|  |  |  |  |
| --- | --- | --- | --- |
|  | **Sub Title : DATABASE APPLICATIONS LABORATORY** | | |
| **Course Code:18CSL56** | **No. of Credits: 0 : 0 : 1**  **(L-T-P)** | **No. of lecture hours/week : 2** |
| **Exam Duration : 3 hours** | **CIE + SEE = 50+50=100** |
|  | | | |
| **Course Objectives:** | **Description** | | |
| 1. Provide a strong formal foundation in database concepts and technology and techniques relating to query processing by SQL.  2. Design and implement a real time database application for a given problem-domain.  3. Demonstrate the use of relational data model and systematic database design approaches covering conceptual design, logical design through the mini project.  4.IntroduceMongoDB, CRUD Operations &itsusage in Enterprise Applications. | | |
|  | | | |
| **COURSE CONTENTS:** | | | |
| **Part A** | 1. Execution of given 3 exercises.  2. Introduction to MongoDB and CRUD Operations.  3. MongoDB Usage in Enterprise Applications. | | |
| **Part B** | Implementation of mini project. | | |
|  | | | |
| **PART – A**  **INSTRUCTIONS:**   1. The exercises are to be solved in an RDBMS environment like Oracle or DB2. 2. Suitable tuples have to be entered so that queries are executed correctly. 3. Relevant queries other than the ones listed along with the exercises may also be asked in the examinations. 4. Questions must be asked based on lots. | | | |
| **1. 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. | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **2**. | **Consider the following schema for Order Database:**  SALESMAN(Salesman\_id, Name, City, Commission)  CUSTOMER(Customer\_id, Cust\_Name, City, Grade, Salesman\_id)  ORDERS(Ord\_No, Purchase\_Amt, Ord\_Date, Customer\_id, Salesman\_id)  Write SQL queries to  1. Count the customers with grades above Bangalore’s average.  2. Find the name and numbers of all salesmen who had more than one customer.  3. List all the 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 12345. All his orders must also be deleted. | | | | | | | | | | | | | |
| **3.** | **Consider the schema for College Database:**  STUDENT(USN, SName, Address, Phone, Gender)  SEMSEC(SSID, Sem, Sec)  CLASS(USN, SSID)  SUBJECT(Subcode, Title, Sem, Credits)  CIEMARKS(USN, Subcode, SSID, CIE1, CIE2, CIE3, FinalCIE)  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 ‘1DA15CS101’ in all subjects.  4. Calculate the FinalCIE (average of best two test marks) and update the corresponding table for all students.  5. Categorize students based on the following criterion:  If FinalCIE = 17 to 20 then CAT = ‘Outstanding’  If FinalCIE< 12 then CAT = ‘Weak’  If FinalCIE = 12 to 16 then CAT = ‘Average’  Give these details only for 8th semester A, B, and C section students. | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | |
| **PART – B**  A mini project should be implemented by the students in teams. The maximum size of a team can be 3 from the same batch. The students have to finalize a project topic by discussing with the faculty. The mini project must be carried out in the college only.  Design a Database application for a particular case study using Visual Basic/Java Script in visual studio /Eclipse Tool.  The tasks when implementing mini project would be:   1. Understand the complete domain knowledge of the application and derive the complete data requirement specification for the mini project. 2. Design the ER diagram for the application. 3. Design Relational Schema diagram for the application. 4. Normalization of the relational design. 5. Implement minimum 5 queries for the application. 6. Documentation & submission of report.   **General guidelines:**   * Database for the project - Oracle / MySQL/ DB2 / SQL Server / MongoDB etc.   **Sample Mini Projects.**   |  |  | | --- | --- | | Inventory Control System. | Placement management system | | Material Requirement Processing. | Library management system | | Hospital Management System. | Web Based User Identification System. | | Railway Reservation System. | Timetable Management System | | Hotel Management System | Personal Information System |   **Note**: In the examination, the marks will be evaluated based on database execution from Part A and project demonstration, project report and viva-voce from Part B. | | | | | | | | | | | | | | |
| **Course Outcomes** | | **Description** | | | | | | | | | | | **RBT Levels** | |
| **CO1** | | Understand, analyze, and effectively explain the underlying concepts of database technologies. | | | | | | | | | | | **L4** | |
| **CO2** | | Use SQL to create, secure, populate, maintain and query a database. | | | | | | | | | | | **L4** | |
| **CO3** | | Design and implement real time applications according to design principles that balance data retrieval performance with data consistency. | | | | | | | | | | | **L5** | |
| **CO4** | | Identify the Core MongoDB Operations. | | | | | | | | | | | **L2** | |
|  | | | | | | | | | | | | | | |
| **CO-PO Mapping** | | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **P06** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | | **PO12** |
| **CO1** | | **3** | **3** |  |  |  |  |  |  |  |  |  | |  |
| **CO2** | | **3** | **3** | **3** |  |  |  |  |  |  |  |  | |  |
| **CO3** | | **3** | **3** | **3** | **3** | **3** |  |  |  | **3** |  |  | |  |
| **CO4** | | **3** |  |  |  | **2** |  |  |  |  |  |  | |  |
| **Strong -3 Medium -2 Weak -1** | | | | | | | | | | | | | | |
| **TEXT BOOKS:**  1. Fundamental of Database Systems by Elmasri and Navathe, 7th Edition, Addison-Wesley,  2015 **ISBN-10:** 0133970779, **ISBN-13:** 978-0133970777 | | | | | | | | | | | | | | |
| **REFERENCE BOOKS:**  1. Database Management Systems by Raghu Ramakrishnan and Johannes Gehrke – 3rd Edition, McGraw-Hill, 2006.  2. An Introduction to Database Systems by C.J. Date, A. Kannan, S. Swamynathan, 8th Edition, Pearson Education,2013.  3. Data Base system Concepts by Silberschatz, Korth and Sudharshan, 5th edition McGraw Hill, 2011. | | | | | | | | | | | | | | |
| **SELF STUDY REFERENCES/WEBLINKS:**  1. <https://www.mongodb.com/>  2. <https://docs.mongodb.com/manual/crud/> | | | | | | | | | | | | | | |

**INTRODUCTION**

**OVERVIEW**

Databases store data and metadata. Data are the individual facts that are used to derive information. Metadata describe the content, quality, condition, availability, and characteristics of data. Database Management Systems (DBMS) is used to modify the data.

There are varieties of database types: Sequential files, Hierarchical databases, Network databases, and Relational databases. The recent database type is Object-Relational database, which is essentially a relational database with some Object properties.

Relational databases become popular because it was easy to modify the schema. It is very easy to add tables and columns to the schema, and doing so does not affect the remainder of the schema, and more important, does not affect the applications that access the database schema. Older databases required the databases to be restructured and the applications to be modified. Avoiding database and application maintenance is an important benefit and the reason for the switch.

Relational databases consist of independent tables. Relational Database Management Software (RDBMS) does not know how the records in the table are related.

**ORACLE**

Oracle is the largest database manufacturer and the second-largest software manufacturer in the world. The company began as a relational database manufacturer. In the beginning, Oracle touted its software as “being able to run on any platform”. This openness has been most attractive to companies; an Oracle has tried to maintain its image as an open product. Oracle was at a good place when industry became extremely interested in moving away from network databases and the mainframe.

Oracle Database 10g is an entry-level database based on the Oracle Database 10g Release 2 code base that's free to develop, deploy, and distribute; fast to download; and simple to administer. Oracle Database is a great starter database for:

* Developers working on PHP, Java, .NET, and Open Source applications
* DBAs who need a starter database for training and deployment
* Independent Software Vendors (ISVs) and hardware vendors who want a starter database
* Educational institutions and students who need database for their curriculum

The main use of the Oracle database system is to store and retrieve data for applications. The Oracle database includes different languages and interfaces that allow programmers to access and manipulate the data in the database. The ANSI standard Structured Query Language (SQL) provides basic functions for data manipulation, transaction control, and record retrieval from the database. However, most end users interact with Oracle through applications that provide an interface that hides the underlying SQL and its complexity.

**BASICS**

**Structured Query Language (SQL)**, which is an ANSI standard language for interacting with relational databases, is the main tool for extracting the information.

A **database** is a representation of a real-world thing called an **Entity.** Examples of entities are vehicles, employees, customers, fish, buildings, and even things such as baseball teams. The database stores facts about the entity in an organized framework, model, or schema. These facts are called **attributes.**

An **Instance** is one occurrence of an entity. Each entity must have an identifier, which is one or more attributes that make each entity instance unique from any other instance. The identifier should contain a value that does not change. Examples of identifiers are student IDs, payroll numbers, or social security numbers. If the entity does not have an attribute that can be used as an identifier, an artificial identifier can be created. The identifier on an entity is often called a **primary key.** A **foreign key** is a set of attributes of the considered table that exists as a primary key attributes in another table. Database records are matched (joined) through the use of primary and foreign keys.

**Normalization** is a process consisting of series of steps, which is used to group the database attributes. The purpose of this design is to ensure that the tables within the database are space efficient and performance efficient.

* Zero Normal Form—each of the relations (tables) has a unique identifier (primary key).
* First Normal Form—Separate the repeating groups of attributes or multi-valued attributes into a relation of their own. Be sure to form composite keys.
* Second Normal Form—Establish full functional dependency by separating out attributes that are not fully dependent on the full primary keys.
* Third Normal Form—Remove transitive dependencies by separating attributes that are dependent on a non key attribute.

MOVIE DATABASE

**1. 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.

SCHEMA DIAGRAM

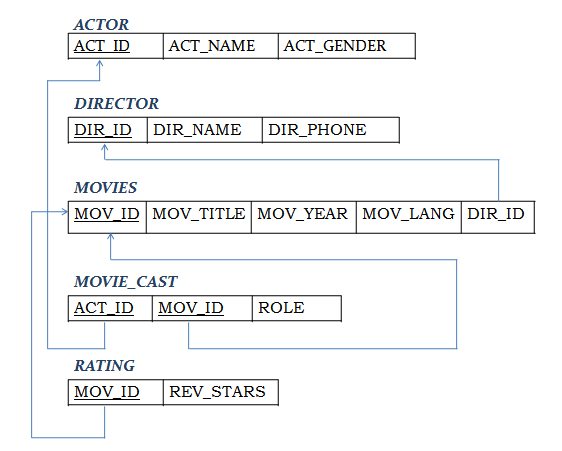


TABLE CREATION

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 VARCHA

R (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));

RECORD INSERTION

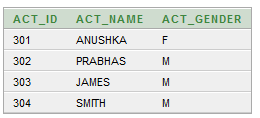
INSERT INTO ACTOR VALUES (301,'ANUSHKA','F');

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

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

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

SELECT \* FROM ACTOR;



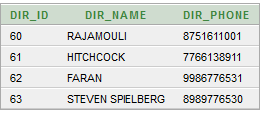
INSERT INTO DIRECTOR VALUES (60,'RAJAMOULI', 8751611001);

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

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

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

SELECT \* FROM DIRECTOR;



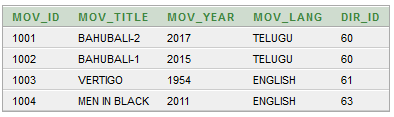
INSERT INTO MOVIES VALUES (1001,'BAHUBALI-2', 2017, 'TELUGU', 60);

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

INSERT INTO MOVIES VALUES (1003,'VERTIGO', 1954, 'ENGLISH', 61);

INSERT INTO MOVIES VALUES (1004,'MEN IN BLACK', 2011, 'ENGLISH', 63);

SELECT \* FROM MOVIES;



INSERT INTO MOVIE\_CAST VALUES (301, 1002, 'HEROINE');

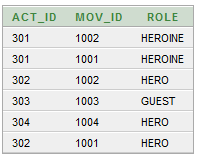
INSERT INTO MOVIE\_CAST VALUES (301, 1001, 'HEROINE');

INSERT INTO MOVIE\_CAST VALUES (302, 1002, 'HERO');

INSERT INTO MOVIE\_CAST VALUES (303, 1003, 'GUEST');

INSERT INTO MOVIE\_CAST VALUES (304, 1004, 'HERO');

SELECT \* FROM MOVIE\_CAST;



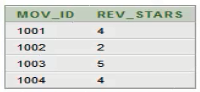
INSERT INTO RATING VALUES (1001, 4);

INSERT INTO RATING VALUES (1002, 2);

INSERT INTO RATING VALUES (1003, 5);

INSERT INTO RATING VALUES (1004, 4);

SELECT \* FROM RATING;



**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');

33Q-1

**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;



**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

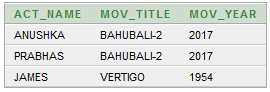
JOIN MOVIE\_CAST C

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;



**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)

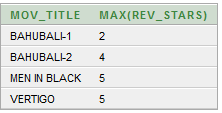
FROM MOVIES

INNER JOIN RATING USING (MOV\_ID)

GROUP BY MOV\_TITLE

HAVING MAX (REV\_STARS)>0

ORDER BY MOV\_TITLE;



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

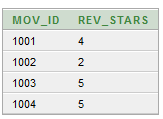
WHERE DIR\_ID IN (SELECT DIR\_ID

FROM DIRECTOR

WHERE DIR\_NAME = 'STEVEN SPIELBERG'));

33Q-51

SELECT \* FROM RATING;



ORDER DATABASE

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

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

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

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

Write SQL queries to

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

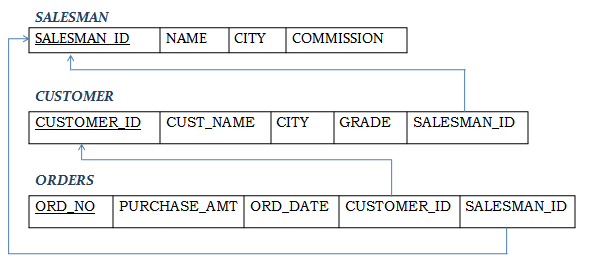
2. Find the name and numbers of all salesmen who had more than one customer.

3. List all the 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 12345. All his orders must also be deleted.

**SCHEMA-DIAGRAM:**



**CREATION OF TABLES:**

CREATE TABLE SALESMAN

(SALESMAN\_ID NUMBER (4),

NAME VARCHAR2 (20),

CITY VARCHAR2 (20),

COMMISSION VARCHAR2 (20),

PRIMARY KEY (SALESMAN\_ID));

CREATE TABLE CUSTOMER

(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 CUSTOMER (CUSTOMER\_ID) ON DELETE CASCADE,

SALESMAN\_ID REFERENCES SALESMAN (SALESMAN\_ID) ON DELETE CASCADE);

**INSERTION OF RECORDS:**

INSERT INTO SALESMAN VALUES (1000, 'JOHN','BANGALORE','25 %');

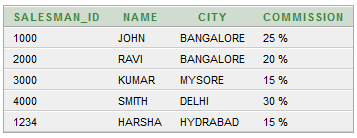
INSERT INTO SALESMAN VALUES (2000, 'RAVI','BANGALORE','20 %');

INSERT INTO SALESMAN VALUES (3000, 'KUMAR','MYSORE','15 %');

INSERT INTO SALESMAN VALUES (4000, 'SMITH','DELHI','30 %');

INSERT INTO SALESMAN VALUES (1234, 'HARSHA','HYDRABAD','15 %');

SELECT \* FROM SALESMAN;



INSERT INTO CUSTOMER VALUES (10, 'PREETHI','BANGALORE', 100, 1000);

INSERT INTO CUSTOMER VALUES (11, 'VIVEK','MANGALORE', 300, 1000);

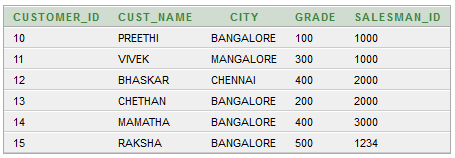
INSERT INTO CUSTOMER VALUES (12, 'BHASKAR','CHENNAI', 400, 2000);

INSERT INTO CUSTOMER VALUES (13, 'CHETHAN','BANGALORE', 200, 2000);

INSERT INTO CUSTOMER VALUES (14, 'MAMATHA','BANGALORE', 400, 3000);

INSERT INTO CUSTOMER VALUES (15, 'RAKSHA','BANGALORE', 500, 1234);

SELECT \* FROM CUSTOMER;



INSERT INTO ORDERS VALUES (50, 5000, '04-MAY-17', 10, 1000);

INSERT INTO ORDERS VALUES (51, 450, '20-JAN-17', 10, 2000);

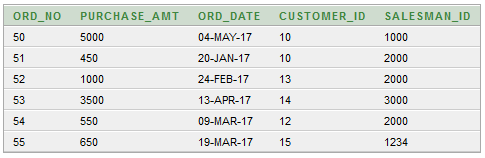
INSERT INTO ORDERS VALUES (52, 1000, '24-FEB-17', 13, 2000);

INSERT INTO ORDERS VALUES (53, 3500, '13-APR-17', 14, 3000);

INSERT INTO ORDERS VALUES (54, 550, '09-MAR-17', 12, 2000);

INSERT INTO ORDERS VALUES (55, 650, '19-MAR-17', 15, 1234);

SELECT \* FROM ORDERS;



**QUERIES**

**1. Count the customers with grades above Bangalore’s average.**

SELECT GRADE, COUNT (DISTINCT CUSTOMER\_ID)

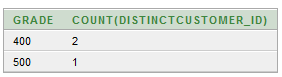
FROM CUSTOMER

GROUP BY GRADE

HAVING GRADE > (SELECT AVG(GRADE)

FROM CUSTOMER

WHERE CITY='BANGALORE');

****

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

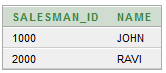
SELECT SALESMAN\_ID, NAME

FROM SALESMAN A

WHERE 1 < (SELECT COUNT (\*)

FROM CUSTOMER

WHERE SALESMAN\_ID=A.SALESMAN\_ID);

****

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

SELECT SALESMAN.SALESMAN\_ID, NAME, CUST\_NAME, COMMISSION

FROM SALESMAN, CUSTOMER

WHERE SALESMAN.CITY = CUSTOMER.CITY

UNION

SELECT SALESMAN\_ID, NAME, 'NO MATCH', COMMISSION

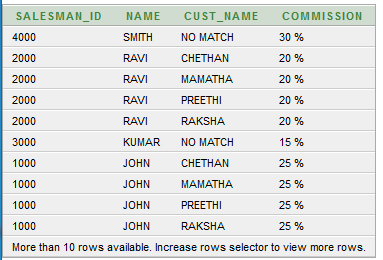
FROM SALESMAN

WHERE NOT CITY = ANY

(SELECT CITY

FROM CUSTOMER)

ORDER BY 2 DESC;

****

**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

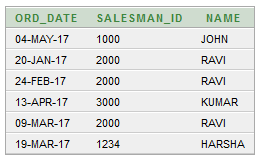
AND B.PURCHASE\_AMT=(SELECT MAX (PURCHASE\_AMT)

FROM ORDERS C

WHERE C.ORD\_DATE = B.ORD\_DATE);

SELECT \* FROM ELITSALESMAN

**22Q-41**

****

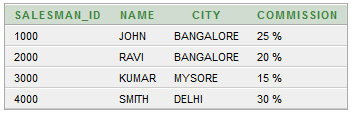
**5. Demonstrate the DELETE operation by removing salesman with id 12345. All his orders must also be deleted.**

DELETE FROM SALESMAN

WHERE SALESMAN\_ID=1234;

SELECT \* FROM SALESMAN;

**22Q-51**

****

COLLEGE DATABASE

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

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

SEMSEC(SSID, Sem, Sec)

CLASS(USN, SSID)

SUBJECT(Subcode, Title, Sem, Credits)

CIEMARKS(USN, Subcode, SSID, CIE1, CIE2, CIE3, FinalCIE)

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 ‘1DA15CS101’ in all subjects.

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

5. Categorize students based on the following criterion:

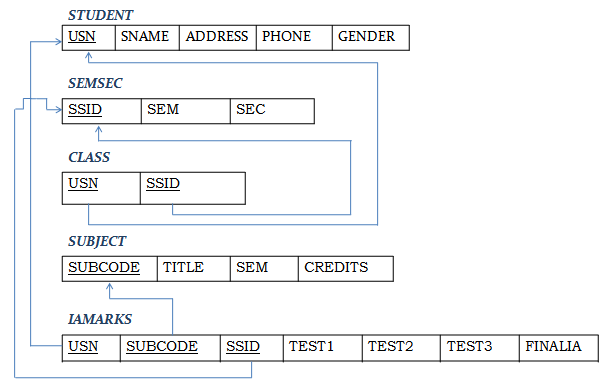
If FinalCIE = 17 to 20 then CAT = ‘Outstanding’

If FinalCIE< 12 then CAT = ‘Weak’

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

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

**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));

RECORD INSERTION

INSERT INTO STUDENT VALUES

('1DA15CS001','AJAY','TUMKUR', 9845091341,'M');

INSERT INTO STUDENT VALUES ('1DA15CS091','CHITRA','DAVANGERE',7696772121,'F');

INSERT INTO STUDENT VALUES

('1DA15CS101','JEEVA','BELLARY', 9944850121,'M');

INSERT INTO STUDENT VALUES ('1DA19CS045','AKASH','BENGALURU',9900211201,'M');

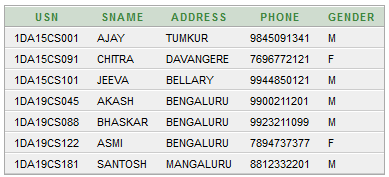
INSERT INTO STUDENT VALUES ('1DA19CS088','BHASKAR','BENGALURU',9923211099,'M');

INSERT INTO STUDENT VALUES

('1DA19CS122','ASMI','BENGALURU', 7894737377,'F');

INSERT INTO STUDENT VALUES ('1DA19CS181','SANTOSH','MANGALURU',8812332201,'M');

SELECT \* FROM STUDENT;



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

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

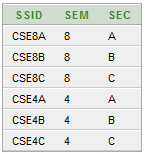
INSERT INTO SEMSEC VALUES ('CSE6C', 6,'C');

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

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

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

SELECT \* FROM SEMSEC;



INSERT INTO CLASS VALUES ('1DA15CS001','CSE6A');

INSERT INTO CLASS VALUES ('1DA15CS091','CSE6B');

INSERT INTO CLASS VALUES ('1DA15CS101','CSE6C');

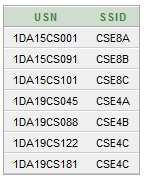
INSERT INTO CLASS VALUES ('1DA19CS045','CSE4A');

INSERT INTO CLASS VALUES ('1DA19CS088','CSE4B');

INSERT INTO CLASS VALUES ('1DA19CS122','CSE4C');

INSERT INTO CLASS VALUES ('1DA19CS181','CSE4C');

SELECT \* FROM CLASS;



INSERT INTO SUBJECT VALUES ('15CS61', 'ME', 6, 4);

INSERT INTO SUBJECT VALUES ('15CS62','CN', 6, 4);

INSERT INTO SUBJECT VALUES ('15CS63','DBMS', 6, 4);

INSERT INTO SUBJECT VALUES ('15CS64','JAVA', 6, 3);

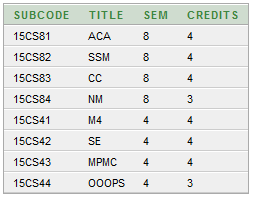
INSERT INTO SUBJECT VALUES ('15CS41','M4', 4, 4);

INSERT INTO SUBJECT VALUES ('15CS42','SE', 4, 4);

INSERT INTO SUBJECT VALUES ('15CS43','MPMC', 4, 4);

INSERT INTO SUBJECT VALUES ('15CS44','OOOPS', 4, 3);

SELECT \* FROM SUBJECT;



INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES

('1DA15CS101','15CS61','CSE6C', 15, 16, 18);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES

('1DA15CS101','15CS62','CSE6C', 12, 19, 14);

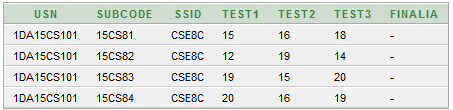
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES

('1DA15CS101','15CS63','CSE6C', 19, 15, 20);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES

('1DA15CS101','15CS64','CSE6C', 20, 16, 19);

SELECT \* FROM IAMARKS;



**QUERIES**

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

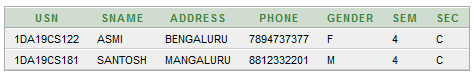
SELECT S.\*, SS.SEM, SS.SEC

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';



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

SELECT SS.SEM, SS.SEC, S.GENDER, COUNT (S.GENDER) AS COUNT

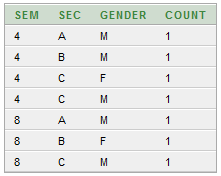
FROM STUDENT S, SEMSEC SS, CLASS C

WHERE S.USN = C.USN AND

SS.SSID = C.SSID

GROUP BY SS.SEM, SS.SEC, S.GENDER

ORDER BY SEM;



**3. Create a view of Test1 marks of student USN ‘1DA15CS101’ in all subjects.**

CREATE VIEW STU\_TEST1\_MARKS\_VIEW

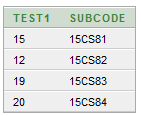
AS

SELECT TEST1, SUBCODE

FROM IAMARKS

WHERE USN = '1DA15CS101';

44Q-31



**4. Calculate the FinalCIE (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;

BEGIN

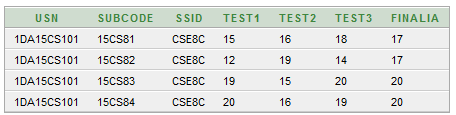
AVGMARKS;

END;

SELECT \* FROM IAMARKS;

44Q-41

44Q-42



**5. Categorize students based on the following criterion:**

**If FinalCIE = 17 to 20 then CAT = ‘Outstanding’**

**If FinalCIE< 12 then CAT = ‘Weak’**

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

**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;

