JAIN
DEEMED-TO-BE UNIVERSITY

FACULTY OF
ENGINEERING
AND TECHNOLOGY

Database Management Systems Lab

4th Semester

Submitted By

Name: Ajay Kumar Mandal

USN: 20BTRCO048

JAIN DEEMED-TO-BE UNIVERSITY
FACULTY OF ENGINEERING AND TECHNOLOGY

Laboratory Certificate

This is to certify That Mr /Ms .....has satisfactorily completed the course of experiments in practical **Database Management Systems Lab** (18CS4SP04L) presented by the Jain (Deemed-to-be University) fourth semester course in laboratory of this college in the year 2021-22.

Date:

Name of the Candidate:.....

USN:.....

Date of Practical Examination.....

Signature of the Teacher
Incharge of the Batch

Valued	
Examiner-1	
Examiner-2	

Contents

Ajay Kumar Mandal
20BTRCO048
CSE – IOT

EXPERIMENT - 1: Create User in Oracle Database and grant and revoke the privileges and use of commit save point and rollback command.

Question-1 (Grant and Revoke)

1. Create a bank table which contain four filed account number, name, balance, mobile number and insertfive rows.

   → create table bank(Account_Number int, Name varchar(20), balance int, Mobile varchar(10));
   → insert into bank values (32890909, 'Roshan Raman Giri', 1000000,'9213489028');
   → insert into bank values (28903748, 'Sanjay Dutt', 100,'9001892789');
   → insert into bank values (37849027, 'Sharukh Khan', 12890098,'9389209832');
   → insert into bank values (83903789, 'Salman Khan', 543903,'7389278327');
   → insert into bank values (65948965, 'Ranveer Singh', 3478902,'73648937');

```
SQL> create table bank(Account_Number int, Name varchar(20), balance int, Mobile varchar(10));

Table created.

SQL> insert into bank values (32890909, 'Roshan Raman Giri', 1000000,'9213489028');

1 row created.

SQL> insert into bank values (28903748, 'Sanjay Dutt', 100,'9001892789');

1 row created.

SQL> insert into bank values (37849027, 'Sharukh Khan', 12890098,'9389209832');

1 row created.

SQL> insert into bank values (83903789, 'Salman Khan', 543903,'7389278327');

1 row created.

SQL> insert into bank values (65948965, 'Ranveer Singh', 3478902,'73648937');

1 row created.

SQL> select * form bank;
select * form bank
         *
ERROR at line 1:
ORA-00923: FROM keyword not found where expected


SQL> select * from bank;

ACCOUNT_NUMBER NAME                 BALANCE MOBILE
-------------- -------------------- ---------- ----------
      32890909 Roshan Raman Giri       1000000 9213489028
      28903748 Sanjay Dutt                 100 9001892789
      37849027 Sharukh Khan            12890098 9389209832
      83903789 Salman Khan               543903 7389278327
      65948965 Ranveer Singh            3478902 73648937
```

2. Create Manager user and assign All privilege to the manager user.

   → create user Manager identified by Manager;
   → grant all privileges to Manager;

3. Create a Raja user and assign Update, select privilege on bank table to the Raja users.

   → create user Raja identified by Raja;
   → grant update, select on bank to Raja;

4. Create a Babu user and assign select privilege on bank table to the Babu users.

   → create user Babu identified by Babu;
   → grant select on bank to Babu;

5. Revoke Update privilege on the bank table from the Raja users.

   → revoke update on bank from Raja;

```
SQL> Create user Manager identified by Manager;

User created.

SQL> grant all privileges to Manager;

Grant succeeded.

SQL> create user Raja identified by Raja;

User created.

SQL> grant update, select on bank to Raja;

Grant succeeded.

SQL> create user Babu identified by Babu;

User created.

SQL> grant select on bank to Babu;

Grant succeeded.

SQL> revoke update on bank from Raja;

Revoke succeeded.
```

Question-2 (Commit, Roll Back and Save Point)

Consider the following customer table and perform the transaction in the following order.

CUSTOMER ID	CUSTOMER NAME	STATE	COUNTRY

1	Akash	Delhi	India
2	Amit	Hyderabad	India
3	Jason	California	USA
4	John	Texas	USA

→ insert into customer values (1,'Akash','Delhi','India');
→ insert into customer values (2,'Amit','Hyderabad','India');
→ insert into customer values (3,'Jason','California','USA');insert into customer values(4,'John','Texas','USA');

```
SQL> create table customer(CUSTOMER_ID int, CUSTOMER_NAME varchar(20), STATE varchar(15), COUNTRY Varcha
r(15));

Table created.

SQL> insert into customer values (1,'Akash','Delhi','India');

1 row created.

SQL> insert into customer values (2,'Amit','Hyderabad','India');

1 row created.

SQL> insert into customer values (3,'Jason','California','USA');

1 row created.

SQL> insert into customer values (4,'John','Texas','USA');

1 row created.

SQL> select * from customer;

CUSTOMER_ID CUSTOMER_NAME        STATE           COUNTRY
----------- -------------------- --------------- ---------------
          1 Akash                Delhi           India
          2 Amit                 Hyderabad       India
          3 Jason                California      USA
          4 John                 Texas           USA
```

1. Delete the customer where State = 'Texas' and commit the transaction.

   → delete from customer where STATE='Texas';
   → commit;

2. Display customer tables.

   → select * from customer;

3. Rollback the transaction and display the customer table.

→ rollback;
→ select * from customer;

4. Create Save point one.

   → savepoint sp1;

5. Insert one record values (5, Karthi, Salem, India)

   → insert into customer values (5,'Karthi,'Salem', 'India');

6. Create Save point two.

   → savepoint sp2;

7. Update the customer state as Bangalore whose id is 1.

   → update customer set STATE='Bangalore' where CUSTOMER_ID=1;

8. Create Save point three.

   → savepoint sp3;

9. Delete the customer details whose id is 3.

   → delete from customer where CUSTOMER_ID=3;

10. Rollback to the Save point two and display customer table.

    → rollback to sp2;

11. Rollback to the save point one and display the customer table.

    → rollback to sp1;

```
SQL> delete from customer where STATE='Texas';

1 row deleted.

SQL> commit;

Commit complete.

SQL> select * from customer;

CUSTOMER_ID CUSTOMER_NAME        STATE           COUNTRY
----------- -------------------- --------------- ---------------
          1 Akash                Delhi           India
          2 Amit                 Hyderabad       India
          3 Jason                California      USA

SQL> rollback;

Rollback complete.

SQL> select * from customer;

CUSTOMER_ID CUSTOMER_NAME        STATE           COUNTRY
----------- -------------------- --------------- ---------------
          1 Akash                Delhi           India
          2 Amit                 Hyderabad       India
          3 Jason                California      USA

SQL> savepoint sp1;

Savepoint created.

SQL> insert into customer values (5,'Karthi,'Salem', 'India');
ERROR:
ORA-01756: quoted string not properly terminated


SQL> insert into customer values (5,'Karthi','Salem', 'India');

1 row created.

SQL> savepont sp2;
SP2-0734: unknown command beginning "savepont s..." - rest of line ignored.
SQL> savepoint sp2;

Savepoint created.

SQL> update customer set STATE='Bangalore' where CUSTOMER_ID=1;

1 row updated.

SQL> savepoint sp3;
```

```
Savepoint created.

SQL> delete from customer where CUSTOMER_ID=3;

1 row deleted.

SQL> rollback to sp2;

Rollback complete.

SQL> select * from customer;

CUSTOMER_ID CUSTOMER_NAME        STATE           COUNTRY
----------- -------------------- --------------- ---------------
          1 Akash                Delhi           India
          2 Amit                 Hyderabad       India
          3 Jason                California      USA
          5 Karthi               Salem           India

SQL> rollback to sp1;

Rollback complete.

SQL> select * from customer;

CUSTOMER_ID CUSTOMER_NAME        STATE           COUNTRY
----------- -------------------- --------------- ---------------
          1 Akash                Delhi           India
          2 Amit                 Hyderabad       India
          3 Jason                California      USA
```

Question 3 (Commit, Roll Back and

Save Point)Scenario-1

1. Create Person table with the ID, NAME and address.

   → create table person (ID int, NAME varchar(20), ADDRESS varchar(20));

2. Insert one row in the person table.

   →

   insert into person values (1, 'Roshan Raman Giri','Nepal');

3. Rollback the transaction and display the person table.

   → rollback;

```
SQL> create table person (ID int, NAME varchar(20), ADDRESS varchar(20));

Table created.

SQL> insert into person values (1, 'Roshan Raman Giri','Nepal');

1 row created.

SQL> rollback;

Rollback complete.

SQL> select * from person;

no rows selected
```

Scenario-2

1. Create Person table with the ID, NAME and address.

   → create table person (ID int, NAME varchar(20), ADDRESS varchar(20));

2. Insert one row in the person table.

   → insert into person values (1, 'Roshan Raman Giri','Nepal');

3. Create Salary table with field ID and salary.

   → create table salary (ID int, Salary int);

4. Rollback the Transaction and display the person table.

   → rollback;

```
SQL> create table person (ID int, NAME varchar(20), ADDRESS varchar(20));

Table created.

SQL> insert into person values (1, 'Roshan Raman Giri','Nepal');

1 row created.

SQL> create table salary (ID int, Salary int);

Table created.

SQL> rollback;

Rollback complete.

SQL> select * from person;

        ID NAME                 ADDRESS
---------- -------------------- --------------------
         1 Roshan Raman Giri    Nepal
```

Justify why the sceneario-1 result is different from the scenario-2.

In both cases before using rollback command table named person is created and only one row is added using insert command but in the scenario-2 another table is created before using rollback command. Scenario-1's resultis different from the result of scenario-2 because in the case of scenario-1 rollback is done after inserting a row and as we know rollback command undo one step, so insertion process is undone after running rollback commandin scenario-1 because of which there aren't any row available in the table. Whereas in the case of scenario-2 rollback is done after creating table salary, so creating table salary is undone after running rollback command andinsertion process is not affected because of which a table with a row is displayed in output.

Ajay Kumar Mandal
20BTRCO048
CSE – IOT

EXPERIMENT - 2: C Create the following: (a) Synonym sequences and Index and (b) Create alter and update views.

Question-1

Create public synonym for the student table and grant the select access rights to the teachers' users for the studenttable using public synonym. Display the student table details using teacher's login.

→ select * from Student;
→ create public synonym st for Student;
→ create user teachers identified by teachers;
→ grant create session to teachers;
→ grant select on st to teachers;
→ connect teachers;
→ password:teachers
→ select *from st;

```
SQL> select * from Student;

      ROLL NAME                 ADDRESS                      THEORY  PARCTICAL
---------- -------------------- ------------------------- ---------- ----------
         1 Ram                  Bangalore                         67         88
         2 Shyam                Mangalore                         55         64
         3 Sita                 Delhi                             90         92
         4 Mohan                Mumbai                            45         52
         5 Sweta                Kathmandu                         73         85

SQL> create public synonym st for Student;

Synonym created.

SQL> create user teachers identified by teachers;
create user teachers identified by teachers
            *
ERROR at line 1:
ORA-65096: invalid common user or role name


SQL> alter session set "_ORACLE_SCRIPT"=true;

Session altered.

SQL> create user teachers identified by teachers;

User created.

SQL> grant create session to teachers;

Grant succeeded.

SQL> grant select on st to teachers;

Grant succeeded.

SQL> connect teachers;
Enter password:
Connected.
SQL> select * from st;

      ROLL NAME                 ADDRESS                      THEORY  PARCTICAL
---------- -------------------- ------------------------- ---------- ----------
         1 Ram                  Bangalore                         67         88
         2 Shyam                Mangalore                         55         64
         3 Sita                 Delhi                             90         92
         4 Mohan                Mumbai                            45         52
         5 Sweta                Kathmandu                         73         85
```

Question-2

1.create a sequence named sequence_1. Sequence will start from 1 and will be incremented by 1 having maximum value

100. Sequence will repeat itself from start value after exceeding 100. create a table named students with columns as idand name.

- → create sequence sequence_1
- → start with 1
- → increment by 1
- → minvalue 0
- → maxvalue 100
- → cycle;

```
SQL> connect / as sysdba
Connected.
SQL> create sequence sequence_1
  2  start with 1
  3  increment by 1
  4  minvalue 0
  5  maxvalue 100
  6  cycle;

Sequence created.
```

CREATE TABLE

students(

ID number(10),

NAME

char(20)

);

Now insert ID values into table using sequences.

INSERT into students

VALUES(sequence_1.nextval,'Ramesh');INSERT

into students VALUES(sequence_1.nextval,'Suresh');

- → create table students(ID number(10), NAME char(20));
- → insert into students value(sequence_1.nextval,'Ramesh');
- → insert into students value(sequence_1.nextval,'Suresh');
- → select * from students;

```
SQL> create table students(ID number(10), NAME char(20));

Table created.

SQL> insert into students values(sequence_1.nextval,'Ramesh');

1 row created.

SQL> insert into students values(sequence_1.nextval,'Suresh');

1 row created.

SQL> select * from students;

        ID NAME
---------- --------------------
         1 Ramesh
         2 Suresh
```

Question-3

1. Create index on employee_name of the customer table.
2. Create a bit map index on the department field.
3. Drop the bitmap index-based index.

→ create index employee_name on customer(CUSTOMER_NAME);
→ create bitmap index department_field on customer(COUNTRY);
→ drop index department_field;

```
SQL> select *from customer;

CUSTOMER_ID CUSTOMER_NAME        STATE           COUNTRY
----------- -------------------- --------------- ---------------
          1 Akash                Delhi           India
          2 Amit                 Hyderabad       India
          3 Jason                California      USA

SQL> create index employee_name on customer(CUSTOMER_NAME);

Index created.

SQL> create bitmap index department_field on customer(COUNTRY);

Index created.

SQL> drop index department_field;

Index dropped.
```

Question-4

1. Create salesman table.

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	0.12

→ create table salesman(salesman_id int, name varchar(20), city varchar(20), commissionfloat(10));
→ insert into salesman values (5001,'James Hoog','New York',0.15);
→ insert into salesman values (5002,'Nail Knite','Paris',0.13);
→ insert into salesman values (5005,'Pit Alex','London',0.11);
→ insert into salesman values (5006,'Mc Lyon','Paris',0.14);
→ insert into salesman values (5007,'Paul Adam','Rome',0.13);
→ insert into salesman values (5003,'Lauson Hen','San Jose',0.12);
→ select * from salesman;

```
SQL> create table salesman(salesman_id int, name varchar(20), city varchar(20), commission float(10));

Table created.

SQL> insert into salesman values (5001,'James Hoog','New York',0.15);

1 row created.

SQL> insert into salesman values (5002,'Nail Knite','Paris',0.13);

1 row created.

SQL> insert into salesman values (5005,'Pit Alex','London',0.11);

1 row created.

SQL> insert into salesman values (5006,'Mc Lyon','Paris',0.14);

1 row created.

SQL> insert into salesman values (5007,'Paul Adam','Rome',0.13);

1 row created.

SQL> insert into salesman values (5003,'Lauson Hen','San Jose',0.12);

1 row created.

SQL> select * from salesman;

SALESMAN_ID NAME                 CITY                 COMMISSION
----------- -------------------- -------------------- ----------
       5001 James Hoog           New York                    .15
       5002 Nail Knite           Paris                       .13
       5005 Pit Alex             London                      .11
       5006 Mc Lyon              Paris                       .14
       5007 Paul Adam            Rome                        .13
       5003 Lauson Hen           San Jose                    .12

6 rows selected.
```

2. Write a query to create a view for all salesmen with columns salesman_id, name, and city.
3. Write a query to find the salesmen of the city New York who achieved the commission more than 13%.

→ create view all_salesman as select salesman_id, name, city from salesman;
→ select * from all_salesman;
→ create view n_salesman as select salesman_id, name, city, commission from salesman wherecity='New York' and commission>0.13;

→ select * from n_salesman;

```
SQL> create view all_salesman as select salesman_id, name, city from salesman;

View created.

SQL> select * from all_salesman;

SALESMAN_ID NAME                 CITY
----------- -------------------- --------------------
       5001 James Hoog           New York
       5002 Nail Knite           Paris
       5005 Pit Alex             London
       5006 Mc Lyon              Paris
       5007 Paul Adam            Rome
       5003 Lauson Hen           San Jose

6 rows selected.

SQL> create view n_salesman as select salesman_id, name, city, commission from salesman where city='New
York', commission>0.13;
create view n_salesman as select salesman_id, name, city, commission from salesman where city='New York'
, commission>0.13

*
ERROR at line 1:
ORA-00933: SQL command not properly ended


SQL> create view n_salesman as select salesman_id, name, city, commission from salesman where city='New
York' and commission>0.13;

View created.

SQL> select * from n_salesman;

SALESMAN_ID NAME                 CITY                COMMISSION
----------- -------------------- -------------------- ----------
       5001 James Hoog           New York                    .15
```

Ajay Kumar Mandal
20BTRCO048
CSE – IOT

EXPERIMENT - 3: Create PL/SQL program using cursors, control structure, exception handling

DESCRIPTION

A cursor is a pointer to this context area. PL/SQL controls the context area through a cursor.

A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the

cursor holds is referred to as the active set.

You can name a cursor so that it could be referred to in a program to fetch and process the

rows returned by the SQL statement, one at a time. There are two types of cursors −

1. Implicit cursors

2. Explicit cursors

Syntax for cursor

Declaring the Cursor

Declaring the cursor defines the cursor with a name and the associated SELECT statement.

For example −

```
CURSOR c_customers IS
SELECT id, name, address FROM customers;
```

Opening the Cursor

Opening the cursor allocates the memory for the cursor and makes it ready for fetching the

rows returned by the SQL statement into it. For example, we will open the above defined

cursor as follows −

```
OPEN c_customers;
```

Fetching the CursorFetching the cursor involves accessing one row at a time. For example, we

will fetch rows

from the above-opened cursor as follows −

```
FETCH c_customers INTO c_id, c_name, c_addr;
```

Closing the Cursor

Closing the cursor means releasing the allocated memory. For example, we will close the

above-opened cursor as follows −

CLOSE c_customers;

QUESTION 1

Create following table and compute gross salary ((Gross Salary=Basic+DA+HRA) for the employee table and print using cursor

EMP-ID	NAME	Basic	DA	HRA
1	Raja	10000	1000	3000
2	SELEVAM	30000	2000	4000
3	BABU	340000	1500	4000
4	GOWDA	340000	2000	3000

Commands

Creating table;

Create table items(e_id int,e_name char(10), e_basic int, e_da int,e_hra int);

insert into items values(1 ,'Raja', 10000, 1000 ,300);

insert into items values(2 ,'SELEVAM', 30000 ,2000 ,400);

insert into items values(3,' BABU ',340000, 1500, 4000);

insert into items values(4,' GOWDA', 340000 ,2000 ,3000);

```
declare
gross items.e_basic%type;
cursor compute is select e_basic+e_da+e_hra as gross_salary from items;
begin
open compute;
loop
fetch compute into gross ;
exit when compute%notfound;
dbms_output.put_line( gross);
end loop;
end;
/
```

Run SQL Command Line

```
SQL> select * from items;

      E_ID E_NAME        E_BASIC       E_DA      E_HRA
---------- ---------- ---------- ---------- ----------
         1 Raja            10000       1000        300
         2 SELEVAM         30000       2000        400
         3  BABU          340000       1500       4000
         4  GOWDA         340000       2000      30000

SQL> declare
  2
  3  gross items.e_basic%type;
  4
  5  cursor compute is select e_basic+e_da+e_hra as gross_salary from items;
  6
  7  begin
  8  open compute;
  9
 10  loop
 11
 12  fetch compute into gross ;
 13
 14   exit when compute%notfound;
 15  dbms_output.put_line( gross);
 16  end loop;
 17  end;
 18  /

PL/SQL procedure successfully completed.

SQL> set serveroutput on;
SQL> /
11300
32400
345500
372000

PL/SQL procedure successfully completed.

SQL>
```

EXCEPTION HANDLING

An exception is an error condition during a program execution. PL/SQL supports programmers to catch such conditions using EXCEPTION block in the program and an appropriate action is taken against the error condition. There are two types of exceptions −

1. System-defined exceptions

2. User-defined exceptions

Syntax for Exception Handling

The general syntax for exception handling is as follows. Here you can list down as many exceptions as you can handle. The default exception will be handled using WHEN others THEN –

DECLARE

<declarations section>

BEGIN

<executable command(s)>

EXCEPTION

<exception handling goes here >

WHEN exception1 THEN

exception1-handling-statements WHEN exception2 THEN

exception2-handling-statements

WHEN exception3 THEN

exception3-handling-statements

WHEN others THEN

exception3-handling-statements

END;

Question-2

Consider the employee management system has the following information 1. employee id, 2. employee name, 3. salary and 4. Type of employee. Create employee table and store the above information using PLSQL program.

Handle appropriate exceptions for the following:

i. Generate exception when employee salary is negative.

ii. Generate the exception when employee type is other than full time or part time

COMMANDS

CREATING TABLE

create table emanagement (e_id int, e_name char(10),e_salary int, e_type char(20));

OPERATION COMMANDS

declare

id emanagement.e_id%type:=&enterid;

name emanagement.e_name%type:='&entername';

salary emanagement.e_salary%type:=&enteramt;

emptype emanagement.e_type%type:='&entertypeojob';

negative_salary exception;

employeetype exception ;

begin

if (salary <0)

then

raise negative_salary;

elsif (emptype<>’fulltime ' and emptype<>'parttime')

```
then
raise employeetype;
else
insert into emanagement  values(id,name,salary,emptype);
end if;
exception
when negative_salary then
dbms_output.put_line('salary is negative');

when employeetype then
dbms_output.put_line('employee type ');
when others then
dbms_output.put_line('other error');

end;
/
```

EXECUTION FOR NEGATIVE SALARY

Run SQL Command Line

```
  6  negative_salary exception;
  7  employeetype exception ;
  8  begin
  9  if (salary <0)
 10  then
 11  raise negative_salary;
 12  elsif((emptype!='fulltime')and(emptype!='parttime'))
 13  then
 14  raise employeetype;
 15  else
 16  insert into emanagement  values(id,name,salary,emptype);
 17  end if;
 18
 19  exception
 20  when negative_salary then
 21  dbms_output.put_line('salary is negative');
 22
 23  when employeetype then
 24  dbms_output.put_line('employee type ');
 25  when others then
 26  dbms_output.put_line('other error');
 27
 28  end;
 29  /
Enter value for enterid: 1
old   2: id emanagement.e_id%type:=&enterid;
new   2: id emanagement.e_id%type:=1;
Enter value for entername: a
old   3: name emanagement.e_name%type:='&entername';
new   3: name emanagement.e_name%type:='a';
Enter value for enteramt: -12332
old   4: salary emanagement.e_salary%type:=&enteramt;
new   4: salary emanagement.e_salary%type:=-12332;
Enter value for entertypeojob: fulltime
old   5: emptype emanagement.e_type%type:='&entertypeojob';
new   5: emptype emanagement.e_type%type:='fulltime';
salary is negative

PL/SQL procedure successfully completed.
```

EXECUTION OF EMPLOYEE TYPE EXCEPTION

Run SQL Command Line

```
 6  negative_salary exception;
 7  employeetype exception ;
 8  begin
 9  if (salary <0)
10  then
11  raise negative_salary;
12  elsif((emptype!='fulltime')and(emptype!='parttime'))
13  then
14  raise employeetype;
15  else
16  insert into emanagement  values(id,name,salary,emptype);
17  end if;
18
19  exception
20  when negative_salary then
21  dbms_output.put_line('salary is negative');
22
23  when employeetype then
24  dbms_output.put_line('employee type ');
25  when others then
26  dbms_output.put_line('other error');
27
28  end;
29  /
Enter value for enterid: 1
old   2: id emanagement.e_id%type:=&enterid;
new   2: id emanagement.e_id%type:=1;
Enter value for entername: a
old   3: name emanagement.e_name%type:='&entername';
new   3: name emanagement.e_name%type:='a';
Enter value for enteramt: 2332
old   4: salary emanagement.e_salary%type:=&enteramt;
new   4: salary emanagement.e_salary%type:=2332;
Enter value for entertypeojob: intern
old   5: emptype emanagement.e_type%type:='&entertypeojob';
new   5: emptype emanagement.e_type%type:='intern';
employee type

PL/SQL procedure successfully completed.
```

Executing with correct details

Run SQL Command Line

```
 14  raise employeetype;
 15  else
 16  insert into emanagement  values(id,name,salary,emptype);
 17  end if;
 18
 19  exception
 20  when negative_salary then
 21  dbms_output.put_line('salary is negative');
 22
 23  when employeetype then
 24  dbms_output.put_line('employee type ');
 25  when others then
 26  dbms_output.put_line('other error');
 27
 28  end;
 29  /
Enter value for enterid: 1
old   2: id emanagement.e_id%type:=&enterid;
new   2: id emanagement.e_id%type:=1;
Enter value for entername: raju
old   3: name emanagement.e_name%type:='&entername';
new   3: name emanagement.e_name%type:='raju';
Enter value for enteramt: 10000
old   4: salary emanagement.e_salary%type:=&enteramt;
new   4: salary emanagement.e_salary%type:=10000;
Enter value for entertypeojob: fulltime
old   5: emptype emanagement.e_type%type:='&entertypeojob';
new   5: emptype emanagement.e_type%type:='fulltime';

PL/SQL procedure successfully completed.

SQL> select * from emanagement;

      E_ID E_NAME       E_SALARY E_TYPE
---------- ---------- ---------- --------------------
         1 raju            10000 fulltime

SQL>
SQL>
SQL>
SQL>
```

Ajay Kumar Mandal
20BTRCO048
CSE – IOT

EXPERIMENT - 4: Create following: (a) Simple Triggers and (b) Package using procedures and functions

Question-1

Create following employee table and perform the following task.

EMP-ID	NAME	Basic	DA	HRA
1	Raja	10000	1000	3000
2	SELEVAM	30000	2000	4000

→ create table emp1(ID int, NAME varchar(10), BASIC number(10), DA number (10), HRA number(10));
→ insert into emp1 values (1,'Raja', 10000, 1000, 3000);
→ insert into emp1 values(2,'SELEVAM', 30000, 2000, 4000);

```
SQL> create table emp1(ID int, NAME varchar(10), BASIC number(10), DA number(10), HRA number(10));

Table created.

SQL> insert into emp1 values (1,'Raja',10000,1000,3000);

1 row created.

SQL> insert into emp1 values (2,'SELEVAM',30000,2000,4000);

1 row created.

SQL> select * from emp1;

        ID NAME            BASIC         DA        HRA
---------- ---------- ---------- ---------- ----------
         1 Raja            10000       1000       3000
         2 SELEVAM         30000       2000       4000
```

a.Create add_employe procedure which will add one employe details in the employee table.

→ create or replace procedure add_employee(e_id emp1.id%type, e_name emp1.name%type, e_basicemp1.basic%type, e_da emp1.da%type, e_hra emp1.hra%type)

→ as
→ begin
→ insert into emp1(id, name, basic, da, hra) values (e_id, e_name, e_basic, e_da, e_hra);
→ end add_employee;
→ /
→ execute add_employee(3,'Roshan',40000,3000,5000);
→ select * from emp1;

```
SQL> create or replace procedure add_employee(e_id emp1.id%type, e_name emp1.name%type, e_basic emp1.bas
ic%type, e_da emp1.da%type, e_hra emp1.hra%type)
  2  as
  3  begin
  4  insert into emp1(id, name, basic, da, hra) values (e_id, e_name, e_basic, e_da, e_hra);
  5  end add_employee;
  6  /

Procedure created.

SQL> execute add_employee(3,'Roshan',40000,3000,5000);

PL/SQL procedure successfully completed.

SQL> select * from emp1;

        ID NAME            BASIC         DA        HRA
---------- ---------- ---------- ---------- ----------
         1 Raja            10000       1000       3000
         2 SELEVAM         30000       2000       4000
         3 Roshan          40000       3000       5000
```

b.Create delete_employe procedure which will delete one employe details the from the employee table.

→ create or replace procedure del_employee(e_id emp1.id%type)
→ is
→ namenotfoundexp EXCEPTION;
→ begin
→ delete from emp1 where id=e_id;
→ if sql%found then
→ dbms_output.put_line(sql%rowcount || 'Employee record(s) got deleted.');
→ else
→ raise namenotfoundexp;
→ end if;
→ EXCEPTION
→ when namenotfoundexp then
→ dbms_output.put_line('No Records Found.');
→ end del_employee;
→ /
→ execure del_employee(3);
→ select * from emp1;

```
SQL> create or replace procedure del_employee(e_id emp1.id%type)
  2  is
  3  namenotfoundexp EXCEPTION;
  4  begin
  5  delete from emp1 where id=e_id;
  6  if sql%found then
  7  dbms_output.put_line(sql%rowcount || 'Employee record(s) got deleted.');
  8  else
  9  raise namenotfoundexp;
 10  end if;
 11  EXCEPTION
 12  when namenotfoundexp then
 13  dbms_output.put_line('No Records Found.');
 14  end del_employee;
 15  /

Procedure created.

SQL> execute del_employee(3);
1Employee record(s) got deleted.

PL/SQL procedure successfully completed.

SQL> select * from emp1;

        ID NAME           BASIC         DA        HRA
---------- ---------- ---------- ---------- ----------
         1 Raja           10000       1000       3000
         2 SELEVAM        30000       2000       4000
```

c.Create display_employee procedure which will display employee details.

→ create or replace procedure display_employee(e_id emp1.ID%type) is
→ cursor c_emp is select * from emp1;
→ r_emp c_emp%rowtype;
→ begin
→ open c_emp;
→ loop
→ fetch c_emp into r_emp;
→ exit when c_emp%notfound;
→ if r_emp.id=e_id then
    → dbms_output.put_line(r_emp.id ||' '||r_emp.name||' has basic='||r_emp.basic||' DA='||r_emp.da||' andHRA='||r_emp.hra);
→ end if;
→ end loop;
→ close c_emp;
→ end display_employee;
→ /
→ execute display_employee(1)
→ execute display_employee(2)

```
SQL> create or replace procedure display_employee(e_id emp1.ID%type) is
  2  cursor c_emp is select * from emp1;
  3  r_emp c_emp%rowtype;
  4  begin
  5  open c_emp;
  6  loop
  7  fetch c_emp into r_emp;
  8  exit when c_emp%notfound;
  9  if r_emp.id=e_id then
 10  dbms_output.put_line(r_emp.id ||'  '||r_emp.name||' has basic='||r_emp.basic||' DA='||r_emp.da||' a
nd HRA='||r_emp.hra);
 11  end if;
 12  end loop;
 13  close c_emp;
 14  end display_employee;
 15  /

Procedure created.

SQL> execute display_employee(1);
1  Raja has basic=10000 DA=1000 and HRA=3000

PL/SQL procedure successfully completed.

SQL> execute display_employee(2);
2  SELEVAM has basic=30000 DA=2000 and HRA=4000

PL/SQL procedure successfully completed.
```

d. Create compute_salary function which will compute the gross salary for the given employee id.

→ create or replace function compute_salary(e_id emp1.id%type) return number is
→ Gross_Salary number(10);
→ cursor c_emp is select * from emp1 where id=e_id;
→ r_emp c_emp%rowtype;
→ begin
→ open c_emp;
→ fetch c_emp into r_emp;
→ Gross_Salary:= r_emp.basic + r_emp.da + r_emp.hra;
→ close c_emp;
→ return Gross_Salary;
→ end compute_salary;
→ /

→ begin
→ dbms_output.put_line('Gross Salary is '||compute_salary(1));
→ end;
→ /

→ begin
→ dbms_output.put_line('Gross Salary is '||compute_salary(2));
→ end;
→ /

```
SQL>  create or replace function compute_salary(e_id emp1.id%type) return number is
  2  Gross_Salary number(10);
  3  cursor c_emp is select * from emp1 where id=e_id;
  4  r_emp c_emp%rowtype;
  5  begin
  6  open c_emp;
  7  fetch c_emp into r_emp;
  8  Gross_Salary:= r_emp.basic + r_emp.da + r_emp.hra;
  9  close c_emp;
 10  return Gross_Salary;
 11  end compute_salary;
 12  /

Function created.

SQL> begin
  2  dbms_output.put_line('Gross Salary is '||compute_salary(1));
  3  end;
  4  /
Gross Salary is 14000

PL/SQL procedure successfully completed.

SQL>
SQL> begin
  2  dbms_output.put_line('Gross Salary is '||compute_salary(2));
  3  end;
  4  /
Gross Salary is 36000

PL/SQL procedure successfully completed.
```

e. Create salary_trigger which will trigger and inform the users whenever you're trying to insert basic salaryis negative.

→ create or replace trigger salary_triggers
→ before insert on emp1 for each row
→ begin
→ if :new.basic<0 or :new.da<0 or :new.hra<0 then
→ dbms_output.put_line('BASIC cannot be Negative');
→ end if;
→ end;
→ /
→ insert into emp1 values(5,'Jewan',40000, 3000, 2000);
→ insert into emp1 values (6,'Gyan',-50000,4000,3000);

```
SQL> create or replace trigger salary_triggers
  2  before insert on emp1 for each row
  3  begin
  4  if :new.basic<0 or :new.da<0 or :new.hra<0 then
  5  dbms_output.put_line('BASIC cannot be Negative');
  6  end if;
  7  end;
  8  /

Trigger created.

SQL> insert into emp1 values(5,'Jewan',40000, 3000, 2000);

1 row created.

SQL> insert into emp1 values (6,'Gyan',-50000,4000,3000);
BASIC cannot be Negative
```

f.Create a package emp and include all procedures in the emp package and invoke the procedure usingpackage.

→ create or replace package p_emp as
→ procedure add_employee(e_id emp1.id%type, e_name emp1.name%type, e_basic emp1.basic%type,e_da emp1.da%type, e_hra emp1.hra%type);
→ procedure del_employee(e_id emp1.id%type);
→ procedure display_employee(e_id emp1.ID%type);
→ end p_emp;
→ /

→ create or replace package body p_emp as
→ procedure add_employee(e_id emp1.id%type, e_name emp1.name%type, e_basic emp1.basic%type,e_da emp1.da%type, e_hra emp1.hra%type) as
→ begin
→ insert into emp1(id, name, basic, da, hra) values (e_id, e_name, e_basic, e_da, e_hra);
→ end add_employee;
→ procedure del_employee(e_id emp1.id%type) is
→ namenotfoundexp EXCEPTION;
→ begin
→ delete from emp1 where id=e_id;
→ if sql%found then
→ dbms_output.put_line(sql%rowcount || 'Employee record(s) got deleted.');
→ else
→ raise namenotfoundexp;
→ end if;
→ EXCEPTION
→ when namenotfoundexp then
→ dbms_output.put_line('No Records Found.');
→ end del_employee;
→ procedure display_employee(e_id emp1.ID%type) is
→ cursor c_emp is select * from emp1;
→ r_emp c_emp%rowtype;
→ begin
→ open c_emp;

→ loop
→ fetch c_emp into r_emp;
→ exit when c_emp%notfound;
→ if r_emp.id=e_id then
    → dbms_output.put_line(r_emp.id ||' '||r_emp.name||' has basic='||r_emp.basic||' DA='||r_emp.da||' andHRA='||r_emp.hra);
→ end if;
→ end loop;
→ close c_emp;
→ end display_employee;
→ end p_emp;
→ /
→ begin
→ p_emp.add_employee(7,'Ishwor',30000,2000,1000);
→ p_emp.del_employee(6);
→ p_emp.display_employee(7);
→ end;
→ /

```
SQL> create or replace package p_emp as
  2  procedure add_employee(e_id emp1.id%type, e_name emp1.name%type, e_basic emp1.basic%type, e_da emp1
.da%type, e_hra emp1.hra%type);
  3  procedure del_employee(e_id emp1.id%type);
  4  procedure display_employee(e_id emp1.ID%type);
  5  end p_emp;
  6  /

Package created.

SQL> create or replace package body p_emp as
  2  procedure add_employee(e_id emp1.id%type, e_name emp1.name%type, e_basic emp1.basic%type, e_da emp1
.da%type, e_hra emp1.hra%type) as
  3  begin
  4  insert into emp1(id, name, basic, da, hra) values (e_id, e_name, e_basic, e_da, e_hra);
  5  end add_employee;
  6  procedure del_employee(e_id emp1.id%type) is
  7  namenotfoundexp EXCEPTION;
  8  begin
  9  delete from emp1 where id=e_id;
 10  if sql%found then
 11  dbms_output.put_line(sql%rowcount || 'Employee record(s) got deleted.');
 12  else
 13  raise namenotfoundexp;
 14  end if;
 15  EXCEPTION
 16  when namenotfoundexp then
 17  dbms_output.put_line('No Records Found.');
 18  end del_employee;
 19  procedure display_employee(e_id emp1.ID%type) is
 20  cursor c_emp is select * from emp1;
 21  r_emp c_emp%rowtype;
 22  begin
 23  open c_emp;
 24  loop
 25  fetch c_emp into r_emp;
 26  exit when c_emp%notfound;
 27  if r_emp.id=e_id then
 28  dbms_output.put_line(r_emp.id ||'  '||r_emp.name||' has basic='||r_emp.basic||' DA='||r_emp.da||' a
nd HRA='||r_emp.hra);
 29  end if;
 30  end loop;
 31  close c_emp;
 32  end display_employee;
 33  end p_emp;
 34  /

Package body created.
```

```
SQL> begin
  2  p_emp.add_employee(7,'Ishwor',30000,2000,1000);
  3  p_emp.del_employee(6);
  4  p_emp.display_employee(7);
  5  end;
  6  /
1Employee record(s) got deleted.
7  Ishwor has basic=30000 DA=2000 and HRA=1000

PL/SQL procedure successfully completed.

SQL> select * from emp1;

        ID NAME           BASIC         DA        HRA
---------- ---------- ---------- ---------- ----------
         1 Raja           10000       1000       3000
         2 SELEVAM        30000       2000       4000
         5 Jewan          40000       3000       2000
         7 Ishwor         30000       2000       1000
```

Ajay Kumar Mandal
20BTRCO048
CSE – IOT

EXPERIMENT - 5 : Create the table for (a) STUDENT database and Insert five records for each attribute and (b) COMPANY database there as 6_7

5 a:

Create the table for (a) STUDENT database consisting 3 tables and insert five records for each attribute.

Table 1:

```
SQL> create table students_name(USN int,NAME varchar(25));

Table created.
```

```
SQL> insert into students_name values('001','Raju');

1 row created.

SQL> insert into students_name values('002','Rakesh');

1 row created.

SQL> insert into students_name values('003','Ramesh');

1 row created.

SQL> insert into students_name values('004','Rudhresh');

1 row created.

SQL> insert into students_name values('005','Rahul');

1 row created.

SQL> select * from students_name;

       USN NAME
---------- -------------------------
         1 Raju
         2 Rakesh
         3 Ramesh
         4 Rudhresh
         5 Rahul
```

Table 2 :

```
SQL> create table students_marks(USN int,total_marks int,percentage int);

Table created.
```

```
SQL> insert into students_marks values('001','890','60');

1 row created.

SQL> insert into students_marks values('002','890','60');

1 row created.

SQL> insert into students_marks values('003','950','60');

1 row created.

SQL> insert into students_marks values('004','800','75');

1 row created.

SQL> insert into students_marks values('005','750','85');

1 row created.

SQL> select * from students_marks;

       USN TOTAL_MARKS PERCENTAGE
---------- ----------- ----------
         1         890         60
         2         890         60
         3         950         60
         4         800         75
         5         750         85
```

Table 3:

```
SQL> create table students_dept(USN int,NAME varchar(25),DEPARTMENT varchar(25));

Table created.

SQL> insert into students_dept values('001','Arvindh','CSE');

1 row created.

SQL> insert into students_dept values('002','Anurag','EC');

1 row created.

SQL> insert into students_dept values('003','Divya','EEE');

1 row created.

SQL> insert into students_dept values('004','Manoj','CSE');

1 row created.

SQL> insert into students_dept values('005','Prashanth','CSE');

1 row created.

SQL> select * from students_dept;

       USN NAME                      DEPARTMENT
---------- ------------------------- -------------------------
         1 Arvindh                   CSE
         2 Anurag                    EC
         3 Divya                     EEE
         4 Manoj                     CSE
         5 Prashanth                 CSE
```

(b) COMPANY database consist of 6 tables and 10 records for each attribute.

Table 1:

```
SQL> create table company_salaries(emp_no int,name varchar(25),salary int);

Table created.

SQL> insert into company_salaries values('001','Divya','25000');

1 row created.

SQL> insert into company_salaries values('002','Dhanush','35000');

1 row created.

SQL> insert into company_salaries values('003','vikas','35000');

1 row created.

SQL> insert into company_salaries values('004','arvind','45000');

1 row created.

SQL> insert into company_salaries values('005','shubha','50000');

1 row created.

SQL> insert into company_salaries values('006','santhosh','50000');

1 row created.

SQL> insert into company_salaries values('007','sanath','50000');

1 row created.

SQL> insert into company_salaries values('008','ragu','50000');

1 row created.

SQL> insert into company_salaries values('009','shamanth','25000');

1 row created.

SQL> insert into company_salaries values('010','prashanth','25000');

1 row created.
```

```
SQL> select * from company_salaries;

    EMP_NO NAME                           SALARY
---------- ------------------------- ----------
         1 Divya                          25000
         2 Dhanush                        35000
         3 vikas                          35000
         4 arvind                         45000
         5 shubha                         50000
         6 santhosh                       50000
         7 sanath                         50000
         8 ragu                           50000
         9 shamanth                       25000
        10 prashanth                      25000
```

Table 2:

```
SQL> create table company_employee(emp_no int,birth_date DATE,name varchar(25),gender varchar(25));

Table created.
```

```
SQL> insert into company_employee values('001','23-JUN-2000','prashanth','M');

1 row created.

SQL> insert into company_employee values('002','01-JUN-1987','pramav','M');

1 row created.

SQL> insert into company_employee values('003','15-MAR-1987','rajesh','M');

1 row created.

SQL> insert into company_employee values('004','15-JAN-1987','ragu','M');

1 row created.

SQL> insert into company_employee values('005','15-FEB-1988','ragu','M');

1 row created.

SQL> insert into company_employee values('006','15-AUG-1950','vaishnavi','F');

1 row created.

SQL> insert into company_employee values('007','25-AUG-1990','sneha','F');

1 row created.

SQL> insert into company_employee values('008','30-JAN-1990','rukhmini','F');

1 row created.

SQL> insert into company_employee values('009','30-MAR-1990','shine shetty','M');

1 row created.

SQL> insert into company_employee values('010','30-SEP-1990','vasuki','M');

1 row created.

SQL> select * from company_employee;

    EMP_NO BIRTH_DAT NAME                      GENDER
---------- --------- ------------------------ ------------------------
         1 23-JUN-00 prashanth                M
         2 01-JUN-87 pramav                   M
         3 15-MAR-87 rajesh                   M
         4 15-JAN-87 ragu                     M
         5 15-FEB-88 ragu                     M
         6 15-AUG-50 vaishnavi                F
         7 25-AUG-90 sneha                    F
         8 30-JAN-90 rukhmini                 F
         9 30-MAR-90 shine shetty             M
        10 30-SEP-90 vasuki                   M
```

Table 3:

```
SQL> create table company_coustomers(C_id int,name varchar(25),city varchar(10));

Table created.

SQL> insert into  company_coustomers values('16854','akash','bengaluru');

1 row created.

SQL> insert into  company_coustomers values('16852','nagaraj','bengaluru');

1 row created.

SQL> insert into  company_coustomers values('16822','nagaveni','bengaluru');

1 row created.

SQL> insert into  company_coustomers values('16832','manoj','tumakuru');

1 row created.
```

```
SQL> insert into  company_coustomers values('16833','rajesh','tumakuru');

1 row created.
```

```
SQL> insert into  company_coustomers values('16890','rudhresh','bengaulru');

1 row created.

SQL> insert into  company_coustomers values('16899','ragu','bengaulru');

1 row created.

SQL> insert into  company_coustomers values('16799','aarush','bengaulru');

1 row created.

SQL> insert into  company_coustomers values('16719','ayush','bengaulru');

1 row created.

SQL> insert into  company_coustomers values('16711','prashanth','bengaulru');

1 row created.

SQL> select * from company_coustomers;

      C_ID NAME                      CITY
---------- ------------------------- ----------
     16854 akash                     bengaluru
     16852 nagaraj                   bengaluru
     16822 nagaveni                  bengaluru
     16832 manoj                     tumakuru
     16833 rajesh                    tumakuru
     16890 rudhresh                  bengaulru
     16899 ragu                      bengaulru
     16799 aarush                    bengaulru
     16719 ayush                     bengaulru
     16711 prashanth                 bengaulru
```

Table 4:

```
SQL> create table company_products(p_id int,p_name varchar(25),category_id int);

Table created.

SQL> insert into  company_products values('16719','Shirt','1150');

1 row created.

SQL> insert into  company_products values('16720','Saree','1250');

1 row created.

SQL> insert into  company_products values('16320','t-Shirt','1230');

1 row created.

SQL> insert into  company_products values('16340','jeans','1240');

1 row created.

SQL> insert into  company_products values('16342','shorts','1220');

1 row created.

SQL> insert into  company_products values('16352','shoes','1228');

1 row created.

SQL> insert into  company_products values('16359','tie','1558');

1 row created.

SQL> insert into  company_products values('11552','slippers','1158');

1 row created.

SQL> insert into  company_products values('14552','blazzers','1128');

1 row created.

SQL> insert into  company_products values('14122','formal shirt','1133');

1 row created.

SQL> select * from company_products;

      P_ID P_NAME                    CATEGORY_ID
---------- ------------------------- -----------
     16719 Shirt                            1150
     16720 Saree                            1250
     16320 t-Shirt                          1230
     16340 jeans                            1240
     16342 shorts                           1220
     16352 shoes                            1228
     16359 tie                              1558
     11552 slippers                         1158
     14552 blazzers                         1128
     14122 formal shirt                     1133

10 rows selected.
```

Table 5 :

```
SQL> create table company_departments(dept_id int,dept_name varchar(25));

Table created.
```

```
SQL> insert into  company_departments values('1411','front_end');

1 row created.

SQL> insert into  company_departments values('1412','back_end');

1 row created.

SQL> insert into  company_departments values('1413','production');

1 row created.

SQL> insert into  company_departments values('1414','testing_1');

1 row created.

SQL> insert into  company_departments values('1415','testing_2');

1 row created.

SQL> insert into  company_departments values('1416','database_management');

1 row created.

SQL> insert into  company_departments values('1417','database_security');

1 row created.

SQL> insert into  company_departments values('1418','cyber_security');

1 row created.

SQL> insert into  company_departments values('1419','web_design');

1 row created.

SQL> insert into  company_departments values('1420','web_dev');

1 row created.

SQL> select * from company_departments;

   DEPT_ID DEPT_NAME
---------- ------------------------
      1411 front_end
      1412 back_end
      1413 production
      1414 testing_1
      1415 testing_2
      1416 database_management
      1417 database_security
      1418 cyber_security
      1419 web_design
      1420 web_dev
```

Table 6 :

```
SQL> create table company_sales_man(id int,name varchar(25),commisson int);

Table created.

SQL> insert into company_sales_man values(30001,'daniel',1);

1 row created.

SQL> insert into company_sales_man values(30002,'oliver',2);

1 row created.

SQL> insert into company_sales_man values(30003,'william',3);

1 row created.

SQL> insert into company_sales_man values(30004,'james',2);

1 row created.

SQL> insert into company_sales_man values(30005,'Matt',3);

1 row created.

SQL> insert into company_sales_man values(30006,'thomas',2);

1 row created.

SQL> insert into company_sales_man values(30007,'steve',1);

1 row created.

SQL> insert into company_sales_man values(30008,'castle',3);

1 row created.

SQL> insert into company_sales_man values(30009,'frank',2);

1 row created.

SQL> insert into company_sales_man values(30010,'peter',1);

1 row created.
```

```
SQL> select * from company_sales_man;

        ID NAME                      COMMISSON
---------- ------------------------ ----------
     30001 daniel                            1
     30002 oliver                            2
     30003 william                           3
     30004 james                             2
     30005 Matt                              3
     30006 thomas                            2
     30007 steve                             1
     30008 castle                            3
     30009 frank                             2
     30010 peter                             1

10 rows selected.
```

Ajay Kumar Mandal
20BTRCO048
CSE – IOT

EXPERIMENT - 6 : Illustrate the use of SELECT statement

EXPERIMENT - 7 : Conditional retrieval - WHERE clause

Create a new employee table (fields and records) based on requirements.REFERENCE TABLE:

JOB_ID	COMMISSION_PCT	EMP_ID	EMP_FNAME	EMP_LNAME	CLIENT	EMP_GENDER	BRANCH	YOB	EMP_SALARY	DEPT_ID
SA_REP	0.5	1	Jim	Cooper	Raaju	Male	3	1965	10000	100
IT_PROG	0.2	2	Michael	Hicks	Aditi	Male	1	1967	5000	102
SH_CLERK	0.1	3	Johnny	Rojas	Amirish	Male	2	1969	3000	101
FI_ACCOUNT	0.2	4	Adriana	Sosa	Vijay	Female	5	1965	20000	100
AD_ASST	0.3	5	Alissa	Cooper	Jayadev	Female	2	1969	22000	102
PR_REP	0.5	6	Selena	May	Hardeep	Female	4	1972	4000	101
SA_MAN	0.2	7	David	Lorentz	Chandni	Male	2	1970	30000	100
SA_MAN	0.2	7	David	Lorentz	Chandni	Male	2	1970	30000	100
AD_PRES	0.1	9	James	Connor	Saahil	Male	5	1973	14000	101
SA_MAN	0.4	10	Katy	Sosa	Dhruv	Female	2	1974	12000	100

Write queries for COMPANY DATABASE:

1. Find all employees

QUERY:

```
SQL> SELECT EMP_FNAME FROM COMPANY;

EMP_FNAME
-------------------------
Jim
Michael
Johnny
Adriana
Alissa
Selena
David
Stalia
James
Katy

10 rows selected.
```

2. Find all clients

QUERY:

```
SQL> SELECT CLIENT from COMPANY;

CLIENT
-------------------------
Raaju
Aditi
Amirish
Vijay
Jayadev
Hardeep
Chandni
Sasha
Saahil
Dhruv

10 rows selected.
```

3. Find the first and last names of all employees

QUERY:

```
SQL> SELECT EMP_FNAME, EMP_LNAME FROM COMPANY;

EMP_FNAME                 EMP_LNAME
------------------------- -------------------------
Jim                       Cooper
Michael                   Hicks
Johnny                    Rojas
Adriana                   Sosa
Alissa                    Cooper
Selena                    May
David                     Lorentz
Stalia                    Coleman
James                     Connor
Katy                      Sosa

10 rows selected.
```

4. Find the first and last names of all employees

QUERY:

```
SQL> SELECT EMP_ID, EMP_FNAME, EMP_GENDER FROM COMPANY WHERE EMP_GENDER='Male';

    EMP_ID EMP_FNAME                EMP_GENDER
---------- ------------------------ ------------------------
         1 Jim                      Male
         2 Michael                  Male
         3 Johnny                   Male
         7 David                    Male
         9 James                    Male
```

5. Find the forename and surnames names of all employees

QUERY:

```
SQL> SELECT EMP_ID, EMP_FNAME, EMP_GENDER FROM COMPANY WHERE EMP_GENDER='Male';

    EMP_ID EMP_FNAME                EMP_GENDER
---------- ------------------------ ------------------------
         1 Jim                      Male
         2 Michael                  Male
         3 Johnny                   Male
         7 David                    Male
         9 James                    Male
```

6. Find all employees at branch 2

QUERY:

```
SQL> SELECT EMP_ID,EMP_FNAME,EMP_LNAME,BRANCH FROM COMPANY WHERE BRANCH=2;

    EMP_ID EMP_FNAME                EMP_LNAME                      BRANCH
---------- ------------------------ ------------------------ ----------
         3 Johnny                   Rojas                             2
         5 Alissa                   Cooper                            2
         7 David                    Lorentz                           2
        10 Katy                     Sosa                              2
```

7. Find all employee's id's and names who were born after 1969 SELECT emp_id, first_name, last_name

QUERY:

```
SQL> SELECT EMP_ID,EMP_FNAME,EMP_LNAME,YOB FROM COMPANY WHERE YOB>1969;

    EMP_ID EMP_FNAME                EMP_LNAME                       YOB
---------- ------------------------ ------------------------ ----------
         6 Selena                   May                            1972
         7 David                    Lorentz                        1970
         8 Stalia                   Coleman                        1975
         9 James                    Connor                         1973
        10 Katy                     Sosa                           1974
```

8. Find all employees who are female & born after 1969 or who make over80000

QUERY:

```
SQL> SELECT EMP_ID,EMP_FNAME,YOB,EMP_SALARY FROM COMPANY WHERE EMP_GENDER='Female' AND YOB>1969 OR EMP_SALARY>80000;

    EMP_ID EMP_FNAME                YOB EMP_SALARY
---------- ------------------------ ---------- ----------
         6 Selena                   1972       4000
         8 Stalia                   1975      10000
        10 Katy                     1974      12000
```

9. Find all employees born between 1970 and 1975

QUERY:

```
SQL> SELECT EMP_ID,EMP_FNAME,YOB FROM COMPANY WHERE YOB>1970 AND YOB<1975;

    EMP_ID EMP_FNAME                       YOB
---------- ------------------------ ----------
         6 Selena                         1972
         9 James                          1973
        10 Katy                           1974
```

10. Find all employees named Jim, Michael, Johnny or David

QUERY:

```
SQL> SELECT EMP_ID, EMP_FNAME FROM COMPANY WHERE EMP_FNAME='Jim' OR EMP_FNAME='Michael' OR EMP_FNAME='Johnny' OR EMP_FNAME='David';

    EMP_ID EMP_FNAME
---------- ------------------------
         1 Jim
         2 Michael
         3 Johnny
         7 David
```

11. Find the number of employees

QUERY:

```
SQL> SELECT COUNT(*) FROM COMPANY;

  COUNT(*)
----------
        10
```

12. Find the average of all employee's salaries

QUERY:

```
SQL> SELECT AVG(EMP_SALARY) FROM COMPANY;

AVG(EMP_SALARY)
---------------
          13000
```

13. Find the sum of all employee's salaries

QUERY:

```
SQL> SELECT SUM(EMP_SALARY) FROM COMPANY;

SUM(EMP_SALARY)
---------------
         130000
```

14. Display the employee_id, first_name, last_name, department_id of employees whose departmet_id=100

QUERY:

```
SQL> SELECT EMP_ID,EMP_FNAME,EMP_LNAME,DEPT_ID FROM COMPANY WHERE DEPT_ID=100;

    EMP_ID EMP_FNAME                EMP_LNAME                     DEPT_ID
---------- ------------------------ ------------------------ ----------
         1 Jim                      Cooper                          100
         4 Adriana                  Sosa                            100
         7 David                    Lorentz                         100
        10 Katy                     Sosa                            100
```

15. Displays the employee_id, job_id, salary of employees whose last_name='Lorentz'.

QUERY:

```
SQL> SELECT EMP_ID,EMP_FNAME,JOB_ID,EMP_SALARY FROM COMPANY WHERE EMP_LNAME='Lorentz';

    EMP_ID EMP_FNAME                JOB_ID                   EMP_SALARY
---------- ------------------------ ------------------------ ----------
         7 David                    SA_MAN                        30000
```

16. Displays the employee_id, first_name, last_name and salary of employees whose salary is greater than or equal to 4000.

QUERY:

```
SQL> SELECT EMP_ID,EMP_FNAME,EMP_LNAME,EMP_SALARY FROM COMPANY WHERE EMP_SALARY>=4000;

    EMP_ID EMP_FNAME                EMP_LNAME                EMP_SALARY
---------- ------------------------ ------------------------ ----------
         1 Jim                      Cooper                        10000
         2 Michael                  Hicks                          5000
         4 Adriana                  Sosa                          20000
         5 Alissa                   Cooper                        22000
         6 Selena                   May                            4000
         7 David                    Lorentz                       30000
         8 Stalia                   Coleman                       10000
         9 James                    Connor                        14000
        10 Katy                     Sosa                          12000

9 rows selected.
```

17. Displays the first_name, last_name , salary and (salary+(salary*commission_pct)) as Net Salary of employees whose NetSalary is in the range 10000 and 15000.

QUERY:

```
SQL> SELECT EMP_FNAME,EMP_LNAME,EMP_SALARY,
  2  (EMP_SALARY+(EMP_SALARY*COMMISSION_PCT)) "Net Salary"
  3  FROM COMPANY
  4  WHERE
  5  (EMP_SALARY+(EMP_SALARY*COMMISSION_PCT))
  6  BETWEEN 10000 AND 15000
  7  AND COMMISSION_PCT>0;

EMP_FNAME                 EMP_LNAME                 EMP_SALARY Net Salary
------------------------- ------------------------- ---------- ----------
Jim                       Cooper                         10000      15000
Stalia                    Coleman                        10000      12000
```

18. Displays the employee_id, first_name, last_name and salary of employees whose salary is greater than or equal to 4000 and less than equal to 6000 where 4000 is the lower limit and 6000 is the upper limit ofthe salary.

QUERY:

```
SQL> SELECT EMP_ID,EMP_FNAME,EMP_LNAME,EMP_SALARY FROM COMPANY WHERE EMP_SALARY BETWEEN 4000 AND 6000;

    EMP_ID EMP_FNAME                 EMP_LNAME                 EMP_SALARY
---------- ------------------------- ------------------------- ----------
         2 Michael                   Hicks                           5000
         6 Selena                    May                             4000
```

19. Displays the employee_id, first_name, last_name and salary of employees whose first_name starting with 'S' and salary greater than orequal to 4000.

QUERY:

```
SQL> SELECT EMP_ID,EMP_FNAME,EMP_LNAME,EMP_SALARY FROM COMPANY WHERE EMP_FNAME LIKE ('S%') AND EMP_SALARY>=4000;

    EMP_ID EMP_FNAME                EMP_LNAME                EMP_SALARY
---------- ------------------------ ------------------------ ----------
         6 Selena                   May                            4000
         8 Stalia                   Coleman                       10000
```

20. Displays the employee_id, first_name, last_name and salary of employees whose first_name starting with 'S' or 'A'

QUERY:

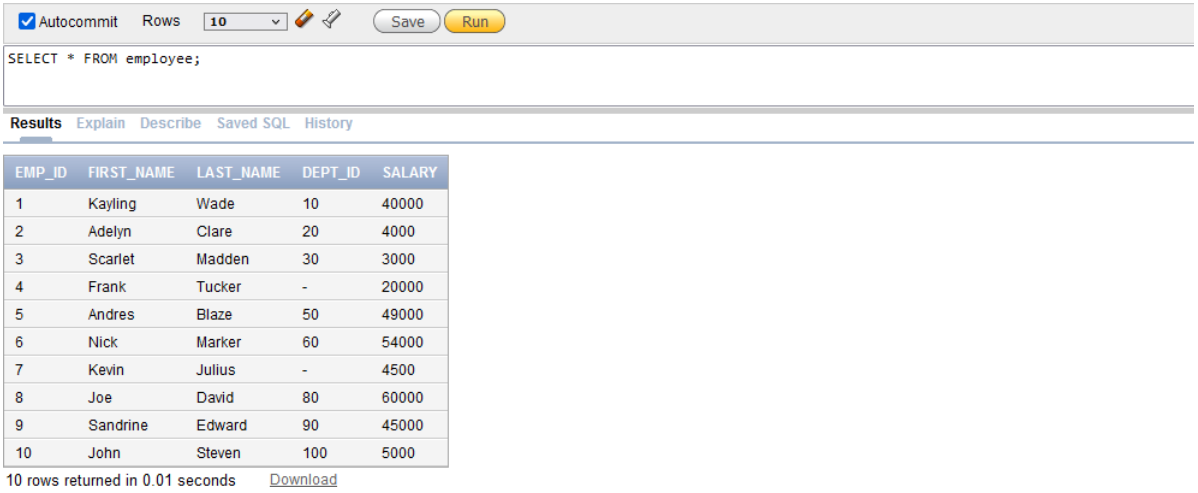
```
SQL> SELECT EMP_ID,EMP_FNAME,EMP_LNAME,EMP_SALARY FROM COMPANY WHERE EMP_FNAME LIKE ('S%') OR EMP_FNAME LIKE ('A%');

    EMP_ID EMP_FNAME                EMP_LNAME                EMP_SALARY
---------- ------------------------ ------------------------ ----------
         4 Adriana                  Sosa                          20000
         5 Alissa                   Cooper                        22000
         6 Selena                   May                            4000
         8 Stalia                   Coleman                       10000
```

Ajay Kumar Mandal
20BTRCO048
CSE – IOT

EXPERIMENT - 8 : Query sorted - ORDER BY clause

Table employee:



Autocommit Rows 10 Save Run

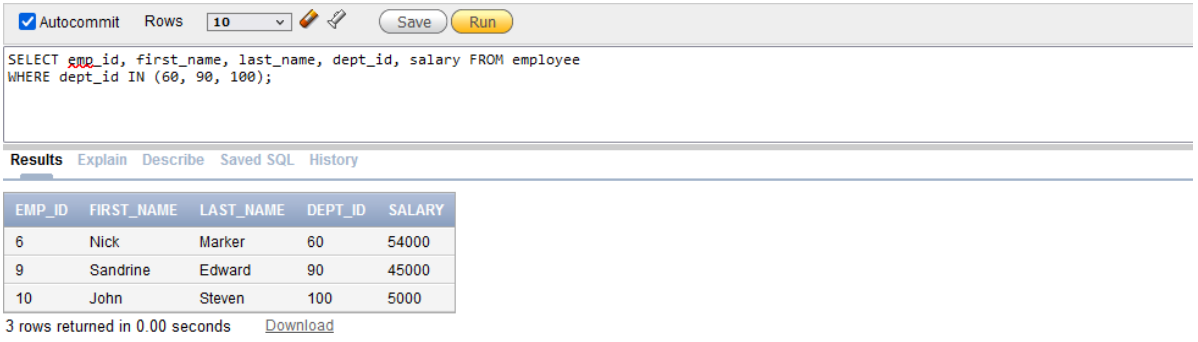
```
SELECT * FROM employee;
```

Results Explain Describe Saved SQL History

EMP_ID	FIRST_NAME	LAST_NAME	DEPT_ID	SALARY
1	Kayling	Wade	10	40000
2	Adelyn	Clare	20	4000
3	Scarlet	Madden	30	3000
4	Frank	Tucker	-	20000
5	Andres	Blaze	50	49000
6	Nick	Marker	60	54000
7	Kevin	Julius	-	4500
8	Joe	David	80	60000
9	Sandrine	Edward	90	45000
10	John	Steven	100	5000

10 rows returned in 0.01 seconds Download

1. Display the employee_id, first_name, last_name, department_id and salary ofemployees whose department_id 60, 90 or 100.



Autocommit Rows 10 Save Run

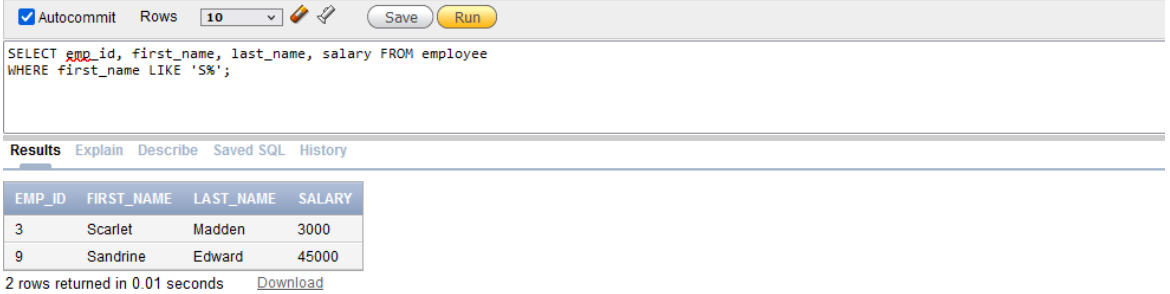
```
SELECT emp_id, first_name, last_name, dept_id, salary FROM employee
WHERE dept_id IN (60, 90, 100);
```

Results Explain Describe Saved SQL History

EMP_ID	FIRST_NAME	LAST_NAME	DEPT_ID	SALARY
6	Nick	Marker	60	54000
9	Sandrine	Edward	90	45000
10	John	Steven	100	5000

3 rows returned in 0.00 seconds Download

2. Display the employee_id, first_name, last_name and salary of employees whosefirst_namestarting with 'S'



Autocommit Rows 10 Save Run

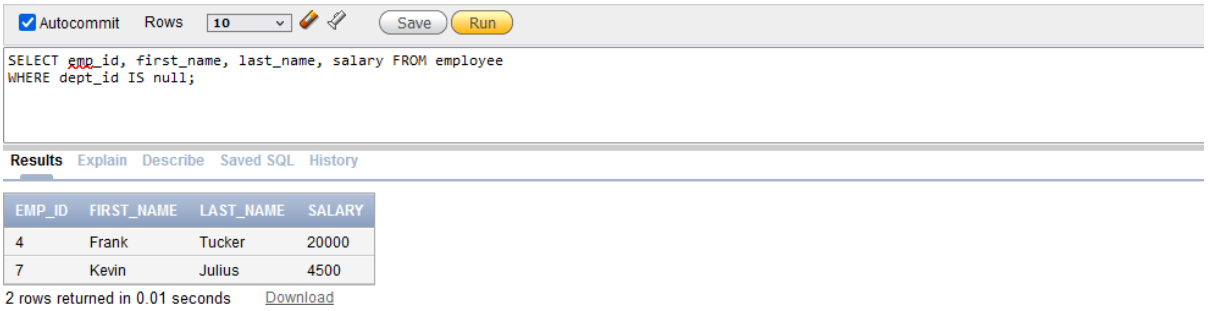
```
SELECT emp_id, first_name, last_name, salary FROM employee
WHERE first_name LIKE 'S%';
```

Results Explain Describe Saved SQL History

EMP_ID	FIRST_NAME	LAST_NAME	SALARY
3	Scarlet	Madden	3000
9	Sandrine	Edward	45000

2 rows returned in 0.01 seconds Download

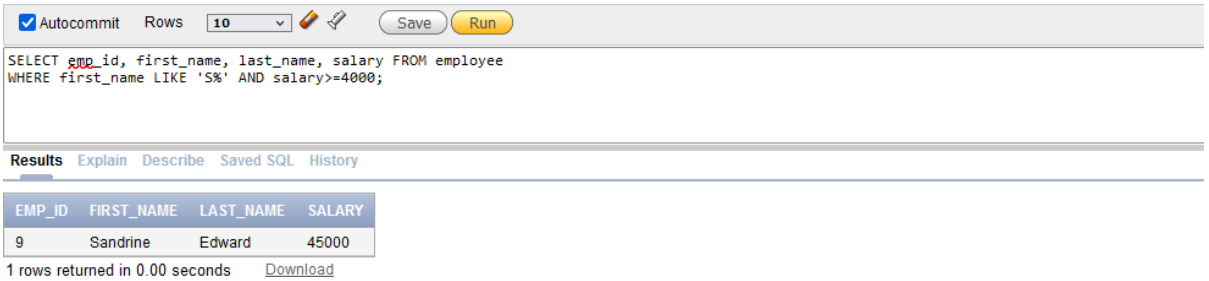
3. Display the employee_id, first_name, last_name and salary of employees whosedepartment_id is null.



EMP_ID	FIRST_NAME	LAST_NAME	SALARY
4	Frank	Tucker	20000
7	Kevin	Julius	4500

2 rows returned in 0.01 seconds Download

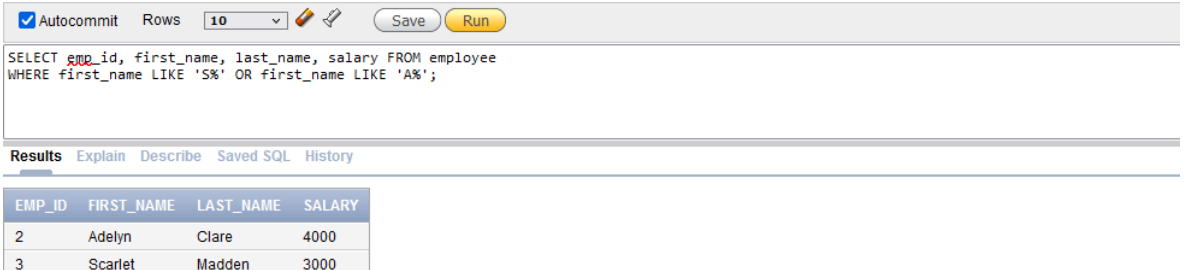
4. Display the employee_id, first_name, last_name and salary of employees whosefirst_name starting with 'S' and salary greater than or equal to 4000.



EMP_ID	FIRST_NAME	LAST_NAME	SALARY
9	Sandrine	Edward	45000

1 rows returned in 0.00 seconds Download

5. Display the employee_id, first_name, last_name and salary of employees whosefirst_name starting with 'S' or 'A'.



EMP_ID	FIRST_NAME	LAST_NAME	SALARY
2	Adelyn	Clare	4000
3	Scarlet	Madden	3000
5	Andres	Blaze	49000
9	Sandrine	Edward	45000

4 rows returned in 0.00 seconds Download

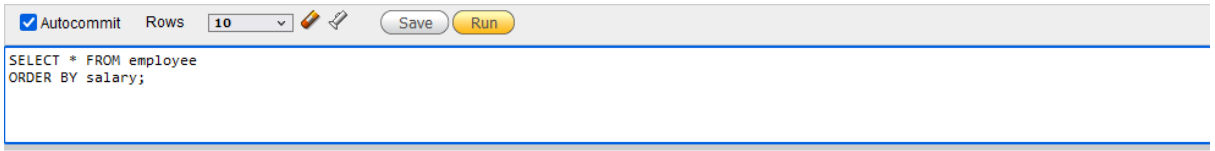
6. Display the employee_id, first_name, last_name and salary of employees except the department_id 90, 60 or 100.

Results Explain Describe Saved SQL History

EMP_ID	FIRST_NAME	LAST_NAME	SALARY
1	Kayling	Wade	40000
2	Adelyn	Clare	4000
3	Scarlet	Madden	3000
5	Andres	Blaze	49000
8	Joe	David	60000

5 rows returned in 0.00 seconds Download

7. Sort and display the employee table by salary of the employee.



Results Explain Describe Saved SQL History

EMP_ID	FIRST_NAME	LAST_NAME	DEPT_ID	SALARY
3	Scarlet	Madden	30	3000
2	Adelyn	Clare	20	4000
7	Kevin	Julius	-	4500
10	John	Steven	100	5000
4	Frank	Tucker	-	20000
1	Kayling	Wade	10	40000
9	Sandrine	Edward	90	45000
5	Andres	Blaze	50	49000
6	Nick	Marker	60	54000
8	Joe	David	80	60000

10 rows returned in 0.00 seconds Download

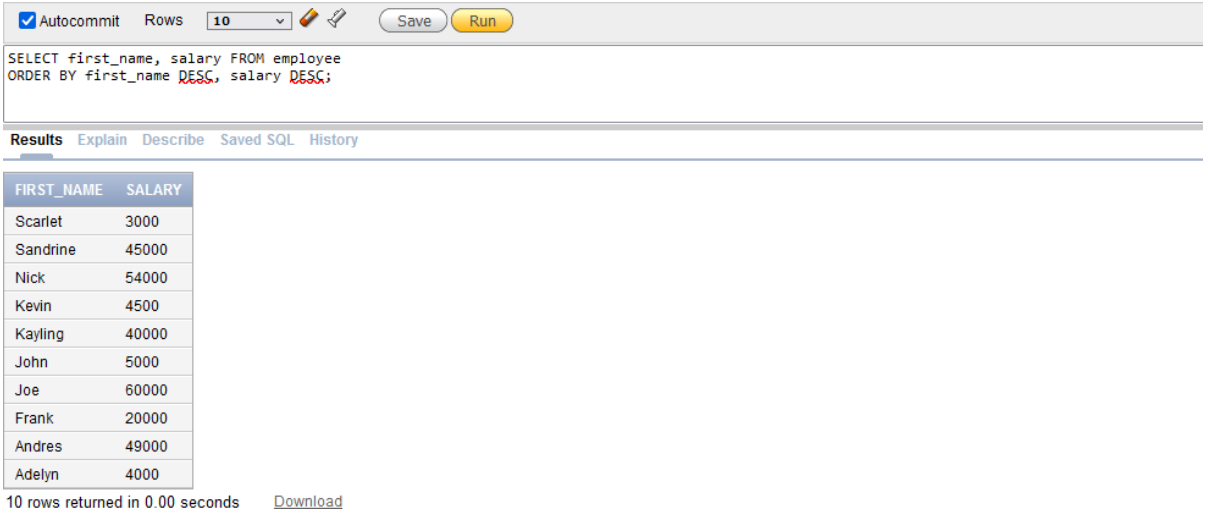
8. Sort and display the employee table by the name and salary.

Results Explain Describe Saved SQL History

EMP_ID	FIRST_NAME	LAST_NAME	DEPT_ID	SALARY
2	Adelyn	Clare	20	4000
5	Andres	Blaze	50	49000
4	Frank	Tucker	-	20000
8	Joe	David	80	60000
10	John	Steven	100	5000
1	Kayling	Wade	10	40000
7	Kevin	Julius	-	4500
6	Nick	Marker	60	54000
9	Sandrine	Edward	90	45000
3	Scarlet	Madden	30	3000

10 rows returned in 0.00 seconds Download

9. Select both name and salary in descending order and display it.



Autocommit Rows 10 Save Run

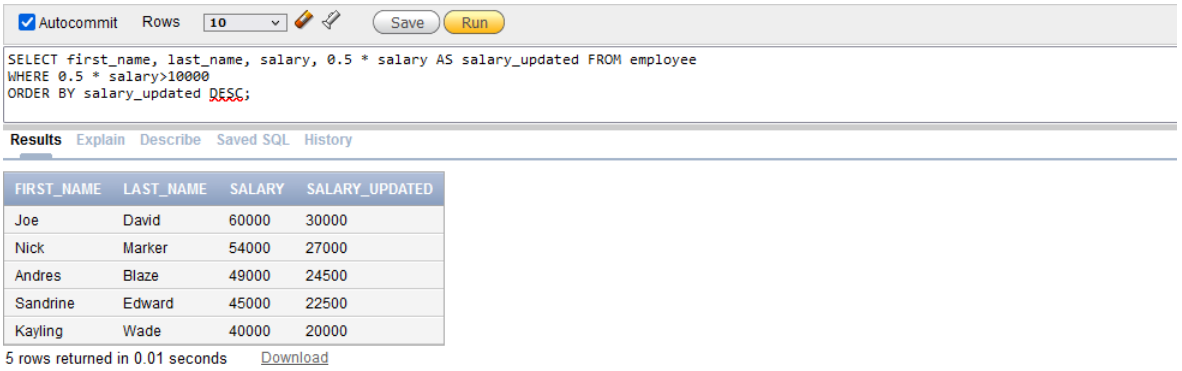
```
SELECT first_name, salary FROM employee
ORDER BY first_name DESC, salary DESC;
```

Results Explain Describe Saved SQL History

FIRST_NAME	SALARY
Scarlet	3000
Sandrine	45000
Nick	54000
Kevin	4500
Kayling	40000
John	5000
Joe	60000
Frank	20000
Andres	49000
Adelyn	4000

10 rows returned in 0.00 seconds Download

10. Display employee name, current salary, and a 20% increase in the salary for only those employees for whom the percentage increase in salary is greater than 10000and in descending order of the increased price.



Autocommit Rows 10 Save Run

```
SELECT first_name, last_name, salary, 0.5 * salary AS salary_updated FROM employee
WHERE 0.5 * salary>10000
ORDER BY salary_updated DESC;
```

Results Explain Describe Saved SQL History

FIRST_NAME	LAST_NAME	SALARY	SALARY_UPDATED
Joe	David	60000	30000
Nick	Marker	54000	27000
Andres	Blaze	49000	24500
Sandrine	Edward	45000	22500
Kayling	Wade	40000	20000

5 rows returned in 0.01 seconds Download

Ajay Kumar Mandal
20BTRCO048
CSE – IOT

EXPERIMENT - 9: Perform following: (a) UNION, INTERSECTION and MINUS operations on tables. and (b) UPDATE, ALTER, DELETE, DROP operations on tables

Table created:

Autocommit Rows 10 Save Run

```
SELECT * FROM students;
```

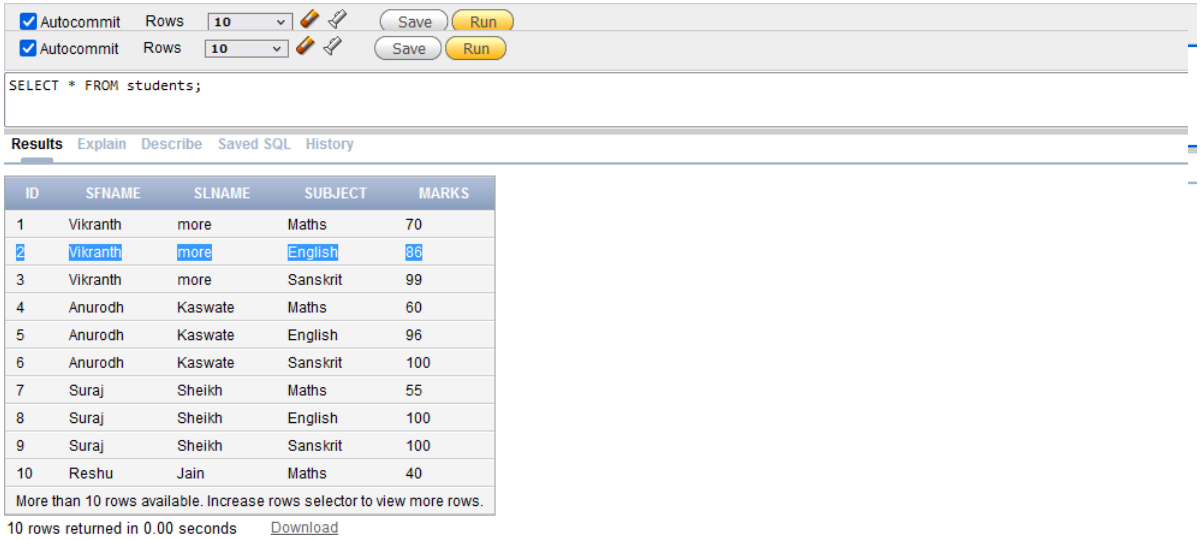
Results Explain Describe Saved SQL History

ID	SFNAME	SLNAME	SUBJECT	MARKS
1	Vikranth	more	Maths	70
2	Vikranth	more	English	80
3	Vikranth	more	Sanskrit	99
4	Anurodh	Kaswate	Maths	60
5	Anurodh	Kaswate	English	96
6	Anurodh	Kaswate	Sanskrit	100
7	Suraj	Sheikh	Maths	55
8	Suraj	Sheikh	English	100
9	Suraj	Sheikh	Sanskrit	100
10	Reshu	Jain	Maths	40

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.01 seconds Download

1. Update the English Mark of Vikrant to 86.



Autocommit Rows 10 Save Run

Autocommit Rows 10 Save Run

```
SELECT * FROM students;
```

Results Explain Describe Saved SQL History

ID	SFNAME	SLNAME	SUBJECT	MARKS
1	Vikranth	more	Maths	70
2	Vikranth	more	English	86
3	Vikranth	more	Sanskrit	99
4	Anurodh	Kaswate	Maths	60
5	Anurodh	Kaswate	English	96
6	Anurodh	Kaswate	Sanskrit	100
7	Suraj	Sheikh	Maths	55
8	Suraj	Sheikh	English	100
9	Suraj	Sheikh	Sanskrit	100
10	Reshu	Jain	Maths	40

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.00 seconds Download

2. Update the SFName to 'Sakthi' for all records whose SFName is “Suraj".

Autocommit Rows 10 Save Run

```
UPDATE students SET sfname='Sakthi'
WHERE sfname='Suraj';
```

Results Explain Describe Saved SQL History

```
3 row(s) updated.
```

0.01 seconds

Autocommit Rows 10 Save Run

```
SELECT * FROM students;
```

Results Explain Describe Saved SQL History

ID	SFNAME	SLNAME	SUBJECT	MARKS
1	Vikranth	more	Maths	70
2	Vikranth	more	English	86
3	Vikranth	more	Sanskrit	99
4	Anurodh	Kaswate	Maths	60
5	Anurodh	Kaswate	English	96
6	Anurodh	Kaswate	Sanskrit	100
7	Sakthi	Sheikh	Maths	55
8	Sakthi	Sheikh	English	100
9	Sakthi	Sheikh	Sanskrit	100
10	Reshu	Jain	Maths	40

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.00 seconds Download

3. Update Subject and Mark of Reshu from Sanskrit 91 to French 82.

Autocommit Rows 10 Save Run

```
UPDATE students SET subject='French', marks=82
WHERE sfname='Reshu' AND subject='Sanskrit';
```

Results Explain Describe Saved SQL History

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

0.01 seconds

Autocommit Rows 15 Save Run

```
SELECT * FROM students;
```

Results Explain Describe Saved SQL History

ID	SFNAME	SLNAME	SUBJECT	MARKS
1	Vikranth	more	Maths	70
2	Vikranth	more	English	86
3	Vikranth	more	Sanskrit	99
4	Anurodh	Kaswate	Maths	60
5	Anurodh	Kaswate	English	96
6	Anurodh	Kaswate	Sanskrit	100
7	Sakthi	Sheikh	Maths	55
8	Sakthi	Sheikh	English	100
9	Sakthi	Sheikh	Sanskrit	100
10	Reshu	Jain	Maths	40
11	Reshu	Jain	English	97
12	Reshu	Jain	French	82

12 rows returned in 0.01 seconds Download

4. Add a column named as Student DOB and insert records.

Autocommit Rows 15 Save Run

```
ALTER TABLE students ADD dob date;
```

Results Explain Describe Saved SQL History

```
Table altered.
```

0.07 seconds

Autocommit Rows 15 Save Run

```
UPDATE students SET dob='01-10-2001' WHERE sfname='Vikranth';
```

Results Explain Describe Saved SQL History

```
3 row(s) updated.
```

0.01 seconds

Autocommit Rows 15 Save Run

```
UPDATE students SET dob='01-02-2000' WHERE sfname='Anurodh';
```

Results Explain Describe Saved SQL History

```
3 row(s) updated.
```

0.01 seconds

Autocommit Rows 15 Save Run

```
UPDATE students SET dob='02-12-2002' WHERE sfname='Sakthi';
```

Results Explain Describe Saved SQL History

```
3 row(s) updated.
```

0.01 seconds

Autocommit Rows 15 Save Run

```
UPDATE students SET dob='02-28-2002' WHERE sfname='Reshu';
```

Results Explain Describe Saved SQL History

```
3 row(s) updated.
```

0.00 seconds

Autocommit Rows 15 Save Run

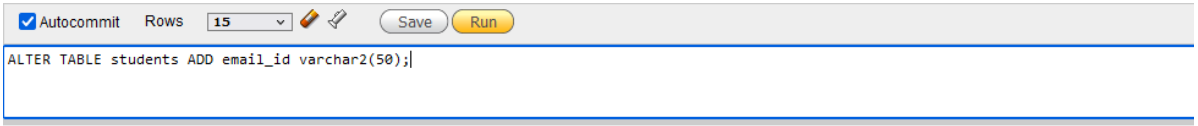
```
SELECT * FROM students;
```

Results Explain Describe Saved SQL History

ID	SFNAME	SLNAME	SUBJECT	MARKS	DOB
1	Vikranth	more	Maths	70	01/10/2001
2	Vikranth	more	English	86	01/10/2001
3	Vikranth	more	Sanskrit	99	01/10/2001
4	Anurodh	Kaswate	Maths	60	01/02/2000
5	Anurodh	Kaswate	English	96	01/02/2000
6	Anurodh	Kaswate	Sanskrit	100	01/02/2000
7	Sakthi	Sheikh	Maths	55	02/12/2002
8	Sakthi	Sheikh	English	100	02/12/2002
9	Sakthi	Sheikh	Sanskrit	100	02/12/2002
10	Reshu	Jain	Maths	40	02/28/2002
11	Reshu	Jain	English	97	02/28/2002
12	Reshu	Jain	French	82	02/28/2002

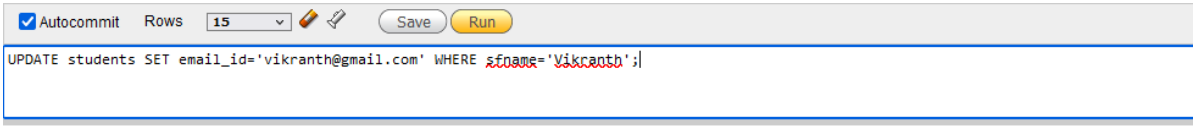
12 rows returned in 0.00 seconds Download

5. Add a column named as email_id and insert records.



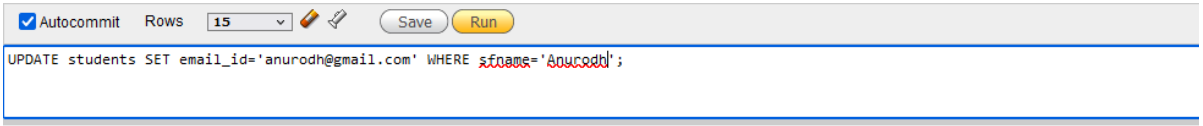

```
Table altered.
```

0.02 seconds



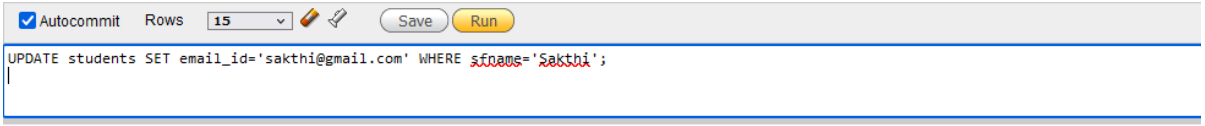

```
3 row(s) updated.
```

0.01 seconds



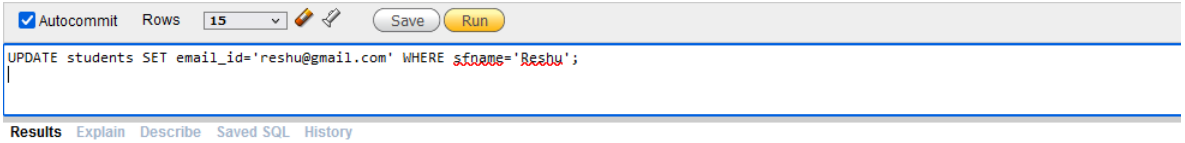

```
3 row(s) updated.
```

0.01 seconds




```
3 row(s) updated.
```

0.00 seconds




```
3 row(s) updated.
```

0.00 seconds

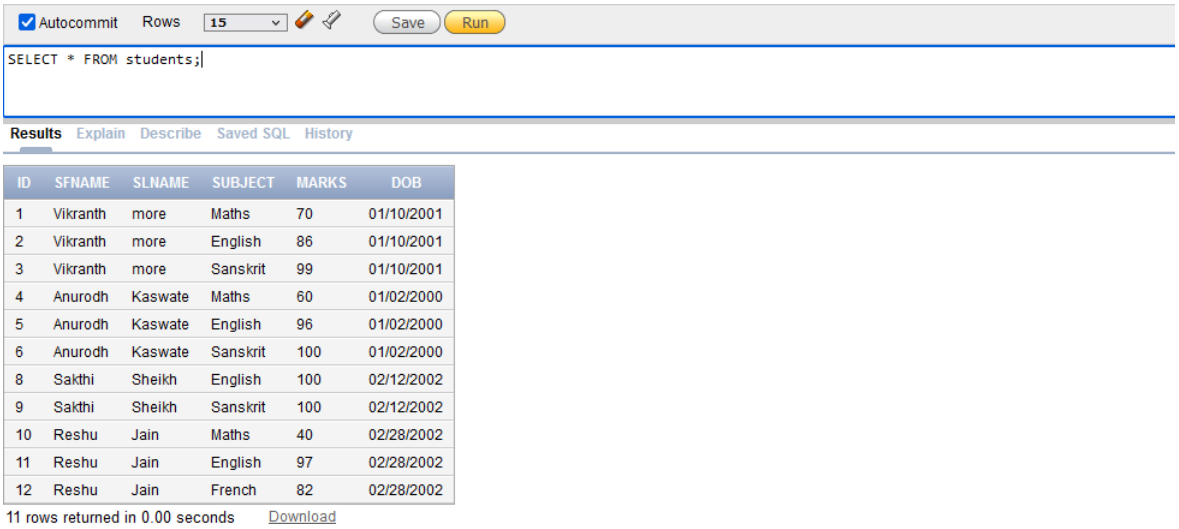
Autocommit Rows 15 Save Run

```
SELECT * FROM students;
```

Results Explain Describe Saved SQL History

ID	SFNAME	SLNAME	SUBJECT	MARKS	DOB
1	Vikranth	more	Maths	70	01/10/2001
2	Vikranth	more	English	86	01/10/2001
3	Vikranth	more	Sanskrit	99	01/10/2001
4	Anurodh	Kaswate	Maths	60	01/02/2000
5	Anurodh	Kaswate	English	96	01/02/2000
6	Anurodh	Kaswate	Sanskrit	100	01/02/2000
7	Sakthi	Sheikh	Maths	55	02/12/2002
8	Sakthi	Sheikh	English	100	02/12/2002
9	Sakthi	Sheikh	Sanskrit	100	02/12/2002
10	Reshu	Jain	Maths	40	02/28/2002
11	Reshu	Jain	English	97	02/28/2002
12	Reshu	Jain	French	82	02/28/2002

12 rows returned in 0.01 seconds Download



ID	SFNAME	SLNAME	SUBJECT	MARKS	DOB
1	Vikranth	more	Maths	70	01/10/2001
2	Vikranth	more	English	86	01/10/2001
3	Vikranth	more	Sanskrit	99	01/10/2001
4	Anurodh	Kaswate	Maths	60	01/02/2000
5	Anurodh	Kaswate	English	96	01/02/2000
6	Anurodh	Kaswate	Sanskrit	100	01/02/2000
8	Sakthi	Sheikh	English	100	02/12/2002
9	Sakthi	Sheikh	Sanskrit	100	02/12/2002
10	Reshu	Jain	Maths	40	02/28/2002
11	Reshu	Jain	English	97	02/28/2002
12	Reshu	Jain	French	82	02/28/2002

11 rows returned in 0.00 seconds Download

8. Rename the marks as student_marks.

Autocommit Rows 15 Save Run

```
ALTER TABLE students RENAME COLUMN marks TO student_marks;
```

Results Explain Describe Saved SQL History

```
Table altered.
```

0.02 seconds

Autocommit Rows 15 Save Run

```
SELECT * FROM students;
```

Results Explain Describe Saved SQL History

ID	SFNAME	SLNAME	SUBJECT	STUDENT_MARKS	DOB
1	Vikranth	more	Maths	70	01/10/2001
2	Vikranth	more	English	86	01/10/2001
3	Vikranth	more	Sanskrit	99	01/10/2001
4	Anurodh	Kaswate	Maths	60	01/02/2000
5	Anurodh	Kaswate	English	96	01/02/2000
6	Anurodh	Kaswate	Sanskrit	100	01/02/2000
8	Sakthi	Sheikh	English	100	02/12/2002
9	Sakthi	Sheikh	Sanskrit	100	02/12/2002
10	Reshu	Jain	Maths	40	02/28/2002
11	Reshu	Jain	English	97	02/28/2002
12	Reshu	Jain	French	82	02/28/2002

11 rows returned in 0.01 seconds Download

9. Delete the row from the table where the student Marks is less than 60.

Autocommit Rows 15 Save Run

```
DELETE FROM students
WHERE student_marks<60;
```

Results Explain Describe Saved SQL History

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

0.01 seconds

Autocommit Rows 15 Save Run

```
SELECT * FROM students;
```

Results Explain Describe Saved SQL History

ID	SFNAME	SLNAME	SUBJECT	STUDENT_MARKS	DOB
1	Vikranth	more	Maths	70	01/10/2001
2	Vikranth	more	English	86	01/10/2001
3	Vikranth	more	Sanskrit	99	01/10/2001
4	Anurodh	Kaswate	Maths	60	01/02/2000
5	Anurodh	Kaswate	English	96	01/02/2000
6	Anurodh	Kaswate	Sanskrit	100	01/02/2000
8	Sakthi	Sheikh	English	100	02/12/2002
9	Sakthi	Sheikh	Sanskrit	100	02/12/2002
11	Reshu	Jain	English	97	02/28/2002
12	Reshu	Jain	French	82	02/28/2002

10 rows returned in 0.00 seconds Download

10. Delete entire Student_table.

Autocommit Rows 15 Save Run

```
DROP TABLE students;
```

Results Explain Describe Saved SQL History

```
Table dropped.
```

0.17 seconds

Autocommit Rows 15 Save Run

```
SELECT * FROM students;
```

Results Explain Describe Saved SQL History

ORA-00942: table or view does not exist

Ajay Kumar Mandal
20BTRCO048
CSE – IOT

EXPERIMENT - 10 : Query multiple tables using JOIN operation.

Sample table: orders

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001

Sample table: customer

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007

Sample table: salesman

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	0.12

- create table orders(ord_no int, purch_amt number(6,2), ord_date date, customer_id int, salesman_id int);
- insert into orders values(70001, 150.5,'2012/10/05', 3005, 5002);
- insert into orders values(70009, 270.65,'2012/09/10', 3001, 5005);
- insert into orders values(70002, 65.26,'2012/10/05', 3002, 5001);
- insert into orders values(70004, 110.5,'2012/08/17', 3009, 5003);
- insert into orders values(70007, 948.5,'2012/09/10', 3005, 5002);
- insert into orders values(70005, 2400.6,'2012/07/27', 3007, 5001);
- insert into orders values(70008, 5760,'2012/09/10', 3002, 5001);
- select * from orders;

```
SQL> create table orders(ord_no int, purch_amt number(6,2), ord_date date, customer_id int, salesman_id
int);

Table created.

SQL> insert into orders values(70001, 150.5,'2012/10/05', 3005, 5002);

1 row created.

SQL> insert into orders values(70009, 270.65,'2012/09/10', 3001, 5005);

1 row created.

SQL> insert into orders values(70002, 65.26,'2012/10/05', 3002, 5001);

1 row created.

SQL> insert into orders values(70004, 110.5,'2012/08/17', 3009, 5003);

1 row created.

SQL> insert into orders values(70007, 948.5,'2012/09/10', 3005, 5002);

1 row created.

SQL> insert into orders values(70005, 2400.6,'2012/07/27', 3007, 5001);

1 row created.

SQL> insert into orders values(70008, 5760,'2012/09/10', 3002, 5001);

1 row created.

SQL> select * from orders;

    ORD_NO  PURCH_AMT ORD_DATE   CUSTOMER_ID SALESMAN_ID
---------- ---------- ---------- ----------- -----------
     70001      150.5 2012-10-05        3005        5002
     70009     270.65 2012-09-10        3001        5005
     70002      65.26 2012-10-05        3002        5001
     70004      110.5 2012-08-17        3009        5003
     70007      948.5 2012-09-10        3005        5002
     70005     2400.6 2012-07-27        3007        5001
     70008       5760 2012-09-10        3002        5001

7 rows selected.
```

- create table customer(customer_id int, cust_name varchar(20), city varchar(20), grade int, salesman_id int);
- insert into customer values (3002, 'Nick Rimando', 'New York',100,5001);
- insert into customer values (3007, 'Brad Davis', 'New York',200,5001);
- insert into customer values (3005, 'Graham Zusi', 'California',200,5002);
- insert into customer values (3008, 'Julian Green', 'London',300,5002);
- insert into customer values (3004, 'Fanian Johnson', 'Paris',300,5006);

- insert into customer values (3009, 'Geoff Cameron', 'Berlin',100,5003);
- insert into customer values (3003, 'JOzy Altidor', 'Moscow',200,5007);
- select * from customer;

```
SQL> create table customer(customer_id int, cust_name varchar(20), city varchar(20), grade int, salesman
_id int);

Table created.

SQL> insert into customer values (3002, 'Nick Rimando', 'New York',100,5001);

1 row created.

SQL> insert into customer values (3007, 'Brad Davis', 'New York',200,5001);

1 row created.

SQL> insert into customer values (3005, 'Graham Zusi', 'California',200,5002);

1 row created.

SQL> insert into customer values (3008, 'Julian Green', 'London',300,5002);

1 row created.

SQL> insert into customer values (3004, 'Fanian Johnson', 'Paris',300,5006);

1 row created.

SQL> insert into customer values (3009, 'Geoff Cameron', 'Berlin',100,5003);

1 row created.

SQL> insert into customer values (3003, 'JOzy Altidor', 'Moscow',200,5007);

1 row created.

SQL> select * from customer;

CUSTOMER_ID CUST_NAME           CITY                    GRADE SALESMAN_ID
----------- ------------------- ------------------- ---------- ----------
       3002 Nick Rimando        New York                   100       5001
       3007 Brad Davis          New York                   200       5001
       3005 Graham Zusi         California                 200       5002
       3008 Julian Green        London                     300       5002
       3004 Fanian Johnson      Paris                      300       5006
       3009 Geoff Cameron       Berlin                     100       5003
       3003 JOzy Altidor        Moscow                     200       5007

7 rows selected.
```

- create table salesman(salesman_id int, name varchar(20), city varchar(20), commission float(10));
- insert into salesman values (5001,'James Hoog','New York',0.15);

- ❼ insert into salesman values (5002,'Nail Knite','Paris',0.13);
- ❼ insert into salesman values (5005,'Pit Alex','London',0.11);
- ❼ insert into salesman values (5006,'Mc Lyon','Paris',0.14);
- ❼ insert into salesman values (5007,'Paul Adam','Rome',0.13);
- ❼ insert into salesman values (5003,'Lauson Hen','San Jose',0.12); ❼ select * from salesman;

```
SQL> create table salesman(salesman_id int, name varchar(20), city varchar(20), commission float(10));

Table created.

SQL> insert into salesman values (5001,'James Hoog','New York',0.15);

1 row created.

SQL> insert into salesman values (5002,'Nail Knite','Paris',0.13);

1 row created.

SQL> insert into salesman values (5005,'Pit Alex','London',0.11);

1 row created.

SQL> insert into salesman values (5006,'Mc Lyon','Paris',0.14);

1 row created.

SQL> insert into salesman values (5007,'Paul Adam','Rome',0.13);

1 row created.

SQL> insert into salesman values (5003,'Lauson Hen','San Jose',0.12);

1 row created.

SQL> select * from salesman;

SALESMAN_ID NAME               CITY               COMMISSION
----------- ------------------ ------------------ ----------
       5001 James Hoog         New York                  .15
       5002 Nail Knite         Paris                     .13
       5005 Pit Alex           London                    .11
       5006 Mc Lyon            Paris                     .14
       5007 Paul Adam          Rome                      .13
       5003 Lauson Hen         San Jose                  .12

6 rows selected.
```

->Write a SQL statement to know which salesman are working for which customer

- ❼ select salesman.name, customer.cust_name
- ❼ from salesman
- ❼ inner join customer on salesman.salesman_id=customer.salesman_id;

```
SQL> select salesman.name, customer.cust_name
  2  from salesman
  3  inner join customer on salesman.salesman_id=customer.salesman_id;

NAME                 CUST_NAME
-------------------- --------------------
James Hoog           Nick Rimando
James Hoog           Brad Davis
Nail Knite           Graham Zusi
Nail Knite           Julian Green
Mc Lyon              Fanian Johnson
Lauson Hen           Geoff Cameron
Paul Adam            JOzy Altidor

7 rows selected.
```

2.Write a SQL statement to find the list of customers who appointed a salesman for their jobs who gets a commission from the company is more than 12%

- ❼ select customer.cust_name
- ❼ from customer
- ❼ inner join salesman on customer.salesman_id=salesman.salesman_id
- ❼ where salesman.commission>0.12;

```
SQL> select customer.cust_name
  2  from customer
  3  inner join salesman on customer.salesman_id=salesman.salesman_id
  4  where salesman.commission>0.12;

CUST_NAME
--------------------
Nick Rimando
Brad Davis
Graham Zusi
Julian Green
Fanian Johnson
JOzy Altidor

6 rows selected.
```

3.Write a SQL statement to make a list in ascending order for the customer who works either through a salesman or by own.

❼select customer.cust_name
❼from customer
❼left join salesman on customer.salesman_id=salesman.salesman_id ❼ order by customer.cust_name;

```
SQL> select customer.cust_name
  2  from customer
  3  left join salesman on customer.salesman_id=salesman.salesman_id
  4  order by customer.cust_name;

CUST_NAME
--------------------
Brad Davis
Fanian Johnson
Geoff Cameron
Graham Zusi
JOzy Altidor
Julian Green
Nick Rimando

7 rows selected.
```

4.Write a SQL statement to make a list in ascending order for the salesmen who works either for one or more customer or not yet join under any of the customers.

- ❼ SELECT salesman.name AS "Salesma", customer.cust_name
- ❼ FROM customer
- ❼ RIGHT OUTER JOIN salesman
- ❼ ON salesman.salesman_id=customer.salesman_id
- ❼ ORDER BY salesman.salesman_id;

```
SQL> SELECT salesman.name AS "Salesma", customer.cust_name
  2  FROM customer
  3  RIGHT OUTER JOIN salesman
  4  ON salesman.salesman_id=customer.salesman_id
  5  ORDER BY salesman.salesman_id;

Salesma              CUST_NAME
-------------------- --------------------
James Hoog           Nick Rimando
James Hoog           Brad Davis
Nail Knite           Graham Zusi
Nail Knite           Julian Green
Lauson Hen           Geoff Cameron
Pit Alex
Mc Lyon              Fanian Johnson
Paul Adam            JOzy Altidor
```

2. Write a SQL statement to make a list for the salesmen who either work for one or more customers or yet to join any of the customer. The customer, may have placed, either one or more orders on or above order amount 2000 and must have a grade, or he may not have placed any order to the associated supplier.
   - ❼ SELECT a.cust_name,a.city,a.grade, b.name AS "Salesman", c.ord_no, c.ord_date, c.purch_amt ❼ FROM customer a
   - ❼ RIGHT OUTER JOIN salesman b
   - ❼ ON b.salesman_id=a.salesman_id
   - ❼ LEFT OUTER JOIN orders c

- ON c.customer_id=a.customer_id
- WHERE c.purch_amt>=2000 ❼ AND a.grade IS NOT NULL;

```
SQL> SELECT a.cust_name,a.city,a.grade, b.name AS "Salesman", c.ord_no, c.ord_date, c.purch_amt
  2  FROM customer a
  3  RIGHT OUTER JOIN salesman b
  4  ON b.salesman_id=a.salesman_id
  5  LEFT OUTER JOIN orders c
  6  ON c.customer_id=a.customer_id
  7  WHERE c.purch_amt>=2000
  8  AND a.grade IS NOT NULL;

CUST_NAME            CITY                     GRADE Salesman
-------------------- -------------------- ---------- --------------------
    ORD_NO ORD_DATE   PURCH_AMT
---------- --------- ----------
Brad Davis           New York                    200 James Hoog
     70005 27-JUL-12     2400.6

Nick Rimando         New York                    100 James Hoog
     70008 10-SEP-12       5760
```

6.Write a SQL statement to make a report with customer name, city, order no. order date, purchase amount for only those customers on the list who must have a grade and placed one or more orders or which order(s) have been placed by the customer who is neither in the list nor have a grade.

- SELECT a.cust_name,a.city, b.ord_no,
- b.ord_date,b.purch_amt AS "Order Amount"
- FROM customer a
- FULL OUTER JOIN orders b
- ON a.customer_id=b.customer_id
- WHERE a.grade IS NOT NULL;

```
SQL> SELECT a.cust_name,a.city, b.ord_no,
  2  b.ord_date,b.purch_amt AS "Order Amount"
  3  FROM customer a
  4  FULL OUTER JOIN orders b
  5  ON a.customer_id=b.customer_id
  6  WHERE a.grade IS NOT NULL;

CUST_NAME            CITY                    ORD_NO ORD_DATE  Order Amount
-------------------- -------------------- ---------- --------- ------------
Graham Zusi          California                70001 05-OCT-12        150.5
Nick Rimando         New York                  70002 05-OCT-12        65.26
Geoff Cameron        Berlin                    70004 17-AUG-12        110.5
Graham Zusi          California                70007 10-SEP-12        948.5
Brad Davis           New York                  70005 27-JUL-12       2400.6
Nick Rimando         New York                  70008 10-SEP-12         5760
Fanian Johnson       Paris
JOzy Altidor         Moscow
Julian Green         London

9 rows selected.
```

7.Write a SQL statement to find the details of a order i.e. order number, order date, amount of order, which customer gives the order and which salesman works for that customer and commission rate he gets for an order.

- SELECT a.ord_no,a.ord_date,a.purch_amt,
- b.cust_name AS "Customer Name", b.grade,
- c.name AS "Salesman", c.commission
- FROM orders a
- INNER JOIN customer b
- ON a.customer_id=b.customer_id
- INNER JOIN salesman c
- ON a.salesman_id=c.salesman_id;

```
SQL> SELECT a.ord_no,a.ord_date,a.purch_amt,
  2  b.cust_name AS "Customer Name", b.grade,
  3  c.name AS "Salesman", c.commission
  4  FROM orders a
  5  INNER JOIN customer b
  6  ON a.customer_id=b.customer_id
  7  INNER JOIN salesman c
  8  ON a.salesman_id=c.salesman_id;

    ORD_NO ORD_DATE   PURCH_AMT Customer Name             GRADE
---------- --------- ---------- -------------------- ----------
Salesman             COMMISSION
-------------------- ----------
     70001 05-OCT-12      150.5 Graham Zusi                 200
Nail Knite                  .13

     70002 05-OCT-12      65.26 Nick Rimando                100
James Hoog                  .15

     70004 17-AUG-12      110.5 Geoff Cameron               100
Lauson Hen                  .12


    ORD_NO ORD_DATE   PURCH_AMT Customer Name             GRADE
---------- --------- ---------- -------------------- ----------
Salesman             COMMISSION
-------------------- ----------
     70007 10-SEP-12      948.5 Graham Zusi                 200
Nail Knite                  .13

     70005 27-JUL-12     2400.6 Brad Davis                  200
James Hoog                  .15

     70008 10-SEP-12       5760 Nick Rimando                100
James Hoog                  .15


6 rows selected.
```

Ajay Kumar Mandal
20BTRCO048
CSE – IOT

EXPERIMENT - 11: Grouping the result of query - GROUP BY clause and HAVING clause

Create a new table having minimum of 10 records based on requirements.

Table Name: Books

Columns: ISBN, Title, Publication Date, Price, Publisher

CODE:

```
Create table books(
ISBN int primary key,
Title Varchar(50),
Publication_date date,
Price int,
Publisher varchar(50)
);
Insert into books(ISBN,title,publication_date,price,publisher) Values(001,'The
Daughter of Snow','2010-09-01',399,'Tulika'); Insert into
books(ISBN,title,publication_date,price,publisher) Values(002,'Murder on the
orient Express','2012-09-01',169,'Collins Crime Club'); Insert into
books(ISBN,title,publication_date,price,publisher)

Values(003,'The lord of the rings','2007-10-11',599,'Allen & Unwin');
Insert into books(ISBN,title,publication_date,price,publisher)
Values(004,'Heart of Darkness','1902-05-16',799,'Blackwoods
Magazine'); Insert into books(ISBN,title,publication_date,price,publisher)
Values(005,'Long Bright River','1901-01-01',499,'Riverhead publication');
```

```
Insert into books(ISBN,title,publication_date,price,publisher)

Values(006,'The Women in White','1859-11-26',469,'Charles Dickens magazine');
Insert into books(ISBN,title,publication_date,price,publisher)

Values(007,'The Girl with the Dragon Tattoo','2015-06-30',499,'Norstedts Förlag');
Insert into books(ISBN,title,publication_date,price,publisher)

Values(008,'A Brief History of Time','2019-02-12',499,'Bantam Dell Publishing Group');
Insert into books(ISBN,title,publication_date,price,publisher)

Values(009,'Bad blood','1776-01-10',299,'Alfred A. Knopf');
Insert into books(ISBN,title,publication_date,price,publisher)
Values(010,'The big sleep','2001-10-21',599,'Alfred
A.Knopf'); Alter table books add catogery Varchar(20);

Update books set catogery= 'Mystery' where ISBN IN(006,007,010);
Update books set catogery= 'Fiction' where ISBN in(001,003,004,005);
Update books set catogery= 'Crime' where ISBN in(002);

Update books set catogery= 'Non-Fiction' where ISBN
 in(008,009); Select * from books;
```

OUTPUT:

Output Input Comments 0 (1.52 sec)

Text

ISBN	title	publication_date	price	publisher	catogery
1	The Daughter of Snow	2010-09-01	399	Tulika	Fiction
2	Murder on the orient Express	2012-09-01	169	Collins Crime Club	Crime
3	The lord of the rings	2007-10-11	599	Allen & Unwin	Fiction
4	Heart of Darkness	1902-05-16	799	Blackwoods Magazine	Fiction
5	Long Bright River	1901-01-01	499	Riverhead publication	Fiction
6	The Women in White	1859-11-26	469	Charles Dickens magazine	Mystery
7	The Girl with the Dragon Tattoo	2015-06-30	499	Norstedts Förlag	Mystery
8	A Brief History of Time	2019-02-12	499	Bantam Dell Publishing Group	Non-Fiction
9	Bad blood	1776-01-10	299	Alfred A. Knopf	Non-Fiction
10	The big sleep	2001-10-21	599	Alfred A.Knopf	Mystery

Write SQL queries for the following statements:

1)Determine how many books are in each category.

Code:

```
select count(ISBN) , catogery from books
group by catogery
order by count(ISBN);
```

Output:

count(ISBN)	catogery
1	Crime
2	Non-Fiction
3	Mystery
4	Fiction

2)Determine how many books are in the Management category.

Code:

```
select count(ISBN) , catogery from books
```

```
group by catogery
having catogery='Management'
order by count(ISBN);
```

Output:

(As there were no books in management so the result is empty)

3)Determine the average book price of each category.

Code:

```
Select count(ISBN) , catogery , avg(price)from books
Group by catogery
Order by count(ISBN);
```

Output:

count(ISBN)	catogery	avg(price)
1	Crime	169.0000
2	Non-Fiction	399.0000
3	Mystery	522.3333
4	Fiction	574.0000

4)List the price of the least expensive book in each category

Code:

```
select catogery ,min(price) from books
group by catogery
order by min(price);
```

Output:

```
catogery        min(price)
Crime   169
Non-Fiction     299
Fiction 399
Mystery 469
```

Ajay Kumar Mandal
20BTRCO048
CSE – IOT

EXPERIMENT - 12: Query multiple tables using NATURAL and OUTER JOIN operation.

QUERY:

```
CREATE TABLE Student_details(
Roll_no INT PRIMARY KEY,
Name VARCHAR(10),
Address VARCHAR(10),
Phone VARCHAR(10),
Age INT(2)
);
INSERT INTO Student_details(
Roll_no ,
Name ,
Address ,
Phone ,
Age )
VALUES
( 1,'Harsh','Delhi','9867543321',18),
( 2,'Pratik','Bihar','9978654432',19),
( 3,'Riyanka’,Siliguri','8169765543',20),
( 4,'Deep','Ramnagar','7271876654',18),
```

(5,'Saptarhi','Kolkata','9382987765',19),

(6,'Dhanraj','Barabajar','8493198876',20),

(7,'Rohit','Balurghat','7514219987',18),

(8,'Niraj','Alipur','9625321198',19);

CODE:

SELECT * from Student_details;

OUTPUT:

Roll_no	Name	Address	Phone	Age
1	Harsh	Delhi	9867543321	18
2	Pratik	Bihar	9978654432	19
3	Riyanka	Siliguri	8169765543	20
4	Deep	Ramnagar	7271876654	18
5	Saptarhi	Kolkata	9382987765	19
6	Dhanraj	Barabajar	8493198876	20
7	Rohit	Balurghat	7514219987	18
8	Niraj	Alipur	9625321198	19

CREATE TABLE StudentCourses(

Course_id INT(2),

Roll_no INT(2)

);

INSERT INTO StudentCourses(

Course_id,

Roll_no)

VALUES

(1,1) ,

(2,2) ,

(2,3) ,

(3,4) ,

(1,5) ,

(4,9) ,

(5,10),

(4,11);

CODE:

select * from StudentCourses;

OUTPUT:

Course_id	Roll_no
1	1
2	2
2	3
3	4
1	5
4	9
5	10
4	11

COMBINED CODE FOR FINDING JOINS:

CREATE TABLE Student_details(

```
Roll_no INT PRIMARY KEY,
Name VARCHAR(10),
Address VARCHAR(10),
Phone VARCHAR(10),
Age INT(2)
);
INSERT INTO Student_details(
Roll_no ,
Name ,
Address ,
Phone ,
Age )
VALUES
( 1,'Harsh','Delhi','9867543321',18),
( 2,'Pratik','Bihar','9978654432',19),
( 3,'Riyanka','Siliguri','8169765543',20),
( 4,'Deep','Ramnagar','7271876654',18),
( 5,'Saptarhi','Kolkata','9382987765',19),
( 6,'Dhanraj','Barabajar','8493198876',20),
( 7,'Rohit','Balurghat','7514219987',18),
```

(8,’Niraj’,’Alipur’,’9625321198’,19);

```
CREATE TABLE StudentCourses(
Course_id INT(1),
Roll_no INT(2)
);
INSERT INTO StudentCourses(
Course_id,
Roll_no )
VALUES
( 1,1) ,
( 2,2) ,
( 2,3) ,
( 3,4) ,
( 1,5) ,
( 4,9) ,
( 5,10),
( 4,11);
```

Natural Join :

```
SELECT * FROM StudentCourses
NATURAL JOIN
Student_details;
```

OUTPUT:

Roll_no	Course_id	Name	Address	Phone	Age
1	1	Harsh	Delhi	9867543321	18
2	2	Pratik	Bihar	9978654432	19
3	2	Riyanka	Siliguri	8169765543	20
4	3	Deep	Ramnagar	7271876654	18
5	1	Saptarhi	Kolkata	9382987765	19

Left Outer Join :

SELECT * FROM Student_details

LEFT JOIN

StudentCourses

ON

(Student_details.Roll_no =StudentCourses.Roll_no);

OUTPUT:

Roll_no	Name	Address	Phone	Age	Course_id	Roll_no	
1	Harsh	Delhi	9867543321	18	1	1	
2	Pratik	Bihar	9978654432	19	2	2	
3	Riyanka	Siliguri	8169765543	20	2	3	
4	Deep	Ramnagar	7271876654	18	3	4	
5	Saptarhi	Kolkata	9382987765	19	1	5	
6	Dhanraj	Barabajar	8493198876	20	NULL	NULL	
7	Rohit	Balurghat	7514219987	18	NULL	NULL	
8	Niraj	Alipur	9625321198	19	NULL	NULL	

Right Outer Join :

SELECT * FROM Student_details

RIGHT JOIN

StudentCourses

ON

(Student_details.Roll_no =StudentCourses.Roll_no);

OUTPUT:

Roll_no	Name	Address	Phone	Age	Course_id	Roll_no	
1	Harsh	Delhi	9867543321	18	1	1	
2	Pratik	Bihar	9978654432	19	2	2	
3	Riyanka	Siliguri	8169765543	20	2	3	
4	Deep	Ramnagar	7271876654	18	3	4	
5	Saptarhi	Kolkata	9382987765	19	1	5	
NULL	NULL	NULL	NULL	NULL	4	9	
NULL	NULL	NULL	NULL	NULL	5	10	
NULL	NULL	NULL	NULL	NULL	4	11	

Full Outer Join :

SELECT * FROM Student_details

FULL JOIN

StudentCourses

ON

(Student_details.Roll_no =StudentCourses.Roll_no);

OUTPUT:

ROLL_NO	NAME	ADDRESS	PHONE	AGE	COURSE_ID	ROLL_NO
1	Harsh	Delhi	9867543321	18	1	1
2	Pratik	Bihar	9978654432	19	2	2
4	Deep	Ramnagar	7271876654	18	3	4
5	Saptarhi	Kolkata	9382987765	19	1	5
6	Dhanraj	Barabajar	8493198876	20	null	null
7	Rohit	Balurghat	7514219987	18	null	null
8	Niraj	Alipur	9625321198	19	null	null
null	null	null	null	null	4	9
null	null	null	null	null	5	10
null	null	null	null	null	4	11