List of experiments:

S.no	List of experiments
1	Creation, altering and droping of tables and inserting rows into a table (use constraints while creating tables) examples using SELECT command.
2	Queries (along with sub Queries) using ANY, ALL, IN, EXISTS, NOTEXISTS, UNION, INTERSET, Constraints. Example:- Select the roll number and name of the student who secured fourth rank in the class.
3	Queries using Aggregate functions (COUNT, SUM, AVG, MAX and MIN), GROUP BY, HAVING and Creation and dropping of Views.
4	Queries using Conversion functions (to_char, to_number and to_date), string functions (Concatenation, lpad, rpad, ltrim, rtrim, lower, upper, initcap, length, substr and instr), date functions (Sysdate, next_day, add_months, last_day, months_between, least, greatest, trunc, round, to_char, to_date)
5	a) Creation of simple PL/SQL program which includes declaration section, executable section and exception –Handling section (Ex. Student marks can be selected from the table and printed for those who secured first class and an exception can be raised if no records were found) b) Insert data into student table and use COMMIT, ROLLBACK and SAVEPOINT in PL/SQL block.
6	Develop a program that includes the features NESTED IF, CASE and CASE expression. The program can be extended using the NULLIF and COALESCE functions.
7	Program development using WHILE LOOPS, numeric FOR LOOPS, nested loops using ERROR Handling, BUILT –IN Exceptions, USE defined Exceptions, RAISEAPPLICATION ERROR.
8	Programs development using creation of procedures, passing parameters IN and OUT of PROCEDURES.
9	Program development using creation of stored functions, invoke functions in SQL Statements and write complex functions.
10	Develop programs using features parameters in a CURSOR, FOR UPDATE CURSOR, WHERE CURRENT of clause and CURSOR variables.
11	Develop Programs using BEFORE and AFTER Triggers, Row and Statement Triggers and INSTEAD OF Triggers
12	Create a table and perform the search operation on table using indexing and non-indexing techniques.

Exercise - 1

1. Creation, altering and droping of tables and inserting rows into a table (use constraints while creating tables) examples using SELECT command.

Aim: To create a table using CREATE TABLE command in SQL.

Syntax:

```
CREATE TABLE tableName
(
 column_1 datatype [ NULL | NOT NULL ],
 column_2 datatype [ NULL | NOT NULL ],
 ...
);
```

Here,

- The parameter **tableName** denotes the name of the table that you are going to create.
- The parameters **column_1**, **column_2**… denote the columns to be added to the table.
- A column should be specified as either **NULL** or **NOTNULL**.

Q1) Create a table 'st' with 'CREATE TABLE' command in SQL which maintains information about the students.

Query:

SQL> create table st(sid int primary key,sname varchar(20),sdept varchar(20),sgrade varchar(5));

Table created**.**

Q2) Describe the structure of a table using 'DESC' command in SQL

Query:

SQL> desc st;

Name	Null?	Type
SID	NOT NULL	NUMBER(38)
SNAME		VARCHAR2(20)
SGRADE		VARCHAR2(5)

Q2) Alter table 'st' with new name 'student'.

```
SQL> alter table st
  2  rename to student;
```

Table altered**.**

Q3) Alter table by adding new columns age and address.

Query:

```
SQL> alter table student
  2  add( age number(2),
  3       address  varchar(20)
  4    );
```

Table altered.

Q4)Modify the student table by changing the columns datatype and size

Query:

```
SQL> alter table student
  2  modify(sdept int,
  3         sname varchar(10));
```

Table altered.

Q5) Rename the columns to name, dept, and grade in student table

Queries:

```
SQL> alter table student
  2  rename column sname to name;
```

Table altered.

```
SQL> alter table student
  2  rename column sdept to dept;
```

Table altered.

```
SQL> alter table student
  2  rename column sgrade to grade;
```

Table altered.

Q6) Alter table 'student' by dropping column

Query:

```
SQL> alter table student drop column dept;
```

Table altered.

Q7) Create a 'course' table using CREATE TABLE command.

Query:

```
SQL> create table course(
            cid  int primary key,
            cname varchar(10)
            );
```

Table created**.**

Q8) Describe the structure of a table using 'DESC' command in SQL

Query:

```
SQL> desc course;

 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CID                                       NOT NULL NUMBER(38)
 CNAME                                              VARCHAR2(10)
```

Q9) Create a table enrollment which describes student enrollment in a course.

Query:

```
SQL> create table enrollment(sid int references student(sid),
  2     cid int references course(cid)
  3  , primary key(sid, cid));

Table created.
```

Q10) Describe the structure of an enrollment table

Query:

```
SQL> desc enrollment;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SID                                       NOT NULL NUMBER(38)
 CID                                       NOT NULL NUMBER(38)
```

Q11) Insert records into the table student

Queries:

<u>Method 1: Inserting all values in the order of column definition</u>

```
SQL> insert into student values(1201, 'ananya', 'a', 18, 'vijayawada');

1 row created.

SQL> insert into student values(1208, 'keerthi', 's', 19, 'guntur');
1 row created.
```

Method 2: Inserting specific columns

```
SQL> insert into student(sid, name, address) values(1210, 'yesoda','vijayawada');

1 row created.

SQL> insert into student(name, grade, sid) values('swetha', 's', 1226);

1 row created.
```

Method 3: Inserting multiple columns

```
SQL> insert into student(sid, name, age, grade, address)
  2  values(&sid, '&name', &age, '&grade', '&address');
Enter value for sid: 1240
Enter value for name: harsha
Enter value for age: 20
Enter value for grade: o
Enter value for address: vijayawada
old   2: values(&sid, '&name', &age, '&grade', '&address')
new   2: values(1240, 'harsha', 20, 'o', 'vijayawada')

1 row created.

SQL> /
Enter value for sid: 1254
Enter value for name: riyaz
Enter value for age: 20
Enter value for grade: a
Enter value for address: guntur
old   2: values(&sid, '&name', &age, '&grade', '&address')
new   2: values(1254, 'riyaz', 20, 'a', 'guntur')

1 row created.
```

Q12) display the records in the table student

```
SQL> select * from student;
      SID NAME       G        AGE ADDRESS
---------- ---------- - ---------- --------------------
      1201 ananya     a         18 vijayawada
      1208 keerthi    s         19 guntur
      1210 yesoda                  vijayawada
      1226 swetha     s
      1240 harsha     o         20 vijayawada
      1254 riyaz      a         20 guntur
```

6 rows selected.

Q13) Update 1226 record.

Query:

```
SQL> update student
  2  set age = 18
  3  where sid = 1226;

1 row updated.
```

Q14) update the address field of 'harsha'

Query:

```
SQL> update student
  2  set address = 'guntur'
  3  where name = 'harsha';

1 row updated.
```

List of records in student relation

```
SQL> select * from student;

      SID NAME       G       AGE ADDRESS
---------- ---------- - ---------- --------------------
      1201 ananya     a        18 vijayawada
      1208 keerthi    s        19 guntur
      1210 yesoda                 vijayawada
      1226 swetha     s        18
      1240 harsha     o        20 guntur
      1254 riyaz      a        20 guntur

6 rows selected.
```

Q15) Display students who are living in Guntur.

Query:

```
SQL> select * from student
  2  where address = 'guntur';

      SID NAME       G       AGE ADDRESS
---------- ---------- - ---------- --------------------
      1208 keerthi    s        19 guntur
      1240 harsha     o        20 guntur
```

1254 riyaz a 20 guntur

3 rows selected.

Q16) Give student details by using Column Aliases

Query:

```
SQL> select sid as rollno, name as sname, grade as g from student
  2 ;
```

Output:

```
   ROLLNO SNAME      G
---------- ---------- -
      1201 ananya     a
      1208 keerthi    s
      1210 yesoda
      1226 swetha     s
      1240 harsha     o
      1254 riyaz      a

6 rows selected.
```

Q17)Insert records into 'course' table

Query:

```
SQL> insert into course values(1, 'dbms');

1 row created.

SQL> insert into course values(2, 'toc');

1 row created.

SQL> insert into course values(3, 'os');

1 row created.

SQL> insert into course values(4, 'jp');

1 row created.

SQL> insert into course values(5, 'ps');

1 row created.
```

```
SQL> select * from course;

       CID CNAME
---------- ----------
         1 dbms
         2 toc
         3 os
         4 jp
         5 ps
```

Q18) Insert records into 'enrollment' table

Queries:

```
SQL> insert into enrollment values(1201, 3);

1 row created.

SQL> insert into enrollment values(1208, 1);

1 row created.

SQL> insert into enrollment values(1201, 2);

1 row created.

SQL> insert into enrollment values(1210, 1)
  2  ;

1 row created.

SQL> insert into enrollment values(1226, 4)
  2  ;

1 row created.

SQL> insert into enrollment values(1226, 2);

1 row created.

SQL> insert into enrollment values(1240, 1);

1 row created.
```

```
SQL> insert into enrollment values(1254, 5);

1 row created.

SQL> insert into enrollment values(1240, 3);

1 row created.
```

List of records in the table 'enrollment':

```
SQL> select * from enrollment;

       SID        CID
---------- ----------
      1201          2
      1201          3
      1208          1
      1210          1
      1226          2
      1226          4
      1240          1
      1240          3
      1254          5

9 rows selected.
```

Student-Course Database is…

Student

SID	NAME	G	AGE	ADDRESS
1201	ananya	a	18	vijayawada
1208	keerthi	s	19	guntur
1210	yesoda			vijayawada
1226	swetha	s	18	
1240	harsha	o	20	guntur
1254	riyaz	a	20	guntur

Course

CID	CNAME
1	dbms
2	toc
3	os
4	jp
5	ps

Enrollment

SID	CID
1201	2
1201	3
1208	1
1210	1
1226	2
1226	4
1240	1
1240	3
1254	5

CID	CNAME
1	dbms
2	toc
3	os
4	jp
5	ps

Enrollment

SID	CID
1201	2
1201	3

1208	1
1210	1
1226	2
1226	4
1240	1
1240	3
1254	5

Q19) Display courses offered to the students.

Query:

SQL> select cname from course;

Output:

CNAME

---

dbms

toc

os

jp

ps

Q20) What is the course id of 'dbms'?

Query:

SQL> select cid from course where cname = 'dbms';

Output:

CID

1

SQL> select cid as code from course where cname = 'dbms';

Output:

CODE

1

Q21) Display the student id who joined in '1' course.

Query:

SQL> select sid from enrollment where cid=1;

Output:

```
       SID
----------
      1208
      1210
      1240
```

Q22) Give the details of the students who joined in '1' course.

Query:

```
SQL> select s.sid, s.name, s.address
  2  from student s, enrollment e
  3  where s.sid = e.sid and e.cid = 1;

       SID NAME       ADDRESS
---------- ---------- --------------------
      1208 keerthi    guntur
      1210 yesoda     vijayawada
      1240 harsha     Guntur
```

Or

```
SQL> select s.sid, s.name, s.address
  2  from  student s,  enrollment e
  3  where  e.cid = 1 and e.sid = s.sid;

       SID NAME       ADDRESS
---------- ---------- --------------------
      1208 keerthi    guntur
      1210 yesoda     vijayawada
      1240 harsha     guntur
```

s.SID	s.NAME	s.G	s.AGE	s.ADDRESS	e.SID	e.CID
1201	ananya	a	18	vijayawada	1201	2
1201	ananya	a	18	vijayawada	1201	3
1201	ananya	a	18	vijayawada	1208	1
……						
1208	keerthi	s	19	guntur	1201	2
1208	keerthi	s	19	guntur	1201	3
1208	keerthi	s	19	guntur	1208	1
……..						
1210	yesoda			vijayawada	1201	2
……						
1210	yesoda			vijayawada	1210	1

1240	harsha	o	20	guntur	1240	1
…….						

Q23) Display the student name, course name and course id who joined in 'dbms' course

Query:

```
SQL> select  s.name, c.cname, c.cid
  2  from  student s,  course c, enrollment e
  3  where  c.cid = e.cid and c.cname = 'dbms'  and e.sid = s.sid;

NAME       CNAME           CID
---------- ---------- ----------
keerthi    dbms             1
yesoda     dbms             1
harsha     dbms             1
```

Q24) Give the names and addresses of the students who joined in 'toc' course.

Query:

```
SQL> select  s.name,  s.address
  2  from student s, course c, enrollment e
  3  where  c.cname='toc' and c.cid = e.cid and e.sid=s.sid;

NAME       ADDRESS
---------- --------------------
ananya     vijayawada
swetha
```

Q25) Give the student details along with courses joined

Query:

```
SQL> select  s.sid, s.name, s.grade, s.age, s.address, c.cname
  2  from student s, course c, enrollment e
  3  where e.sid = s.sid and e.cid = c.cid;
```

Output:

```
       SID NAME       G        AGE ADDRESS            CNAME
---------- ---------- - ---------- -------------------- ----------
      1201 ananya     a         18 vijayawada          toc
      1201 ananya     a         18 vijayawada          os
      1208 keerthi    s         19 guntur              dbms
      1210 yesoda               vijayawada             dbms
```

1226	swetha	s	18	toc
1226	swetha	s	18	jp
1240	harsha	o	20 guntur	dbms
1240	harsha	o	20 guntur	os
1254	riyaz	a	20 guntur	ps

9 rows selected.

Q26) Drop table student table.

Query:

```
SQL> drop table student;
drop table student
          *
ERROR at line 1:
ORA-02449: unique/primary keys in table referenced by foreign keys
```

Note: To drop a parent table we need to drop child table first.

Q27) Drop the table enrollment.

```
SQL> drop table enrollment;
```

Output:
Table dropped.

Q28) Drop the table course.

```
SQL>drop table course;
```

Output:
Table dropped.

Drop:

- **Removes or deletes - 1) all data in the table/relation -**
- **2) deletes structure/schema of the relation from memory**
- **Rollback can't be applied**

Truncate:

-removes all the records from the relation.

-structure/schema remains same

Can apply rollback(means undo) command

Exercise - 2

2.Queries (along with sub Queries) using ANY, ALL, IN, EXISTS, NOTEXISTS, UNION, INTERSECT, Constraints. Example:- Select the roll number and name of the student who secured fourth rank in the class.

Aim: To write queries (along with sub Queries) using ANY, ALL, IN, EXISTS, NOTEXISTS, UNION, INTERSECT, Constraints.

Q1) UNION: Find the names of sailors who have reserved a red or a green boat.

Query: Using 'UNION'

```
SQL> select s.sname
  2  from sailors s, reserves r, boats b
  3  where s.sid = r.sid and r.bid = b.bid and b.color = 'red'
  4  union
  5  select s2.sname
  6  from sailors s2, boats b2, reserves r2
  7  where s2.sid = r2.sid and r2.bid = b2.bid and b2.color ='green';
```

Output:

```
SNAME
----------
dustin
horatio
lubber
```

Query:Using 'or'

```
SQL> select distinct s.sname
  2  from sailors s, reserves r, boats b
  3  where s.sid = r.sid and r.bid = b.bid and (b.color = 'red' or b.color = 'green');
```

Output:

```
SNAME
----------
lubber
dustin
horatio
```

Q2) UNION: Find all sids of sailors who have a rating of 10 or reserved boat number 1.

Query:

```
SQL> select s.sid
  2  from sailors s
  3  where s.rating = 10
  4  union
  5  select r.sid
  6  from reserves r
  7  where r.bid = 1;
```

Output:

```
     SID
----------
      58
      71
```

Q3) INTERSECT: Find the names of sailor's who have reserved both a red and a green boat.
Query:

Query:

```
SQL> select s.sname
  2  from sailors s, reserves r, boats b
  3  where s.sid = r.sid and r.bid = b.bid and b.color = 'red'
  4  intersect
  5  select s2.sname
  6  from sailors s2, boats b2, reserves r2
  7  where s2.sid = r2.sid and r2.bid = b2.bid and b2.color = 'green';
```

Output:

```
SNAME
----------
dustin
horatio
lubber
```

Q4)IN: Find the names of sailors who have reserved boat 103 using IN Operator.

Query:

```
SQL> select s.sname from sailors s where s.sid
  2  in (select r.sid from reserves r where r.bid = 103 );
```

Output:

```
SNAME
----------
dustin
lubber
horatio
```

Q5) NOT IN: Find the names of sailors who have not reserved boat 103 using NOT IN Operator.

Query:

```
SQL> select s.sname
  2  from sailors s
  3  where s.sid
  4  not in (select r.sid from reserves r where r.bid = 103 );
```

Output:
SNAME
---
brutus
andy
rusty
horatio
zorba
art
bob

7 rows selected.

Q6)EXISTS: Find the names of sailors who have reserved boat number 103 using EXISTS Operator.

Query:
```
SQL> select s.sname
  2  from sailors s
  3  where  exists (select * from reserves r where r.bid = 103 and r.sid = s.sid );
```

Output:
SNAME
---
dustin
lubber
horatio

Q7) NOT EXISTS: Find the names of sailors who have not reserved boat number 103 using NOT EXISTS Operator.

Query:
```
SQL> select s.sname
  2  from sailors s
  3   where not exists (select * from reserves r where r.bid = 103 and r.sid =s.sid);
```
Output:
SNAME
---
brutus
andy
rusty
horatio
zorba
art
bob

7 rows selected.

Q8) ANY: Find sailors whose rating is better than some sailor called 'BOB' using ANY Operator.

Query:

```
SQL> select s.sid
  2  from sailors s
  3  where s.rating > any (select s2.rating
  4  from sailors s2 where s2.sname = 'bob' );
```

Output:

```
     SID
----------
      58
      71
      74
      31
      32
      64
      22

7 rows selected.
```

Q9)ALL: Find the sailor's with the highest rating using ALL Operator.

Query:

```
SQL> select s.sid
  2  from sailors s
  3  where s.rating >= all ( select s2.rating from sailors s2 );
```

Output:

```
     SID
----------
      58
      71
```

Exercise - 3

3.Queries using Aggregate functions (COUNT,SUM, AVG, MAX and MIN), GROUP BY, HAVING and Creation and dropping of Views**.**

Aim: To write SQL Queries using Aggregate functions (COUNT,SUM, AVG, MAX and MIN), GROUP BY, HAVING and Creation and dropping of Views.

Create table product

SQL> create table product(pno int primary key ,pname varchar(30),price float,quantity int);

Table created.

Insert into product

SQL> insert into product values(1,'dairy milk',60,2);

1 row created.

SQL> insert into product values(2,'good day',25,4);

1 row created.

SQL> insert into product values(3,'boost',10,6);

1 row created.

SQL> insert into product values(4,'maggi',5,10);

1 row created.

SQL> insert into product values(5,'book',20,20);

1 row created.

Select from product

SQL> select * from product;

PNO	PNAME	PRICE	QUANTITY
1	dairy milk	60	2
2	good day	25	4
3	boost	10	6
4	maggi	5	10
5	book	20	20

Count function

```
SQL> select count(price) from product;

COUNT(PRICE)
------------
         5

SQL> select count(quantity) from product;

COUNT(QUANTITY)
---------------
           5
```

Sum function

```
SQL> select sum(price) from product;

SUM(PRICE)
----------
      120

SQL> select sum(quantity) from product;

SUM(QUANTITY)
-------------
         42
```

Avg function

```
SQL> select avg(price) from product;

AVG(PRICE)
----------
       24

SQL> select avg(quantity) from product;

AVG(QUANTITY)
-------------
         8.4
```

Max function

```
SQL> select max(price) from product;

MAX(PRICE)
```

```
----------
      60

SQL> select max(quantity) from product;

MAX(QUANTITY)
-------------
        20
```

Min function

```
SQL> select min(price) from product;

MIN(PRICE)
----------
       5

SQL> select min(quantity) from product;

MIN(QUANTITY)
-------------
         2
```

Group by

Create table employ(sid int,name varchar(20),dept varchar(10),sal float);

Table created

Select * from employ

Sid	name	dept	sal
1	ayisha	ece	6000
2	sindhu	it	50000
3	sai	it	80000
4	lalli	ece	8000

SQL>select dept,sum(sal) from employ group by dept;

Dept	sum(sal)
It	130000
Ece	14000

Having

SQL>select dept,sum(sal) from employ group by dept having sum(sal)>25000;

Dept sum(sal)
It 130000

Exercise - 4

4.Queries using Conversion functions (to_char,to_number and to_date),stringfunctions(Concatenation, lpad, rpad, ltrim, rtrim, lower,upper, initcap, length, substr and instr), datefunctions (Sysdate, next_day, add_months,last_day, months_between, least, greatest, trunc,round, to_char, to_date)

Aim: To write Queries using Conversion functions (to_char,to_number and to_date) in SQL

Conversion functions

1.to_char

Create table db(dob date)

Select to_char(dob,'month dd,year');

To_char(dob,'month dd,year')

May 21,nineteen ninety-six

2.to_number

SQL> select to_number('234.87') from dual;

TO_NUMBER('234.87')

234.87

3.to_date

SQL> select to_date('jan 21 1998','month dd,yy') from dual;

TO_DATE('

21-JAN-98

Aim: To write Queries using stringfunctions(Concatenation, lpad, rpad, ltrim, rtrim, lower,upper, initcap, length, substr and instr) in SQL

String functions

1.lower

```
SQL> select lower('SAI') from dual;

LOW
---
sai
```

2.upper

```
SQL> select upper('sai') from dual;

UPP
---
SAI
```

3.concat

```
SQL> select concat('hello','sai') from dual;

CONCAT('
--------
hellosai
```

4.initcap

```
SQL> select initcap('sai') from dual;

INI
---
Sai
```

5.length

```
SQL> select length('sai') from dual;

LENGTH('SAI')
-------------
           3
```

6.instr

```
SQL> select instr('ruhanika','a') from dual;

INSTR('RUHANIKA','A')
---------------------
                    4
```

7.substr

```
SQL> select substr('ruhanika',5) from dual;

SUBS
----
nika
```

8.lpad

```
SQL> select lpad('ruhanika',10,'****') from dual;

LPAD('RUHA
----------
**ruhanika
```

9.rpad

```
SQL> select rpad('ruhanika',10,'****') from dual;

RPAD('RUHA
----------
ruhanika**
```

10.ltrim

```
SQL> select ltrim('      ruhanika') from dual;

LTRIM('R
--------
ruhanika
```

11.rtrim

```
SQL> select rtrim('      ruhanika         ') from dual;
```

```
RTRIM('RUHANIK
--------------
    ruhanika
```

Aim: To write Queries using datefunctions (Sysdate, next_day, add_months,last_day, months_between, least, greatest, trunc,round, to_char, to_date) in SQL

Date functions

1.sysdate

```
SQL> select sysdate from dual;

SYSDATE
---------
20-NOV-15
```

2.last_day

```
SQL> select last_day(sysdate) from dual;

LAST_DAY(
---------
30-NOV-15
```

3.next_day

```
SQL> select next_day('20-nov-2015','friday') from dual;

NEXT_DAY(
---------
27-NOV-15
```

4.add_months

```
SQL> select add_months(sysdate,2) from dual;

ADD_MONTH
---------
20-JAN-16
```

5.months_between

```
SQL> select months_between('20-nov-2015','20-jan-2016') from dual;

MONTHS_BETWEEN('20-NOV-2015','20-JAN-2016')
-------------------------------------------
```

-2

6.least

```
SQL> select least(10,11,12) from dual;

LEAST(10,11,12)
---------------
          10

SQL> select least('s','f','a') from dual;

L
-
a
```

7.greatest

```
SQL> select greatest(10,11,12) from dual;

GREATEST(10,11,12)
------------------
            12

SQL> select greatest('s','f','a') from dual;

G
-
s
```

8.ground

```
SQL> select round(21.088) from dual;

ROUND(21.088)
-------------
         21

SQL> select trim(21.088) from dual;

TRIM(2
------
21.088
```

Exercise – 5.i

5.i) Creation of simple PL/SQL program which includes declaration section, executable section and exception –Handling section (Ex. Student marks can be selected from the table and printed for those who secured first class and an exception can be raised if no records were found)

Aim: To create a PL/SQL program

Create table student (sid, sname, sclass);

PL/SQL Program:

```
declare
        stu_id number;
        stu_name varchar(20);
        cursor stu_cur is
select sid,sname
from student
where sclass='first';
Begin
Open stu_cur;
Loop
fetch stu_cur into stu_id,stu_name;
exit when stu_cur%notfound;
dbms_output.put_line('student_id: '||  stu_id  || 'student_name:'|| stu_name);
end loop;
close stu_cur;
end;
/
```

Output:

```
student_id: 1student_name:abhi
student_id: 2student_name:sai
student_id: 5student_name:ish

PL/SQL procedure successfully completed.
```

5.ii) Insert data into student table and use COMMIT, ROLLBACK and SAVEPOINT in PL/SQL block.

Create table stu(name varchar(10),branch varchar(10));

Insert into stu values(‘sai’,’it’);

Savepoint h;

Savepoint created

```
SQL> set serveroutput on;
SQL> begin
        savepoint g;
        insert into stu values('ruhi','cse');
        exception
        when dup_val_on_index then
        rollback to g;
        commit;
        end;
        /
```

SQL>Rollback to h;

Rollback completed

6)Aim: Develop a program that includes the features Nested if,Case and case expression.The Program can be extended using the NULL if, and COALESCE functions.

(1)CASE:

Syntax:

```
select case("column_name")
when "value1" then "result1"
when "value2" then "result2"
....
[else "resultN"]
end
from "table-name";
```

program:

```
select store_name,case store_name
when 'Newyork' then sales*2
when 'chicago' then sales*3
else sales
end
"new sales"
txn_date
from store_information;
```

(2)SEARCHED CASE EXPRESSION:

program:

```
select store_name,txn_date,case
when sales>=8000 then 'congrats get a gift coupon'
when sales>=2000 then 'Thanks for shopping'
else 'Good day'
end
"sales status"
from store_information;
```

(3)COALESCE:

v

This returns the first non-NULL expression among its arguements.

syntax:

```
coalesce("expression1","expression2",...);
```

(4)NULL IF:

Takes two arguements.If the two arguments are equal,then NULL is returned.otherwise the first arguement is returned.

syntax:

```
select column_name,NULLIF(argument1,arguement2) from tabe_name;
```

(5)PROGRAM FOR CASE:

```
declare
grade char(1);
begin
grade:='a';
case
when grade='a' then
```

```
dbms_output.put_line('Excellent');
when grade='b' then
dbms_output.put_line('very good');
when grade='c' then
dbms_output.put_line('good');
when grade='d' then
dbms_output.put_line('fair');
when grade='f' then
dbms_output.put_line('poor');
else
dbms_output.put_line('No such grade');
end case;
end;
/
```

7) Program development using WHILE LOOPS, numeric FOR LOOPS, nested loops using ERROR Handling, BUILT –IN Exceptions, USE defined Exceptions, RAISE- APPLICATION ERROR.
PL/SQL

1) AIM: Addition at run time

```
Declare
a number;
b number;
c number;
Begin
a:=&a;
b:=&b;
c:=&c;
dbms_output.put_line('sum of' || a || 'and' || b || 'is' || c);
end;
/
```

output:

```
Enter value for a: 10
old   6: a:=&a;
new   6: a:=10;
Enter value for b: 10
old   7: b:=&b;
new   7: b:=10;
Enter value for c: 10
old   8: c:=&c;
new   8: c:=10;
```

sum of10and10is10

PL/SQL procedure successfully completed.

2) AIM: Simple loop to get sum of 100 numbers

```
Declare
a number;
s1 number default 0;
begin
a:=1;
loop
s1:=s1+a;
exit when(a=100);
a:=a+1;
end loop;
dbms_output.put_line('sum between 1 to 100 is'  || s1);
end;
/
```

output:

sum between 1 to 100 is5050

PL/SQL procedure successfully completed.

3) AIM: While Loop for sum of 100 odd numbers

Program:

```
SQL> declare
n number;
endvalue number;
sum1 number default 0;
begin
endvalue:=&endvalue;
n:=1;
while(n<endvalue)
loop
sum1:=sum1+n;
n:=n+2;
end loop;
```

```
dbms_output.put_line('sum of odd numbers between 1 and ' || endvalue || 'is' || sum1);
end;
/
```

Output:

```
Enter value for endvalue: 19
old   6: endvalue:=&endvalue;
new   6: endvalue:=19;
sum of odd numbers between 1 and 19 is 81

PL/SQL procedure successfully completed.
```

4) AIM: if else for finding maximum of three numbers

Program:

```
SQL> declare
a number;
b number;
c number;
begin
a:=&a;
b:=&b;
c:=&c;
if(a>b)and(a>c)then
dbms_output.put_line(' a is maximum');
elsif(b>a)and(b>c)then
dbms_output.put_line('b is maximum');
else
dbms_output.put_line('c is maximum');
end if;
end;
/
```

Output:

```
Enter value for a: 10
old   6: a:=&a;
new   6: a:=10;
Enter value for b: 4
```

```
old   7: b:=&b;
new   7: b:=4;
Enter value for c: 19
old   8: c:=&c;
new   8: c:=19;
c is maximum

PL/SQL procedure successfully completed.
```

5) AIM: select column from table employ by using memory variable

Program:

```
Declare
Mvsalary number(10,2);
Begin
Select salary into mvsalary
From
Employ
Where ename=’sai’;
Dbms_output.put_line(‘the salary of employ is’ || to_char9mvsalary));
End;
/
```

Output:

```
The salary of employ is 50000

PL/SQL procedure successfully completed.
```

8)AIM: Programs development using creation of procedures,passing paramneters IN and OUT of procedures.

```
->create table enquiry(enqno1 number,fname varchar2(30));
->insert into enquiry values(111,'sai');
->insert into enquiry values(112,'sindhu');
```

/*program*/

```
create procedure findname(enquiryno1 IN number,fname1 OUT varchar2)
is
fname2 varchar2(30);
```

```
begin
select fname into fname2
from enquiry
where enqno1=enquiryno1;
fname1:=fname2;
exception
when no_data-found then
raise_application_error(-20100,'The given number is not present');
end;
/
```

/*calling procedure*/

```
declare
enqno2 number(5);
fname2 varchar2(30);
begin
enqno2:=111;
findname(enqno2,fname2);
dbms_output.put_line(fname2);
end;
/
```

output: sai

9)AIM:Program development using creation of stored functions,invoke functions in SQL statements and write complex functions.

```
->create table dept(deptno int,dname varchar(10));
->insert into dept values(1219,'sai');
```

/*program*/

```
create or replace function getname(dno number)
return varchar2 as
fname1 varchar2(30);
begin
select dname
into fname1
from dept
where deptno=dno;
return(fname1);
exception
when no_data_found then
raise_application_error(-20100,'The dno is present');
end;
/
```

/*calling function*/

```
declare
fname2 varchar2(30);
deptno2 number(5);
begin
deptno:=1219;
fname2:=getname(dno);
dbms_output.put_line(fname2);
end;
/
```

output: sai

10) Develop programs using features parameters in a CURSOR, FOR UPDATE CURSOR, WHERE CURRENT of clause and CURSOR variables.

CURSORS

AIM: create an employ table and retrieve the name of employ whose salary is

Greater than 25000 by PL/SQL

```
Create table employ(eid,ename,salary);
```

Program:

```
declare
  emp_rec varchar(30);
  cursor emp_cur is
  select ename
  from employ where salary>25000;
  Begin
  Open emp_cur;
  Loop
  fetch emp_cur into emp_rec;
  exit when emp_rec%notfound;
  dbms_output.put_line(emp_rec);
  end loop;
  close emp_cur;
  end;
  /
```

Output:

Sindhu

Sai

Satya

PL/SQL procedure successfully completed.

11)AIM:Develop programs using before and after triggers,row and statement triggers and instead of triggers.

(1)Create a trigger

```
create or replace trigger trg2
after insert or delete or update
on dept1
for each row
when(new.deptno>0)
begin
dbms_output.put_line('trigger fired');
end;
/
```

(2)Insert

```
->insert into dept values('sindhu',30);
*trigger fired*
*1 row created*
```

(3)UPADTE

```
->udpate dept1 set deptno=19 where dname='sindhu';
*trigger fired*
*1 row updated*
```

(4)DELETE

```
->delete from dept where deptno=30;
*trigger fired*
*1 row deletd*
```