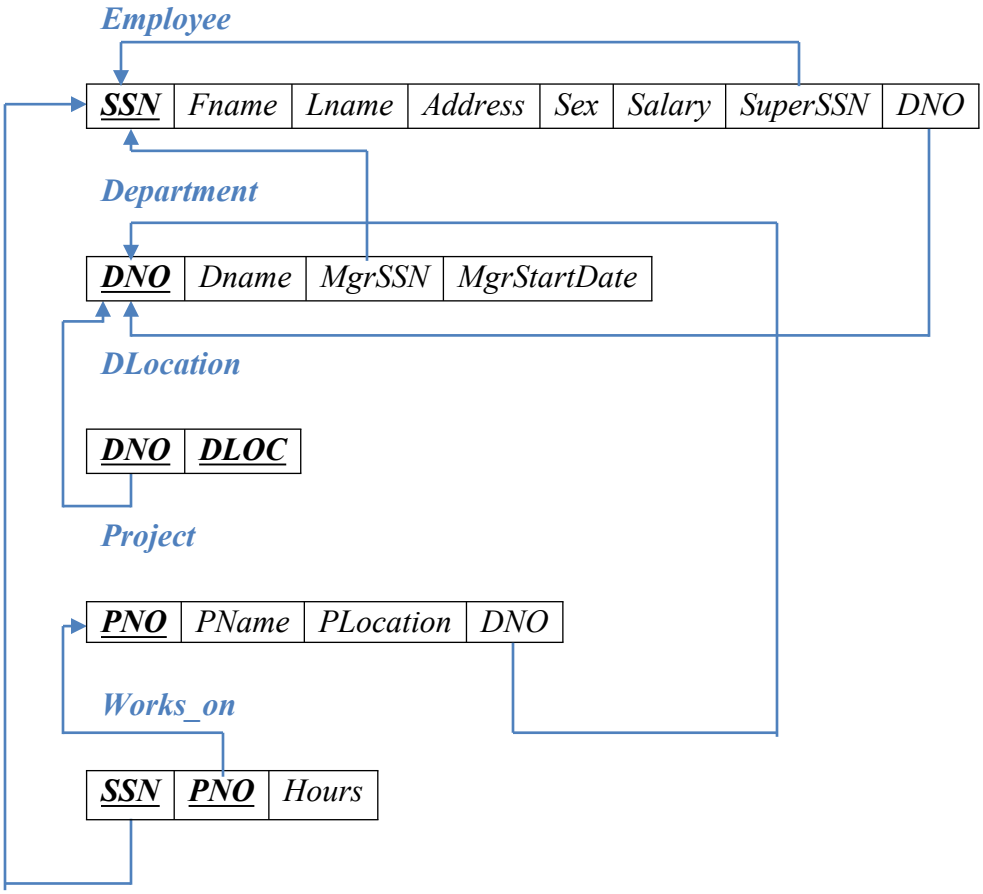

Table Creation

```
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’,’AMCEC01’);
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='AMCEC01';

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, 'AMCEC01', 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')
                    MINUS (SELECT PNO
                     FROM WORKS_ON
                     WHERE E.SSN=SSN));
```

```
FNAME                LNAME
-------------------- --------------------
JAMES                SMITH
```

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
```

Sample list of Mini Projects

- Bus Reservation System
- Hostel Management System
- Art Gallery Management System
- Human Resource Management
- Online Blood Bank
- Online Car Rental
- Online Pizza Ordering System
- Hospital Management System
- Multiplex Theatre Booking System
- Airline Reservation System
- Railway Reservation
- Student Voting System
- Hotel Management System
- Police & Criminal Record Management System
- Library Management System
- A&C Medicare
- Employee Payroll Management System
- Training & Placement Management System
- Online Voting System
- Banking Management System
- College Event Management
- Restaurant Billing System
- Hotel Database Management System
- Railway Management System
- Pharmacy Management System
- ATM Transaction Management System
- Pet Shop Management System
- Online Quiz System
- Inventory Control System
- IT Training Group Database Management System
- Wholesale Management System
- Salary Management System
- Etc..etc…

Most likely Viva Questions

1. What is SQL?

Structured Query Language

2. What is database?

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

3. What is DBMS?

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

4. What is a Database system?

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

5. Advantages of DBMS?

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

6. Disadvantage in File Processing System?

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

7. Describe the three levels of data abstraction?

There are three levels of abstraction:

- Physical level: The lowest level of abstraction describes how data are stored.

- Logical level*:* The next higher level of abstraction, describes what data are stored in database and what relationship among those data.
- View level:The highest level of abstraction describes only part of entire database.

8. Define the "integrity rules"

There are two Integrity rules.

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

9. What is extension and intension?

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

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

10. What is Data Independence?

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

Two types of Data Independence:

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

NOTE: Logical Data Independence is more difficult to achieve

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

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

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

12. What is Data Model?

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

13. What is E-R model?

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

14. What is Object Oriented model?

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

15. What is an Entity?

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

16. What is an Entity type?

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

17. What is an Entity set?

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

18. What is an Extension of entity type?

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

19. What is an attribute?

It is a particular property, which describes the entity.

20. What is a Relation Schema and a Relation?

A relation Schema denoted by R(A1, A2, …, An) is made up of the relation name R and the list of attributes A_i that it contains. A relation is defined as a set of tuples. Let r be the relation which contains set tuples (t1, t2, t3, ...,tn). Each tuple is an ordered list of n-values t=(v1,v2, ..., vn).

21. What is degree of a Relation?

It is the number of attribute of its relation schema.

22. What is Relationship?

It is an association among two or more entities.

23. What is Relationship set?

The collection (or set) of similar relationships.

24. *What is Relationship type?*

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

25. What is degree of Relationship type?

It is the number of entity type participating.

26. What is DDL (Data Definition Language)?

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

27. What is VDL (View Definition Language)?

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

28. What is SDL (Storage Definition Language)?

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

29. What is Data Storage - Definition Language?

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

30. What is DML (Data Manipulation Language)?

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

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

31. What is DML Compiler?

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

32. What is Relational Algebra?

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

33. What is Relational Calculus?

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

34. What is normalization?

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

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

35. What is Functional Dependency?

A Functional dependency is denoted by X ⟶ Y between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuple that can form a relation state r of R. **What is Lossless join property?**

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

36. What is 1 NF (Normal Form)?

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

37. What is Fully Functional dependency?

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

38. What is 2NF?

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

39. What is 3NF?

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

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

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

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

→

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

41. What is 4NF?

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

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

42. What is 5NF?

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

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

43. What is Domain-Key Normal Form?

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

44. What is a Partial key and Compound key?

Partial Key:

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

Compound Key**:**

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

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

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

46. What is join dependency and inclusion dependency?

Join Dependency:

A Join dependency is generalization of Multivalued dependency.A JD {R1, R2, ...,Rn} is said to hold over a relation R if R1, R2, R3, ..., Rn is a lossless-join decomposition of R . There is no set of sound and complete inference rules for JD.

Inclusion Dependency:

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

47. What is durability in DBMS?

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

48. What do you mean by atomicity and aggregation?

Atomicity:

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

Aggregation*:*

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

49. What is a Phantom Deadlock?

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

50. What is a checkpoint and when does it occur?

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

51. What are the different phases of transaction?

Different phases are

- Analysis phase
- Redo Phase

- Undo phase

52. What do you mean by flat file database?

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

53. What is "transparent DBMS"?

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

54. What is a query?

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

55. What do you mean by Correlated subquery?

Subqueries, or nested queries, are used to bring back a set of rows to be used by the parent query. Depending on how the subquery is written, it can be executed once for the parent query or it can be executed once for each row returned by the parent query. If the subquery is executed for each row of the parent, this is called a *correlated subquery*.

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

Addition, deletion and modification.

57. What are the unary operations in Relational Algebra?

PROJECTION and SELECTION.

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

No.

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

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

59. Name the sub-systems of a RDBMS

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

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

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

61. What is the job of the information stored in data-dictionary?The information in the data dictionary validates the existence of the objects, provides access to them, and maps the actual physical storage location.

62. How do you communicate with an RDBMS?

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

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

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

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

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

- Database files
- Control files
- Redo logs

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

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

65. What is ROWID?

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

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

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

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

66. What is database Trigger?

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

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

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

68. Spurious tuples may occur due to

i. Bad normalization

ii. Theta joins

iii. Updating tables from join

a) i& ii b) ii & iii

c) i& iii d) ii & iii

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

69. A dominant entity is the entity

a) on the N side in a 1 : N relationship

b) on the 1 side in a 1 : N relationship

c) on either side in a 1 : 1 relationship

d) nothing to do with 1 : 1 or 1 : N relationship

(b) on the 1 side in a 1 : N relationship

70. What is Storage Manager?

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

71. What is Buffer Manager?

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

72. What is Transaction Manager?

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

73. What is File Manager?

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

74. What is Authorization and Integrity manager?

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

75. What are stand-alone procedures?

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

76. What are cursors give different types of cursors.

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

- *Implicit*
- *Explicit*

77. What do you understand by dependency preservation?

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

$$((\Pi_{R1}(F)) \cup \ldots \cup (\Pi_{Rn}(F)))^{+} = F^{+}$$

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

Content Beyond Syllabus

Part A. For connecting java application with the mysql database, you need to follow 5 steps to perform database connectivity.

In this example we are using MySql as the database. So we need to know following informations for the mysql database:

1. **Driver class**: The driver class for the mysql database is com.mysql.jdbc.Driver.
2. **Connection URL**: The connection URL for the mysql database **is jdbc:mysql://localhost:3306/sonoo** where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we may also use IP address, 3306 is the port number and sonoo is the database name. We may use any database, in such case, you need to replace the sonoo with your database name.
3. **Username**: The default username for the mysql database is root.
4. **Password**: Password is given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.

Let's first create a table in the mysql database, but before creating table, we need to create database first.

1. create database sonoo;
2. use sonoo;
3. create table emp(id int(10),name varchar(40),age int(3));

Example to Connect Java Application with mysql database

In this example, sonoo is the database name, root is the username and password.

```
import java.sql.*;
class MysqlCon{
public static void main(String args[]){
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(  "jdbc:mysql://localhost:3306/sonoo","root","ro
ot");  //here sonoo is database name, root is username and password
Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery("select * from emp");
while(rs.next())
System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));
con.close();
}catch(Exception e){ System.out.println(e);}
```

```
} }
```

To connect java application with the mysql database mysqlconnector.jar file is required to be loaded.

Two ways to load the jar file:

1. paste the mysqlconnector.jar file in jre/lib/ext folder
2. set classpath

1) paste the mysqlconnector.jar file in JRE/lib/ext folder:

Download the mysqlconnector.jar file. Go to jre/lib/ext folder and paste the jar file here.

2) set classpath:

There are two ways to set the classpath:

- temporary
- permanent

How to set the temporary classpath

open command prompt and write:

1. C:>set classpath=c:\folder\mysql-connector-java-5.0.8-bin.jar;.;

How to set the permanent classpath

Go to environment variable then click on new tab. In variable name write classpath and in variable value paste the path to the mysqlconnector.jar file by appending mysqlconnector.jar;.; as C:\folder\mysql-connector-java-5.0.8-bin.jar;.;

Part B. **PL/SQL If**

Example of PL/SQL If Statement

```
1. DECLARE
2.   a number(3) := 500;
3. BEGIN
4.   -- check the boolean condition using if statement
5.   IF( a < 20 ) THEN
6.     -- if condition is true then print the following
7.     dbms_output.put_line('a is less than 20 ' );
8.   ELSE
9.     dbms_output.put_line('a is not less than 20 ' );
10.  END IF;
11.  dbms_output.put_line('value of a is : ' || a);
12. END;
```

PL/SQL Loop

The PL/SQL loops are used to repeat the execution of one or more statements for specified number of times. These are also known as iterative control statements.

There are 4 types of PL/SQL Loops.

1. Basic Loop / Exit Loop
2. While Loop
3. For Loop
4. Cursor For Loop

Example of PL/SQL EXIT Loop

```
1. DECLARE
2. i NUMBER := 1;
3. BEGIN
4. LOOP
5. EXIT WHEN i>10;
6. DBMS_OUTPUT.PUT_LINE(i);
7. i := i+1;
8. END LOOP;
9. END;
```