


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

	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.	

	<p>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)</p> <p>Write SQL queries to</p> <ol style="list-style-type: none"> 1. Count the customers with grades above Bangalore's average. 2. Find the name and numbers of all 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.	<p>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)</p> <p>Write SQL queries to</p> <ol style="list-style-type: none"> 1. List all the student details studying in fourth semester 'C' section. 2. Compute the total number of male and female students in each semester and in each section. 3. Create a view of Test1 marks of student USN '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' <p>Give these details only for 8th semester A, B, and C section students.</p>
<p style="text-align: center;">PART – B</p> <p>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.</p> <p>Design a Database application for a particular case study using Visual Basic/Java Script in visual studio /Eclipse Tool.</p> <p>The tasks when implementing mini project would be:</p> <ol style="list-style-type: none"> 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	PO6	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)

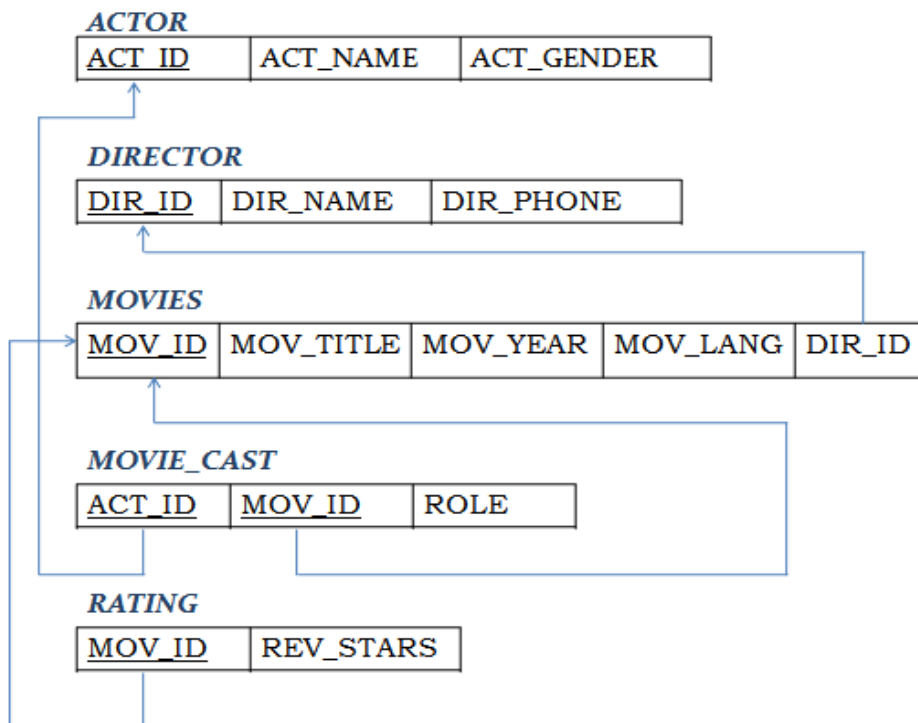
MOVIES(Mov_id, Mov_Title, Mov_Year, Mov_Lang, Dir_id)

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 VARCHAR  
  A (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;
```

ACT_ID	ACT_NAME	ACT_GENDER
301	ANUSHKA	F
302	PRABHAS	M
303	JAMES	M
304	SMITH	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);
```

```
SELECT * FROM DIRECTOR;
```

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

```
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;
```

MOV_ID	MOV_TITLE	MOV_YEAR	MOV_LANG	DIR_ID
1001	BAHUBALI-2	2017	TELUGU	60
1002	BAHUBALI-1	2015	TELUGU	60
1003	VERTIGO	1954	ENGLISH	61
1004	MEN IN BLACK	2011	ENGLISH	63

```
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;
```


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

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
BAHUBALI-2

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;
```

ACT_NAME	MOV_TITLE	MOV_YEAR
ANUSHKA	BAHUBALI-2	2017
PRABHAS	BAHUBALI-2	2017
JAMES	VERTIGO	1954

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

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

MOV_TITLE	MAX(REV_STARS)
BAHUBALI-1	2
BAHUBALI-2	4
MEN IN BLACK	5
VERTIGO	5

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'));
```

```
1 row(s) updated.
```

SELECT * FROM RATING;

MOV_ID	REV_STARS
1001	4
1002	2
1003	5
1004	5

ORDER DATABASE

2. Consider the following schema for Order Database:

SALESMAN(Salesman_id, Name, City, Commission)

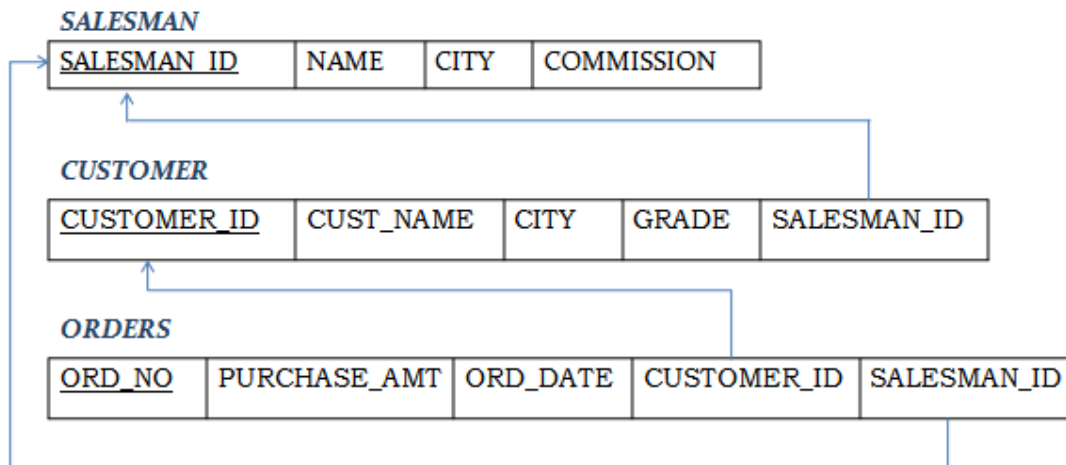
CUSTOMER(Customer_id, Cust_Name, City, Grade, Salesman_id)

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;
```

SALESMAN_ID	NAME	CITY	COMMISSION
1000	JOHN	BANGALORE	25 %
2000	RAVI	BANGALORE	20 %
3000	KUMAR	MYSORE	15 %
4000	SMITH	DELHI	30 %
1234	HARSHA	HYDRABAD	15 %

```
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;
```

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
15	RAKSHA	BANGALORE	500	1234

```
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;
```

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
55	650	19-MAR-17	15	1234

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');
```

GRADE	COUNT(DISTINCTCUSTOMER_ID)
400	2
500	1

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

```
SELECT SALESMAN_ID, NAME
```

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

SALESMAN_ID	NAME
1000	JOHN
2000	RAVI

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;
```

SALESMAN_ID	NAME	CUST_NAME	COMMISSION
4000	SMITH	NO MATCH	30 %
2000	RAVI	CHETHAN	20 %
2000	RAVI	MAMATHA	20 %
2000	RAVI	PREETHI	20 %
2000	RAVI	RAKSHA	20 %
3000	KUMAR	NO MATCH	15 %
1000	JOHN	CHETHAN	25 %
1000	JOHN	MAMATHA	25 %
1000	JOHN	PREETHI	25 %
1000	JOHN	RAKSHA	25 %
More than 10 rows available. Increase rows selector to view more rows.			

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
```

```
View created.
```

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
19-MAR-17	1234	HARSHA

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;
```

```
1 row(s) deleted.
```

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

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

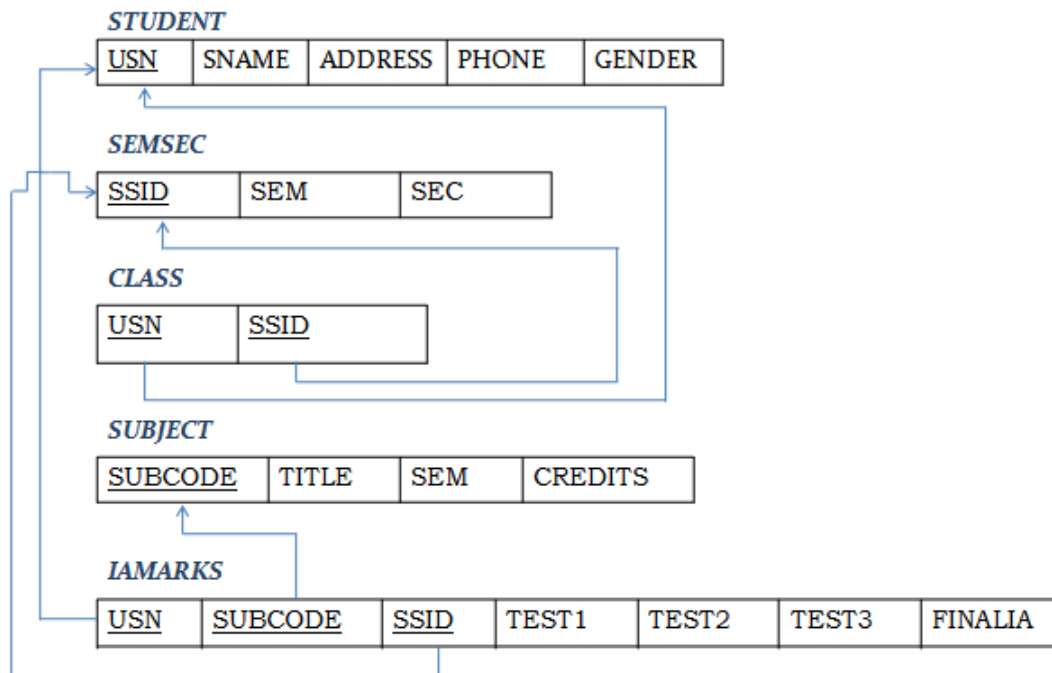


TABLE CREATION

```
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;
```

USN	SNAME	ADDRESS	PHONE	GENDER
1DA15CS001	AJAY	TUMKUR	9845091341	M
1DA15CS091	CHITRA	DAVANGERE	7696772121	F
1DA15CS101	JEEVA	BELLARY	9944850121	M
1DA19CS045	AKASH	BENGALURU	9900211201	M
1DA19CS088	BHASKAR	BENGALURU	9923211099	M
1DA19CS122	ASMI	BENGALURU	7894737377	F
1DA19CS181	SANTOSH	MANGALURU	8812332201	M

```
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;
```

SSID	SEM	SEC
CSE8A	8	A
CSE8B	8	B
CSE8C	8	C
CSE4A	4	A
CSE4B	4	B
CSE4C	4	C

```
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;
```

USN	SSID
1DA15CS001	CSE8A
1DA15CS091	CSE8B
1DA15CS101	CSE8C
1DA19CS045	CSE4A
1DA19CS088	CSE4B
1DA19CS122	CSE4C
1DA19CS181	CSE4C

```
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;
```

SUBCODE	TITLE	SEM	CREDITS
15CS81	ACA	8	4
15CS82	SSM	8	4
15CS83	CC	8	4
15CS84	NM	8	3
15CS41	M4	4	4
15CS42	SE	4	4
15CS43	MPMC	4	4
15CS44	OOOPS	4	3

```
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;
```

USN	SUBCODE	SSID	TEST1	TEST2	TEST3	FINALIA
1DA15CS101	15CS81	CSE8C	15	16	18	-
1DA15CS101	15CS82	CSE8C	12	19	14	-
1DA15CS101	15CS83	CSE8C	19	15	20	-
1DA15CS101	15CS84	CSE8C	20	16	19	-

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	GENDER	SEM	SEC
1DA19CS122	ASMI	BENGALURU	7894737377	F	4	C
1DA19CS181	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  
WHERE S.USN = C.USN AND  
SS.SSID = C.SSID  
GROUP BY SS.SEM, SS.SEC, S.GENDER  
ORDER BY SEM;
```

SEM	SEC	GENDER	COUNT
4	A	M	1
4	B	M	1
4	C	F	1
4	C	M	1
8	A	M	1
8	B	F	1
8	C	M	1

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';
```

View created.

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

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;
```

Procedure created.

Statement processed.

USN	SUBCODE	SSID	TEST1	TEST2	TEST3	FINALIA
1DA15CS101	15CS81	CSE8C	15	16	18	17
1DA15CS101	15CS82	CSE8C	12	19	14	17
1DA15CS101	15CS83	CSE8C	19	15	20	20
1DA15CS101	15CS84	CSE8C	20	16	19	20

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;

USN	SNAME	ADDRESS	PHONE	GENDER	CAT
1DA15CS101	JEEVA	BELLARY	9944850121	M	OUTSTANDING
1DA15CS101	JEEVA	BELLARY	9944850121	M	OUTSTANDING
1DA15CS101	JEEVA	BELLARY	9944850121	M	OUTSTANDING
1DA15CS101	JEEVA	BELLARY	9944850121	M	OUTSTANDING