Department of Computer Science and Engineering

KPR Institute of Engineering and Technology

LABORATORY MANUAL

U21CS304

DATABASE MANAGEMENT SYSTEMS LABORATORY

VISION AND MISSION OF THE INSTITUTION

Vision

To become a premier institute of academic excellence by imparting technical, intellectual and professional skills to students for meeting the diverse need of the industry, society, the nation and the world at large

Mission

- ❖ Commitment to offer value-based education and enhancement of practical skills
- ❖ Continuous assessment of teaching and learning process through scholarly activities
- ❖ Enriching research and innovation activities in collaboration with industry and institute of repute
- ❖ Ensuring the academic process to uphold culture, ethics and social responsibility

VISION AND MISSION OF THE DEPARTMENT

Vision

To foster the students by providing learner centric teaching environment, continuous learning, research and development to become thriving professionals and entrepreneurs to excel in the field of computer science and contribute to the society.

Mission

The Mission of the Department is to

- ❖ Providing value-based education and contented learning experience to the students.
- ❖ Educating the students with the state of art technologies and cultivating their proficiency in analytical and designing skills.
- ❖ Enabling the students to achieve a successful career in Computer Science and Engineering or related fields to meet the changing needs of various stakeholders.
- ❖ Guiding the students in research by nurturing their interest in continuous learning towards serving the society and the country

Program Educational Objectives (PEOs)

The Program Educational Objectives (PEOs) of the Computer Science and Engineering (CSE) represent major accomplishments that the graduates are expected to achieve after three to five years of graduation.

PEO1: Obtain knowledge in cutting edge technologies in the field of computer science, necessary to solve real time problems through value-based education.

PEO2: Possess skills for team building, leadership quality and ethical values necessary to function productively and professionally.

PEO3: Develop innovative ideas to establish themselves as professionals and entrepreneurs in computing industry.

PEO4: Continue to learn new technologies through higher studies and research.

Program Outcomes (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering Fundamentals and an engineering specialization to the solution of complex engineering problems.
2. **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.
3. **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.
4. **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.
5. **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.
6. **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.
7. **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.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **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.
11. **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.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

Program Specific Outcomes (PSOs)

Graduates of Computer Science and Engineering will be able to

PSO 1: An ability to identify and analyse data management system like data acquisition, big data so as to facilitate the students in solving problems using the techniques of data analytics.

PSO 2: An ability to apply design and development principles of hardware and software in emerging technology environments like cloud computing and cyber forensics.

RUBRICS FOR ASSESSMENT

Criteria		Excellent (4 Marks)	Good (3 Marks)	Adequate (2 Marks)	Inadequate (1 Mark)
A. Preparation & Observation	**Criterion #1** Ability to setup and conduct experiments	Able to develop contingency or alternative plans and anticipate problems during experiment.	Able to develop contingency or alternative plans.	Able to use theoretical framework, measurement techniques, testing apparatus or model.	Unable to identify theoretical framework, measurement techniques, testing apparatus or model.
	Criterion #2 Ability to take measurements / readings and present data	Able to formulate, controls and evaluate alternatives of the experiment. Able to evaluate data and relate to engineering phenomena for decision-making.	Able to evaluate data and relate to engineering phenomena for decision-making.	Able to apply constraint and assumption into the experimental design. Able to conduct experiment correctly and collect data.	Unable to discuss experimental processes and protocols
B. Results & Interpretation	**Criterion #3** Ability to analyze the data theoretically and logically to conclude experimental results	Able to combine /organize more than one set of data, interpret data and make meaningful conclusion.	Able to evaluate or compare data and make meaningful conclusion	Able to select and use and apply appropriate techniques or methods to analyse the data.	Unable to select and describe the techniques or methods of analyzing the data.
	Criterion #4 Ability to interpret and discuss any discrepancies between theoretical and experimental results	Able to verify and/or validate several sets of data and relates to engineering phenomena for decision making.	Able to verify and/or validate data and relate to engineering phenomena for decision making.	Able to identify and verify how results relate/differ from theory or previous results	Unable to identify how results relate/differ from theory or previous results.
C. Viva Voce	**Criterion #5** Demonstrate the ability to respond effectively to questions	Able to listen carefully and respond to questions appropriately; is able to explain and interpret results to the teacher	Able to listen carefully and respond to questions appropriately	Misunderstand the questions and does not respond appropriately to the teacher, or has some trouble in answering questions	Unable to listen carefully to questions and does not provide an appropriate answer, or is unable to answer questions

LIST OF EXPERIMENTS

Prepared by **Approved by**

Mr.Mohan M

Ms.Avani Chandran

CYCLE OF EXPERIMENTS

CYCLE – I

S.NO	NAME OF THE EXPERIMENTS
01	Conceptual Database design using E-R model – case study
02	Implementation of SQL commands DDL, DCL, TCL
03	Queries to demonstrate implementation of various integrity and key constrains
04	Practice on various DML commands to write a query to interact with database
05	Practice on and aggregate functions and views

CYCLE – II

S.NO	NAME OF THE EXPERIMENTS
06	Implement joins, nested queries and stored procedures
07	Practice on procedural extensions (Functions, Cursors, Triggers)
08	Document Database creation using MongoDB
09	Mini Project (App development using oracle DB) i. Campus Management System ii. Library Management System iii. Student information system iv. Hall Booking System v. Online Exam Registration system vi. Stock maintenance system vii. Event Registration System viii. Passport automation system ix. Blood bank Management system x. E-ticketing for Airline reservation System

EX.NO: 01	CONCEPTUAL DATABASE DESIGN USING E-R MODEL – CASE STUDY
DATE:	

AIM:

To design an E-R diagram for the selected case study.

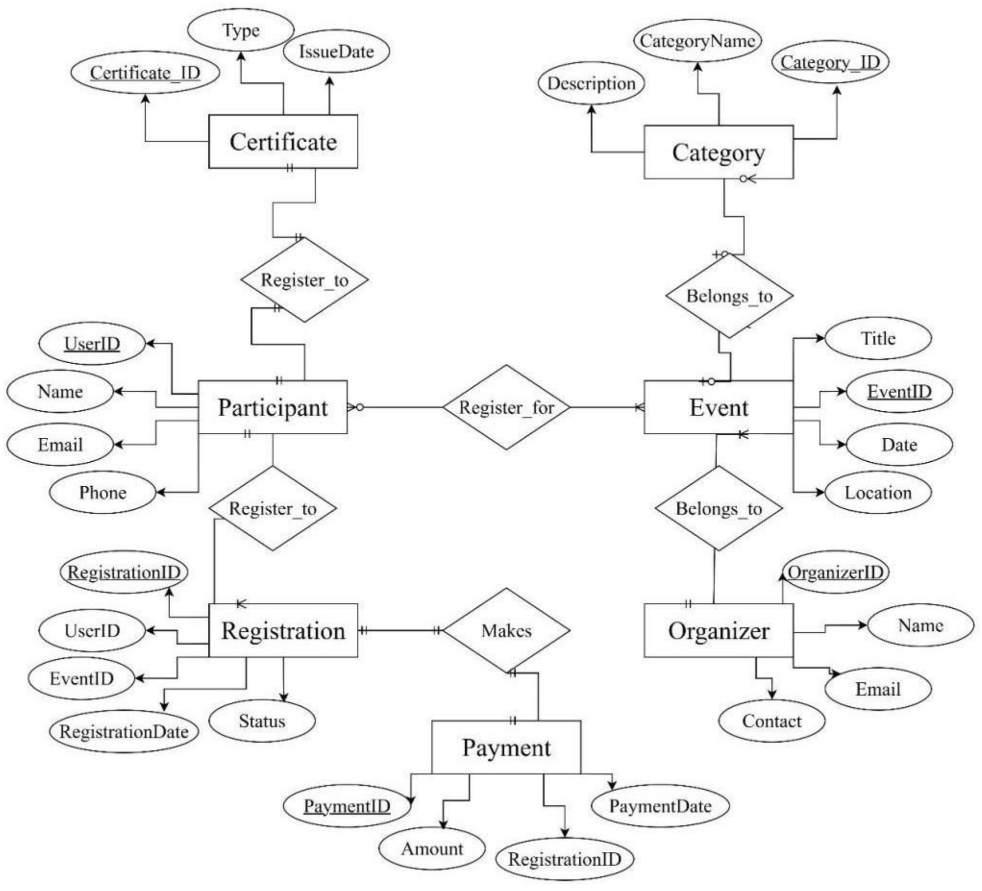
PROCEDURE:

The main steps to be involved are

1. Read and Analyze the Case Study
2. Identify entity, relationship and their attributes
3. Draw the necessary components of the entity, relationship and their attributes
4. Identify constraints, types of entity, attributes and relationships.
5. Draw the necessary components for constraints, types of entity, attributes and relationships.
6. Design the complete E-R model.

CASE STUDY: Event Registration System

E-R diagram for Event Registration System



RESULT:

Thus, the E-R diagram for Event Registration System has been created successfully.

EX.NO: 02	IMPLEMENTATION OF SQL COMMANDS DDL, DCL, TCL
DATE:	

AIM:

To implement DDL, DCL and TCL commands for the given problem statements.

PROCEDURE:

The main steps to be involved are

1. Read the problem statement.
2. Open the oracle terminal.
3. Write the necessary query for the given problem statement.
4. Execute the query
5. Save the results.

EXERCISES: Event Registration System

Data Definition Language (DDL) Commands:

1. **Create the table:**

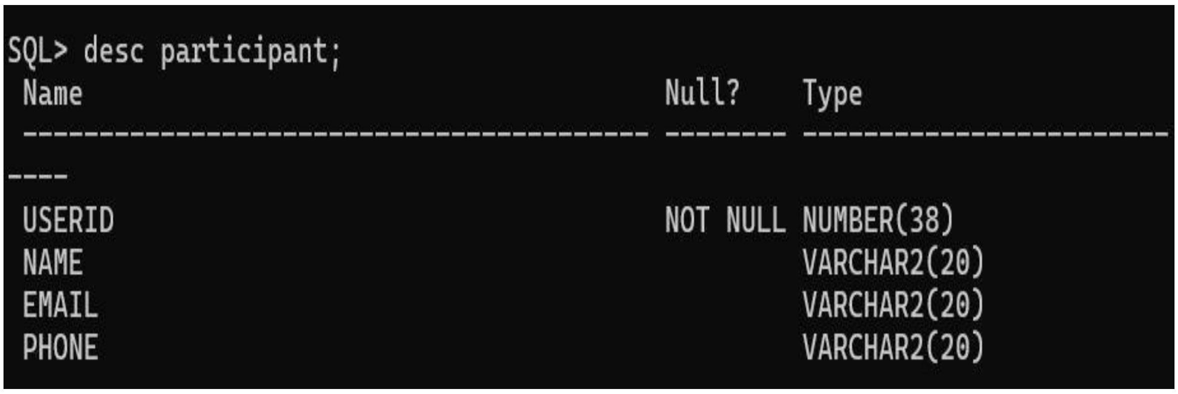
Queries:

- create table participant (UserID INT PRIMARY KEY,Name VARCHAR(20),Email VARCHAR(20),Phone VARCHAR(20));
- Create table event(eventid int primary key,title varchar(20),location varchar(20),event_date date);
- create table organizer(organizerID int primary key,name varchar(20),Email varchar(20),contact varchar(20));
- create table registration(registrationID int primary key,userID int,EventID int,registrationDate date,Status varchar(20));
- create table payment(paymentID int primary key,amount int ,registrationID int,paymentDate date);
- create table certificate(certificateID int primary key,type varchar(20),issueDate date);
- create table category(CategoryID int primary key, categoryName varchar(20),description varchar(20));

2. **Displaying the Structure of the above table:**

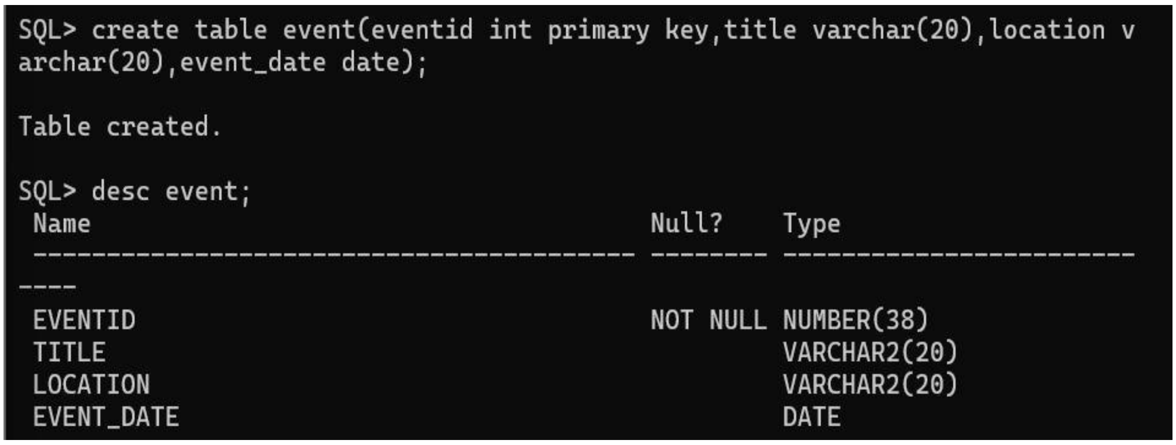
Queries:

desc participant;



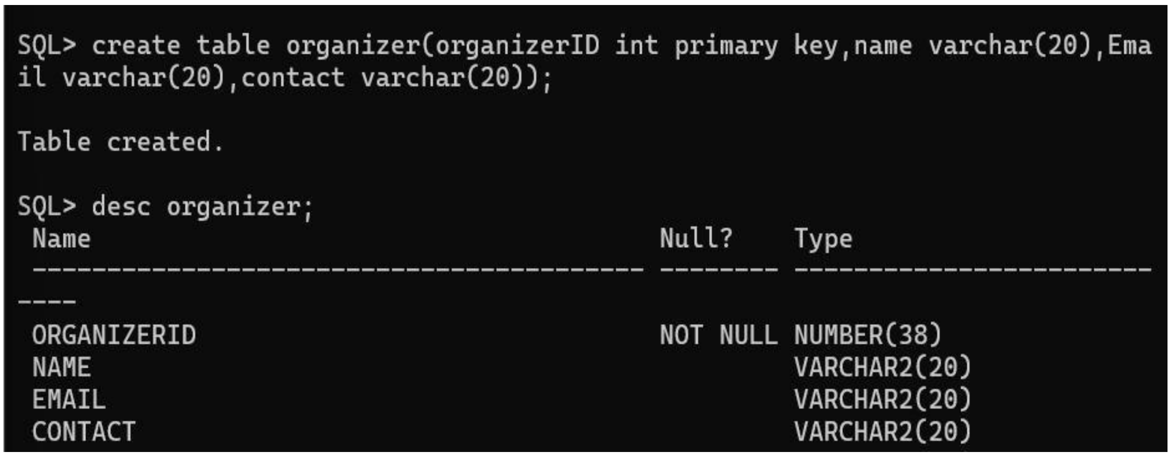
```
SQL> desc participant;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
----
 USERID                                    NOT NULL NUMBER(38)
 NAME                                               VARCHAR2(20)
 EMAIL                                              VARCHAR2(20)
 PHONE                                              VARCHAR2(20)
```

desc event;



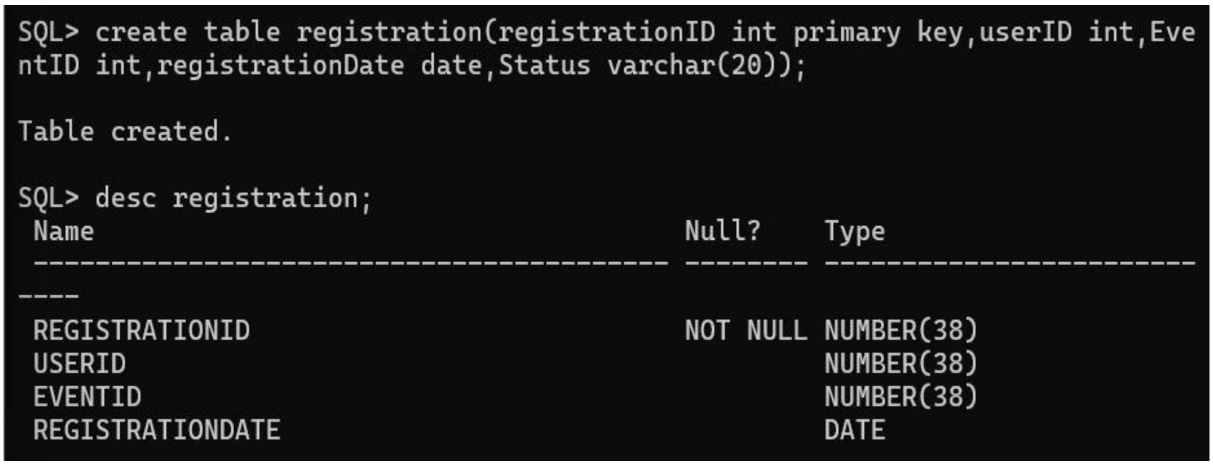
```
SQL> create table event(eventid int primary key,title varchar(20),location v
archar(20),event_date date);

Table created.

SQL> desc event;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
----
 EVENTID                                   NOT NULL NUMBER(38)
 TITLE                                              VARCHAR2(20)
 LOCATION                                           VARCHAR2(20)
 EVENT_DATE                                         DATE
```

desc organizer;



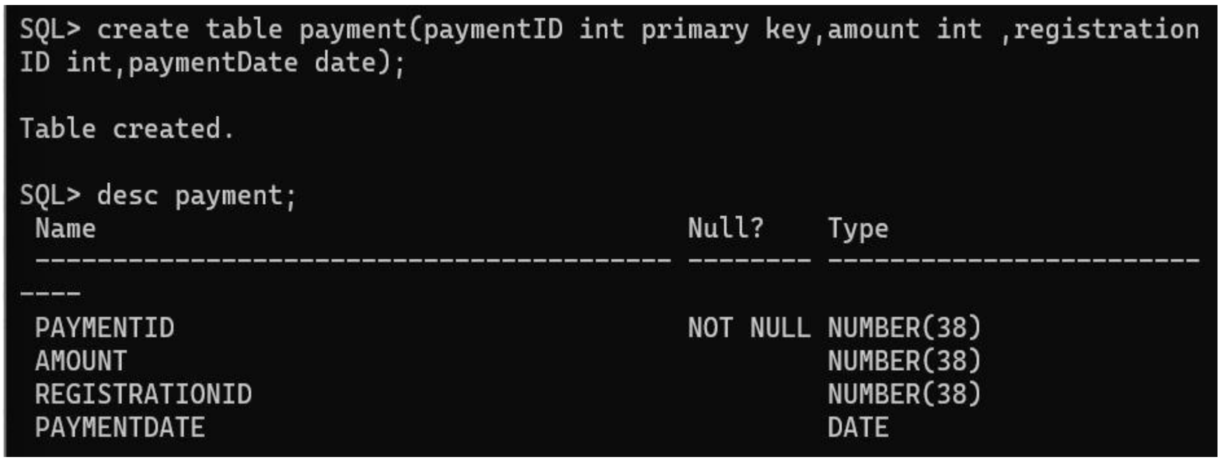
```
SQL> create table organizer(organizerID int primary key,name varchar(20),Ema
il varchar(20),contact varchar(20));

Table created.

SQL> desc organizer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
----
 ORGANIZERID                               NOT NULL NUMBER(38)
 NAME                                               VARCHAR2(20)
 EMAIL                                              VARCHAR2(20)
 CONTACT                                            VARCHAR2(20)
```

desc registration;



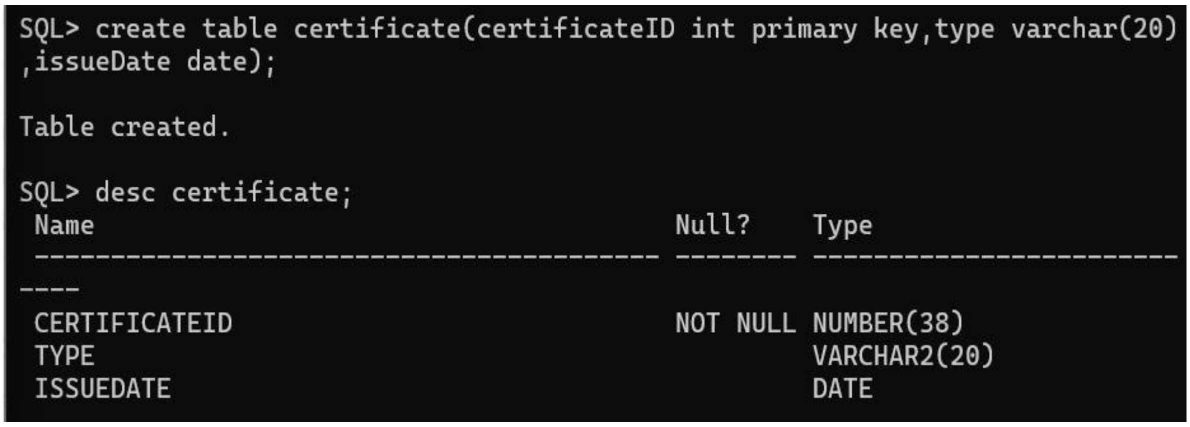
```
SQL> create table registration(registrationID int primary key,userID int,Eve
ntID int,registrationDate date,Status varchar(20));

Table created.

SQL> desc registration;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
----
 REGISTRATIONID                            NOT NULL NUMBER(38)
 USERID                                             NUMBER(38)
 EVENTID                                            NUMBER(38)
 REGISTRATIONDATE                                   DATE
```

desc payment;



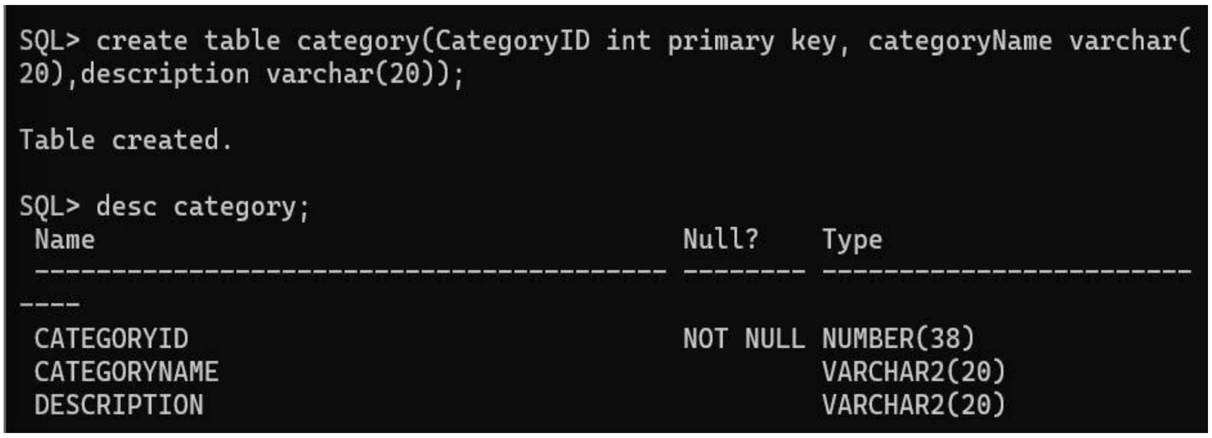
```
SQL> create table payment(paymentID int primary key,amount int ,registration
ID int,paymentDate date);

Table created.

SQL> desc payment;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
----
 PAYMENTID                                 NOT NULL NUMBER(38)
 AMOUNT                                             NUMBER(38)
 REGISTRATIONID                                     NUMBER(38)
 PAYMENTDATE                                        DATE
```

desc certificate;



```
SQL> create table certificate(certificateID int primary key,type varchar(20)
,issueDate date);

Table created.

SQL> desc certificate;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
----
 CERTIFICATEID                             NOT NULL NUMBER(38)
 TYPE                                               VARCHAR2(20)
 ISSUEDATE                                          DATE
```

desc category;

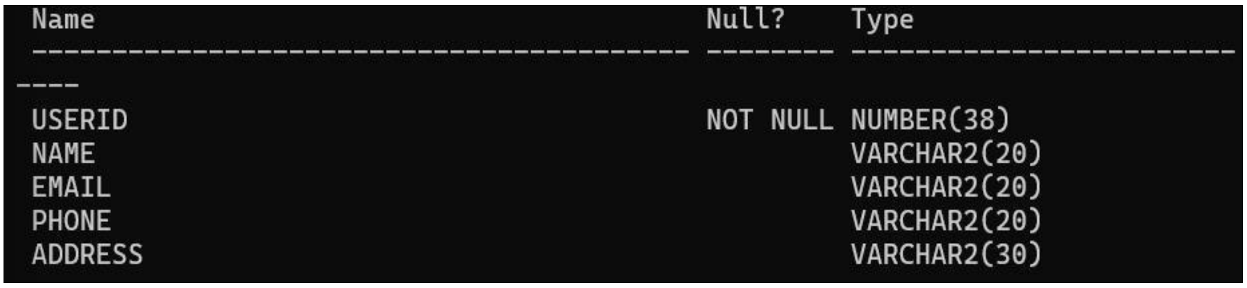


```
SQL> create table category(CategoryID int primary key, categoryName varchar(
20),description varchar(20));

Table created.

SQL> desc category;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
----
 CATEGORYID                                NOT NULL NUMBER(38)
 CATEGORYNAME                                       VARCHAR2(20)
 DESCRIPTION                                        VARCHAR2(20)
```

3. **Add column ‘address’ to table participant:**

   <u>Query:</u>

   alter table participant add address varchar(30);

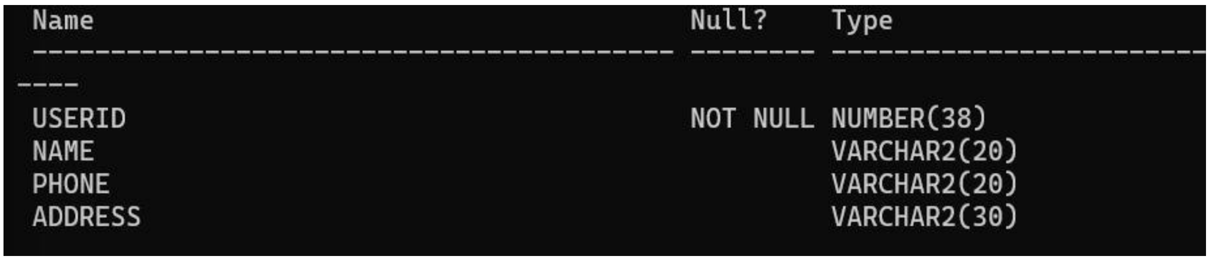


```
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
----
 USERID                                    NOT NULL NUMBER(38)
 NAME                                               VARCHAR2(20)
 EMAIL                                              VARCHAR2(20)
 PHONE                                              VARCHAR2(20)
 ADDRESS                                            VARCHAR2(30)
```

4. **Drop column email from table participant:**

   <u>Query:</u>

   alter table participant drop(email);

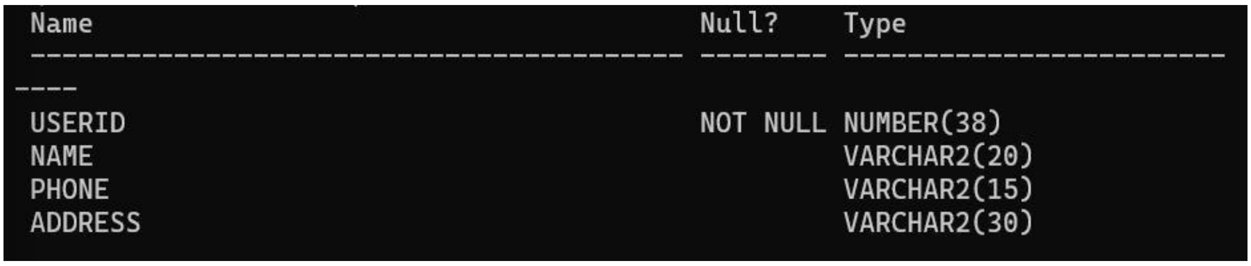


```
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
----
 USERID                                    NOT NULL NUMBER(38)
 NAME                                               VARCHAR2(20)
 PHONE                                              VARCHAR2(20)
 ADDRESS                                            VARCHAR2(30)
```

5. **Modify phone from table participant:**

   <u>Query:</u>

   alter table participant modify(phone varchar(15));

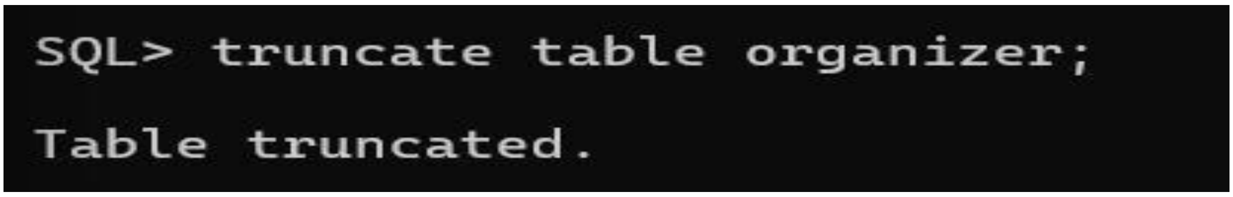


```
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
----
 USERID                                    NOT NULL NUMBER(38)
 NAME                                               VARCHAR2(20)
 PHONE                                              VARCHAR2(15)
 ADDRESS                                            VARCHAR2(30)
```

6. **Truncate organizer table:**

   <u>Query:</u>

   truncate table organizer;

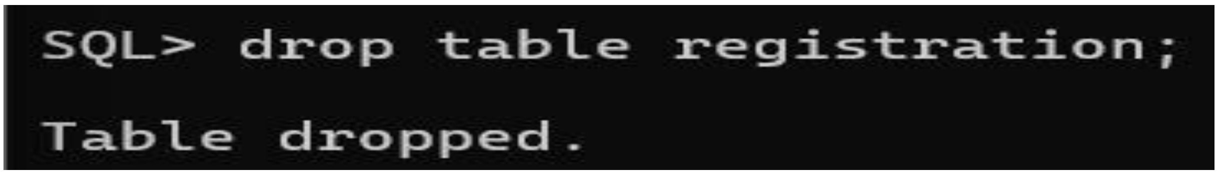


```
SQL> truncate table organizer;

Table truncated.
```

7. **Drop registration table:**

   <u>Query:</u>

   drop table registration;

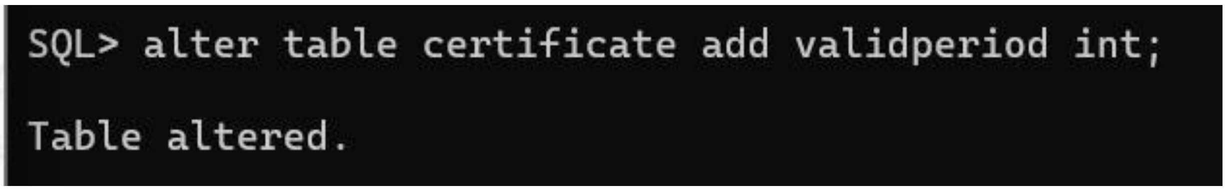


```
SQL> drop table registration;

Table dropped.
```

8 **Add validperiod to certificate table:**

Query:

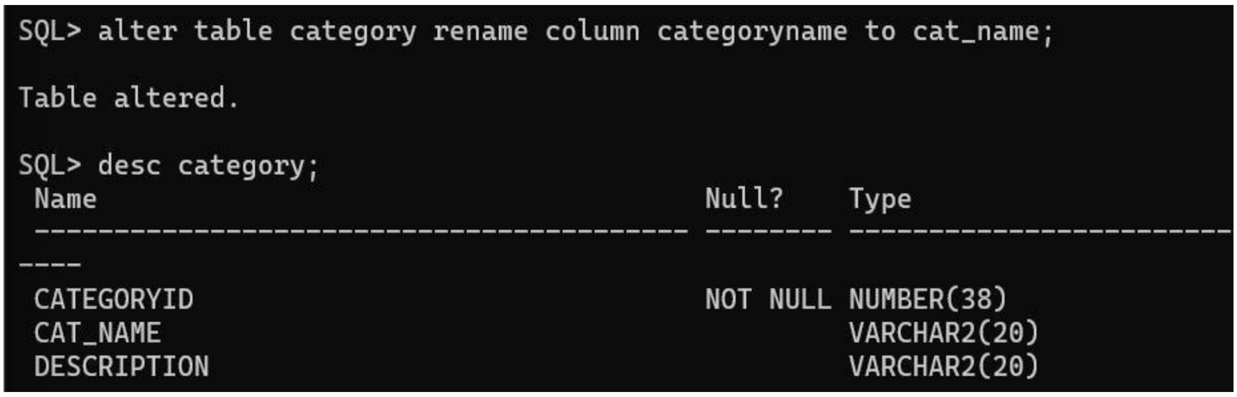
alter table certificate add validperiod int;



```
SQL> alter table certificate add validperiod int;

Table altered.
```

9 **Rename category table:**

Query:



```
SQL> alter table category rename column categoryname to cat_name;

Table altered.

SQL> desc category;
 Name                                      Null?    Type
 ----------------------------------------- -------- -------------------------
----
 CATEGORYID                                NOT NULL NUMBER(38)
 CAT_NAME                                           VARCHAR2(20)
 DESCRIPTION                                        VARCHAR2(20)
```

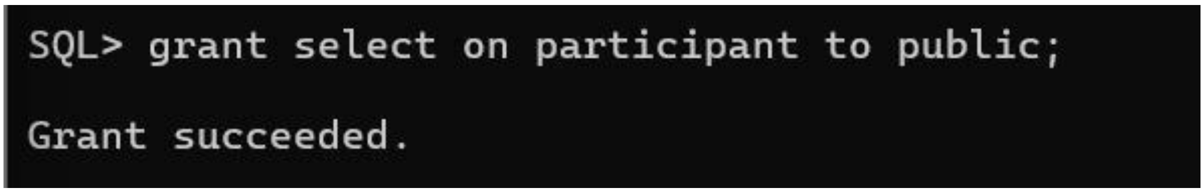
Data Control Language (DCL) Commands:

GRANT:

1. **Grant the permission for the participant table to public:**

Query:

grant select on participant to public;



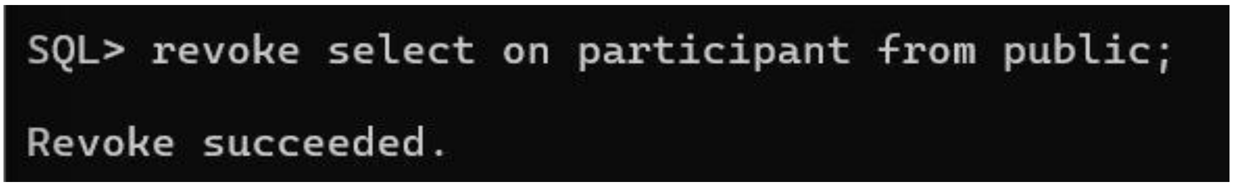
```
SQL> grant select on participant to public;

Grant succeeded.
```

REVOKE:

1. **Revoke the SELECT permission on the participant table from public:**

   Query:

   revoke select on participant from public;



```
SQL> revoke select on participant from public;

Revoke succeeded.
```

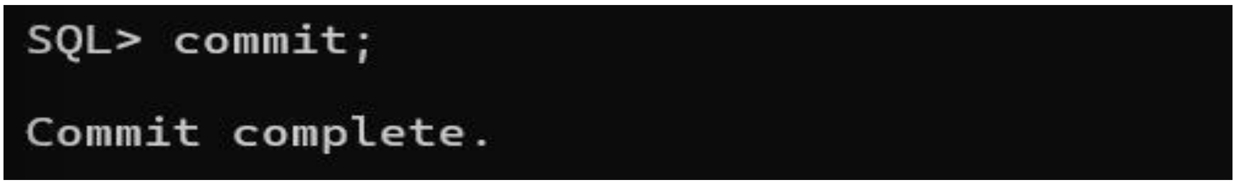
Transaction Control Language (TCL) Commands:

COMMIT:

1. **Perform commit operation:**

   Query:

   insert into participant values(101,'John','957863456','Tiruppur');



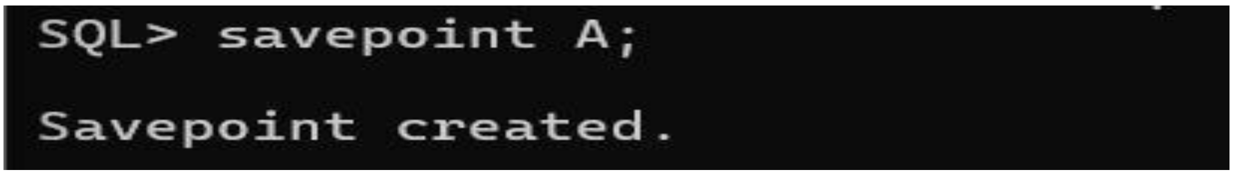
```
SQL> commit;

Commit complete.
```

SAVEPOINT:

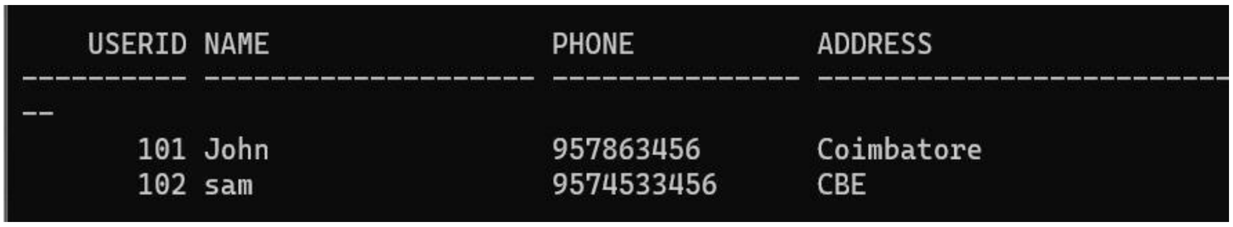
1. **Create a savepoint of name A where userid='101':**

   Query:

   update participant set address='Coimbatore' where userid='101';
   savepoint A;



```
SQL> savepoint A;

Savepoint created.
```



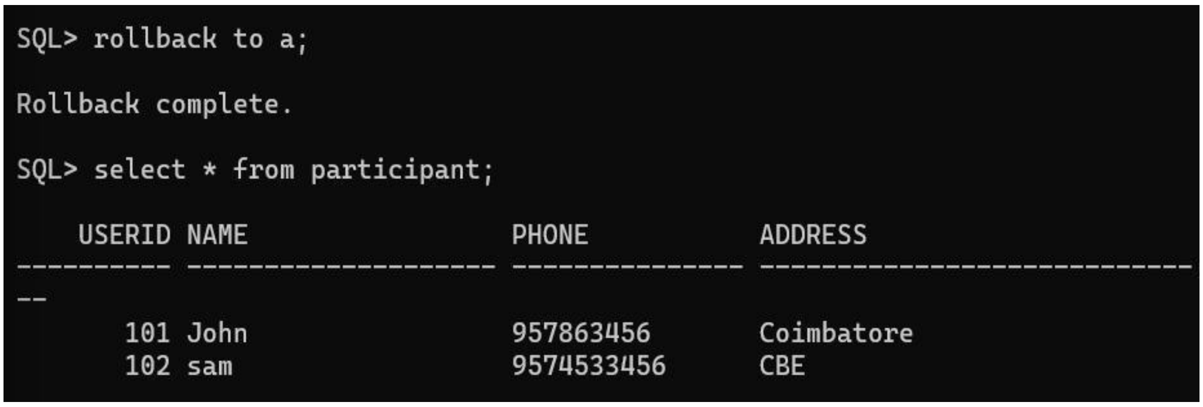
```
    USERID NAME                 PHONE           ADDRESS
---------- -------------------- --------------- -------------------------
--
       101 John                 957863456       Coimbatore
       102 sam                  9574533456      CBE
```

ROLLBACK:

1. **Write a query which returns to savepoint 'A':**

   Query:

   rollback to A;
   select * from participant;



```
SQL> rollback to a;

Rollback complete.

SQL> select * from participant;

    USERID NAME                 PHONE           ADDRESS
---------- -------------------- --------------- ----------------------------
--
       101 John                 957863456       Coimbatore
       102 sam                  9574533456      CBE
```

RESULT:

Thus the various DDL,DCL,TCL commands for the given database have been executed successfully.

EX.NO: 3	QUERIES TO DEMONSTRATE IMPLEMENTATION OF VARIOUS INTEGRITY AND KEY CONSTRAINS
DATE:	

AIM:

To implement various integrity constraints in the given Database.

PROCEDURE:

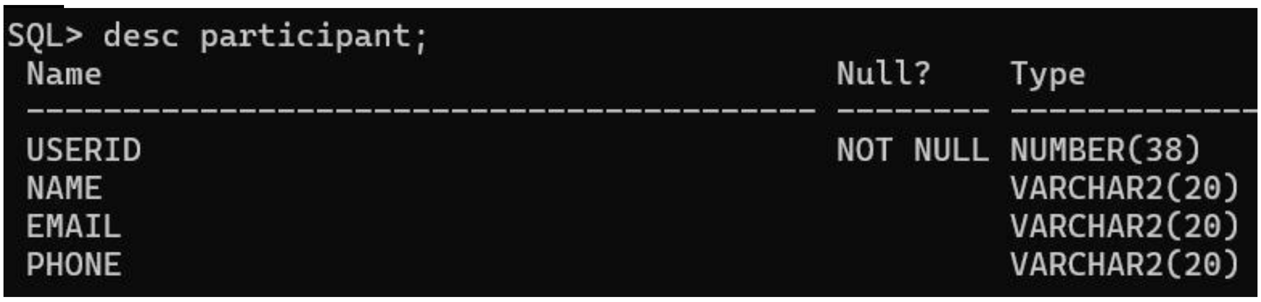
The main steps to be involved are

1. Read the problem statement.
2. Open the oracle terminal.
3. Write the necessary query for the given problem statement.
4. Execute the query
5. Save the results.

EXERCISES: EVENT REGISTRATION SYSTEM

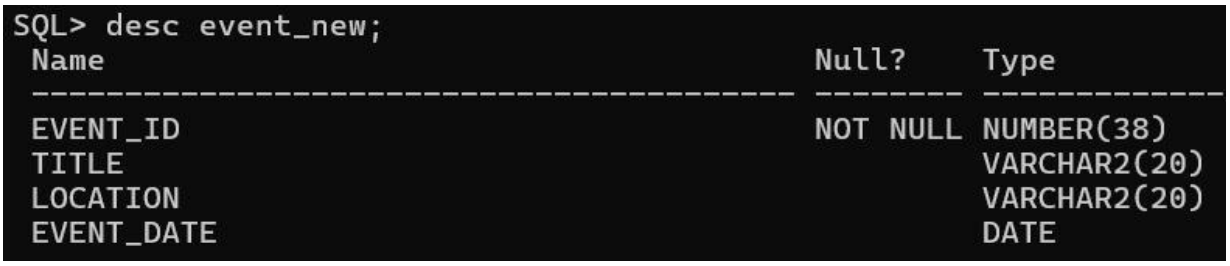
PRIMARY KEY:

alter table participant add primary key (userid);



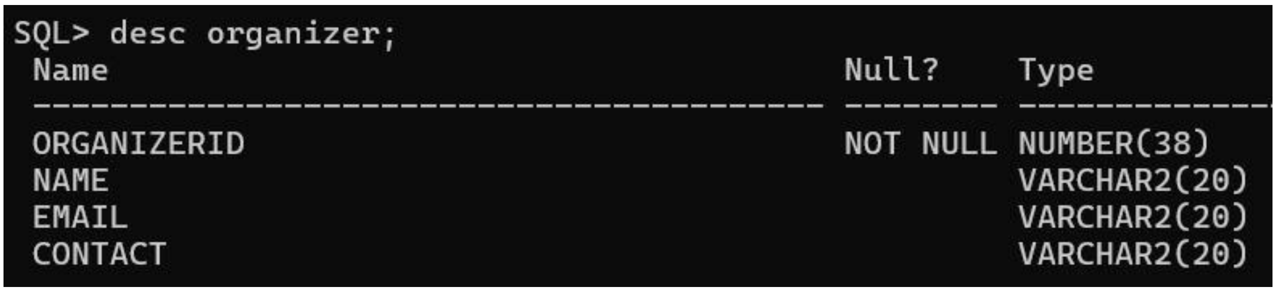
```
SQL> desc participant;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
 USERID                                    NOT NULL NUMBER(38)
 NAME                                               VARCHAR2(20)
 EMAIL                                              VARCHAR2(20)
 PHONE                                              VARCHAR2(20)
```

alter table event_new add primary key (event_id);

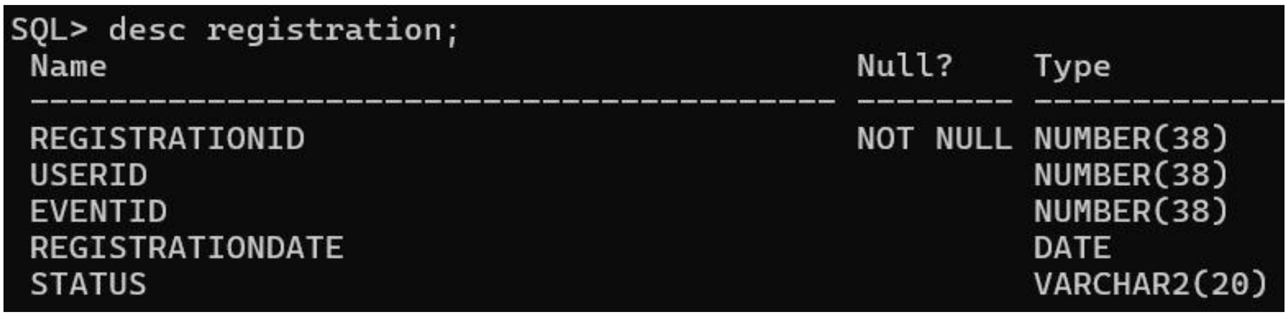


```
SQL> desc event_new;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
 EVENT_ID                                  NOT NULL NUMBER(38)
 TITLE                                              VARCHAR2(20)
 LOCATION                                           VARCHAR2(20)
 EVENT_DATE                                         DATE
```

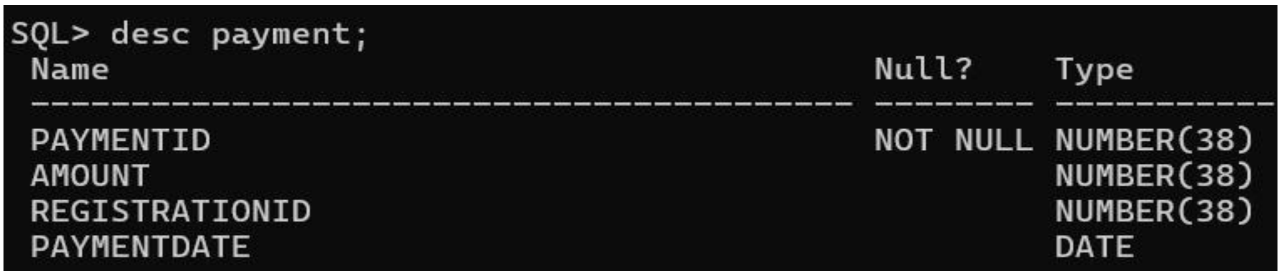
alter table organizer add primary key (organizerid);


```
SQL> desc organizer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORGANIZERID                               NOT NULL NUMBER(38)
 NAME                                               VARCHAR2(20)
 EMAIL                                              VARCHAR2(20)
 CONTACT                                            VARCHAR2(20)
```

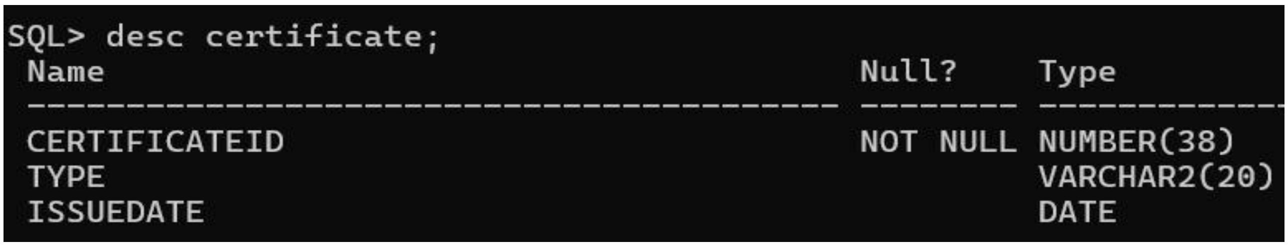
alter table registration add primary key (registrationid);


```
SQL> desc registration;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 REGISTRATIONID                            NOT NULL NUMBER(38)
 USERID                                             NUMBER(38)
 EVENTID                                            NUMBER(38)
 REGISTRATIONDATE                                   DATE
 STATUS                                             VARCHAR2(20)
```

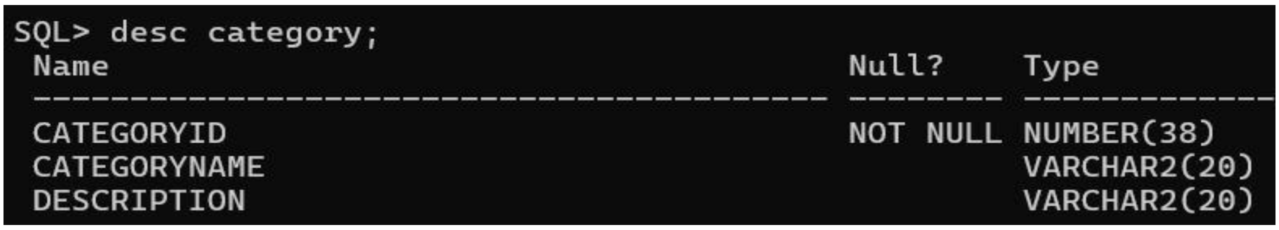
alter table payment add primary key (paymentid);


```
SQL> desc payment;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PAYMENTID                                 NOT NULL NUMBER(38)
 AMOUNT                                             NUMBER(38)
 REGISTRATIONID                                     NUMBER(38)
 PAYMENTDATE                                        DATE
```

alter table certificate add primary key (certificateid);

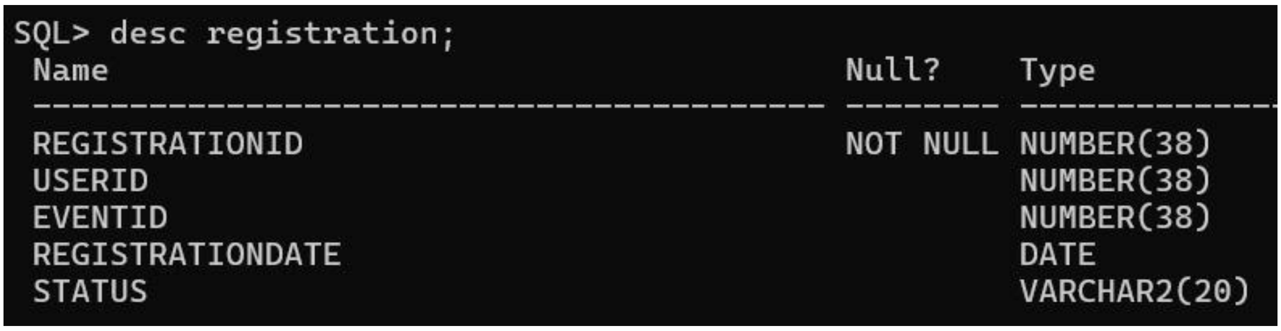

```
SQL> desc certificate;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CERTIFICATEID                             NOT NULL NUMBER(38)
 TYPE                                               VARCHAR2(20)
 ISSUEDATE                                          DATE
```

alter table category add primary key (categoryid);


```
SQL> desc category;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CATEGORYID                                NOT NULL NUMBER(38)
 CATEGORYNAME                                       VARCHAR2(20)
 DESCRIPTION                                        VARCHAR2(20)
```

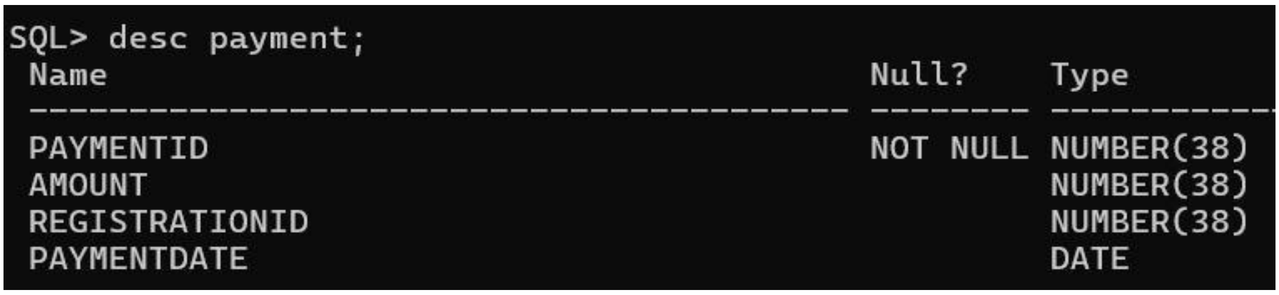
foreign key:

alter table registration add foreign key (userid) references participant(userid);



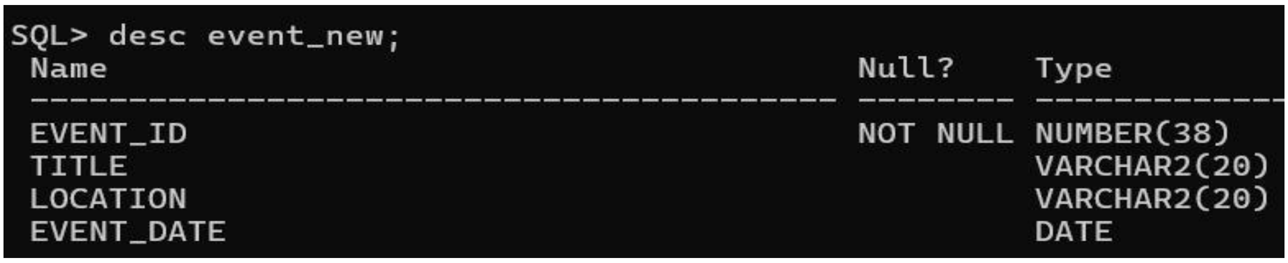
```
SQL> desc registration;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 REGISTRATIONID                            NOT NULL NUMBER(38)
 USERID                                             NUMBER(38)
 EVENTID                                            NUMBER(38)
 REGISTRATIONDATE                                   DATE
 STATUS                                             VARCHAR2(20)
```

alter table payment add foreign key (registrationid) references registration(registrationid);



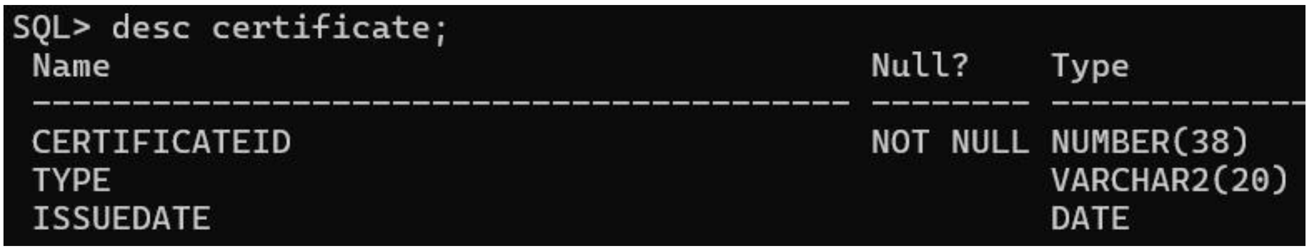
```
SQL> desc payment;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PAYMENTID                                 NOT NULL NUMBER(38)
 AMOUNT                                             NUMBER(38)
 REGISTRATIONID                                     NUMBER(38)
 PAYMENTDATE                                        DATE
```

alter table event_new add foreign key (event_id) references event_new(event_id);



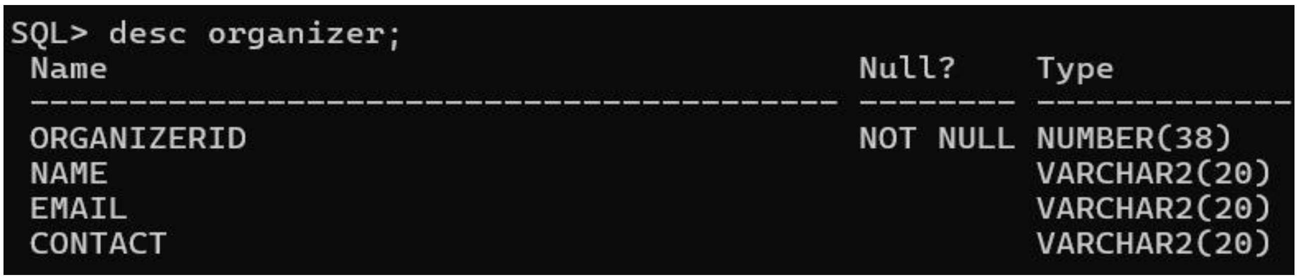
```
SQL> desc event_new;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EVENT_ID                                  NOT NULL NUMBER(38)
 TITLE                                              VARCHAR2(20)
 LOCATION                                           VARCHAR2(20)
 EVENT_DATE                                         DATE
```

alter table certificate add foreign key (certificateid) references certificate(certificateid);



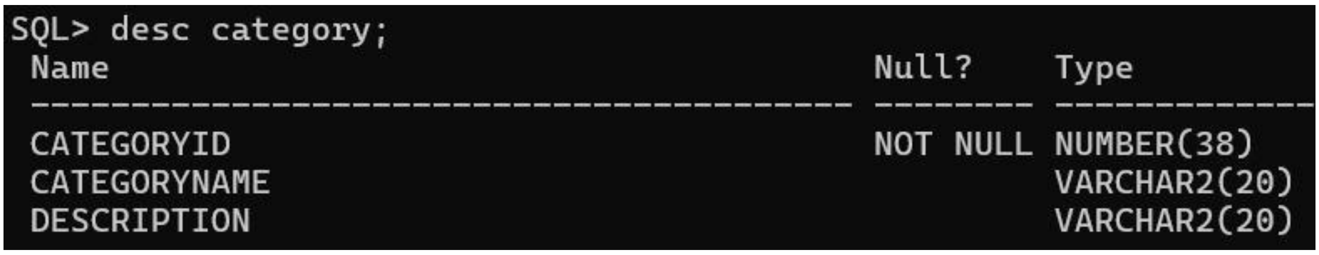
```
SQL> desc certificate;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CERTIFICATEID                             NOT NULL NUMBER(38)
 TYPE                                               VARCHAR2(20)
 ISSUEDATE                                          DATE
```

alter table organizer add foreign key (organizerid) references organizer(organizerid);



```
SQL> desc organizer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ORGANIZERID                               NOT NULL NUMBER(38)
 NAME                                               VARCHAR2(20)
 EMAIL                                              VARCHAR2(20)
 CONTACT                                            VARCHAR2(20)
```

alter table category add foreign key (categoryid) references category(categoryid);


```
SQL> desc category;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
 CATEGORYID                                NOT NULL NUMBER(38)
 CATEGORYNAME                                       VARCHAR2(20)
 DESCRIPTION                                        VARCHAR2(20)
```

<u>unique key:</u>

<u>queries:</u>

alter table participant add unique (phone);

alter table event_new add unique (title);

alter table organizer add unique (email);

alter table category add unique (categoryname);

alter table certificate add unique (type);

<u>not null constraint:</u>

<u>queries:</u>

alter table participant modify email varchar(20) not null;
alter table event_new modify title varchar(20) not null;
alter table organizer modify contact varchar(20) not null;
alter table category modify descriptionvarchar(20) not null; alter table
certificate modify type varchar(20) not null;

<u>check constraint:</u>

<u>query:</u>

alter table payment add check (amount > 0);

RESULT:

Thus the various integrity constraint commands for the given database have been executed successfully.

EX.NO: 4	PRACTICE ON VARIOUS DML COMMANDS TO WRITE A QUERY TO INTERACT WITH DATABASE
DATE:	

AIM:

To implement various DML commands to interact with the given Database.

PROCEDURE:

The main steps to be involved are

1. Read the problem statement.
2. Open the oracle terminal.
3. Write the necessary query for the given problem statement.
4. Execute the query.
5. Save the results.

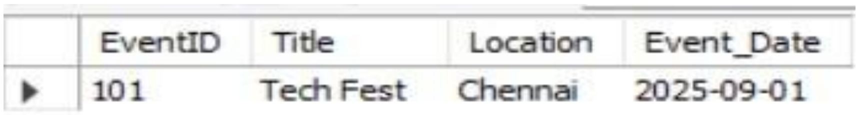
EVENT MANAGEMENT SYSTEM:

DML COMMANDS:

1)Insert a single row into the above tables.

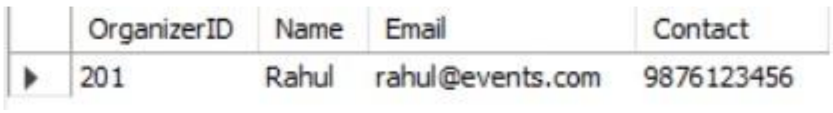
Query:

insert into event (eventid, title, location, event_date) values (101, 'tech fest', 'chennai', '2025-09-01');



	EventID	Title	Location	Event_Date
▶	101	Tech Fest	Chennai	2025-09-01

insert into organizer (organizerid, name, email, contact) values (201, 'rahul','rahul@events.com', '9876123456');

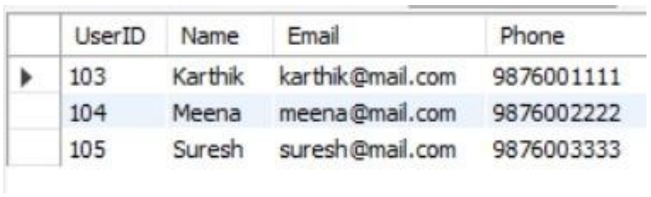


	OrganizerID	Name	Email	Contact
▶	201	Rahul	rahul@events.com	9876123456

2)Insert multiple rows to the above tables.

Query:

```
insert into participant (UserID, Name, Email, Phone)
values
        (103, 'Karthik', 'karthik@mail.com', '9876001111'),
        (104, 'Meena', 'meena@mail.com', '9876002222'),
        (105, 'Suresh', 'suresh@mail.com', '9876003333')select * from dual;
```

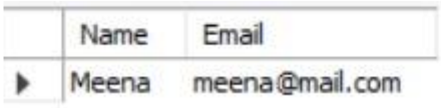


	UserID	Name	Email	Phone
▶	103	Karthik	karthik@mail.com	9876001111
	104	Meena	meena@mail.com	9876002222
	105	Suresh	suresh@mail.com	9876003333

3)Select the data from the tables.

1) Show the Name,Email in the table participant.

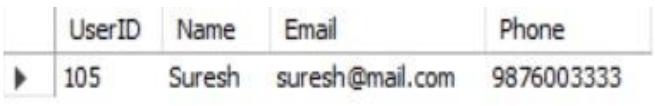
```
select Name, Email
from participant
where UserId='104';
```



	Name	Email
▶	Meena	meena@mail.com

2) Show the data in table participant where UserId=105

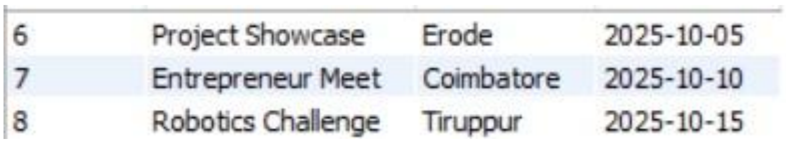
```
select * from participant where userID=105;
```



	UserID	Name	Email	Phone
▶	105	Suresh	suresh@mail.com	9876003333

3)Show only the eventid>5 in the table event.

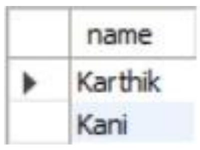
```
select * from event where eventid>5;
```



6	Project Showcase	Erode	2025-10-05
7	Entrepreneur Meet	Coimbatore	2025-10-10
8	Robotics Challenge	Tiruppur	2025-10-15

4) Select the students whose name like 'K%' in table participant.

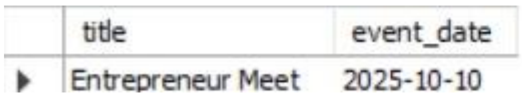
select name from participant where name like 'K%';



name
Karthik
Kani

5) Show the details of eventid=7 from the table event.

select title,event_date from event where eventid=7;



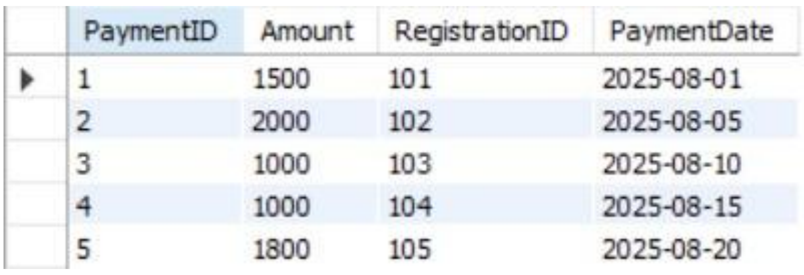
title	event_date
Entrepreneur Meet	2025-10-10

4) Update the values in the tables. (Single & Multi Rows)

1)Update the amount of the payment in payment table.

Query:

update payment set amount =1000 where paymentid=4;

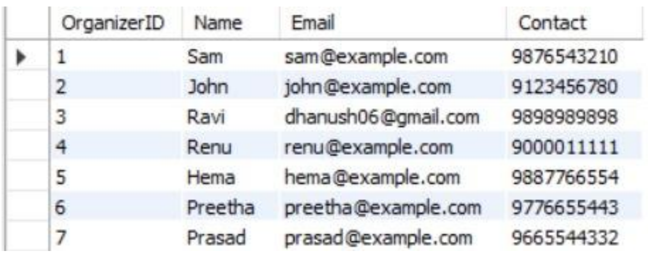


PaymentID	Amount	RegistrationID	PaymentDate
1	1500	101	2025-08-01
2	2000	102	2025-08-05
3	1000	103	2025-08-10
4	1000	104	2025-08-15
5	1800	105	2025-08-20

2)Update the email of the organizer whose organizerid=3.

Query:

update organizer set email='dhanush06@gmail.com'
where organizerid=3;



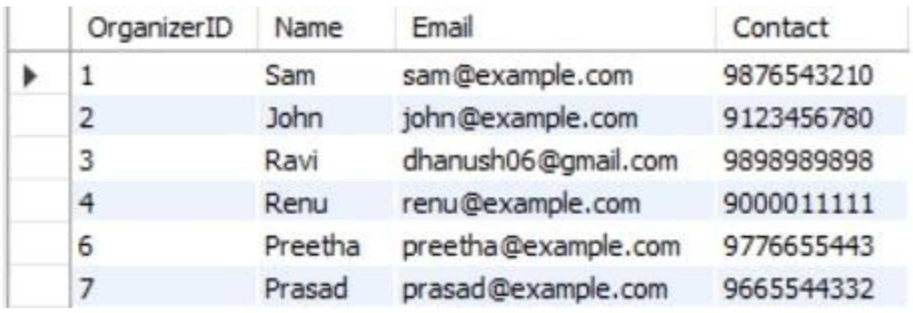
OrganizerID	Name	Email	Contact
1	Sam	sam@example.com	9876543210
2	John	john@example.com	9123456780
3	Ravi	dhanush06@gmail.com	9898989898
4	Renu	renu@example.com	9000011111
5	Hema	hema@example.com	9887766554
6	Preetha	preetha@example.com	9776655443
7	Prasad	prasad@example.com	9665544332

5)Delete the values in the tables.

1)Delete the row in organizer whose name is hema.

Query:

delete from organizer where name='hema';



OrganizerID	Name	Email	Contact
1	Sam	sam@example.com	9876543210
2	John	john@example.com	9123456780
3	Ravi	dhanush06@gmail.com	9898989898
4	Renu	renu@example.com	9000011111
6	Preetha	preetha@example.com	9776655443
7	Prasad	prasad@example.com	9665544332

RESULT:

Thus, the various DML commands for the given problem statements in the database have been executed successfully.

EX.NO: 5	**PRACTICE ON AGGREGATE FUNCTIONS AND VIEWS**
DATE:	

AIM:

To implement various aggregate functions and views with the given Database.

SYSTEM CONFIGURATION:

❖ Ram	: 2 GB
❖ Processor	: i3 / Core 2 Quad / Core 2 Duo
❖ Operating system	: Window 7 / Window XP Service Pack 3
❖ Software	: Oracle 11g / MySQL

PROCEDURE:

The main steps to be involved are

1. Read the problem statement.
2. Open the oracle terminal.
3. Write the necessary query for the given problem statement.
4. Execute the query
5. Save the results

BOOKS DATABASE:

1. List avg, max and min copies of the books.
2. List the name of the books starts by d and count more than 5.
3. List minimum no book copies.
4. Create a view from single table containing all columns from the base table.
5. Create a view from single table with selected columns.
6. Create a view from two tables with all columns.

LIBRARY DATABASE:

1. List avg,max and min price of the books.
2. List the name of the books starts by d and price more than 5.
3. Create a view from two tables with selected columns.
4. Use DML operations (insert,delete,update) on the above created views.
5. Create a synonym for a table created by other user.
6. Use DML operations (insert,delete,update) on the above created synonym.
7. Create a sequence to generate unique value for tid field in thelibrary table while inserting.

STUDENT DATABASE:

1. List avg,max and min fee of the courses.
2. List the name of the students who paid the fee between xxxxx and yyyyy dates
3. Create a sequence for Reg.No of Student table.
4. Use sequence in DML.
5. Alter the sequence.
6. Drop the sequence

VIVA QUESTIONS:

1. List significance of views.
2. Give the syntax for synonym.
3. How do you alter a sequence?
4. List the advantages of sequence.
5. What is meant by CACHE and NOCACHE?
6. How do you alter a sequence?
8. How do you find MAX and MIN?
9. What are the sorting clauses available?

RESULT:

Thus, the various commands on aggregate functions and views for the given problem statements in the database have been executed successfully.

EX.NO: 6	IMPLEMENT JOINS, NESTED QUERIES AND STORED PROCEDURES
DATE:	

AIM:

To implement various joins, nested queries and stored procedures with the given Database.

SYSTEM CONFIGURATION:

❖ Ram	: 2 GB
❖ Processor	: i3 / Core 2 Quad / Core 2 Duo
❖ Operating system	: Window 7 / Window XP Service Pack 3
❖ Software	: Oracle 11g / MySQL

PROCEDURE:

The main steps to be involved are

1. Read the problem statement.
2. Open the oracle terminal.
3. Write the necessary query for the given problem statement.
4. Execute the query
5. Save the results

BOOKS DATABASE:

1. Find the details of the author who has the maximum no of book copies.
2. Find the details of the author who has the maximum no of book copies in “XXXXX” branch.
3. Find the details of the author whose book has been borrowed in maximum nos.
4. Find the details of the author whose book has been borrowed in maximum nos by customers in the address “XXXXXX”.
5. Find the details of the borrower whose borrowed duration is the maximum in the particular library branch “XXXXXX”
6. Display the book details in descending based on no of books.
7. Display maximum no of days borrowed for each title.

LIBRARY DATABASE:

1. Display the book details ordered after the date “XXXXX” by the customers from the address “YYYYYY”.
2. Display the return details of the costliest book ordered by customers from “XXXXXX” address and ordered after the date “YYYYY”
3. Find the Maximum, minimum and average price of books returned by “XXXXXXX” customers.

4. Find the cheapest book in each year.
5. Find the details of order of the cheapest books in each year.

STORED PROCEDURES:

1. Consider the relations PASSENGER (PID, PNAME, DOB, GENDER) BOOK_TICKET (PID, ROUTENO, JOURNEY_DATE, SEAT_NO)
   Create a PL / SQL stored procedure that accepts journey_date and displays list of passengers booked ticket on that date.
2. Consider the following employee and department tables.
   EMPLOYEE(empno, ename, designation, manager, hiredate, salary, commission,deptno)
   Write a procedure to update the salaries by given amount.
3. Consider the following relations for a transport management system application: BUS (ROUTENO, SOURCE, DESTINATION)
   DRIVER (DID, DNAME, DOB, GENDER)
   ASSIGN_ROUTE (DID, ROUTENO, JOURNEY_DATE)
   Create a procedure that displays the details of all drivers

VIVA QUESTIONS:

1. What is a nested query?
2. What is join query?
3. Can we give an output of an query as an input to another query?
4. What is Cartesian product?
5. What is spurious tuple?
6. How do restrict duplicates when joining two tables?
7. What is a natural join?
8. Give example for left outer join.
9. Explain about right outer join.
10. What is division operation?
11. What is a stored procedure?
12. What is a stand-alone procedure?
13. What are the modes of parameters that can be passed to a procedure?
14. How to find the list of procedures that were available in the database?
15. What are the purposes of stored procedure?
16. Give the syntax of procedure

RESULT: Thus, the various commands on joins, nested queries and stored procedures in the database have been executed successfully.

EX.NO: 7	PRACTICE ON PROCEDURAL EXTENSIONS (FUNCTIONS, CURSORS, TRIGGERS)
DATE:	

AIM:

To implement various procedural extensions (Functions, Cursors, Triggers) with the given Database.

SYSTEM CONFIGURATION:

❖ Ram	: 2 GB
❖ Processor	: i3 / Core 2 Quad / Core 2 Duo
❖ Operating system	: Window 7 / Window XP Service Pack 3
❖ Software	: Oracle 11g / MySQL

PROCEDURE:

The main steps to be involved are

1. Read the problem statement.
2. Open the oracle terminal.
3. Write the necessary query for the given problem statement.
4. Execute the query
5. Save the results

FUNCTIONS:

1. Consider the relation stu_details (reg_no, stu_name, DOB, address, city). Write a pl/sqlprogram to find the address of a particular student using functions.
2. Consider the relation mark_details (reg_no, mark1, mark2, mark3, total). Write a pl/sql program to find the sum &avg marks of all the student using procedures.
3. For the relation emp_details (emp_no, emp_name, DOB, address, doj, mobile_no, dept_no, salary). Write a pl/sql program to display the salary of a particular employee using functions
4. consider the relation Phone_book(ph_no,name,door_no,street,place). Write a pl/sql program to find the address of a particular customer using functions.
5. Consider the relations

SAILOR (SID, NAME, DOB, GENDER)

BOAT (BID, BTYPE, BNAME, COLOR)

SAILS (SID, BID, DOT, SHIFT)

Create a PL / SQL stored function that accepts SID and returns the name of sailor

6. Consider the following relations for an order processing

application: CUSTOMER (CID, NAME)

PRODUCT (PCODE, PNAME, UNIT_PRICE)

CUST_ORDER (OCODE, ODATE, CID)
ORDER_PRODUCT (OCODE, PCODE, NOU)
Create a function that accepts PCODE, Unit_Price and NOU. Calculate the total_cost of the ordered product. Return the total_cost.

7. Consider the following relational schema for a Loan database application: Customer (Custid, Custname, Age, phno)
Loan (Loanid, Amount, Custid)
Develop a function named Customer_Loan which accepts Loanid as input and displays Custid, CustName and loan_amount.

CURSORS:

1. Write a PL SQL cursor block to update the due date of the book_loans table for those who have borrowed books after the date “XXXXX” and display no rows affected.
2. Copy customers table using PL SQL cursors.
3. Implement PL SQL cursor while,for loops and %ROWCOUNT to count the no of rows affected by the following operations.
   i. Update the book quantity
   ii. Increase the course fee for a particular department.

TRIGGERS:

1. Consider the table EMPLOYEE(empno, ename, designation, manager, hiredate, salary, commission,
deptno). Write a trigger to ensure that salary of an employee is always greater than the commission.
2. Consider the following relational schema for a banking database application: CUSTOMER (CID, CNAME)
BRANCH (BCODE, BNAME)
ACCOUNT (ANO, ATYPE, BALANCE, CID, BCODE)
An account can be a savings account or a current account. Check ATYPE in ‘S’ or ‘C’. A customer can have both types of accounts.
TRANSACTION (TID, ANO, TTYPE, TDATE, TAMOUNT)
TTYPE CAN BE ‘D’ OR ‘W’
D- Deposit; W – Withdrawal
Develop a database trigger that will update the value of BALANCE in ACCOUNT table when a record is inserted in the transaction table. Consider the following cases:
i. If TTYPE =’D’ the value of BALANCE in the ACCOUNT table must be

incremented by the value of TAMOUNT

ii. If TTYPE ='W' the value of BALANCE in the ACCOUNT table
must be decremented by the value of TAMOUNT.

If a minimum balance of Rs. 2000/- will be maintained for a savings account and
a minimum balance of Rs. 5000/- will be maintained for a current account else
appropriate messages must be displayed

3. Consider the following relational schema for a Loan database
application: Customer (Custid, Custname, Age, phno)
Loan (Loanid, Amount, Custid)

Create a database trigger that will not permit a customer to get more than 3 loans.

4. Consider the following relational schema for a Sales database
application: Product (Prodid, Prodesc, Price, Stock)
Purchase (Purid, Proid, qty, supplierName)
Sales (Saleid, Proid, qty, custname)

Create a Trigger which reduces the stock of Product that is been inserted in sales and print if it is out of stock (stock <Reord)

5. Consider the following relational schema for Products Order database
application: Products (p_id, p_name, retail_price, qty_on_hand)
Orders (order_id, order_date)
Order_details (order_number, product_number, qty_ordered)

Create a database TRIGGER, which deletes the order from Orders table, AFTER the deletion of corresponding order_number in Order_details

VIVA QUESTIONS:

1. What is RAISE_APPLICATION_ERROR?
2. Give the sql statement to list the triggers associated with a particular table.
3. How to disable and enable triggers?
4. How to drop a trigger?
5. What are two virtual tables available during database trigger execution?
6. What happens if a procedure that updates a column of table X is called in a database trigger of the same table?
7. Write the order of precedence for validation of a column in a table?
8. What are the instances when triggers are appropriate?
9. How do you use trigger AFTER keyword?
10. Why do we need trigger restrictions?
11. How a function can be called from a SELECT statement?

12. Can function return a value?
13. Name the tables where the characteristics of functions are stored?
14. How many values can be returned by functions?
15. Is it possible to create recursive function?
16. What is a PL SQL Cursor?
17. Define explicit cursor
18. Give the significance of implicit cursor
19. What is For Loop cursor statement?
20. Give the advantages and applications of cursors in a DB.
21. Give the syntax for PL SQL for loop.
22. Give the syntax for PL SQL function.
23. Give the syntax for PL SQL while loop?
24. How do you declare a variable?
25.1How do you start a block?

RESULT: Thus, the various problem statements on Functions, Cursors, Triggers in the database have been implemented successfully.

EX.NO: 8	**DOCUMENT DATABASE CREATION USING MONGODB**
DATE:	

AIM:

To implement various procedural extensions (Functions, Cursors, Triggers) with the given Database.

SYSTEM CONFIGURATION:

❖ Ram	: 2 GB
❖ Processor	: i3 / Core 2 Quad / Core 2 Duo
❖ Operating system	: Window 7 / Window XP Service Pack 3
❖ Software	: MongoDB

PROCEDURE:

The main steps to be involved are

1. Read the problem statement.
2. Open the MongoDB terminal.
3. Write the necessary query for the given problem statement.
4. Execute the query
5. Save the results

STUDENT DATABSE:

1. Implemet MongoDB CRUD operation on the student database.Perform text search operation in the student data base for the substring “xxxx”,”yyyy”,”zzzz”.

VIVA QUESTIONS:

1. What is a Document in MongoDB?
2. What is a Collection in MongoDB?
3. What is the Mongo Shell?
4. What are some features of MongoDB?
5. How do you Update a Document?

RESULT: Thus, the various CRUD operation on the MongoDB database have been implemented successfully.

EX.NO: 9	MINI PROJECT (APP DEVELOPMENT USING ORACLE DB)
DATE:	

AIM:

To implement various procedural extensions (Functions, Cursors, Triggers) with the given Database.

SYSTEM CONFIGURATION:

❖ Ram	: 2 GB
❖ Processor	: i3 / Core 2 Quad / Core 2 Duo
❖ Operating system	: Window 7 / Window XP Service Pack 3
❖ Software	: Visual Studio/ Android Studio /Oracle 11g / MySQL /JDK 7

PROCEDURE:

The main steps to be involved are

1. Read the problem statement.
2. Perform requirement analysis
3. Design the database, user interface models.
4. Perform database connection
5. Test the project
6. Implement the Project.

SAMPLE PROJECT

CREATE MOBILE BACKEND

A mobile backend is the server-side companion to your mobile application. It provides secure access to services like storage, notifications, user management, and custom APIs.

To create a mobile backend do the following:

- Log in your MCS instance and click the menu and select Mobile Backends.
- Click the New Mobile Backend button.
- Enter MCS_QL_Test and click Create.

Select Settings and copy the following to a handy place on your system.

- Mobile Backend ID
- Anonymous Key
- Base URL

 Keep your Backend open because you will be using it in the next steps.

CREATE STORAGE COLLECTION

Now you'll create a storage collection in MCS and add an image to it. This collection will serve as container for your mobile application.

- In the left navigation bar of the mobile backend, click Storage.
- Click New Collection.
- Create a collection called Customer_Images.
- In the New Collection dialog, select Shared.
- Click Create
- In the Read-Only field, type FIF_Technician.
- This ensures that only users with that role have access to that collection, including the user you've set up in the previous part.
- Select the Content tab.
- Click Upload Files and use the file chooser to select a file to upload, preferably an image.
- After the object is loaded, you'll see it listed on the page.
- Copy the ID of that object and paste it somewhere handy. You'll need it a bit later.
- Click Mobile Backends and open the MCS_QL_Test backend you created in the Create Mobile Backend part.
- Click Storage from the Mobile Backend left menu and then click Select Collections.
- Type in the Collections field Customer_Images and then click Select to associate this collection with your Mobile Backend.

CREATE A TEST USER

In this part you create a test user to test the running application.

If you did not leave your mobile backend open in a new tab, go to the mobile backend by doing the following in MCS: Click the side menu, Select Applications, Mobile Backends, then select your mobile backend, and click Open.

- In the left navbar of your mobile backend, click Mobile User Management.
- Click Users and then click New User.
- Enter bob as Username.
- Enter a valid Email address. Your temporary password will be sent to the email address you enter.
- Enter Bob as First Name and Jones as Last Name.
- Select the FIF_Technician role from the Roles box by clicking on the Roles text box.
- Click Create

CONFIGURE THE APP

In this section, we'll set up your application to work with MCS.

Use the user name and password you created in the Get Test User step.

MCS allows you to use either OAuth or BasicAuth for authentication. For this lab, we use BasicAuth.

Note: The credentials differ by environment. An environment is a runtime container that holds your mobile backends, APIs, and other artifacts. You typically start work in a development environment, and eventually deploy your artifacts to a staging or production environment.

Create a new directory as follows: C:\Projects\MCS\Android\<YourName>

Unpack the Getting started project that you downloaded.

Open Android Studio and select Import project (Eclipse, ADT, Gradle, etc.)

Select build.gradle in the Select Eclipse or Gradle Project Import dialog box.

Wait until the project import and gradle sync process finish.

In the project, open main/assets/oracle_mobile_cloud_config.xml and replace the following items:

- Your_Mobile_Backend_Name.
- Your_Base_Url
- Your_Mobile_Backend_ID
- Your_Anonymous_Key

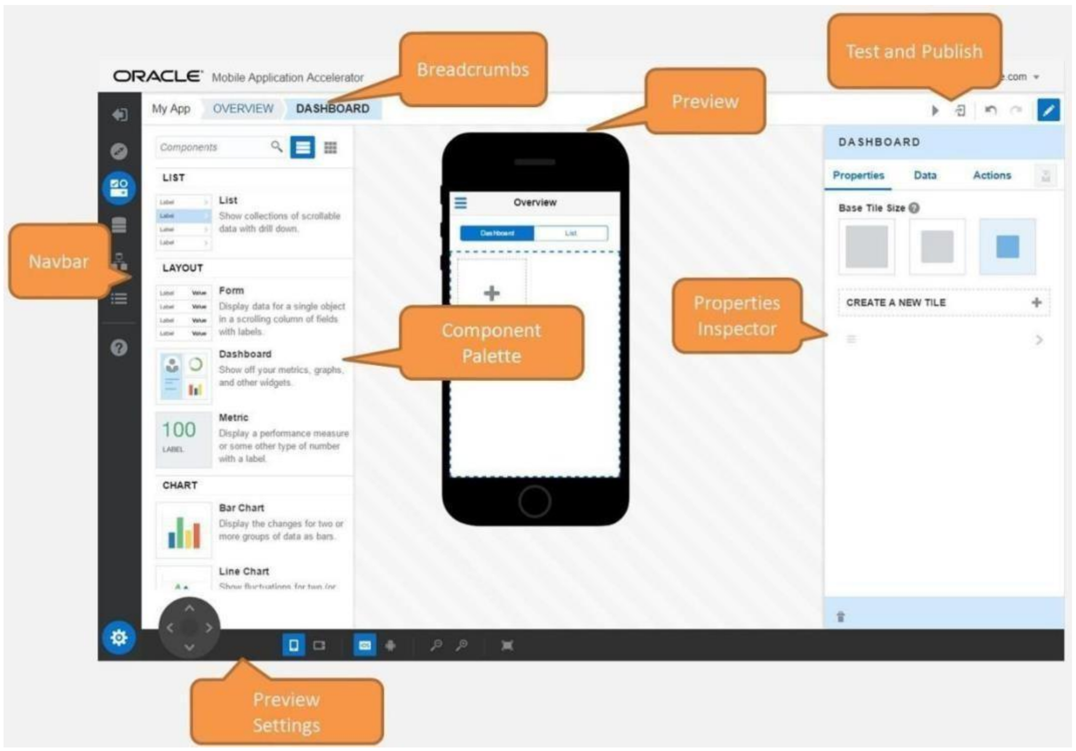
```
<?xml version="1.0" encoding="utf-8" ?>
   <mobileBackends>
   <!--Here are the settings for a single mobileBackend.
   For those apps with multiple mobileBackends, you might need to add
   settings one by one below-->
   <mobileBackend>
      <!--Add the name and version of your mobileBackend as values here-->
      <mbeName>YOUR_MOBILE_BACKEND_NAME</mbeName>
      <mbeVersion>1.0</mbeVersion>
      <!--Set to true if the current mobileBackend is the default one-->
      <default>true</default>
      <!--The base URL goes here-->
      <baseUrl>YOUR_BASE_URL</baseUrl>
      <!--Set it true if you want to get analytics information from UI-->
      <enableAnalytics>true</enableAnalytics>
      <!--Set it true if you want to get logging information in Logger-->
      <enableLogger>true</enableLogger>
      <authorization>
         <offlineAuthenticationEnabled>true</offlineAuthenticationEnabled>
         <authenticationType>basic</authenticationType>
         <basic>
            <mobileBackendID>YOUR_MOBILE_BACKEND_ID</mobileBackendID>
            <anonymousKey>YOUR_ANONYMOUS_KEY</anonymousKey>
         </basic>
       </authorization>
    </mobileBackend>
</mobileBackends>
```

RUN THE APP

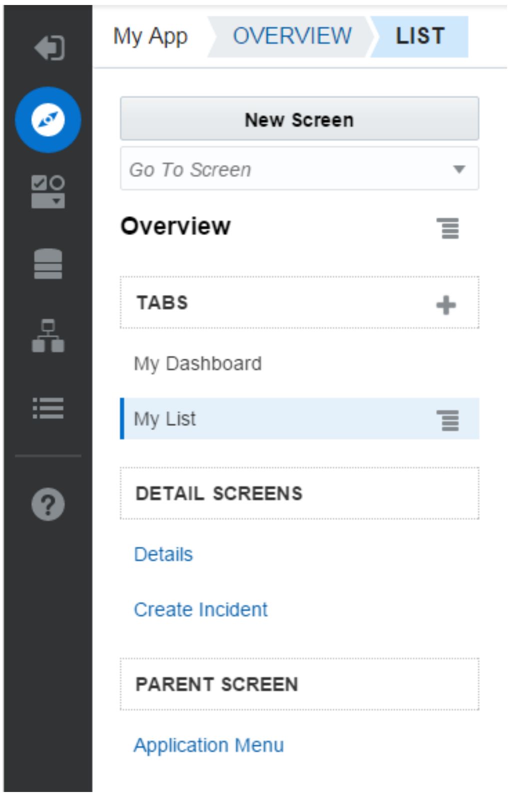
Before running the app in Android studio:

- Click Tools
- Click Android
- Click AVD Manager
- In the AVD Manager dialog box, click Create Virtual Device...
- Select Nexus 5X and click Next.
- Accept the default values and click Next.
- Verify the configuration and change it as needed and click Finish when ready.
- Your device must appear in Your Virtual Devices dialog box.

- From the Actions column, click the Start button depicted as a green arrowhead.
- In Android Studio, choose Run -> Run 'Getting Started'.
- Android Studio will build the application and then launch it in the Android emulator.
- Select the Nexus 5X device you created in the AVD Manager, and click OK.
- In the running application enter bob as user name and the password you received by email, and click Sign In.
- Click Download Incident and then Download Photo.



Description of the illustration composer



VIVA QUESTIONS:

1. What is database driver?
2. What is JDBC?
3. What is ODBC?
4. What is a place holder?
5. What is an IDE?
6. What are the major components of DB connection?
7. How to authenticate application to DB connection?
8. Design a schema for the given application.
9. Draw ER diagram for the given application.
10. What is an API?

RESULT:

Thus, the Customer Login portal App has been Implemented successfully.