EXPERIMENT 1:

Aim: Write SQL queries to CREATE TABLES for various databases using DDL commands (i.e.

CREATE, ALTER, DROP, TRUNCATE).

```
SQL> set sqlprompt'cse504>';
cse504>create table student(stu_id number,stu_name varchar2(20));

Table created.

cse504>alter table student add st_city varchar2(30);

Table altered.

cse504>desc student;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STU_ID                                             NUMBER
 STU_NAME                                           VARCHAR2(20)
 ST_CITY                                            VARCHAR2(30)
```

```
se504>alter table student drop column st_city;

able altered.

se504>select * from student;

o rows selected

se504>drop table student;

able dropped.

se504>create table student (stu_id number primary key,stu_name varchar2(30));

able created.
```

EXPERIMENT 2: Write SQL queries to MANIPULATE TABLES for various databases using DML commands(i.e. INSERT, SELECT, UPDATE, DELETE,).

```
cse504>alter table student drop column st_city;

Table altered.

cse504>select * from student;

no rows selected

cse504>drop table student;

Table dropped.

cse504>create table student (stu_id number primary key,stu_name varchar2(30));

Table created.

cse504>insert into student values (&id,'&name');
Enter value for id: 504
Enter value for name: joshika
old   1: insert into student values (&id,'&name')
new   1: insert into student values (504,'joshika')

1 row created.

cse504>select * from student;

    STU_ID STU_NAME
---------- ------------------------------
       504 joshika
```

EXPERIMENT 3: Write SQL queries to create VIEWS for various databases (i.e. CREATE VIEW, UPDATE VIEW, ALTER VIEW, and DELETE VIEW).

```
cse504>update student set stu_name='koushik' where stu_id=504;

1 row updated.

cse504>select * from student;

    STU_ID STU_NAME
---------- ------------------------------
       504 koushik

cse504>select stu_id from student;

    STU_ID
----------
       504

cse504>CREATE TABLE dept(dept_name VARCHAR2(30),building VARCHAR2(10),budget NUMBER(12,2));

Table created.

cse504>CREATE VIEW dept_view AS SELECT dept_name,budget from dept;

View created.

cse504>select * from dept_view;

no rows selected

cse504>insert into dept values('ece','a-block',66000);

1 row created.

cse504>select * from dept_view;

DEPT_NAME                      BUDGET
------------------------------ ----------
ece                             66000

cse504>insert into dept values('eee','a-block',76000);
```

experiment 4: Write SQL queries to perform RELATIONAL SET OPERATIONS (i.e. UNION, UNION ALL, INTERSECT, MINUS, CROSS JOIN, NATURAL JOIN).

```
cse504>insert into dept values('csd','a-block',86000);

1 row created.

cse504>select * from dept_view;

DEPT_NAME                      BUDGET
------------------------------ ----------
ece                             66000
eee                             76000
csd                             86000

cse504>update dept_view set budget=budget*1.5;

3 rows updated.

cse504>select * from dept_view;

DEPT_NAME                      BUDGET
------------------------------ ----------
ece                             99000
eee                            114000
csd                            129000

cse504>delete from dept_view where dept_name='eee';

1 row deleted.

cse504>select * from dept_view;

DEPT_NAME                      BUDGET
------------------------------ ----------
ece                             99000
csd                            129000

cse504>DROP VIEW dept_view;

View dropped.
```

EXPERIMENT 5: Write SQL queries to perform AGGREGATE OPERATIONS (i.e. SUM, COUNT, AVG, MIN, MAX).

```
cse504>run
  1  CREATE TABLE employee(
  2  name VARCHAR(10),
  3  age NUMBER,
  4  subject VARCHAR(15),
  5  marks NUMBER
  6* )

Table created.

cse504>insert into employee values('chaithu',13,'maths',30);

1 row created.

cse504>insert into employee values('roopa',14,'physics',40);

1 row created.

cse504>insert into employee values('rola',15,'civics',50);

1 row created.

cse504>insert into employee values('joshna',16,'es',60);

1 row created.

cse504>insert into employee values('joshika',17,'se',70);

1 row created.

cse504>select * from employee;

NAME              AGE SUBJECT              MARKS
---------- ---------- --------------- ----------
chaithu            13 maths                   30
roopa              14 physics                 40
rola               15 civics                  50
joshna             16 es                      60
joshika            17 se                      70
```

SUM:

```
cse504>run
  1* select sum(marks) from employee

SUM(MARKS)
----------
       250
```

AVG:

```
cse504>select avg(marks) from employee;

AVG(MARKS)
----------
        50
```

MIN:

```
cse504>select min(marks) from employee;

MIN(MARKS)
----------
        30
```

MAX:

```
cse504>select max(marks) from employee;

MAX(MARKS)
----------
        70
```

COUNT:

```
cse504>select count(marks) from employee;

COUNT(MARKS)
------------
           5
```

EXPERIMENT 6: Write SQL queries to perform JOIN OPERATIONS (i.e. CONDITIONAL JOIN, EQUI JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN)

```
cse504>run
  1  CREATE TABLE employee(
  2  name VARCHAR(10),
  3  age NUMBER,
  4  subject VARCHAR(15),
  5  marks NUMBER
  6* )

Table created.

cse504>insert into employee values('chaithu',13,'maths',30);

1 row created.

cse504>insert into employee values('roopa',14,'physics',40);

1 row created.

cse504>insert into employee values('rola',15,'civics',50);

1 row created.

cse504>insert into employee values('joshna',16,'es',60);

1 row created.

cse504>insert into employee values('joshika',17,'se',70);

1 row created.

cse504>select * from employee;

NAME              AGE SUBJECT              MARKS
---------- ---------- --------------- ----------
chaithu            13 maths                   30
roopa              14 physics                 40
rola               15 civics                  50
joshna             16 es                      60
joshika            17 se                      70
```

```
cse504>insert into EMP values('joshna',16,'CSE');

1 row created.

cse504>insert into EMP values('joshika',17,'ECEE');

1 row created.

cse504>insert into EMP values('jyshnavi',18,'ECE');

1 row created.

cse504>select * from student;

no rows selected

cse504>select * from EMP;

NAME          ROLL_NO DEPT
---------- ---------- ----------
joshna             16 CSE
joshika            17 ECEE
jyshnavi           18 ECE

cse504>create table library(
  2  roll_no NUMBER,
  3  book VARCHAR(10)
  4  );

Table created.
```

```
cse504>insert into library values(17,'ECE');

1 row created.

cse504>insert into library values(18,'cse');

1 row created.

cse504>insert into library values(19,'csm');

1 row created.

cse504>select * from library;

   ROLL_NO BOOK
---------- ----------
        17 ECE
        18 cse
        19 csm
```

JOIN:

```
cse504>select * from emp JOIN library on emp.roll_no = library.roll_no;

NAME          ROLL_NO DEPT          ROLL_NO BOOK
---------- ---------- ---------- ---------- ----------
joshika           17 ECEE               17 ECE
jyshnavi          18 ECE                18 cse
```

```
cse504>select * from emp JOIN library using (roll_no);

   ROLL_NO NAME       DEPT       BOOK
---------- ---------- ---------- ----------
        17 joshika    ECEE       ECE
        18 jyshnavi   ECE        cse
```

LEFT OUTER JOIN:

```
cse504>select * from emp NATURAL LEFT OUTER JOIN LIBRARY;

   ROLL_NO NAME       DEPT       BOOK
---------- ---------- ---------- ----------
        17 joshika    ECEE       ECE
        18 jyshnavi   ECE        cse
        16 joshna     CSE
```

RIGHT OUTER JOIN:

```
cse504>select * from emp NATURAL RIGHT OUTER JOIN LIBRARY;

   ROLL_NO NAME       DEPT       BOOK
---------- ---------- ---------- ----------
        17 joshika    ECEE       ECE
        18 jyshnavi   ECE        cse
        19                       csm
```

NATURAL FULL OUTER JOIN:

```
cse504>select * from emp NATURAL FULL OUTER JOIN LIBRARY;

   ROLL_NO NAME       DEPT       BOOK
---------- ---------- ---------- ----------
        17 joshika    ECEE       ECE
        18 jyshnavi   ECE        cse
        19                       csm
        16 joshna     CSE

cse504>
```

EXPERIMENT 7: Write SQL queries to perform SPECIAL OPERATIONS (i.e. ISNULL, BETWEEN, LIKE, IN,

EXISTS).

```
cse504>create table students_in(
  2  name varchar2(10) not null,
  3  r_no varchar2(15) not null,
  4  branch varchar2(15) not null,
  5  block varchar2(5) not null,
  6  fee number not null,
  7  primary key(name))
  8  /

Table created.

cse504>insert into students_in values('jagadeesh',101,'cse','B',250000);

1 row created.

cse504>insert into students_in values('jagan',102,'cse','B',350000);

1 row created.

cse504>insert into students_in values('jayanth',103,'cse','A',350000);

1 row created.

cse504>insert into students_in values('jogi',104,'cse','A',550000);

1 row created.

cse504>insert into students_in values('jignu',105,'cse','B,450000);
ERROR:
ORA-01756: quoted string not properly terminated


cse504>insert into students_in values('jignu',105,'cse','B',450000);

1 row created.

cse504>insert into students_in values('junaid',107,'cse',' ',850000);

1 row created.

cse504>insert into students_in values('jeevan',108,'cse',' ',950000);

1 row created.
```

```
cse504>select * from students_in;

NAME       R_NO            BRANCH          BLOCK        FEE
---------- --------------- --------------- ----- ----------
jagadeesh  101             cse             B         250000
jagan      102             cse             B         350000
jayanth    103             cse             A         350000
jogi       104             cse             A         550000
jignu      105             cse             B         450000
junaid     107             cse                       850000
jeevan     108             cse                       950000

7 rows selected.

cse504>select * from students_in WHERE branch is null;

no rows selected
```

IS NULL:

```
cse504>select * from students_in WHERE branch is null;

no rows selected
```

NOT NULL:

```
cse504>select * from students_in WHERE branch is not null;

NAME       R_NO            BRANCH          BLOCK        FEE
---------- --------------- --------------- ----- ----------
jagadeesh  101             cse             B         250000
jagan      102             cse             B         350000
jayanth    103             cse             A         350000
jogi       104             cse             A         550000
jignu      105             cse             B         450000
junaid     107             cse                       850000
jeevan     108             cse                       950000

7 rows selected.
```

BETWEEN:

```
cse504>select * from students_in WHERE fee BETWEEN 2000 AND 10000000;

NAME       R_NO            BRANCH          BLOCK        FEE
---------- --------------- --------------- ----- ----------
jagadeesh  101             cse             B         250000
jagan      102             cse             B         350000
jayanth    103             cse             A         350000
jogi       104             cse             A         550000
jignu      105             cse             B         450000
junaid     107             cse                       850000
jeevan     108             cse                       950000

7 rows selected.
```

LIKE:

```
cse504>select * from students_in WHERE branch LIKE 'cse%';

NAME       R_NO            BRANCH          BLOCK        FEE
---------- --------------- --------------- ----- ----------
jagadeesh  101             cse             B         250000
jagan      102             cse             B         350000
jayanth    103             cse             A         350000
jogi       104             cse             A         550000
jignu      105             cse             B         450000
junaid     107             cse                       850000
jeevan     108             cse                       950000

7 rows selected.

cse504>select * from students_in WHERE block LIKE 'b%';

no rows selected

cse504>select * from students_in WHERE block LIKE 'A%';

NAME       R_NO            BRANCH          BLOCK        FEE
---------- --------------- --------------- ----- ----------
jayanth    103             cse             A         350000
jogi       104             cse             A         550000

cse504>select * from students_in WHERE EXISTS (select name from students_in);

NAME       R_NO            BRANCH          BLOCK        FEE
---------- --------------- --------------- ----- ----------
jagadeesh  101             cse             B         250000
jagan      102             cse             B         350000
jayanth    103             cse             A         350000
jogi       104             cse             A         550000
jignu      105             cse             B         450000
junaid     107             cse                       850000
jeevan     108             cse                       950000

7 rows selected.
```

EXPERIMENT 8: Write SQL queries to perform ORACLE BUILT-IN FUNCTIONS (i.e. DATE, TIME).

```
cse504>create table names(
  2  first_name varchar(20) not null,
  3  second_name varchar(20) not null,
  4  );
)
*
ERROR at line 4:
ORA-00904: : invalid identifier


cse504>ed
Wrote file afiedt.buf

  1  create table names(
  2  first_name varchar(20) not null,
  3  last_name varchar(20) not null
  4* )
cse504>run
  1  create table names(
  2  first_name varchar(20) not null,
  3  last_name varchar(20) not null
  4* )

Table created.

cse504>insert all
  2  into names values ('chaithu','agraharam')
  3  into names values ('joshika','tadimarri')
  4  into names values ('manjula','talari')
  5  into names values ('jaswanth','santimalla')
  6  select * from dual;

4 rows created.
```

Lower,initcap,concat,upper:

```
cse504>select lower (first_name) from names;

LOWER(FIRST_NAME)
--------------------
chaithu
joshika
manjula
jaswanth

cse504>select upper(first_name) from names;

UPPER(FIRST_NAME)
--------------------
CHAITHU
JOSHIKA
MANJULA
JASWANTH

cse504>select initcap(first_name) from names;

INITCAP(FIRST_NAME)
--------------------
Chaithu
Joshika
Manjula
Jaswanth

cse504>select concat(first_name,last_name) from names;

CONCAT(FIRST_NAME,LAST_NAME)
------------------------------------
chaithuagraharam
joshikatadimarri
manjulatalari
jaswanthsantimalla
```

Substring:

```
cse504>select substr(first_name,1,14) from names;

SUBSTR(FIRST_NAME,1,14)
------------------------------------------------------
chaithu
joshika
manjula
jaswanth

cse504>select length(first_name) from names;

LENGTH(FIRST_NAME)
------------------
                 7
                 7
                 7
                 8
```

Insert,trim:

```
cse504>select instr(first_name,'Jo') from names;

INSTR(FIRST_NAME,'JO')
----------------------
                     0
                     0
                     0
                     0

cse504>select TRIM( 'A' from first_name) from names;

TRIM('A'FROMFIRST_NA
--------------------
chaithu
joshika
manjula
jaswanth
```

Round ,date, month,nextday,mod:

```
cse504>select ROUND(11.111,2) from dual;

ROUND(11.111,2)
---------------
          11.11

cse504>select MOD(11,2)
  2  FROM dual;

 MOD(11,2)
----------
         1

cse504>select sysdate from dual;

SYSDATE
---------
08-JAN-24

cse504>select months_between(sysdate,'19-dec-2024')from dual;

MONTHS_BETWEEN(SYSDATE,'19-DEC-2024')
-------------------------------------
                            -11.33541

cse504>select add_months(sysdate,19)from dual;

ADD_MONTH
---------
08-AUG-25

cse504>select next_day(sysdate,monday)from dual;
select next_day(sysdate,monday)from dual
                        *
ERROR at line 1:
ORA-00904: "MONDAY": invalid identifier


cse504>
cse504>select next_day(sysdate,'monday')from dual;

NEXT_DAY(
---------
15-JAN-24
```

Last day,timestamp:

```
cse504>select last_day(sysdate)from dual;

LAST_DAY(
---------
31-JAN-24

cse504>select current_timestamp(3) from dual;

CURRENT_TIMESTAMP(3)
---------------------------------------------------------------------------
08-JAN-24 02.29.17.295 PM +05:30
```

EXPERIMENT 9: Write SQL queries to perform KEY CONSTRAINTS (i.e. PRIMARY KEY, FOREIGN KEY, UNIQUE NOT NULL, CHECK, DEFAULT).

```
cse504>create table order1(
  2  id number primary key,
  3  product_name varchar2(50) not null,
  4  quantity number
  5  );

Table created.

cse504>insert into order1 values(4,' ',13);

1 row created.

cse504>create table emp1(
  2  id number primary key,
  3  name varchar2(50) not null,
  4  e_mail varchar2(50) unique
  5  );

Table created.

cse504>insert into emp1 values(501,'joshika','joshikanarayanaswamy@gmail.com');
```

```
cse504>create table parts(
  2  part_id number primary key,
  3  part_name varchar2(40) not null,
  4  buy_price number(9,2) check(buy_price>0)
  5  );

Table created.

cse504>insert into parts values(101,'agarbathi',897);

1 row created.

cse504>create table parts12(
  2  part_name varchar2(40) not null,
  3  id number primary key,
  4  country varchar2(20) default 'ind'
  5  );

Table created.

cse504>insert into parts12 values('arjun',101,'usa');

1 row created.

cse504>insert into parts12 values('aravindh',101,'uk');
insert into parts12 values('aravindh',101,'uk')
*
ERROR at line 1:
ORA-00001: unique constraint (LE504.SYS_C008253) violated


cse504>select * from parts12;

PART_NAME                                          ID COUNTRY
---------------------------------------- ---------- --------------------
arjun                                             101 usa
```

EXPERIMENT 10: Write a PL/SQL program for calculating the factorial of a given number.

```
SQL> run
  1  declare
  2  fac number :=1;
  3  n number :=4;
  4  begin
  5  while n>0 loop
  6  fac:=n*fac;
  7  n:=n-1;
  8  end loop;
  9  DBMS_OUTPUT.PUT_LINE(fac);
 10* end;

PL/SQL procedure successfully completed.
```

EXPERIMENT 11: Write a PL/SQL program for finding the given number is prime number or not.

```
cse le504>RUN
  1  DECLARE
  2  n NUMBER;
  3  i NUMBER;
  4  temp NUMBER;
  5  BEGIN
  6  n := 13;
  7  i := 2;
  8  temp := 1;
  9  FOR i IN 2..n/2
 10  LOOP
 11  IF MOD(n, i) = 0
 12  THEN
 13  temp := 0;
 14  EXIT;
 15  END IF;
 16  END LOOP;
 17  IF temp = 1
 18  THEN
 19  DBMS_OUTPUT.PUT_LINE(n||' is a prime number');
 20  ELSE
 21  DBMS_OUTPUT.PUT_LINE(n|| ' is not a prime number');
 22  END IF;
 23* END;

PL/SQL procedure successfully completed.
```

EXPERIMENT 12: Write a PL/SQL program for displaying the Fibonacci series up to an integer.

```
cse le504>RUN
  1  DECLARE
  2  FIRST NUMBER := 0;
  3  SECOND NUMBER := 1;
  4  TEMP NUMBER;
  5  N NUMBER := 5;
  6  I NUMBER;
  7  BEGIN
  8  DBMS_OUTPUT.PUT_LINE('SERIES:');
  9  DBMS_OUTPUT.PUT_LINE(FIRST);
 10  DBMS_OUTPUT.PUT_LINE(SECOND);
 11  FOR I IN 2..N
 12  LOOP
 13  TEMP:=FIRST+SECOND;
 14  FIRST := SECOND;
 15  SECOND := TEMP;
 16  DBMS_OUTPUT.PUT_LINE(TEMP);
 17  END LOOP;
 18* END;

PL/SQL procedure successfully completed.
```

EXPERIMENT 13: Write PL/SQL program to implement Stored Procedure on table.

```
se504>create table sailor1(
 2  id number primary key,
 3  name varchar2(30) not null
 4  );
```

```
  6* end;
cse504>run
  1  create or replace procedure insertuser(id IN NUMBER,name IN VARCHAR2)
  2  as
  3  begin
  4  insert into sailor1 values(id,name);
  5  DBMS_OUTPUT.PUT_LINE('record inserted succesfully');
  6* end;

Procedure created.

cse504>declare
  2  co number;
  3  begin
  4  insertuser(11,'rani');
  5  select count(*) into co from sailor1;
  6  DBMS_OUTPUT.PUT_LINE(co||' record is inserted succesfully');
  7  end;
  8  /

PL/SQL procedure successfully completed.

cse504>drop procedure insertuser;

Procedure dropped.
```

EXPERIMENT 14: Write PL/SQL program to implement Stored Procedure on table.

```
cse504>create table section(
  2  id number primary key,
  3  course_name varchar2(20) not null,
  4  strength number not null
  5  );

Table created.

cse504>insert all
  2  into section values(1,'cse',50)
  3  into section values(2,'csm',60)
  4  into section values(4,'csd',70)
  5  select * from dual;

3 rows created.

cse504>set server out on
SP2-0158: unknown SET option "server"
cse504>set verify off
cse504>create or replace function totalstrength return number
  2  as
  3  total number:=0;
  4  begin
  5  select sum(strength)into total from section;
  6  return total;
  7  end;
  8  /

Function created.
```

```
cse504>run
  1  declare
  2  answer number;
  3  begin
  4  answer:=totalstrength();
  5  DBMS_OUTPUT.PUT_LINE('Total strength of students is '||answer);
  6* end;

PL/SQL procedure successfully completed.
```

EXPERIMENT15: Write PL/SQL program to implement Trigger on table.

```
le504>r
  1  DECLARE
  2  c_id customers.id%type;
  3  c_name customers.name%type;
  4  c_age customers.age%type;
  5  CURSOR c_customers IS
  6  SELECT id,name,age FROM customers;
  7  BEGIN
  8  OPEN c_customers;
  9  LOOP
 10  FETCH c_customers INTO c_id,c_name,c_age;
 11  EXIT WHEN c_customers%notfound;
 12  DBMS_OUTPUT.PUT_LINE(c_id||' '||c_name||' '||c_age);
 13  END LOOP;
 14  CLOSE c_customers;
 15* END;

PL/SQL procedure successfully completed.
```

```
le504>RUN
  1  DECLARE
  2  tot_rows NUMBER;
  3  BEGIN
  4  UPDATE customers SET salary=salary*1.5;
  5  IF sql%notfound THEN
  6  DBMS_OUTPUT.PUT_LINE('No customers updated');
  7  ELSIF sql%found THEN
  8  tot_rows := sql%rowcount;
  9  DBMS_OUTPUT.PUT_LINE(tot_rows||' customers updated');
 10  END IF;
 11* END;

PL/SQL procedure successfully completed.

le504>INSERT ALL
  2  INTO customers VALUES (101,'ram',21,60000)
  3  INTO customers VALUES (102,'ramu',22,65000)
  4  INTO customers VALUES (103,'ramesh',23,70000)
  5  INTO customers VALUES (104,'rajesh',24,75000)
  6  SELECT * FROM dual;

4 rows created.
```

```
  2  BEFORE UPDATE ON instructor
  3  FOR EACH ROW
  4  WHEN (NEW.ID = OLD.ID)
  5  DECLARE
  6  sal_diff number;
  7  BEGIN
  8  sal_diff := :NEW.salary - :OLD.salary;
  9  dbms_output.put_line('Old salary: ' || :OLD.salary);
 10  dbms_output.put_line('New salary: ' || :NEW.salary);
 11  dbms_output.put_line('salary difference: ' || sal_diff);
 12  END;
 13  /

Trigger created.
```

EXPERIMENT 16: Write PL/SQL program to implement Cursor on table.

```
cse504>run
  1  declare
  2  tot_rows NUMBER;
  3  BEGIN
  4  UPDATE customers set salary=salary*1.5;
  5  if sql%notfound then
  6  DBMS_OUTPUT.PUT_LINE('no customers updated');
  7  elsif sql%notfound then
  8  tot_rows := sql%rowcount;
  9  DBMS_OUTPUT.PUT_LINE(tot_rows||' customers updated');
 10  end if;
 11* end;

PL/SQL procedure successfully completed.

cse504>insert all
  2  into customers values (501,'ram',22,600000)
  3  into customers values (502,'ramu',23,650000)
  4  into customers values (503,'ramesh',24,700000)
  5  select * from dual;

3 rows created.
```

```
cse504>run
  1  declare
  2  c_id customers.id%type;
  3  c_name customers.name%type;
  4  c_age customers.age%type;
  5  cursor c_customers is
  6  select id,name,age from customers;
  7  begin
  8  open c_customers;
  9  loop
 10  fetch c_customers into c_id,c_name,c_age;
 11  exit when c_customers%notfound;
 12  dbms_output.put_line(c_id||' '||c_name||' '||c_age);
 13  end loop;
 14  close c_customers;
 15* end;

PL/SQL procedure successfully completed.
```