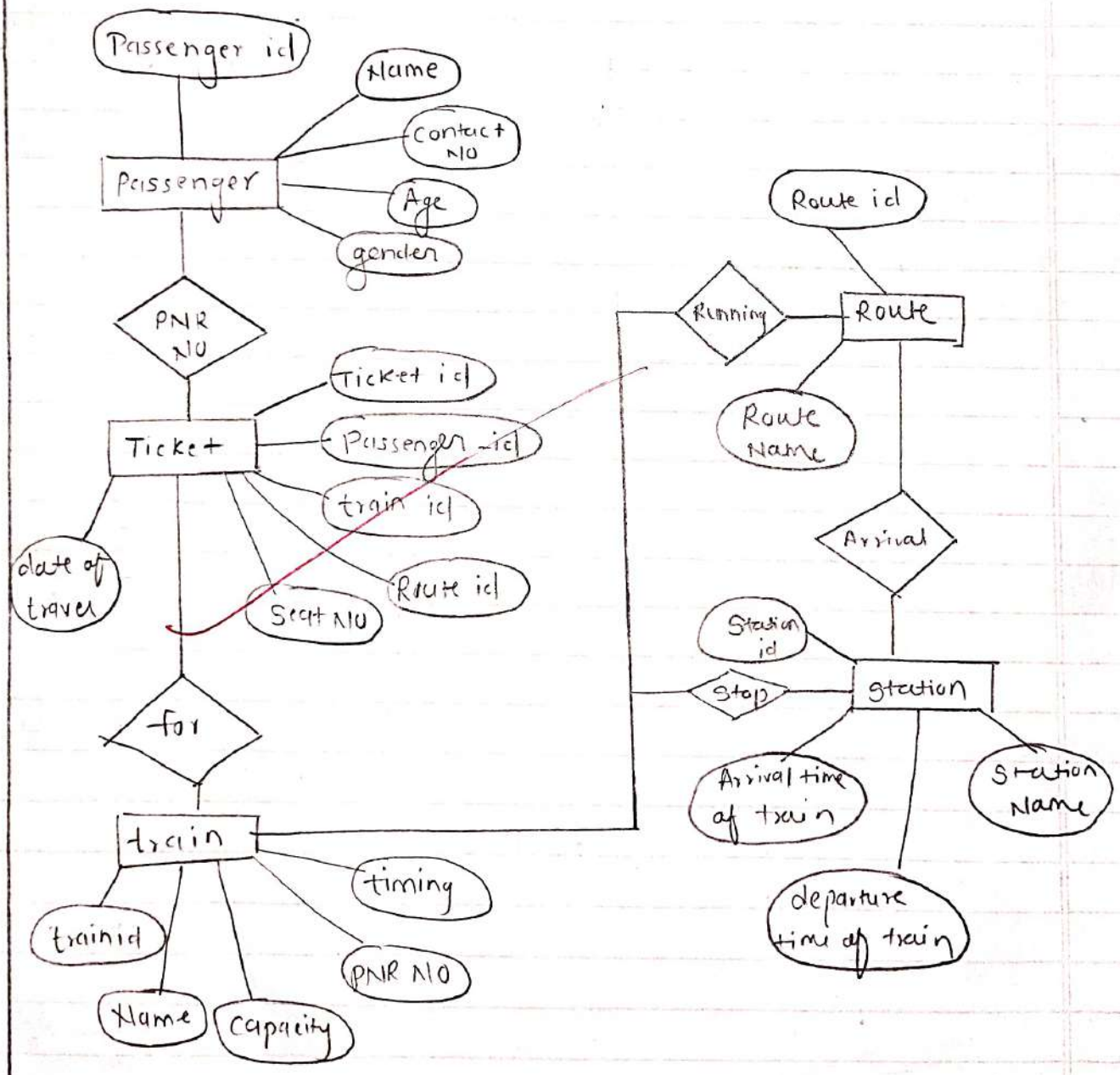
PRACTICAL - 1

- **Aim:** Database design using E-R Model for Railway reservation system and library information system.

E-R Model

(1) Railway reservation System

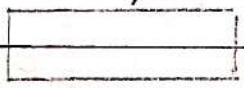
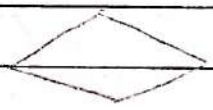
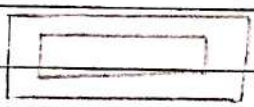


Page No.
Date.

PRACTICAL-01

Aim:- Database design using E-R Model for: Railway reservation system and library information system

Theory:- The Entity Relational Model is a model for identifying entities to be represented in the database and representation of how those entities are related. The ER data model specifies enterprise schema that represents the overall logical structure of a database graphically. The Entity Relationship diagram explains the relationship among the entities present in the ~~dabo~~ database. Symbols used in ER model are:-

Figures	Symbols	Represents
1) Rectangle		Entities in ER model
2) Ellipse		Attributs in ER model
3) diamond		Relationships among Entities
4) line		Attributes to Entities & other relation
5) double Rectangle		Weak Entity

ER Models → A) For Railway Reservation System

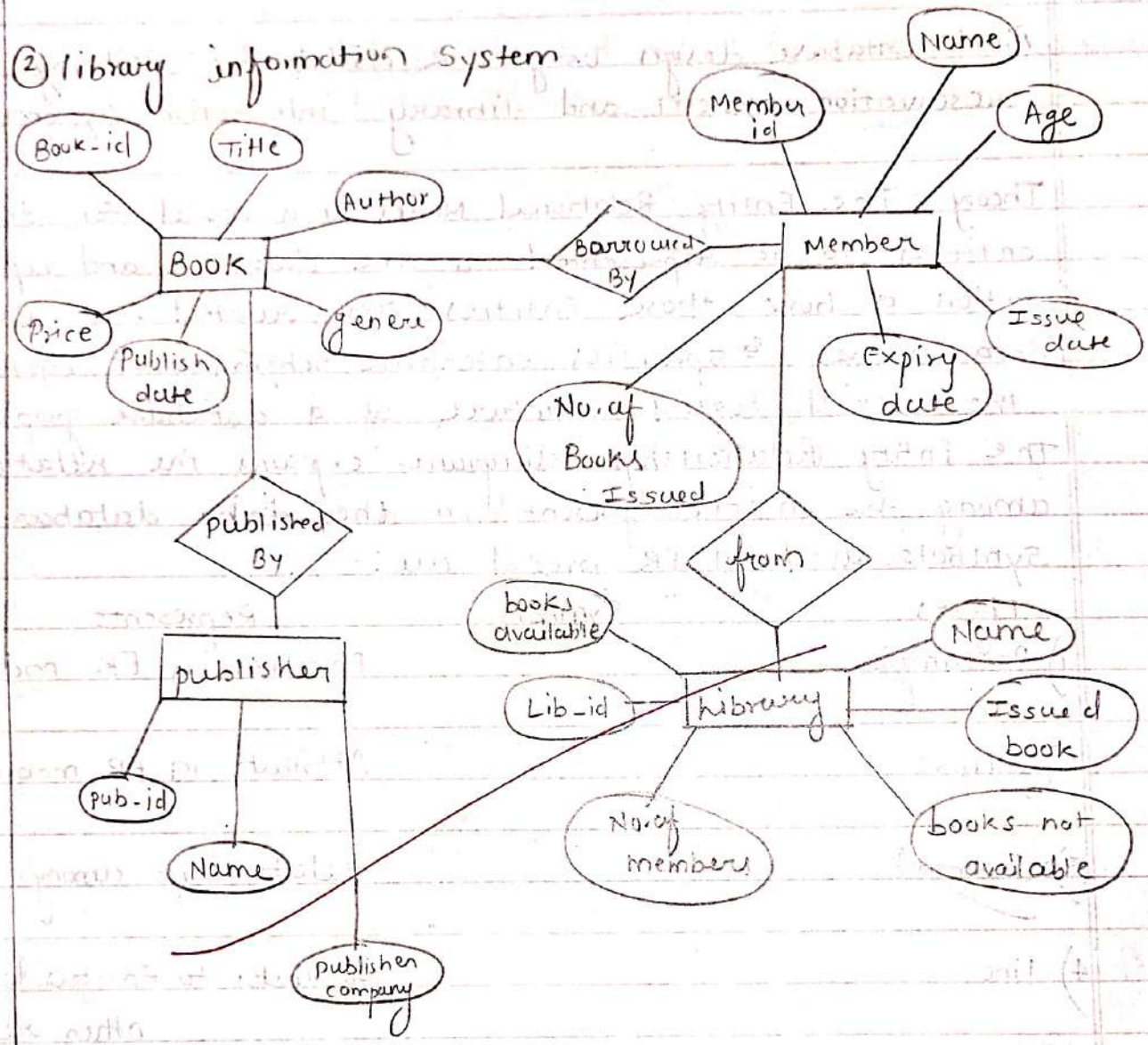
(1) Passanger:-

Attributes:- Passegar_ID, Name, Contact number, gender, Age, PNR NO.

(2) Train:-

Attributes:- train_id, PNR NO, Name, capacity

(2) library information System



* Conclusion:- Thus, we have understand the concept of ER model by taking two examples & we have succesfully done it.

Page No.
Date.

(3) Station :-
Attributes :- station_id, Name, Arrival time, departure time.
(4) Route :-
Attributes :- Route_id, Route_name.
(5) Ticket :-
Attributes :- ticket_id, passanger_id, date of travelling, train_id, seat no, Route_id, PNR no.

(3) For library information system.
(1) Book
Attributes :- book_id, Title, Author, Price, Available, publish date, genere
(2) Member :-
Attributes :- Member_id, name, Issue date, Expiry date, no. of book issued, Age
(3) Publishes :-
Attributes :- Pub. id, name, pub_company
(4) Library :-
Attributes :- no. of members, books available, books not available, Issued book, Name, Lib_id.

Conclusion :- Thus, we have understand the concept of ER model by taking two examples and we have successfully done it.

Page No.
Date.

PRACTICAL NO: 02

Aim:- Mapping of E-R model to relational schema and creation of tables using DDL (Data Definition language).

Theory:- Data Defination Language (DDL): is a subset of SQL that used to create & modify the structure of database objects. These objects includes Tables, Views, indexes, and stores procedures. DDL commands are used to create the new objects include tables, modify existing objects and drop objects. DDL commands are typically executed in a transactional manner. This means if a DDL command fails the database will be rolled back to the state it was in before the command was executed.

These are some of the most common DDL commands→

(1) CREATE:- It is used to create new database objects.
(2) ALTER:- It is used to modify existing database objects
(3) DROP:- It is used to delete database objects
(4) TRUNCATE:- It is used to delete all data from a table, but the table structure is not deleted.
(5) RENAME:- It is used to rename old table to new table name.

Create a schema for employee with following attributes Name, salary and Age.

(1) Create a schema

```
SQL > create table sayali19
(Age int, name varchar(25), salary int);
```

→ Table created.

Page No.
Date.

(2) Rename the schema.
SQL > rename Sayali19 to emp19;
→ renamed.

(3) Alter table to add revised salary column.
SQL > Alter table emp19 add rev-sal int;
→ table Altered.

(4) Describe table
SQL > Desc emp19;
→

Name	NULL?	Type
name		varchar(30)
salary		number
age		number
rev-salary		number

(5) Alter table to drop revised salary attribute:-
SQL > Alter table emp19 drop column rev_sal;
→ Table Altered.

(6) Alter table to modify name varchar(40);
SQL > Alteredtable emp19 modify name varchar(40)

(7) Describe Altered table.
→

Name	NULL?	Type
name		varchar(40)
salary		Number
age		Number

(8) Insert datas in table.

Page No.
Date.

SQL> Insert into emp19 values
("Sayali", 35000, 20);
SQL> Insert into emp19 values
("Teju", 37000, 28);

(9) Display the information in table
SQL > select * from emp19 ;

Name	salary	age
Sayali	35000	20
Teju	37000	28

(10) Perform truncate table:
SQL> Truncate table emp19;
(11) Display information of table
SQL > select * from emp19 ;
→ empty set
(12) Drop the table,
SQL > Alter table
SQL > Drop table emp19;
→ Table dropped
(13) Describe the table.
SQL > Desc emp19;
→ emp19 doesn't exist.

* Conclusion :-
Thus we have learned about the DDL in detail & practice the DDL given queries in SQL plus.

Pthawal

Page No.
Date.

PRACTICAL : 03

Aim:- Modification of database objects using DDL & DML.

Theory:- The DML is an abbreviation of data Manipulation language. The DML commands in structured query language change the data present in the SQL database. We can easily access, store, modify, update and delete the existing records from the database using DML commands. following are the four main DML commands in SQL.

(1) SELECT:- It shows the records of the specific table. It also shows the particular record of a particular column using 'where'.

(2) INSERT:- It allows users to insert data in database tables.

(3) UPDATE:- It allows users to update & modify existing data.

(4) DELETE:- It allows user to remove single or multiple existing records from the database tables.

Q Change according to given queries after creating a patient database with following attributes p_id, name, age, gender, doctor_name, bill, discount amount & disease & phone_number.

(1) Create table for patient database

```
SQL> create table p19
(pid int, pname varchar(40), age int, gender varchar(10),
docname varchar(40), bill int, damount int, disease varchar(40),
phno int)
```

(2) insert values in database 19

```
SQL> insert into p19 values
(1, 'Sayali', 20, 'Female', 'ds deshpandey', 7000, 2000, 'sugar',
7018994989);
```

Page No.
Date.

SQL> insert into pig values.
(2, 'Sejal', 23, 'Female', dr vikey', 10000, 4000', tb, 944551034);

(3) Display all values of database:- SQL> select * from pig

pid	pname	age	gender	docname	Bill amount	disease	phno.
1	Sayali	20	Female	dr. Deshpandey	7000	2000	sugar	7018994989
2	Sejal	23	Female	dr. vikey	10000	4000	tb	944551034

(4) display patient name and doctor name.
SQL> Select pname. docname from pig;

pname	docname
Sayali	dr. Deshpandey
sejal	dr. Vikey

(5) display Patient treated by Dr. Deshpandey.
SQL> select pname from pig where 'dr Deshpandey';

pname
Sayali

(6) display record of patient getting discount of Rs 2000
SQL> select pid, pname, age, gender from pig where damount = 2000;

pid	pname	age	gender
1	Sayali	20	Female

(7) display patient information who paid bill more than Rs 5000

SQL> select pid, pname from pig where bill > 5000;

pid	pname
2	Sejal

PAGE NO.:
DATE.: / / 20

(8) display record of patient whose age is less than 21
SQL> Select pid, pname, age from p19 where age < 21;

pid	pname	age
1	Sayali	20

(9) Update the discount amount for patient id 2.
SQL> update p19 set damount = 3000 where pid = 2;
1 row updated.

(10) Update docname & phone number for pid = 2.
SQL> update p19 set docname = dr. deshpandey, phno = 7058549102
where pid = 2;

(11) Chage patient name to sita whose id is 1.
SQL> update p19 set pname = 'Sita', gender = 'female'
where pid = 2;
→ 1 row updated.

(12) delete the record of pid = 001
SQL> delete from p19 where pid = 1;
→ 1 row deleted

(13) delete the record of patient treated by dr. deshpandey
SQL> delete from p19 where docname = 'dr Deshpandey';
→ 1 row deleted

Conclusion :- Thus, we have understand all the concept of DML language & its commands & successfully executed it

Teacher's Signature

Page No.
Date.

PRACTICAL NO: 04

Aim: Querying the databased on various inbuilt functions (Date function, Aggregate function & many more)

Theory:- SQL aggregations ~~on~~ ~~null~~ function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value. It is also used to summarize the data. The following are aggregation functions.

(1) COUNT:- It is used to calculate the number of rows in database. It can work on both numeric and non-numeric data types.

(2) SUM:- It is used to calculate the sum of all selected.

(3) AVG → It is used to calculate the avg value of the columns.

(4) Max:- It is used to find the maximum value of a certain column. It selects the largest value of all selected values.

(5) Min:- It is used to find the minimum value of a certain column.

The 'Like' operator is used in a 'where' clause to search for a specified pattern in a column. There are two wild cards often used in conjunction with the 'like' operator.

→ The percent sign (%) represents zero, one or multiple characters.

→ The underscore sign (_) represents one, single characters.

Page No.
Date.

Consider the following tables.

Employee (Emp_id, Emp_name, Job_name, Manager_id, Hire_date, Salary, deptno)

Department (Deptno, Dname, MGRSSN)

Project (Pname, Pno, Plocation, Deptno)

emp_id	emp_name	job_name	manager_id	hire_dak	Salary	depno
68319	Kayling	President	68319	1991-11-18	6000	1001
66928	Blaze	Manager	68319	1991-05-01	2750	3001
67832	Clare	Manager	68319	1991-06-09	2550	1001
65646	Jonas	Manager	65646	1991-04-02	2957	2001
67858	Scarlet	analyst	65646	1991-04-19	3100	2001
69062	frank	analyst	69062	1991-12-03	3100	2001
63679	Sandrine	clerk	86928	1991-12-18	900	2001
64989	adelyn	Salesman	66928	1991-02-20	1700	3001
65271	wade	Salesman	66928	1991-02-22	1350	3001
66564	madden	Salesman	66928	1991-09-28	1350	3001
68454	Tucker	Salesman	67858	1991-09-08	1600	3001
68736	adnres	clerk	66928	1997-05-23	1200	2001
69000	Julius	clerk	67832	1991-12-03	1050	3001

Department Table

deptno	dname	location.
1001	Accounting	New York
2001	Research	Dallas
3001	Sales	chicago.

Page No.
Date.

- Project Table

Pname	Pno	Plocation	deptno.
P_1	111	New York	1001
P_2	112	Dallas	2001
P_3	113	Chicago	3001
P_4	114	Denmark	2001
P_5	115	Paris	1001

- Queries:-

1] Display all the details of the records whose employee name starts with 'A'

command:-

```
select * from employee where emp_name like 'a%';
```

2] count the total projects in the project Table

command:-

```
select count (pname) from project;
```

3] Determine the max and min salary.

command:

```
select min (salary) from employee;
select max (salary) from employee;
```

4] Find How many departments are available in Department table.

command:-

```
select count (depno) from department;
```

5] Find the number of projects being handled in department no. 1001

command:-

```
select sum (pno) from project whose deptno = '1001';
```

Query 1

Output :-

emp_id	Empname	jobname	Managerid	Hiredate	Salary	deptno
64989	Adelyn	salesman	66928	20-02-91	1700	3001
68736	admin	clerk	67858	23-05-91	1200	2001

Query 2

Output :-

count (pname)
5

Query 3

Output :-

Min (salary)
900

Max (salary)
25500

Query 4

Output :-

count (deptno)
3

Query 5

output :-

sum (fno)
226.

Page No.
Date.

6] Find the average of salaries of employees having job title as a manager.

Command:

```
Select avg (salary) from employee where job name = 'Manager';
```

7] Find the Employee names whose name contains 5 characters

Command:

```
Select emp_name from Employee where length (emp_name)=5;
```

8] Display the employee names who salary range between 3000 to 6000.

Command:-

```
select * from employee where salary between 3000 and 6000;
```

9] Display the Employee name, job name hire date for the employee where hire date must be in 'dd-mon-yyyy'

command:-

```
Select emp_name, job_name, TO_CHAR.
(hire_date, 'DD-MON-YYYY') As formatted_hire_date from
employee)
```

Query 6

Output :-

avg (salary)
10402.3.33

Query 7

Output :-

emp-name
Blaze
Clare
Jonay
frank

Query 8

output :-

Empid	Empname	Jobname	Manager_id	Hiredate	Salary	dept_no
68319	Kayling	President	68319	18-Nov-91	6000	1001
67858	scarlet	Analyst	65646	19-April-97	3100	2001
69060	frank	Analyst	65646	03-dec-91	3100	2001

Query 9 :

Output

emp_name	Job-name	hire-date
Kayling	President	18-NOV-1991
Blaze	Manager	01-MAY-1991
Clare	Manager	09-JUNE-1991
Jonay	Manager	02-APR-1991
scarlet	Analyst	19-APR-1991
frank	Analyst	03-DEC-1991
sandsing	clerk	18-DEC-1991
Adelyn	salesman	20-FEB-1991
wade	Salesman	22-FEB-1991
Madden	Salesman	28-SEP-1991
Tucker	salesman	08-SEP-1991
adrees	clerk	23-MAY-1991
Tulius.	clerk	03-DEC-1991

Page No.
Date.

10] Display the salary of the employees with the currency sign & two decimal positions

Command:

```
select Concat ('$', FORMAT ( salary, 2)) AS formatted_
salary
from employee;
```

Conclusion:-

Thus, we have successfully executed the queries based on various inbuilt functions.

Pranali

Query 10.

Output - Formatted Salary

$ 6000.00	$ 1350.00
$ 2750.00	$ 1600.00
$ 2550.00	$ 1200.00
$ 2957.00	$ 1050.00
$ 3100.00	
$ 3100.00	
$ 3100.00	
$ 900.00	
$ 1700.00	

Conclusion :- Thus, we have successfully executed the queries based on various inbuilt functions

PRACTICAL NO. 5

OBJECTIVE (AIM) OF THE EXPERIMENT

Querying the Database based on Set, Arithmetic(and Logical operator.(AND,OR,BETWEEN,NOT,LIKE, Addition,Multiplication,Subtraction,Division)

PROCEDURE

a) Procedure for doing the experiment:

Step no.	Details of the step
1	**Set Operators:** The Set operator combines the result of 2 queries into a single result. The following are the operators: • Union • Union all • Intersect • Minus
2	**The rules to which the set operators are strictly adhere to :** The queries which are related by the set operators should have a same number of column and column definition. Such query should not contain a type of long. Labels under which the result is displayed are those from the first select statement.

b) SQL commands:

Union: Returns all distinct rows selected by both the queries
Syntax:
Query1 Union Query2;

Union all: Returns all rows selected by either query including the duplicates.
Syntax:
Query1 Union all Query2;

Intersect: Returns rows selected that are common to both queries.
Syntax:
Query1 Intersect Query2;

Minus: Returns all distinct rows selected by the first query and are not by the second
Syntax:
Query1 minus Query2;

EXCEPT
EXCEPT clause in SQL Server is working as like MINUS operation in Oracle. EXCEPT query returns all rows which are in the first query but those are not returned in the second query.

c) Queries:

UNION:
Q1: Display all the dept numbers available with the dept and emp tables avoiding duplicates.

Solution:

1. Use select from clause.
2. Use union select clause to get the result.

Ans:

SQL> select deptno from emp union select deptno from dept;

```
DEPTNO
-------
1
2
12
30
40
```

1. Get the names of employees who are married or earn over 30,000.

```
SQL> SELECT EMP_NAME FROM EMP WHERE MARITAL_STATUS = 'M'
  UNION
  SELECT EMP_NAME FROM EMP WHERE SALARY > 30000;

EMP_NAME
--------
Brown
Green
Jarvis
Jones
```

2. Get the names of departments with budgets in excess of 140,000 or that are managed by employee E8.

```
SQL> SELECT DEPT_NAME FROM DEPT WHERE BUDGET > 140000
   UNION
  SELECT DEPT_NAME FROM DEPT WHERE MANAGER_NO = 'E8';

DEPT_NAME
--------------
Accounts
Sales
Transport
```

3. Find the Project Numbers of projects which have a deadline before 01-Jan-2008 or have employee E3 working on them.

```
SQL> SELECT PROJ_NO FROM PROJ WHERE DEADLINE < '01-JAN-2008'
   UNION
  SELECT PROJ_NO FROM ALLOC WHERE EMP_NO = 'E3';

PR
--
```

P2

4. List the Employee Numbers of employees who either manage the Sales Department or work on project P4.

```
SQL> SELECT MANAGER_NO FROM DEPT WHERE DEPT_NAME = 'Sales'
  UNION
  SELECT EMP_NO FROM ALLOC WHERE PROJ_NO = 'P4';

MA
--
E4
E5
E6
E9
```

5. Get the names of employees with their salaries and of departments with their budgets.

```
SQL> SELECT EMP_NAME, SALARY FROM EMP
   UNION
  SELECT DEPT_NAME, BUDGET FROM DEPT;

EMP_NAME        SALARY
--------------- ----------
Accounts         95000
Brown            38500
Evans            11000
Fletcher         12000
Green            38500
Jarvis           21000
Jones            12000
Production      100000
Roberts          20000
Sales           250000
Transport       150000
11 rows selected.
```

Q6: **Display all the dept numbers available with the dept and emp tables.**
Solution:
1. Use select from clause. 2. Use union all in select clause to get the result.
Ans:
SQL> select deptno from emp union all select deptno from dept;

```
DEPTNO
-------
1
2
2
1
12
1
2
30
40
9 rows selected.
```

INTERSECT:

1. Get the names of employees who are married and earn more than £15,000.

```
SQL> SELECT EMP_NAME FROM EMP WHERE MARITAL_STATUS = 'M'
  INTERSECT
  SELECT EMP_NAME FROM EMP WHERE SALARY > 15000;
EMP_NAME
--------------
Jarvis
```

2. Get the names of departments not managed by employee E5 that have budgets of more than£96,000.

```
SQL> SELECT DEPT_NAME FROM DEPT WHERE MANAGER_NO <> 'E5'
  INTERSECT
  SELECT DEPT_NAME FROM DEPT WHERE BUDGET > 96000;

DEPT_NAME
--------------
Production
Transport
```

3. List the EmployeeNumbers of department managers who are paid less than £12,500.

```
SQL> SELECT MANAGER_NO FROM DEPT
  INTERSECT
 SELECT EMP_NO FROM EMP WHERE SALARY < 12500;

MA
--
E2
```

4. Get the Project Numbers of projects that started after 10-Jun-2005 and have employee E4 working on them.

```
SQL> SELECT PROJ_NO FROM PROJ WHERE START_DATE > '10-Jun-2005'
  INTERSECT
 SELECT PROJ_NO FROM ALLOC WHERE EMP_NO = 'E4';
PR
--
P4
```

5. Get the Employee Numbersof managers who are also working on projects.

```
SQL> SELECT MANAGER_NO FROM DEPT
   INTERSECT
  SELECT EMP_NO FROM ALLOC;

MA
--
E2
E5
```

MINUS:

1. Get the names of employees known to be single who do not earn more than £13,000.

```
SQL> SELECT EMP_NAME FROM EMP WHERE MARITAL_STATUS = 'S'
   MINUS
 SELECT EMP_NAME FROM EMP WHERE SALARY > 13000;

EMP_NAME
--------------
Evans
Fletcher
```

2. Get the salaries of every employee apart from those working for department D2.

```
SQL> SELECT SALARY FROM EMP
   MINUS
```

```
SELECT SALARY FROM EMP WHERE DEPT_NO = 'D2';

SALARY
----------
     11000
     12000
     20000
     21000
```

3. Find the EmployeeNumbers of employees who do not manage a department.

```
SQL> SELECT EMP_NO FROM EMP
  MINUS
  SELECT MANAGER_NO FROM DEPT;

EM
-- E4
E6 E9
```

4. Get the Employee Numbers of those employees who are not working on any projects.

```
SQL> SELECT EMP_NO FROM EMP
  2      MINUS
  3   SELECT EMP_NO FROM ALLOC;

EM
-- E3
E8
```

5. Get the Employee Numbers of employees paid more than £15,000apart from those who manage departments with a budget of £100,000 or less

```
SQL> SELECT EMP_NO FROM EMP WHERE SALARY > 15000
  2      MINUS
  3   SELECT MANAGER_NO FROM DEPT WHERE BUDGET <= 100000; EM
-- E5
E6
```

Q6: Display all the dept numbers available in emp and not in dept tables and vice versa.

Solution:

1. Use select from clause.
2. Use minus in select clause to get the result. Ans:

```
SQL> select deptno from emp minus select deptno from dept;

DEPTNO
-------
12
SQL> select deptno from dept minus select deptno from emp; DEPTNO
-------
30
40
```

d) **Result:**

Thus the set operations using DML Commands was successfully performed and executed.

SQL Operators

The operators are symbols (and keywords) that are used to perform operations with values.

These operators are used with SQL clauses such as: `SELECT`, `WHERE`, `ON` etc.

The operators in SQL can be categorized as:

- Arithmetic operators
- Comparison operators
- Logical operators

SQL Arithmetic Operators

Arithmetic operators perform simple arithmetic operations such as addition, subtraction, multiplication etc.

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Divide
%	Modulo (Remainder)

Addition Operator

```
-- returns new column named total_amount which is
-- 100 added to the amount field
SELECT item, amount, amount+100 AS total_amount
FROM Orders;
```

```
Run Code
```

Subtraction Operator

```
-- returns new column named offer_price which is
-- 20 subtracted to the amount field
SELECT item, amount, amount-20 AS offer_price
FROM Orders;
Run Code
```

Multiplication Operator

```
-- returns new column named total_amount  which is
-- 4 multiplied to the amount field
SELECT item, amount, amount*4 AS total_amount
FROM Orders;
Run Code
```

Division Operator

```
-- returns new column named half_amount which is
-- divided by 2 to the amount field
SELECT item, amount, amount/2 AS half_amount
FROM Orders;
Run Code
```

Modulo (Remainder) Operator

```
-- returns 1 which is remainder
SELECT 10 % 3 AS result;
Run Code
```

Consider the following Tables:

EMPLOYEE(Emp_id, EMP_name,Job_name,Manager_id,Hire_date,Salary,Deptno)

DEPARTMENT(Deptno, Dname, MGRSSN)

PROJECT(Pname,Pno,Plocation,Deptno)

emp_id	emp_name	job_name	manager_id	hire_date	salary	E_Bonus	dep_no
68319	KAYLING	PRESIDENT		1991-11-18	6000.00	300.00	1001
66928	BLAZE	MANAGER	68319	1991-05-01	2750.00	200.00	3001
67832	CLARE	MANAGER	68319	1991-06-09	2550.00	200.00	1001
65646	JONAS	MANAGER	68319	1991-04-02	2957.00	200.00	2001
67858	SCARLET	ANALYST	65646	1997-04-19	3100.00	250.00	2001
69062	FRANK	ANALYST	65646	1991-12-03	3100.00	250.00	2001
63679	SANDRINE	CLERK	69062	1990-12-18	900.00	150.00	2001
64989	ADELYN	SALESMAN	66928	1991-02-20	1700.00	180.00	3001
65271	WADE	SALESMAN	66928	1991-02-22	1350.00	180.00	3001

66564	MADDEN	SALESMAN	66928	1991-09-28	1350.00	180.00	3001
68454	TUCKER	SALESMAN	66928	1991-09-08	1600.00	180.00	3001
68736	ADNRES	CLERK	67858	1997-05-23	1200.00	150.00	2001
69000	JULIUS	CLERK	66928	1991-12-03	1050.00	150.00	3001
69324	MARKER	CLERK	67832	1992-01-23	1400.00	150.00	1001

Department Table

deptno	dname	Citylocation	dCountry
1001	Accounting	New York	United States of America,
2001	Research	Dallas	United States
3001	Sales	Chicago	United States of America
4001	Marketing	Los Angeles	United States

Project Table

Pno	Pname	PCitylocation	PCountry
111	P_1	New York	United States of America,
112	P_2	Dallas	United States
113	P_3	Chicago	United States of America
114	P_4	Denmark	northern Europe
115	P_5	Paris	France
116	P_6	Chicago	United States of America

Write a query for the following:-

Q1. Display all the Departments and Projects available.

Q2. Display the Locations of Departments and Projects.

Q3. Display the Project's Locations which are not the Department's Locations.

Q4. Display the Department's Locations which are also Project's Locations.

Q5. Display the cities of United States of America in which Projects are been designed and also display their respective Departments.

Q6. Display the Countries and cities for projects P_1 and P_2 & Departments Accounting and Marketing.

Q7. Display those Cities which are same for Projects and Departments.

Q8. Display Project numbers and Department numbers for which country is United States.

Q9. Find the names of the projects and Departments which have city as Chicago.

Q10 Display the details for projects and Departments which don't have country as Northern Europe.

Q11 Get details of the Employees with the largest Salary

Q12. Display the Total Salary of Employees including Bonus

Q13. Display the Salaries if it is increased by 5 times more than original Salaries of Employees who work as Analyst

Q14. Display the Salaries of all Employees who are paying 10 % of their total salary for Social Cause

```
mysql> select dname from dept_itA21 union select pname from proj_itA21;
+------------+
| dname      |
+------------+
| accounting |
| research   |
| sales      |
| marketing  |
| p1         |
| p2         |
| p3         |
| p4         |
| p5         |
| p6         |
+------------+
10 rows in set (0.01 sec)

mysql> select cityloca from dept_itA21 union select cityloca from proj_itA21;
+-------------+
| cityloca    |
+-------------+
| new york    |
| dallas      |
| chicago     |
| los angeles |
| denmark     |
| paris       |
+-------------+
6 rows in set (0.00 sec)

mysql> select cityloca from proj_itA21 left join dept_itA21 using (cityloca) where dept_itA21.cityloca i
s null;
+-----------+
| cityloca  |
+-----------+
| denmark   |
| paris     |
+-----------+
2 rows in set (0.01 sec)

mysql> select cityloca from dept_itA21 intersect select cityloca from proj_itA21;
+-----------+
| cityloca  |
+-----------+
| new york  |
| dallas    |
| chicago   |
+-----------+
3 rows in set (0.00 sec)

mysql> select p.cityloca,d.dname from proj_itA21 p, dept_itA21 d
    -> where p.cityloca='USA' and d.cityloca='USA';
+----------+-------------+
| cityloca | dname       |
+----------+-------------+
| new york | sales       |
| new york | accounting  |
| chicago  | sales       |
| chicago  | accounting  |
| dallas   | sales       |
| dallas   | accounting  |
+----------+-------------+
6 rows in set (0.00 sec)
```

```
mysql> select cityloca from proj_itA21
    -> intersect
    -> select cityloca from dept_itA21;
+----------+
| cityloca |
+----------+
| new york |
| dallas   |
| chicago  |
+----------+
3 rows in set (0.00 sec)

mysql> select p.cityloca,p.country from proj_itA21 p
    -> where p.pname in ('p1','p2') union
    -> select d.cityloca,d.country from dept_itA21 d
    -> where  d.dname in ('accounting','marketing');
+-------------+---------------+
| cityloca    | country       |
+-------------+---------------+
| new york    | USA           |
| dallas      | united states |
| los angeles | united states |
+-------------+---------------+
3 rows in set (0.00 sec)

mysql> select p.pname,d.dname from proj_itA21 p, dept_itA21 d
    -> where p.cityloca='chicago' and d.cityloca='chicago';
+-------+-------+
| pname | dname |
+-------+-------+
| p3    | sales |
| p6    | sales |
+-------+-------+
2 rows in set (0.00 sec)

mysql> select  p.pno, d.deptno from proj_itA21 p, dept_itA21 d
    -> where p.country='USA' and d.country='USA';
+------+--------+
| pno  | deptno |
+------+--------+
|  111 |   3001 |
|  112 |   1001 |
|  113 |   3001 |
|  113 |   1001 |
|  116 |   3001 |
|  116 |   1001 |
+------+--------+
6 rows in set (0.00 sec)
```

```
mysql> select * from proj_itA21 left join dept_itA21 using (country) where proj_itA21.country!='northern
europe';
+---------------+------+-------+----------+--------+------------+-------------+
| country       | pno  | pname | cityloca | deptno | dname      | cityloca    |
+---------------+------+-------+----------+--------+------------+-------------+
| USA           |  111 | p1    | new york |   3001 | sales      | chicago     |
| USA           |  111 | p1    | new york |   1001 | accounting | new york    |
| united states |  112 | p2    | dallas   |   4001 | marketing  | los angeles |
| united states |  112 | p2    | dallas   |   2001 | research   | dallas      |
| USA           |  113 | p3    | chicago  |   3001 | sales      | chicago     |
| USA           |  113 | p3    | chicago  |   1001 | accounting | new york    |
| france        |  115 | p5    | paris    |   NULL | NULL       | NULL        |
| USA           |  116 | p6    | chicago  |   3001 | sales      | chicago     |
| USA           |  116 | p6    | chicago  |   1001 | accounting | new york    |
+---------------+------+-------+----------+--------+------------+-------------+
9 rows in set (0.00 sec)

mysql> select * from emp_itA21 where salary=(select max(salary) from emp_itA21);
+-------+---------+-----------+-----------+------------+--------+--------+--------+
| eid   | ename   | jobname   | managerid | hiredate   | salary | ebonus | deptno |
+-------+---------+-----------+-----------+------------+--------+--------+--------+
| 68319 | kayling | president |     68219 | 1991-11-18 |   6000 |    300 |   1001 |
+-------+---------+-----------+-----------+------------+--------+--------+--------+
1 row in set (0.01 sec)

mysql> select eid,sum(salary+ebonus) from emp_itA21 group by eid;
+-------+---------------------+
| eid   | sum(salary+ebonus)  |
+-------+---------------------+
| 68319 |                6300 |
| 66928 |                2950 |
| 67832 |                2750 |
| 65646 |                3157 |
| 67858 |                3350 |
| 69062 |                3350 |
| 63679 |                1050 |
| 64989 |                1800 |
| 65271 |                1530 |
| 66564 |                1530 |
| 68454 |                1530 |
| 68736 |                1350 |
| 69000 |                1200 |
| 69324 |                1550 |
+-------+---------------------+
14 rows in set (0.01 sec)
```

Page No.
Date.

PRACTICAL : 05

Aim : Querying the database based on set Arithmetic and logical operators.

Theory :- The SQL set operation is used to continue the two or more SQL SELECT statement and its types are.

1] Union : It is used to ~~contin~~ combine the result of two or more SQL select queries.
- Union operation eliminates the duplicate rows from its resultant.

2] Union All : This operation is equal to the Union operation. It return the set without remaining duplication sorting the data.

3] Intersect : It is used to combine two select statements, the intersect operation returns the common rows from both the select statement.
- In the intersect operation, the number of datatype and column must be the same.
- It has no duplicate and it arrange the data in ascending order by default.

4] Minus :- It combines the result of two select statements. Minus operator is used to display the rows which are present in the first query but absent in the second query.
- It has no duplicates & data arranged in ascending order by default.

Page No.
Date.

Arithmetic operators:

1. + → used to perform addition on values
2. − → used to perform substraction
3. / → operators works with "All" keywords audit calculates division
4. * → used to multiply data values
5. % → Modulus is used to get the remainder when data is divided by another.

Logical operators:

1. AND → It compares two boolean as expressions and return true when both expressions are true.
2. OR → It compares two boolean as expressions and returns true when one of the expression is true.
3. NOT → Not takes a single Boolean as an Argument and changes its value from false to true or from true to false.

Conclusion :-

Thus, we have understand the commands we have successfully demonstrate them.

Pravin

Practical Number: # 6

Title of the Exercise : Querying the database based on join opertation

a)Simple join and Self join b)Outer join and Inner join

Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To perform nested Queries and joining Queries using DML command.

b) Procedure for doing the experiment:

Step no.	Details of the step
1	**Relating Data through Join Concept** The purpose of a join concept is to combine data spread across tables. A join is actually performed by the „where□ clause which combines specified rows of tables. Syntax; select columns from table1, table2 where logical expression; **Types of Joins** 1. Simple Join 2. Self Join 3. Outer Join 4. Inner Join
2	1. Simple Join **a) Equi-join:** A join, which is based on equalities, is called equi-join. **b) Non Equi-join:** It specifies the relationship between **Table Aliases** Table aliases are used to make multiple table queries shorted and more readable. We give an alias name to the table in the „from□ clause and use it instead of the name throughout the query.
3	**Self join:** Joining of a table to itself is known as self-join. It joins one row in a table to another. It can compare each row of the table to itself and also with other rows of the same table.
	Outer Join: It extends the result of a simple join. An outer join returns all the rows returned by simple join as well as those rows from one table that do not match any row from the table. The symbol (+) represents outer join. **Inner join:** Inner join returns the matching rows from the tables that are being joined

c) Simple Join

a) Equi-join

Example: select * from item, cust where item.id=cust.id;

b) Non Equi-join

Example: select * from item, cust where item.id<cust.id;

Self join

Example: select * from emp x ,emp y where x.salary >= (select avg(salary) from x.emp where x. deptno =y.deptno);

Outer Join

Example: select ename, job, dname from emp, dept where emp.deptno (+) = dept.deptno;

d) Queries:

1. Use select from clause.
2. Use like operator to match job and in select clause to get the result.

Ans:

```
SQL> select ename,sal from emp where sal>(select min(sal) fromemp where job like 'A%');
ENAME                SAL
--------------------
Arjun               12000
Gugan               20000
Karthik             15000
```

Q2: Issue a query to find all the employees who work in the same job as Arjun.

Ans:

```
SQL> select * from emp;
   EMPNO ENAME          JOB          DEPTNO      SAL
-------------------------------------------
       1 Mathi          AP             1      10000
       2 Arjun          ASP            2      12000
       3 Gugan          ASP            2      20000
       4 Karthik        AP             1      15000
SQL> select ename from emp where job=(select job from emp where ename='Arjun');
ENAME
----------
Arjun
Gugan
```

Q3: Issue a query to display information about employees who earn more than any employee in dept 1.

Ans:

```
SQL> select * from emp where sal>(select max(sal) from emp where empno=1);

   EMPNO ENAME          JOB          DEPTNO      SAL
-------------------------------------------
       2 Arjun          ASP            2      12000
       3 Gugan          ASP            2      20000
       4 Karthik        AP             1      15000
```

JOINS

Tables used

```
SQL> select * from emp;

   EMPNO ENAME          JOB          DEPTNO      SAL
-------------------------------------------
       1 Mathi          AP             1      10000
       2 Arjun          ASP            2      12000
       3 Gugan          ASP            2      20000
       4 Karthik        AP             1      15000

SQL> select * from dept;

   DEPTNO DNAME          LOC
-----------------------------
       1 ACCOUNTING     NEW YORK
```

2 RESEARCH DALLAS
30 SALES CHICAGO
40 OPERATIONS BOSTON

EQUI-JOIN

Q4: Display the employee details, departments that the departments are same in both the emp and dept.

Solution:

1. Use select from clause. 2. Use equi join in select clause to get the result.

Ans:

```
SQL> select * from emp,dept where emp.deptno=dept.deptno;
```

EMPNO	ENAME	JOB	DEPTNO	SAL	DEPTNO	DNAME	LOC
1	Mathi	AP	1	10000	1	ACCOUNTING	NEW YORK
2	Arjun	ASP	2	12000	2	RESEARCH	DALLAS
3	Gugan	ASP	2	20000	2	RESEARCH	DALLAS
4	Karthik	AP	1	15000	1	ACCOUNTING	NEW YORK

NON-EQUIJOIN

Q5: Display the employee details, departments that the departments are not same in both the emp and dept.

Solution:

1. Use select from clause. 2. Use non equi join in select clause to get the result.

Ans:

SQL> select * from emp,dept where emp.deptno!=dept.deptno;

EMPNO	ENAME	JOB	DEPTNO	SAL	DEPTNO	DNAME	LOC
2	Arjun	ASP	2	12000	1	ACCOUNTING	NEW YORK
3	Gugan	ASP	2	20000	1	ACCOUNTING	NEW YORK
1	Mathi	AP	1	10000	2	RESEARCH	DALLAS

EMPNO	ENAME	JOB	DEPTNO	SAL	DEPTNO	DNAME	LOC
4	Karthik	AP	1	15000	2	RESEARCH	DALLAS
1	Mathi	AP	1	10000	30	SALES	CHICAGO
2	Arjun	ASP	2	12000	30	SALES	CHICAGO

EMPNO	ENAME	JOB	DEPTNO	SAL	DEPTNO	DNAME	LOC
3	Gugan	ASP	2	20000	30	SALES	CHICAGO
4	Karthik	AP	1	15000	30	SALES	CHICAGO
1	Mathi	AP	1	10000	40	OPERATIONS	BOSTON

EMPNO	ENAME	JOB	DEPTNO	SAL	DEPTNO	DNAME	LOC
2	Arjun	ASP	2	12000	40	OPERATIONS	BOSTON
3	Gugan	ASP	2	20000	40	OPERATIONS	BOSTON
4	Karthik	AP	1	15000	40	OPERATIONS	BOSTON

12 rows selected.

LEFTOUT-JOIN

Tables used

SQL> select * from stud1;

Regno	Name	Mark2	Mark3	Result
101	john	89	80	pass
102	Raja	70	80	pass
103	Sharin	70	90	pass
104	sam	90	95	pass

SQL> select * from stud2;

NAME	GRA
john	s
raj	s
sam	a
sharin	a

Q6: Display the Student name and grade by implementing a left outer join.

Ans: SQL> select stud1.name,grade from stud1 left outer join stud2 on stud1.name=stud2.name;

Name	Gra
john	s
raj	s
sam	a
sharin	a
smith	null

RIGHTOUTER-JOIN

Q7: Display the Student name, register no, and result by implementing a right outer join.

Ans:

SQL> select stud1.name, regno, result from stud1 right outer join stud2 on stud1.name = stud2.name;

Name	Regno	Result
john	101	pass
raj	102	pass
sam	103	pass
sharin	104	pass

Rollno	Name	Mark1	Mark2	Total
1	sindu	90	95	185
2	arul	90	90	180

FULLOUTER-JOIN

Q8: Display the Student name register no by implementing a full outer join.

Ans:

SQL> select stud1.name, regno from stud1 full outer join stud2 on (stud1.name= stud2.name);

Name	Regno
john	101
raj	102
sam	103
sharin	104

SELFJOIN

Q9: Write a query to display their employee names Ans:

```
SQL> select distinct ename from emp x, dept y where x.deptno=y.deptno;
ENAME
----------
Arjun
Gugan
Karthik
Mathi
```

Q10: Display the details of those who draw the salary greater than the average salary. Ans:

```
SQL> select distinct * from emp x where x.sal >= (select avg(sal) from emp);
     EMPNO ENAME      JOB              DEPTNO        SAL
---------- ---------- ------------------ ---------- ----------
         3 Gugan      ASP                 2         20000
         4 Karthik    AP                  1         15000
        11 kavitha    designer            12        17000
```

e) Result:

Thus the nested Queries and join Queries was performed successfully and executed.

Consider the following Tables:
EMPLOYEE(Emp_id, EMP_name,Job_name,Manager_id,Hire_date,Salary,Deptno)
DEPARTMENT(Deptno, Dname, MGRSSN)
PROJECT(Pname,Pno,Plocation,Deptno)

emp_id	emp_name	job_name	manager_id	hire_date	salary	E_Bonus	dep_no
68319	KAYLING	PRESIDENT		1991-11-18	6000.00	300.00	1001
66928	BLAZE	MANAGER	68319	1991-05-01	2750.00	200.00	3001
67832	CLARE	MANAGER	68319	1991-06-09	2550.00	200.00	1001
65646	JONAS	MANAGER	68319	1991-04-02	2957.00	200.00	2001
67858	SCARLET	ANALYST	65646	1997-04-19	3100.00	250.00	2001
69062	FRANK	ANALYST	65646	1991-12-03	3100.00	250.00	2001
63679	SANDRINE	CLERK	69062	1990-12-18	900.00	150.00	2001
64989	ADELYN	SALESMAN	66928	1991-02-20	1700.00	180.00	3001
65271	WADE	SALESMAN	66928	1991-02-22	1350.00	180.00	3001
66564	MADDEN	SALESMAN	66928	1991-09-28	1350.00	180.00	3001
68454	TUCKER	SALESMAN	66928	1991-09-08	1600.00	180.00	3001
68736	ADNRES	CLERK	67858	1997-05-23	1200.00	150.00	2001
69000	JULIUS	CLERK	66928	1991-12-03	1050.00	150.00	3001
69324	MARKER	CLERK	67832	1992-01-23	1400.00	150.00	1001

Department Table

deptno	dname	Citylocation	dCountry
1001	Accounting	New York	United States of America,
2001	Research	Dallas	United States
3001	Sales	Chicago	United States of America
4001	Marketing	Los Angeles	United States

Project Table

Pno	Pname	PCitylocation	Dept No
111	P_1	New York	1001
112	P_2	Dallas	1001
113	P_3	Chicago	2001
114	P_4	Denmark	2001
115	P_5	Paris	3001
116	P_6	Chicago	3001
117	P_7	Paris	4001

Write a query for the following:-

Q.1 Display the max salaries for each designation ordered in descending order.

Q.2 Display the employees where salary is more than their manager.

Q.3 Display the project details for sales department.

Q.4 Display the name and salaries of employees working in department at location Chicago.

Q.5 Find the project location for employees working in department Research.

Q.6 Display the names of departments having same project location.

Q.7 Display the employee details who working on project p_3 and p_6.

Q.8Display the department names handling more than one project.

```
mysql> select salary, jobname from emp_itA21 where salary in (select max(salary) from emp_itA21 group by jobname) order by salary desc;
+--------+-----------+
| salary | jobname   |
+--------+-----------+
|   6000 | president |
|   3100 | analyst   |
|   3100 | analyst   |
|   2457 | manager   |
|   1700 | salesmen  |
|   1400 | clerk     |
+--------+-----------+
6 rows in set (0.01 sec)

mysql> select ename from emp_itA21 where salary>(select max(salary) from emp_itA21 where jobname='manager');
+---------+
| ename   |
+---------+
| kayling |
| scarlet |
| frank   |
+---------+
3 rows in set (0.00 sec)

mysql> select * from proj1_itA21 where deptno=3001;
+------+-------+-----------+--------+
| pno  | pname | pcityloca | deptno |
+------+-------+-----------+--------+
|  115 | p5    | paris     |   3001 |
|  116 | p6    | chicago   |   3001 |
+------+-------+-----------+--------+
2 rows in set (0.00 sec)

mysql> select ename,salary from emp_itA21 where deptno=3001;
+---------+--------+
| ename   | salary |
+---------+--------+
| blaze   |   2750 |
| adelyn  |   1700 |
| wade    |   1350 |
| madden  |   1350 |
| tucker  |   1350 |
| julius  |   1050 |
+---------+--------+
6 rows in set (0.00 sec)

mysql> select  pcityloca from proj1_itA21 where deptno=2001;
+-----------+
| pcityloca |
+-----------+
| chicago   |
| denmark   |
+-----------+
2 rows in set (0.00 sec)
```

```
mysql> select dname from dept_itA21 d, proj1_itA21 p where d.cityloca=p.pcityloca;
+------------+
| dname      |
+------------+
| accounting |
| research   |
| sales      |
| sales      |
+------------+
4 rows in set (0.00 sec)

mysql> select * from emp_itA21
    -> join proj1_itA21 on
    -> emp_itA21.deptno=proj1_itA21.deptno
    -> where proj1_itA21.pname in ('p3','p6');
+-------+--------+----------+-----------+------------+--------+--------+--------+-------+----------+--------+
| eid   | ename  | jobname  | managerid | hiredate   | salary | ebonus | deptno | pno   | pname    | pcityloca | deptno |
+-------+--------+----------+-----------+------------+--------+--------+--------+-------+----------+--------+
| 66928 | blaze  | manager  |     68319 | 1991-05-01 |   2750 |    200 |   3001 |   116 | p6       | chicago   |   3001 |
| 65646 | jones  | manager  |     68319 | 1991-04-02 |   2957 |    200 |   2001 |   113 | p3       | chicago   |   2001 |
| 67858 | scarlet| analyst  |     65646 | 1997-04-19 |   3100 |    250 |   2001 |   113 | p3       | chicago   |   2001 |
| 69062 | frank  | analyst  |     65646 | 1991-12-03 |   3100 |    250 |   2001 |   113 | p3       | chicago   |   2001 |
| 63679 | sandrine| clerk   |     69062 | 1990-12-18 |    900 |    150 |   2001 |   113 | p3       | chicago   |   2001 |
| 64989 | adelyn | salesman |     66928 | 1991-02-20 |   1700 |    180 |   3001 |   116 | p6       | chicago   |   3001 |
| 65271 | wade   | salesman |     66928 | 1991-02-22 |   1350 |    180 |   3001 |   116 | p6       | chicago   |   3001 |
| 66564 | madden | salesman |     66928 | 1991-09-28 |   1350 |    180 |   3001 |   116 | p6       | chicago   |   3001 |
| 68454 | tucker | salesman |     66928 | 1991-09-08 |   1350 |    180 |   3001 |   116 | p6       | chicago   |   3001 |
| 68736 | adnres | clerk    |     67858 | 1997-05-23 |   1200 |    150 |   2001 |   113 | p3       | chicago   |   2001 |
| 69000 | julius | clerk    |     66928 | 1991-12-03 |   1050 |    150 |   3001 |   116 | p6       | chicago   |   3001 |
+-------+--------+----------+-----------+------------+--------+--------+--------+-------+----------+--------+
11 rows in set (0.00 sec)
```

PRACTICAL NO : 06

Page No.
Date

Aim :- Querying the database based on join Operation
a) Simple Join and self Join
b) Outer Join and Inner Join
To perform nested queries & joining queries using DML command
queries :-

1] Display the max salaries for each designator ordered in descending order

```
=> Select job_name, Max(salary) as "Max salary" From
Empbyee_IT_A-19
Group by job_name
Order by "Max salary" desc;
```

2] Display the employees where salary is more than manager

```
=> Select e1_emp_id, e1.emp_name, e1_salary, e1.manger_id
From Employee_IT_A-19 e1
Join Employee_IT_A-19 e2 on e1.manager_id = e2.emp_id
where e1.salary > e2.salary;
```

3] Display the project details for sales department

```
=> Select P.Pname, P.Pno, P.PCitylocation, P.PCountry
From Project_IT_A-19 P
where P.PCitylocation in (
    select Citylocation from Department_IT_A-19
    where dname = 'sales');
```

Teacher's Signature

Page No.
Date

Q4] Display the name and salaries of employees working in department at location chicago.

⇒ Select E.emp-name, E.Salary
from Employee_IT_A_19 E
Inner join Department_IT_A_19 D on E.depno = D.dept.no
where D.citylocation = 'chicago';

Q5] Find the project location for employees working in department Research.

⇒ Select Distinct P.Pcitylocation
from Project_IT_A_19 P
Inner join Department_IT_A_19 D
on P.Pcitylocation = D.Citylocation
where D.dname = 'research';

Q6] Display the names of department having same project location.

=> Select distinct D1.dname
from Department_IT_A_19 D1
where D1.citylocation in (
select P.Pcitylocation from Project_IT_A_19 P
where P.Pcountry = 'United states of America';

Teacher's Signature

Page No.
Date

Q7] Display the employee details who working on project p_3 and p_6

⇒
```
Select E.emp_id, E.emp_name, E.job_name
from Employee_IT_A_19 E
where E.emp_id in (
  Select distinct E1.emp_id
  from Employee_IT_A_19 E1
  where E1.emp_id = "P_3" or E1.emp_id = "P_8");
```

Q8] Display the department names handling more than one project.

⇒
```
Select D.dname from Department_IT_A_19 D
where D.dept_no in (
  select D1.dept_no from Project_IT_A_19 P
  Inner join Department_IT_A_19 D1 on
  P.PCity location = D1.city location
  group by D1.dept.no
  having count (P.Pname) > 1
);
```

Conclusion:-

Thus, we have executed queries based on join operation.

Teacher's Signature

PRACTICAL NO. 7

Title of the Exercise : Querying the Database based nesting and applying various Clauses.

OBJECTIVE (AIM) OF THE EXPERIMENT

To perform nested Queries and joining Queries using DML command.

a) Procedure:

Step no.	Details of the step
1	**Nested Queries:** Nesting of queries one within another is known as a nested queries. **Sub queries** The query within another is known as a sub query. A statement containing sub query is called parent statement. The rows returned by sub query are used by the parent statement.
2	**Types** **1. Sub queries that return several values** Sub queries can also return more than one value. Such results should be made use along with the operators in and any. **2. Multiple queries** Here more than one sub query is used. These multiple sub queries are combined by means of „and□ & „or□ keywords **3. Correlated sub query** A sub query is evaluated once for the entire parent statement whereas a correlated Sub query is evaluated once per row processed by the parent statement.

b) SQL Commands:

Nested Queries:

A subquery is a `SELECT` statement written within parentheses and nested inside another statement. Here's an example that looks up the IDs for grade event rows that correspond to tests (`'T'`) and uses them to select scores for those tests

```
SELECT * FROM score
WHERE event_id IN (SELECT event_id FROM grade_event WHERE category = 'T');
```

Subqueries can return different types of information:

- A scalar subquery returns a single value.
- A column subquery returns a single column of one or more values.
- A row subquery returns a single row of one or more values.
- A table subquery returns a table of one or more rows of one or more columns.

Subquery results can be tested in different ways:

- Scalar subquery results can be evaluated using relative comparison operators such as = or <.

- IN and NOT IN test whether a value is present in a set of values returned by a subquery.
- ALL, ANY, and SOME compare a value to the set of values returned by a subquery.
- EXISTS and NOT EXISTS test whether a subquery result is empty.

A scalar subquery is the most restrictive because it produces only a single value. But as a consequence, scalar subqueries can be used in the widest variety of contexts. They are applicable essentially anywhere that you can use a scalar operand, such as a term of an expression, as a function argument, or in the output column list. Column, row, and table subqueries that return more information cannot be used in contexts that require a single value.

Subqueries can be correlated or uncorrelated. This is a function of whether a subquery refers to and is dependent on values in the outer query.

You can use subqueries with statements other than SELECT. However, for statements that modify tables (DELETE, INSERT, REPLACE, UPDATE, LOAD DATA), MySQL enforces the restriction that the subquery cannot select from the table being modified.

In some cases, subqueries can be rewritten as joins. You might find subquery rewriting techniques useful to see whether the MySQL optimizer does a better job with a join than the equivalent subquery.

The following sections discuss the kinds of operations you can use to test subquery results, how to write correlated subqueries, and how to rewrite subqueries as joins

1) Subqueries with Relative Comparison Operators

The =, <>, >, >=, <, and <= operators perform relative-value comparisons. When used with a scalar subquery, they find all rows in the outer query that stand in particular relationship to the value returned by the subquery. For example, to identify the scores for the quiz that took place on '2012-09-23', use a scalar subquery to determine the quiz event ID and then match score table rows against that ID in the outer SELECT:

Example:

```
SELECT * FROM score
WHERE event_id =

(SELECT event_id FROM grade_event
   WHERE date = '2012-09-23' AND category = 'Q');
```

Use of scalar subqueries with relative comparison operators is handy for solving problems for which you'd be tempted to use an aggregate function in a WHERE clause. For example, to determine which of the presidents in the president table was born first, you might try this statement:

Example:

1. `SELECT * FROM president WHERE birth = MIN(birth);`

2. ```
   SELECT * FROM president
   WHERE birth = (SELECT MIN(birth) FROM president);
   ```

3. ```
   SELECT * FROM score WHERE event_id = 5
   AND score > (SELECT AVG(score) FROM score WHERE event_id = 5);
   ```

If a subquery returns a single row, you can use a row constructor to compare a set of values (that is, a tuple) to the subquery result. This statement returns rows for presidents who were born in the same city and state as John Adams

Example:

```
mysql> SELECT last_name, first_name, city, state FROM president
    -> WHERE (city, state) =
```

```
    -> (SELECT city, state FROM president
    -> WHERE last_name = 'Adams' AND first_name = 'John');
```

```
+-----------+------------+----------+-------+
| last_name | first_name | city     | state |
+-----------+------------+----------+-------+
| Adams     | John       | Braintree | MA   |
| Adams     | John Quincy | Braintree | MA   |
+-----------+------------+----------+-------+
```

You can also use ROW(city, state) notation, which is equivalent to (city, state). Both act as row constructors.

2) IN and NOT IN Subqueries

The IN and NOT IN operators can be used when a subquery returns multiple rows to be evaluated in comparison to the outer query. They test whether a comparison value is present in a set of values. IN is true for rows in the outer query that match any row returned by the subquery. NOT IN is true for rows in the outer query that match no rows returned by the subquery. The following statements use IN and NOT IN to find those students who have absences listed in the absence table, and those who have perfect attendance (no absences):

```
mysql> SELECT * FROM student
    -> WHERE student_id IN (SELECT student_id FROM absence);
```

```
+-------+-----+------------+
| name  | sex | student_id |
+-------+-----+------------+
| Kyle  | M   |          3 |
| Abby  | F   |          5 |
| Peter | M   |         10 |
| Will  | M   |         17 |
| Avery | F   |         20 |
+-------+-----+------------+
```

```
mysql> SELECT * FROM student
    -> WHERE student_id NOT IN (SELECT student_id FROM absence);
```

```
+--------+-----+------------+
| name   | sex | student_id |
+--------+-----+------------+
| Megan  | F   |          1 |
```

```
| Joseph    | M   |          2 |
| Katie     | F   |          4 |
| Nathan    | M   |          6 |
| Liesl     | F   |          7 |
+-----------+-----+------------+
```

IN and NOT IN also work for subqueries that return multiple columns. In other words, you can use them with table subqueries. In this case, use a row constructor to specify the comparison values to test against each column:

```
mysql> SELECT last_name, first_name, city, state FROM president
    -> WHERE (city, state) IN
    -> (SELECT city, state FROM president
    -> WHERE last_name = 'Roosevelt');
+-----------+------------+----------+-------+
| last_name | first_name | city     | state |
+-----------+------------+----------+-------+
| Roosevelt | Theodore   | New York | NY    |
| Roosevelt | Franklin D. | Hyde Park | NY    |
+-----------+------------+----------+-------+
```

IN and NOT IN actually are synonyms for = ANY and <> ALL, which are covered in the next section.

3) ALL, ANY, and SOME Subqueries

The ALL and ANY operators are used in conjunction with a relative comparison operator to test the result of a column subquery. They test whether the comparison value stands in particular relationship to all or some of the values returned by the subquery. For example, <= ALL is true if the comparison value is less than or equal to every value that the subquery returns, whereas <= ANY is true if the comparison value is less than or equal to any value that the subquery returns. SOME is a synonym for ANY.

This statement determines which president was born first by selecting the row with a birth date less than or equal to all the birth dates in the president table (only the earliest date satisfies this condition):

```
mysql> SELECT last_name, first_name, birth FROM president
    -> WHERE birth <= ALL (SELECT birth FROM president);
+---------------+---------------+---------------+
| last_name     | first_name    | birth         |
+---------------+---------------+---------------+
| Washington    | George        | 1732-02-22    |
+---------------+---------------+---------------+
```

Less usefully, the following statement returns all rows because every date is less than or equal to at least one other date (itself):

```
mysql> SELECT last_name, first_name, birth FROM president
    -> WHERE birth <= ANY (SELECT birth FROM president);
+---------------+---------------+---------------+
| last_name     | first_name    | birth         |
+---------------+---------------+---------------+
| Washington    | George        | 1732-02-22    |
| Adams         | John          | 1735-10-30    |
| Jefferson     | Thomas        | 1743-04-13    |
| Madison       | James         | 1751-03-16    |
| Monroe        | James         | 1758-04-28    |
```

When ALL, ANY, or SOME are used with the = comparison operator, the subquery can be a table subquery. In this case, you test return rows using a row constructor to provide the comparison values.

```
mysql> SELECT last_name, first_name, city, state FROM president
    -> WHERE (city, state) = ANY
    -> (SELECT city, state FROM president
```

```
    -> WHERE last_name = 'Roosevelt');
+--------------+------------+--------------+---------+
| last_name    | first_name | city         | state   |
+--------------+------------+--------------+---------+
| Roosevelt    | Theodore   | New York     | NY      |
| Roosevelt    | Franklin D.| Hyde Park    | NY      |
+--------------+------------+--------------+---------+
```

As mentioned in the previous section, `IN` and `NOT IN` are shorthand for `= ANY` and `<> ALL`. That is, `IN` means "equal to any of the rows returned by the subquery" and `NOT IN` means "unequal to all rows returned by the subquery."

4) EXISTS and NOT EXISTS Subqueries

The EXISTS and NOT EXISTS operators merely test whether a subquery returns any rows. If it does, EXISTS is true and NOT EXISTS is false. The following statements show some trivial examples of these subqueries. The first returns 0 if the absence table is empty, the second returns 1:

```
SELECT EXISTS (SELECT * FROM absence);
SELECT NOT EXISTS (SELECT * FROM absence);
```

EXISTS and NOT EXISTS actually are much more commonly used in correlated subqueries.

With EXISTS and NOT EXISTS, the subquery uses * as the output column list. There's no need to name columns explicitly, because the subquery is assessed as true or false based on whether it returns any rows, not based on the particular values that the rows might contain. You can actually write pretty much anything for the subquery column selection list, but if you want to make it explicit that you're returning a true value when the subquery succeeds, you might write it as SELECT 1 rather than SELECT *.

5) Correlated Subqueries

Subqueries can be uncorrelated or correlated:

- An uncorrelated subquery contains no references to values from the outer query, so it could be executed by itself as a separate statement. For example, the subquery in the following statement is uncorrelated because it refers only to the table t1 and not to t2:

```
SELECT j FROM t2 WHERE j IN (SELECT i FROM t1);
```

- A correlated subquery does contain references to values from the outer query, and thus is dependent on it. Due to this linkage, a correlated subquery cannot be executed by itself as a separate statement. For example, the subquery in the following statement is true for each value of column j in t2 that matches a column i value in t1:

```
SELECT j FROM t2 WHERE (SELECT i FROM t1 WHERE i = j);
```

The following EXISTS subquery identifies matches between the tables—that is, values that are present in both. The statement selects students who have at least one absence listed in the absence table:

```
SELECT student_id, name FROM student WHERE EXISTS
(SELECT * FROM absence WHERE absence.student_id = student.student_id);
```

NOT EXISTS identifies nonmatches—values in one table that are not present in the other. This statement selects students who have no absences:

```
SELECT student_id, name FROM student WHERE NOT EXISTS
(SELECT * FROM absence WHERE absence.student_id = student.student_id);
```

6) Subqueries in the FROM Clause

Subqueries can be used in the FROM clause to generate values. In this case, the result of the subquery acts like a table. A subquery in the FROM clause can participate in joins, its values can be tested in the

WHERE clause, and so forth. With this type of subquery, you must provide a table alias to give the subquery result a name:

```
mysql> SELECT * FROM (SELECT 1, 2) AS t1 INNER JOIN (SELECT 3, 4) AS t2;
+---+---+---+---+
| 1 | 2 | 3 | 4 |
+---+---+---+---+
| 1 | 2 | 3 | 4 |
+---+---+---+---+
```

c) Queries:

1. Get the names of employees who have a below average salary?

```
SQL> SELECT EMP_NAME
  2  FROM EMP
  3  WHERE SALARY <
  4  ( SELECT AVG( SALARY ) FROM EMP );

EMP_NAME

Jones
Roberts
Jarvis
Fletcher
Evans
```

2. Get the names of employees who have a greater than average salary?

```
SQL> SELECT EMP_NAME
  2  FROM EMP
  3  WHERE SALARY >
  4  ( SELECT AVG( SALARY ) FROM EMP );

EMP_NAME

Brown
Green
```

3. Get the names of employees who are paid exactly the avarage salary ?

```
SQL> SELECT EMP_NAME
  2  FROM EMP
  3  WHERE SALARY =
  4  ( SELECT AVG( SALARY ) FROM EMP );

no rows selected
```

4. Get the names of Employee is the most highly paid ?

```
SQL> SELECT EMP_NAME
```

```
  2  FROM EMP
  3  WHERE SALARY =
  4  ( SELECT MAX( SALARY ) FROM EMP );


EMP_NAME

Brown
Green
```

5. Get the names of Employee is the least paid?

```
SQL> SELECT EMP_NAME

  2  FROM EMP
  3  WHERE SALARY =
```

```
  4   ( SELECT MIN( SALARY ) FROM EMP );

EMP_NAME
--------------
Evans
```

6. Get the names of Employee whose salary exceeds the lowest salary by more than 4,000 ?

```
SQL> SELECT EMP_NAME
  2  FROM EMP
  3  WHERE SALARY >
  4  ( SELECT (4000 + MIN( SALARY ) ) AS BOUNDARY
  5     FROM EMP );

EMP_NAME
--------------
Roberts
Jarvis
Brown
Green
```

7. Get the names of employees for the departments that have only one employee

```
SQL> SELECT DEPT_NAME FROM DEPT
  2  WHERE 1 =
  3  ( SELECT COUNT(*) FROM EMP
  4    WHERE EMP.DEPT_NO = DEPT.DEPT_NO );

DEPT_NAME
--------------
Transport
```

8. Get the deadlines of projects that have more than 3 employees working on them

```
SQL> SELECT DEADLINE FROM PROJ
  2  WHERE 3 <
  3  ( SELECT COUNT(*) FROM ALLOC
  4    WHERE ALLOC.PROJ_NO = PROJ.PROJ_NO );

DEADLINE
------------
01-JAN-09
```

9. Get the names of employees who work in departments such that their salary is less than 10% of their department's budget

```
SQL> SELECT EMP_NAME FROM EMP
  2  WHERE SALARY <
  3  ( SELECT BUDGET/10 FROM DEPT
  4    WHERE EMP.DEPT_NO = DEPT.DEPT_NO );

EMP_NAME
--------------
Evans
```

10. Get the details of the projects not worked on by employee E2.

```
SQL> SELECT * FROM PROJ
  2  WHERE PROJ_NO IN
  3    (SELECT PROJ_NO FROM ALLOC
  4     WHERE EMP_NO = 'E2');
```

```
PR START_DAT DEADLINE

P2 21-JAN-05 30-SEP-07
```

11. Get the names of employees not allocated to projects.

```
SQL> SELECT EMP_NAME FROM EMP

  2  WHERE EMP_NO  NOT IN
  3  ( SELECT EMP_NO FROM

ALLOC ); EMP_NAME
    --------------
Rob
ert
s
Jar
vis
```

12. Get the details of all employees who are not departmental managers.

```
SQL> SELECT * FROM EMP

  2  WHERE EMP_NO  NOT IN
  3  ( SELECT MANAGER_NO FROM DEPT );

EM EMP_NAME        SALARY M DE

   --- -------------- ----------...   ---

E9 Fletcher         12000 S D1

E4 Evans            11000 S D5

E6 Green            38500 ? D2
```

13. Get the details of employees whose salary is less than that of all those employees who work on project P2.

```
SQL> SELECT * FROM EMP

  2  WHERE SALARY <ALL
  3  ( SELECT SALARY
  4    FROM EMP  NATURAL JOIN  ALLOC
  5    WHERE PROJ_NO = 'P2' );


EM EMP_NAME        SALARY M DE

   --- -------------- ----------...   ---

E4 Evans            11000 S D5
```

14. Get details of the department with the largest budget;

```
SQL> SELECT * FROM
                DEPT
2  WHERE BUDGET >=ALL
  3   ( SELECT BUDGET FROM
DEPT ); DE DEPT_NAME MA
--- -------------- --- ----------
    BUDGET

D2 Sales        E5      250000
```

Querying the Database based nesting and applying various Clauses.

Create the following Tables:

EMPLOYEE(Emp_id, EMP_name,Job_name,Manager_id,Hire_date,Salary,Deptno)

DEPARTMENT(Deptno, Dname, MGRSSN)

PROJECT(Pname,Pno,Plocation,Deptno)

Q1. Enter the following data into the Employee Table:

emp_id	emp_name	job_name	manager_id	hire_date	salary		dep_no
68319	KAYLING	PRESIDENT		1991-11-18	6000.00		1001
66928	BLAZE	MANAGER	68319	1991-05-01	2750.00		3001
67832	CLARE	MANAGER	68319	1991-06-09	2550.00		1001
65646	JONAS	MANAGER	68319	1991-04-02	2957.00		2001
67858	SCARLET	ANALYST	65646	1997-04-19	3100.00		2001
69062	FRANK	ANALYST	65646	1991-12-03	3100.00		2001
63679	SANDRINE	CLERK	69062	1990-12-18	900.00		2001
64989	ADELYN	SALESMAN	66928	1991-02-20	1700.00		3001
65271	WADE	SALESMAN	66928	1991-02-22	1350.00		3001
66564	MADDEN	SALESMAN	66928	1991-09-28	1350.00		3001
68454	TUCKER	SALESMAN	66928	1991-09-08	1600.00		3001
68736	ADNRES	CLERK	67858	1997-05-23	1200.00		2001
69000	JULIUS	CLERK	66928	1991-12-03	1050.00		3001
69324	MARKER	CLERK	67832	1992-01-23	1400.00		1001

Department Table

deptno	dname	Citylocation	dCountry
1001	Accounting	New York	United States of America,
2001	Research	Dallas	United States
3001	Sales	Chicago	United States of America
4001	Marketing	Los Angeles	United States

Project Table

Pno	Pname	PCitylocation	PCountry
111	P_1	New York	United States of America,
112	P_2	Dallas	United States
113	P_3	Chicago	United States of America
114	P_4	Denmark	northern Europe
115	P_5	Paris	France
116	P_6	Chicago	United States of America

Q1. Display sum of salaries of each job.

Q2. Display the details of employees sorting the salary in increasing order.

Q3. Display number of employees working in each department and their department name.

Q4. Display lowest paid employee details under each manager.

Q5. Show the record of employee earning salary greater than 1600 in each department.

Q6. Display the employee names and id of the department having more than five employees.

Q7. Display the employee of each department who don't earn more than 1000$.

Q8. Find out how many employees work under the name of salesman.

Q9. Display the total employees who work as manager with increasing order of their salaries.

Q10. Find the location of department accounting and no of employees working in that department.

Q11. Display the employee of each department with highest salary employee at the top of each list.

Q12. Display the highest and lowest salary employee of each department.

```
mysql> select sum(salary),jobname from emp_itA21 group by jobname;
+-------------+-----------+
| sum(salary) | jobname   |
+-------------+-----------+
|        6000 | president |
|        8257 | manager   |
|        6200 | analyst   |
|        4550 | clerk     |
|        5750 | salesman  |
+-------------+-----------+
5 rows in set (0.01 sec)

mysql> select * from emp_itA21 order by salary asc;
+-------+----------+-----------+-----------+------------+--------+--------+--------+
| eid   | ename    | jobname   | managerid | hiredate   | salary | ebonus | deptno |
+-------+----------+-----------+-----------+------------+--------+--------+--------+
| 63679 | sandrine | clerk     |     69062 | 1990-12-18 |    900 |    150 |   2001 |
| 69000 | julius   | clerk     |     66928 | 1991-12-03 |   1050 |    150 |   3001 |
| 68736 | adnres   | clerk     |     67858 | 1997-05-23 |   1200 |    150 |   2001 |
| 65271 | wade     | salesman  |     66928 | 1991-02-22 |   1350 |    180 |   3001 |
| 66564 | madden   | salesmen  |     66928 | 1991-09-28 |   1350 |    180 |   3001 |
| 68454 | tucker   | salesman  |     66928 | 1991-09-08 |   1350 |    180 |   3001 |
| 69324 | marker   | clerk     |     67832 | 1992-01-23 |   1400 |    150 |   1001 |
| 64989 | adelyn   | salesman  |     66928 | 1991-02-20 |   1700 |    180 |   3001 |
| 67832 | clare    | manager   |     68319 | 1991-06-09 |   2550 |    200 |   1001 |
| 66928 | blaze    | manager   |     68319 | 1991-05-01 |   2750 |    200 |   3001 |
| 65646 | jonas    | manager   |     68319 | 1991-04-02 |   2957 |    200 |   2001 |
| 67858 | scarlet  | analyst   |     65646 | 1997-04-19 |   3100 |    250 |   2001 |
| 69062 | frank    | analyst   |     65646 | 1991-12-03 |   3100 |    250 |   2001 |
| 68319 | kayling  | president |     68210 | 1991-11-18 |   6000 |    300 |   1001 |
+-------+----------+-----------+-----------+------------+--------+--------+--------+
14 rows in set (0.01 sec)

mysql> select d.dname,count(e.ename) as emp_count from dept_itA21 d
    -> left join emp_itA21 e on d.deptno=e.deptno
    -> group by d.dname order by d.dname;
+------------+-----------+
| dname      | emp_count |
+------------+-----------+
| accounting |         3 |
| marketing  |         6 |
| research   |         5 |
| sales      |         6 |
+------------+-----------+
4 rows in set (0.00 sec)

mysql> select e.eid,e.ename,d.deptno,e.salary
    -> from emp_itA21 e
    -> join dept_itA21 d on e.deptno=d.deptno
    -> where e.salary < 1000;
+-------+----------+--------+--------+
| eid   | ename    | deptno | salary |
+-------+----------+--------+--------+
| 63679 | sandrine |   2001 |    900 |
+-------+----------+--------+--------+
1 row in set (0.00 sec)
```

```
mysql> select d.deptno,d.dname, e.ename,e.salary
    -> from dept_itA21 d
    -> join emp_itA21 e on d.deptno=e.deptno
    -> where salary>1600;
+--------+------------+---------+--------+
| deptno | dname      | ename   | salary |
+--------+------------+---------+--------+
|   1001 | accounting | kayling |   6000 |
|   3001 | sales      | blaze   |   2750 |
|   1001 | accounting | clare   |   2550 |
|   2001 | research   | jonas   |   2957 |
|   2001 | research   | scarlet |   3100 |
|   2001 | research   | frank   |   3100 |
|   3001 | sales      | adelyn  |   1700 |
+--------+------------+---------+--------+
7 rows in set (0.00 sec)

mysql> select e.eid,e.ename from emp_itA21 e
    -> join dept_itA21 d on e.deptno=d.deptno
    -> where ( e.eid)>5;
+-------+----------+
| eid   | ename    |
+-------+----------+
| 68319 | kayling  |
| 66928 | blaze    |
| 67832 | clare    |
| 65646 | jonas    |
| 67858 | scarlet  |
| 69062 | frank    |
| 63679 | sandrine |
| 64989 | adelyn   |
| 65271 | wade     |
| 66564 | madden   |
| 68454 | tucker   |
| 68736 | adnres   |
| 69000 | julius   |
| 69324 | marker   |
+-------+----------+
14 rows in set (0.00 sec)
```

```
mysql> select d.deptno, d.dname, e.eid, e.ename, e.salary
    -> from dept_itA21 d
    -> join emp_itA21 e on d.deptno=e.deptno
    -> where e.salary<=1000;
+--------+----------+-------+----------+--------+
| deptno | dname    | eid   | ename    | salary |
+--------+----------+-------+----------+--------+
|   2001 | research | 63679 | sandrine |    900 |
+--------+----------+-------+----------+--------+
1 row in set (0.00 sec)

mysql> select count(ename) from emp_itA21 where jobname='salesman';
+--------------+
| count(ename) |
+--------------+
|            4 |
+--------------+
1 row in set (0.00 sec)

mysql> select e.ename, e.salary from emp_itA21 e where jobname='manager'
    -> order by salary desc;
+-------+--------+
| ename | salary |
+-------+--------+
| jones |   2957 |
| blake |   2750 |
| clare |   2550 |
+-------+--------+
3 rows in set (0.00 sec)

mysql> select d.cityloca, count(e.eid)
    -> from dept_itA21 d
    -> join emp_itA21 e on d.deptno=e.deptno
    -> where d.dname='accounting'
    -> group by d.cityloca;
+----------+--------------+
| cityloca | count(e.eid) |
+----------+--------------+
| new york |            3 |
+----------+--------------+
1 row in set (0.00 sec)
```

```
mysql> select d.deptno,d.dname,e.eid,e.ename,e.salary
    -> from dept_itA21 d
    -> join emp_itA21 e on d.deptno=e.deptno
    -> where e.salary=(select max(salary) from emp_itA21 where deptno =d.deptno)
    -> order by d.deptno, e.salary desc;
+--------+------------+-------+---------+--------+
| deptno | dname      | eid   | ename   | salary |
+--------+------------+-------+---------+--------+
|   1001 | accounting | 68319 | kayling |   6000 |
|   2001 | research   | 67858 | scarlet |   3100 |
|   2001 | research   | 69062 | frank   |   3100 |
|   3001 | sales      | 66928 | blaze   |   2750 |
+--------+------------+-------+---------+--------+
4 rows in set (0.00 sec)

mysql> select d.deptno,d.dname,'highest' as salary_type,e1.ename as emp_name,e1.salary as emp_salary
    -> from dept_itA21 d
    -> join emp_itA21 e1 on d.deptno=e1.deptno
    -> where e1.salary=(select max(salary) from emp_itA21 where deptno=d.deptno)
    -> union all
    -> select d.deptno,d.dname,'lowest' as salary_type,e2.ename as emp_name,e2.salary as emp_salary
    -> from dept_itA21 d
    -> join emp_itA21 e2 on d.deptno=d.deptno
    -> where e2.salary=(select min(salary) from emp_itA21 where deptno=d.deptno)
    -> order by salary_type;
+--------+------------+-------------+----------+------------+
| deptno | dname      | salary_type | emp_name | emp_salary |
+--------+------------+-------------+----------+------------+
|   1001 | accounting | highest     | kayling  |       6000 |
|   3001 | sales      | highest     | blaze    |       2750 |
|   2001 | research   | highest     | scarlet  |       3100 |
|   2001 | research   | highest     | frank    |       3100 |
|   2001 | research   | lowest      | sandrine |        900 |
|   3001 | sales      | lowest      | julius   |       1050 |
|   1001 | accounting | lowest      | marker   |       1400 |
+--------+------------+-------------+----------+------------+
7 rows in set (0.01 sec)
```

Teacher's Signature _____

PRACTICAL NO : 07

Page No.
Date

Aim: To perform nested queries and joining queries using DML command.

Queries:-

1] Display sum of salaries of each job

⇒ Select job_name, sum(salary) as Total_Salary
from Employee_IT_A_19
Group by job_name;

2] Display the details of employees sorting the salary in increasing order.

⇒ Select * from Employee_IT_A_19
ordered by Salary asc;

3] Display number of employees working in each department and their department name.

⇒ select D.dname as dept_name,
count(E.Emp_id) as employee_count
from Department_IT_A_19 D
Left join Employee_IT_A_19 E on
D.deptno = E.deptno
Group by D.dname
order by D.dname;

Teacher's Signature

Page No.
Date

Q4] Display lowest paid employees details under each managers.

=> select * from Employee_IT_A_19 where salary = (select min(salary) from Employee_IT_A_19 where Job_name = 'manager');

Q5] Show the record of employee earning salary greater than 1600 in each department.

=> Select D.dname, D.dname, E.emp_name, E.salary from Department_IT_A_19 D join Employee_IT_A_19 E on D.dept_no . E.depno where salary > 1600;

Q6] Display the employee-names and id of the department having more than five employees.

=> Select D.dept_no, D.dname, E.emp_name, E.salary from Department_IT_A_19 D Join Employee_IT_A_19 E on D.dept_no = E.depno where salary > 1600;

Select E.emp_id, E.emp_name from Employee_IT_A_19 join Department_IT_A_19 on E.dep_no = D.dep.no where (E.emp_id) > 5;

Teacher's Signature

Page No.
Date

Q7] Display the employee of each department who don't earn more than 1000 $

=> Select D.dept.no, D.dname, E.emp_id, E.emp.name, E.salary
From Department_IT_A_19 D
join Employee_IT_A_19 E on D.dept_no.= E.dep_no
where E.salary <=1000;

Q8] Find out how many employees work under name of Salesman

=> select count (*) from Employee_I.T_A_19
where job_name = 'salesman';

Q9] Display the total employees who work as manager with increasing order of their salaries.

=> select E.emp_name, E.salary from Employee_IT_A_19 E where job name 'manager'.
order by salary desc;

Q10] Find the location of department according and no. of employees working in that department.

→ Select D.city, count (E.emp-id)
from Department_IT_A_19 D
join Employee_IT_A_19 E on D. Dept_no=E.dept_no
where D.dname = ~~'according~~ 'accounting'
group by D.city;

Teacher's Signature

Page No.
Date

Q11] Display the employee of each department with highest salary employee at the top of each list.

⇒
```
Select D.dept_no, D.dname, E.emp_id,
E.emp_name, E.salary
From Department_IT_A_19 D
join Employee_IT_A_19 E on D.dept_no = E.dep no.
where E.salary = (select max(salary) from
Employee_IT_A_19 where dept.no = D.dept_no)
order by D.dept.no, E.salary desc;
```

Q12] Display the highest to lowest salary empbyee of each department

⇒
```
Select D.dept_no, D.dname, 'Highest' as Salary_Type E1.Emp_name,
E1.Salary as Employee salary from Department_IT_A_19 D
join Employee_IT_A_19 E1 on D.dept_no = E1.dept_no
where E1.salary = (select max (salary) from Employee_IT_A_19
where dep.no = D.dept no UNION ALL select
D.dept.no, D.dname, 'Lowest' as salary_Type, E2.emp_name
as Employee_Name, E2.Salary as Employee salary from Department
_IT_A_19 D
join Employee_IT_A_19 E2 on dept-no = D.dept_no.
```

Conclusion :-

Thus, we executed query using DML.

Dhanda.

Teacher's Signature

Practical Number: 8

Title of the Exercise : DATA CONTROL LANGUAGE(DCL), TRANSACTION CONTROL LANGUAGE (TCL) COMMANDS

Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To study the various data language commands (DCL, TCL) and implements them on the database.

PROCEDURE

b) Procedure for doing the experiment:

Step no.	Details of the step
1	**DCL COMMAND** The DCL language is used for controlling the access to the table and hence securing the database. DCL is used to provide certain privileges to a particular user. Privileges are rights to be allocated.
2	The privilege commands are namely, Grant and Revoke
3	The various privileges that can be granted or revoked are, Select Insert Delete Update References Execute All
4	**GRANT COMMAND:** It is used to create users and grant access to the database. It requires database administrator (DBA) privilege, except that a user can change their password. A user can grant access to their database objects to other users.
5	**REVOKE COMMAND:** Using this command , the DBA can revoke the granted database privileges from the user.
6	**TCL COMMAND** **COMMIT:** command is used to save the Records. **ROLL BACK:** command is used to undo the Records. **SAVE POINT** command is used to undo the Records in a particular transaction.

c) SQL Commands DCL Commands GRANT COMMAND

Grant < database_priv [database_priv…..] > to <user_name> identified by <password> [,<password…..];

Grant <object_priv> | All on <object> to <user | public> [With Grant Option];

REVOKE COMMAND

Revoke <database_priv> from <user [, user] >;

Revoke <object_priv> on <object> from < user | public >;

<database_priv> -- Specifies the system level priveleges to be granted to the users or roles. This includes create / alter / delete any object of the system.

<object_priv> -- Specifies the actions such as alter / delete / insert / references / execute / select / update for tables.

<all> -- Indicates all the priveleges.

[With Grant Option] – Allows the recipient user to give further grants on the objects.

The priveleges can be granted to different users by specifying their names or to all users by using the "Public" option.

TCL COMMANDS:

Syntax:

SAVEPOINT: SAVEPOINT <SAVE POINT NAME>;

ROLLBACK: ROLL BACK <SAVE POINT NAME>;

COMMIT: Commit;

d) Queries:

Tables Used:

Consider the following tables namely "DEPARTMENTS" and "EMPLOYEES"

Their schemas are as follows ,

Departments (dept _no , dept_ name , dept_location);

Employees (emp_id , emp_name , emp_salary);

Q1: Develop a query to grant all privileges of employees table into departments table

Ans:

SQL> Grant all on employees to departments; Grant succeeded.

Q2: Develop a query to grant some privileges of employees table into departments table

Ans:

SQL> Grant select, update , insert on departments to departments with grant option; Grant succeeded.

Q3: Develop a query to revoke all privileges of employees table from departments table

Ans:

SQL> Revoke all on employees from departments; Revoke succeeded.

Q4: Develop a query to revoke some privileges of employees table from departments table

Ans:

SQL> Revoke select, update , insert on departments from departments; Revoke succeeded.

Q5: Write a query to implement the save point

Ans:

SQL> SAVEPOINT S1;

Savepoint created.

SQL> select * from emp;

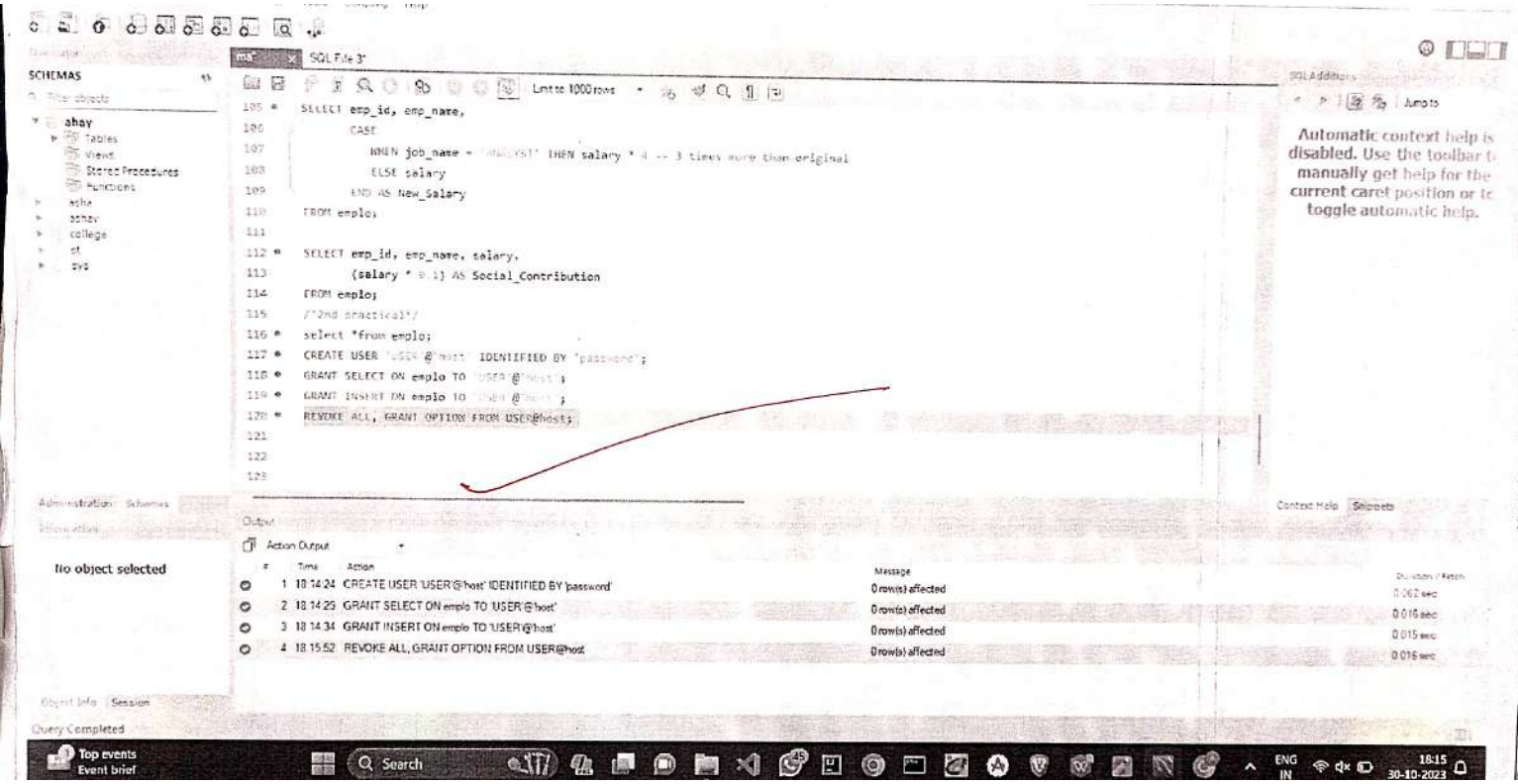
EMPNO	ENAME	JOB	DEPTNO	SAL
1	Mathi	AP	1	10000
2	Arjun	ASP	2	15000
3	Gugan	ASP	1	15000
4	Karthik	Prof	2	30000

SQL> INSERT INTO EMP VALUES(5,'Akalya','AP',1,10000);

1 row created.

SQL> select * from emp;

EMPNO	ENAME	JOB	DEPTNO	SAL
1	Mathi	AP	1	10000
2	Arjun	ASP	2	15000


SCHEMAS
ahay
Tables
Views
Stored Procedures
Functions
esha
sohay
college
st
sys
SQL File 3*
Limit to 1000 rows
SELECT emp_id, emp_name,
CASE
WHEN job_name = 'ANALYST' THEN salary * 3 -- 3 times more than original
ELSE salary
END AS New_Salary
FROM emplo;
SELECT emp_id, emp_name, salary,
(salary * 0.1) AS Social_Contribution
FROM emplo;
/*2nd practical*/
select *from emplo;
CREATE USER 'USER'@'host' IDENTIFIED BY 'password';
GRANT SELECT ON emplo TO 'USER'@'host';
GRANT INSERT ON emplo TO 'USER'@'host';
REVOKE ALL, GRANT OPTION FROM USER@host;
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Administration
Schemas
No object selected
Object Info
Session
Context Help
Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 18:14:24 CREATE USER 'USER'@'host' IDENTIFIED BY 'password' 0 row(s) affected 0.062 sec
2 18:14:29 GRANT SELECT ON emplo TO 'USER'@'host' 0 row(s) affected 0.016 sec
3 18:14:34 GRANT INSERT ON emplo TO 'USER'@'host' 0 row(s) affected 0.015 sec
4 18:15:52 REVOKE ALL, GRANT OPTION FROM USER@host 0 row(s) affected 0.016 sec
Query Completed
Top events
Event brief
Search
ENG
IN
18:15
30-10-2023

PRACTICAL NO: 08

Page No.
Date.

Aim: To study the various data language commands (DCL, TCL) and implements them on the database

Theory :- DCL command / DCL language is used to control the acess to the table and hence securing the database. DCL is used to provide certain privileges to a particular user.

Privileges are right to be allocated.

The privileges commands are namely Grant and Remoke.

Grant command : It is used to create User and grant acess to the database. It requires database administator (DBA) Privelige, except that a user can change their password. A user can grant acess to their database object to the other users.

Revoke command :- Using this command the DBA can revoke the grant database privilege for the user.

TCL Command :-

- commit :- it is used to save the Records
- Roll back :- command is used to undo the Records
- Save point :- command is used to undo the records in a particular transaction.

Conclusion :- Thus, we have learn the DCL and TCL command in the SQL successfully.

Pranali

3 Gugan	ASP	1	15000
4 Karthik	Prof	2	30000
5 Akalya	AP	1	10000

Q6: Write a query to implement the rollback
Ans:

```
SQL> rollback s1;
SQL> select * from emp;
```

EMPNO	ENAME	JOB	DEPTNO	SAL
1	Mathi	AP	1	10000
2	Arjun	ASP	2	15000
3	Gugan	ASP	1	15000
4	Karthik	Prof	2	30000

Q6: Write a query to implement the commit
Ans:

```
SQL> COMMIT;
Commit complete.
```

e)Result
The DCL,TCL commands was performed successfully and executed.

Practical Number: 09

Title of the Exercise : TRIGGER
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To create triggers for various events such as insertion, updation, etc.,

PROCEDURE

a) PL/SQL Syntax:
TRIGGER

A Trigger is a stored procedure that defines an action that the database automatically take when some database-related event such as Insert, Update or Delete occur.

TRIGGER VS. PROCEDURE VS CURSOR

TRIGGER	PROCEDURES	CURSORS
These are named PL/SQL blocks.	These are named PL/SQL blocks.	These are named PL/SQL blocks.
These are invoked automatically.	User as per need invokes these.	These can be created both explicitly and implicitly.
These can"t take parameters.	These can take parameters.	These can take parameters.
These are stored in database.	These are stored in database.	These are not stored in database.

TYPES OF TRIGGERS

The various types of triggers are as follows,

- **Before**: It fires the trigger before executing the trigger statement.
- **After**: It fires the trigger after executing the trigger statement.
- **For each row**: It specifies that the trigger fires once per row.
- **For each statement**: This is the default trigger that is invoked. It specifies that the trigger fires once per statement.

VARIABLES USED IN TRIGGERS

- :new
- :old

These two variables retain the new and old values of the column updated in the database. The values in these variables can be used in the database triggers for data manipulation

Row Level Trigger vs. Statement Level Trigger:

Row Level Trigger	Statement Level Trigger
These are fired for each row affected by the DML statement.	These are fired once for the statement instead of the no of rows modified by it.
These are used for generating/checking the values begin inserted or updated.	These are used for generated the summary information.

```
SQL> CREATE OR REPLACE TRIGGER before_insert_employee
  2  BEFORE INSERT ON EMP
  3  FOR EACH ROW
  4  BEGIN
  5    IF :NEW.SALARY < 0 THEN
  6      RAISE_APPLICATION_ERROR(-20001, 'Salary cannot be negative.');
  7    END IF;
  8  IF :NEW.MANAGER_ID IS NULL THEN
  9      :NEW.MANAGER_ID := 0; -- Set to a default manager ID if not specified.
 10    END IF;
 11    INSERT INTO EMPLOYEE_AUDIT (EMP_ID, ACTION, ACTION_DATE)
 12    VALUES (:NEW.EMP_ID, 'INSERT', SYSDATE);
 13  END;
 14  /

Warning: Trigger created with compilation errors.
```

Q2. Create a trigger after into department table .

```
SQL> CREATE OR REPLACE TRIGGER after_insert_department
  2  AFTER INSERT ON DEP
  3  FOR EACH ROW
  4  BEGIN
  5  INSERT INTO DEPARTMENT_AUDIT (DEPTNO, DNAME, LOCATION, INSERT_DATE)
  6    VALUES (:NEW.DEPTNO, :NEW.DNAME, :NEW.LOCATION, SYSDATE);
  7  END;
  8  /

Warning: Trigger created with compilation errors.
```

Q3. Drop a trigger on department table.

```
SQL> DROP TRIGGER after_insert_department;

Trigger dropped.
```

Q4. Create a trigger on project table before update.

```
SQL> CREATE OR REPLACE TRIGGER before_update_project
  2  BEFORE UPDATE ON PROJECT
  3  FOR EACH ROW
  4  BEGIN
  5    IF :NEW.END_DATE < SYSDATE THEN
  6      RAISE_APPLICATION_ERROR(-20001, 'Cannot update the project end date to a past date.');
  7    END IF;
  8  END;
  9  /

Warning: Trigger created with compilation errors.
```

Q5. Create a trigger on employee table for update.

```
SQL> CREATE OR REPLACE TRIGGER after_update_employee
  2  AFTER UPDATE ON EMP
  3  FOR EACH ROW
  4  BEGIN
  5    INSERT INTO EMPLOYEE_AUDIT (EMP_ID, ACTION, ACTION_DATE)
  6    VALUES (:NEW.EMP_ID, 'UPDATE', SYSDATE);
  7  END;
  8  /

Warning: Trigger created with compilation errors.
```

Q6. Create a trigger on employee table and drop it.

```
SQL> CREATE OR REPLACE TRIGGER after_insert_employee
  2  AFTER INSERT ON EMP
  3  FOR EACH ROW
  4  BEGIN
  5    INSERT INTO EMPLOYEE_AUDIT (EMP_ID, ACTION, ACTION_DATE)
  6    VALUES (:NEW.EMP_ID, 'INSERT', SYSDATE);
  7  END;
  8  /

Warning: Trigger created with compilation errors.

SQL> DROP TRIGGER after_insert_employee;

Trigger dropped.
```

Before trigger vs. after trigger

Before Triggers	After Triggers
Before triggers are fired before the DML statement is actually executed.	After triggers are fired after the DML statement has finished execution.

Sytax:

Create or replace trigger <trg_name> Before /After Insert/Update/Delete
[of column_name, column_name….]
on
<table_name>
[for each row]
[when
condition] begin
---statement
end;

Q1: Create a trigger that insert current user into a username column of an existing table

b) Procedure for doing the experiment:

Step no.	Details of the step
1	Create a table itstudent4 with name and username as arguments
2	Create a trigger for each row that insert the current user as user name into a table
3	Execute the trigger by inserting value into the table

d) Program:

```
SQL> create table itstudent4(name varchar2(15),username varchar2(15));
Table created.
SQL> create or replace trigger itstudent4 before insert on itstudent4 for each
 row 2  declare
 3  name varchar2(20);
 4  begin
 5  select user into name from
 dual; 6  :new.username:=name;
 7 end;
 8  /
Trigger created.
```

e) Output:

```
SQL> insert into itstudent4 values('&name','&username');
Enter value for name: akbar
Enter value for username: ranjani
old 1: insert into itstudent4 values('&name','&username')
new 1: insert into itstudent4 values('akbar','ranjani')
1 row
created.
SQL> /
Enter value for name: suji
```

```
Enter value for username:
priya
old 1: insert into itstudent4 values('&name','&username')
new 1: insert into itstudent4 values('suji','priya')
1 row created.
SQL> select * from itstudent4;

NAME        USERNAME

akbar       SCOTT
suji        SCOTT
```

Q2: Create a Simple Trigger that does not allow Insert Update and Delete Operations on the Table

d) Program:

Table used:

```
SQL> select * from itempls;
ENAME        EID
             SALARY
----------------------
xxx          11    10000
yyy          12    10500
zzz          13    15500
```

Trigger:

```
SQL> create trigger ittrigg before insert or update or delete on itempls for each row
 2 begin
 3 raise_application_error(-20010,'You cannot do manipulation');
 4 end;
 5
 6 /
Trigger created.
```

e) Output:

```
SQL> insert into itempls values('aaa',14,34000);
insert into itempls values('aaa',14,34000)
       *
ERROR at line
1:
ORA-20010: You cannot do manipulation
ORA-06512: at "STUDENT.ITTRIGG", line
2
ORA-04088: error during execution of trigger 'STUDENT.ITTRIGG'

SQL> delete from itempls where ename='xxx';
delete from itempls where ename='xxx'
       *
ERROR at line
1:
ORA-20010: You cannot do manipulation
ORA-06512: at "STUDENT.ITTRIGG", line
2
ORA-04088: error during execution of trigger 'STUDENT.ITTRIGG'
```

```
SQL> update itempls set eid=15 where ename='yyy';
update itempls set eid=15 where ename='yyy'
       *
ERROR at line 1:
ORA-20010: You cannot do manipulation
ORA-06512: at "STUDENT.ITTRIGG", line
2
ORA-04088: error during execution of trigger 'STUDENT.ITTRIGG'
```

Q3: Create a Trigger that raises an User Defined Error Message and does not allow updating and Insertion

d) Program:

Table used:

```
SQL> select * from itempls;
ENAME          EID
               SALARY
----------------------.
xxx            11    10000
yyy            12    10500
zzz            13    15500
```

Trigger:

```
SQL> create trigger ittriggs before insert or update of salary on itempls for each row
 2 declare
 3  triggsal itempls.salary%type;
 4  begin
 5  select salary into triggsal from itempls where eid=12;
 6 if(:new.salary>triggsal or :new.salary<triggsal)then
 7 raise_application_error(-20100,'Salary has not been changed');
 8 end if;
 9 end;
10 /
Trigger created.
```

e) Output:

```
SQL> insert into itempls values ('bbb',16,45000);
insert into itempls values ('bbb',16,45000)
       *
ERROR at line 1:
ORA-04098: trigger 'STUDENT.ITTRIGGS' is invalid and failed re-validation
SQL> update itempls set eid=18 where ename='zzz';
update itempls set eid=18 where ename='zzz'
       *
ERROR at line 1:
ORA-04298: trigger 'STUDENT.ITTRIGGS' is invalid and failed re-validation
```

Q4: develop a query to Drop the Created Trigger

Ans:

```
SQL> drop trigger ittrigg;
Trigger dropped.
```

f) Result:

Thus the creation of triggers for various events such as insertion, updating, etc., was performed and executed successfully.

Practical Number: 10

Title of the Exercise : **VIEWS**
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT
To create and manipulate various database objects of the Table using views

PROCEDURE

a) Procedure for doing the experiment:

Step no.	Details of the step
1	**Views:** A view is the tailored presentation of data contained in one or more table and can also be said as restricted view to the data"s in the tables. A view is a "virtual table" or a "stored query" which takes the output of a query and treats it as a table. The table upon which a view is created is called as base table.
2	A view is a logical table based on a table or another view. A view contains no data of its own but is like a window through which data from tables can be viewed or changed. The tables on which a view is based are called base tables. The view is stored as a SELECT statement in the data dictionary
3	**Advantages of a view:** a. Additional level of table security. b. Hides data complexity. c. Simplifies the usage by combining multiple tables into a single table. d. Provides data"s in different perspective.
4	**Types of view:** Horizontal -> enforced by where cause Vertical -> enforced by selecting the required columns

a) SQL Commands

Creating and dropping view:

Syntax:

Create [or replace] view <view name> [column alias names] as <query> [with <options> conditions];
Drop view <view name>;

Example:

Create or replace view empview as select * from emp;
Drop view empview;

b) Queries: Tables used:

```
SQL> select * from emp;
     EMPNO ENAME      JOB          DEPTNO       SAL
----------------------------------------------------
         1 Mathi      AP                1      10000
         2 Arjun      ASP               2      12000
         3 Gugan      ASP               2      20000
         4 Karthik    AP                1      15000
```

AIM : To create and manipulate various database objects of the Table using views

Q1. Display the details of employees containing emp_id, emp_name , job_name , manager_id , hire_date , salary and include only those employees who work as salesman.

```
SQL> SELECT EMP_ID,EMP_NAME, JOB_NAME, MANAGER_ID,HIRE_DATE,SALARY FROM EMP WHERE JOB_NAME = 'SALEMAN';

    EMP_ID EMP_NAME           JOB_NAME           MANAGER_ID HIRE_DATE
---------- ------------------ ------------------ ---------- ---------
    SALARY
----------
     65271 WADE               SALEMAN                 66928 22-FEB-91
      1350

     66564 MADDEN             SALEMAN                 66928 28-SEP-91
      1350

     68454 TUCKER             SALEMAN                 66928 08-SEP-91
```

Q2. Create a separate table for department having country as United states.

```
SQL> CREATE TABLE DEP1 AS
  2  SELECT * FROM DEP WHERE COUNTRY = 'US';

Table created.

SQL> SELECT * FROM DEP1;

     DEPNO DNAME              CITY               COUNTRY
---------- ------------------ ------------------ --------------------
      1001 ACCOUNTING         NEW YORK           US
      2001 RESEARCH           DALLAS             US
      3001 SALES              CHICAGO            US
      4001 MARKETING          LOS ANGELES        US
```

Q3. The details of employees is needed for extracting the employees having salary more than 10,000.Provide the details.

```
SQL> SELECT
  2       emp_id,
  3       emp_name,
  4       job_name,
  5       manager_id,
  6       hire_date,
  7       salary
  8  FROM
  9       EMP
 10  WHERE
 11       salary > 10000;

no rows selected
```

Q4. The project details are required for some purpose. Provide the data about the project location.

```
SQL> SELECT PNO, CITY
  2  FROM PROJECT;

       PNO CITY
---------- --------------------
       111 NEW YORK
       112 DALLAS
       113 CHICAGO
       114 DENMARK
       115 PARIS
       116 CHICAGO
       117 PARIS

7 rows selected.
```

Q5. Drop the views on departments.

```
SQL> CREATE VIEW department_view AS
  2  SELECT DEPNO, DNAME, LOCATION
  3  FROM DEP;
SELECT DEPNO, DNAME, LOCATION
                     *
ERROR at line 2:
ORA-00904: "LOCATION": invalid identifier


SQL> CREATE VIEW department_view AS
  2  SELECT DEPNO, DNAME, CITY
  3  FROM DEP;

View created.

SQL> DROP VIEW department_view;

View dropped.
```

Q1: The organization wants to display only the details of the employees those who are ASP. (Horizontal portioning)

Solution:

1. Create a view on emp table named managers
2. Use select from clause to do horizontal partioning

Ans:

```
SQL> create view empview as select * from emp where job='ASP';
View created.
SQL> select * from empview;
   EMPNO ENAME   JOB       DEPTNO  SAL
---------- ---------- ---------- ----------
         2 Arjun      ASP              2   12000
         3 Gugan      ASP              2   20000
```

Q2: The organization wants to display only the details like empno, empname, deptno, deptname of the employees. (Vertical portioning)

Solution:

1. Create a view on emp table named general 2. Use select from clause to do vertical partioning

Ans:

```
SQL> create view empview1 as select ename,sal from emp;
View created.
```

Q3: Display all the views generated.

Ans:

```
SQL> select * from tab;
TNAME                TABTYPE  CLUSTERID

DEPT                 TABLE
EMP                  TABLE
EMPVIEW              VIEW
EMPVIEW1             VIEW
```

Q4: Execute the DML commands on the view created.

Ans:

```
SQL> select * from empview;
   EMPNO ENAME          JOB         DEPTNO      SAL
---------- ---------- ---------- ----------
         2 Arjun      ASP              2   12000
         3 Gugan      ASP              2   20000
```

Q5: Drop a view.

Ans: SQL> drop view empview1;
View dropped.

c) **Result:**

Thus the creation and manipulate various database objects of the Table using views was successfully executed.