Query 2

```
create table  students (roll_no int primary key, name varchar(20), department
varchar(20));

insert into students values (1, 'Ashwin', 'CSE'), (2, 'Aarav', 'CSE'),
(3, 'Anadhu', 'CSE');

select * from students;

alter table students modify column roll_no varchar(20);

desc students;

alter table students add(mark int);

update students set mark = 50 where roll_no < 4;

select * from students;
```

Output

```
+---------+--------+------------+
| roll_no | name   | department |
+---------+--------+------------+
| 1       | Ashwin | CSE        |
| 2       | Aarav  | CSE        |
| 3       | Anadhu | CSE        |
+---------+--------+------------+

+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| roll_no    | varchar(20) | NO   | PRI | NULL    |       |
| name       | varchar(20) | YES  |     | NULL    |       |
| department | varchar(20) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+

+---------+--------+------------+------+
| roll_no | name   | department | mark |
+---------+--------+------------+------+
| 1       | Ashwin | CSE        |   50 |
| 2       | Aarav  | CSE        |   50 |
| 3       | Anadhu | CSE        |   50 |
+---------+--------+------------+------+
```

Query 3 (CONSTRAINTS)

```
create table department (d_id int primary key, d_name varchar(20) not null);

insert into department values(1, 'CSE'), (2, 'ECE') , (3, 'EEE');

select * from department;

create table employee (e_id int primary key, e_name varchar(20) not null,
e_desig varchar(20), e_dept int, foreign key(e_dept) references
department(d_id));

insert into employee values(1, 'Vinod', 'HOD', '1'), (2, 'Rajesh', 'HOD', 2),
(3, 'Sreeja', 'HOD', 3);

select * from employee;
```

Output

```
+------+--------+
| d_id | d_name |
+------+--------+
|    1 | CSE    |
|    2 | ECE    |
|    3 | EEE    |
+------+--------+

+------+--------+---------+--------+
| e_id | e_name | e_desig | e_dept |
+------+--------+---------+--------+
|    1 | Vinod  | HOD     |      1 |
|    2 | Rajesh | HOD     |      2 |
|    3 | Sreeja | HOD     |      3 |
+------+--------+---------+--------+
```

Query 4 (DDL COMMANDS)

```
create table student (roll_no int primary key, name varchar(20), d_name
varchar(20));

insert into student values(1, 'Aarav', 'CSE'), (2, 'Ashwin', 'CSE'), (3,
'Anathan', 'CSE');

select * from student;

alter table student rename column roll_no to reg_no;

select * from student;

alter table student modify column reg_no varchar(20);

desc student;

alter table student add(mark int);

update student set mark = 80 where reg_no < 4;

create table department (name varchar(20), hod varchar(20));

insert into department values('CSE', 'Vinod'), ('ECE', 'Rajesh'),
('EEE', 'Sreeja');

select * from department;

truncate department;

select * from department;

drop table department;

select * from department;
```

Output

```
+--------+---------+--------+
| roll_no | name    | d_name |
+--------+---------+--------+
|      1 | Aarav   | CSE    |
|      2 | Ashwin  | CSE    |
|      3 | Anathan | CSE    |
+--------+---------+--------+
+--------+---------+--------+
| reg_no | name    | d_name |
+--------+---------+--------+
|      1 | Aarav   | CSE    |
|      2 | Ashwin  | CSE    |
|      3 | Anathan | CSE    |
+--------+---------+--------+

+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| reg_no | varchar(20) | NO   | PRI | NULL    |       |
| name   | varchar(20) | YES  |     | NULL    |       |
| d_name | varchar(20) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+


+--------+---------+--------+------+
| reg_no | name    | d_name | mark |
+--------+---------+--------+------+
| 1      | Aarav   | CSE    |   80 |
| 2      | Ashwin  | CSE    |   80 |
| 3      | Anathan | CSE    |   80 |
+--------+---------+--------+------+

+------+--------+
| name | hod    |
+------+--------+
| CSE  | Vinod  |
| ECE  | Rajesh |
| EEE  | Sreeja |
+------+--------+
```

After truncation: `Empty set (0.00 sec)`

After dropping the table: `Table 'lab.department' doesn't exist`

Query 5 (DML COMMANDS)

```
create table employee (e_id int primary key, name varchar(20), salary float(10,
2));

insert into employee values(1, 'Ashwin', 100000), (2, 'Aarav', 100000), (3,
'Anantha', 50000);

select * from employee;

update employee set salary = 100000 where e_id = 3;

select * from employee;

delete from employee where name = ‘Aarav’;

select * from employee;
```

Output

```
+------+---------+-----------+
| e_id | name    | salary    |
+------+---------+-----------+
|    1 | Ashwin  | 100000.00 |
|    2 | Aarav   | 100000.00 |
|    3 | Anantha |  50000.00 |
+------+---------+-----------+

+------+---------+-----------+
| e_id | name    | salary    |
+------+---------+-----------+
|    1 | Ashwin  | 100000.00 |
|    2 | Aarav   | 100000.00 |
|    3 | Anantha | 100000.00 |
+------+---------+-----------+

+------+---------+-----------+
| e_id | name    | salary    |
+------+---------+-----------+
|    1 | Ashwin  | 100000.00 |
|    3 | Anantha | 100000.00 |
+------+---------+-----------+
```

Query 6 (BUILT IN AGGREGATE FUNCTIONS)

```
create table student (roll_no int primary key, f_name varchar(20), l_name
varchar(20), mark float(4, 2), fee float(7,2));

insert into student values (1, 'Aarav', 'R', 99, 35000), (2, 'Manas', 'Manoj',
80, 75000), (3, 'Ashwin', 'Wilson', 90, 35000);

select * from student;

select count(*) from student;

select min(mark) from student;

select max(mark) from student;

select f_name, mark from student where mark = (select min(mark) from student) or
mark = (select max(mark) from student);

select f_name as first_rank from student where mark = (select max(mark) from
student);

select sum(fee) from student;

select date_format(now(), '%M %D %Y');
```

Output

```
+---------+--------+--------+-------+----------+
| roll_no | f_name | l_name | mark  | fee      |
+---------+--------+--------+-------+----------+
|       1 | Aarav  | R      | 80.00 | 35000.00 |
|       2 | Manas  | Manoj  | 80.00 | 75000.00 |
|       3 | Ashwin | Wilson | 80.00 | 35000.00 |
+---------+--------+--------+-------+----------+

+----------+
| count(*) |
+----------+
|        3 |
+----------+

+-----------+
| min(mark) |
+-----------+
|     80.00 |
+-----------+
+-----------+
| max(mark) |
+-----------+
|     99.00 |
+-----------+

+--------+-------+
| f_name | mark  |
+--------+-------+
| Aarav  | 99.00 |
| Manas  | 80.00 |
+--------+-------+

+------------+
| first_rank |
+------------+
| Aarav      |
+------------+

+-----------+
| sum(fee)  |
+-----------+
| 145000.00 |
+-----------+

+--------------------------------+
| date_format(now(), '%M %D %Y') |
+--------------------------------+
| September 17th 2024            |
+--------------------------------+
```

Query 7 (ORDER BY, GROUP BY, HAVING CLAUSES)

```
create table cd (s_no int primary key, state varchar(20), year int, month
varchar(20), no_of_infections int, death int);

insert into cd values(1, 'Kerala', 2004, 'July', 50, 12), (2, 'Goa', 2005,
'June', 58, 8), (3, 'Bihar', 2008, 'March', 51, 6), (4, 'UP', 2008, 'May', 25,
7);

select * from cd;

select avg(death) from cd where year = 2008;

select state, death from cd where death > 10;

select state, max(death) from cd where year = 2004  group by state having
max(death) > 10;

select * from cd order by state desc;
```

Output

```
+------+--------+------+-------+------------------+-------+
| s_no | state  | year | month | no_of_infections | death |
+------+--------+------+-------+------------------+-------+
|    1 | Kerala | 2004 | July  |               50 |    12 |
|    2 | Goa    | 2005 | June  |               58 |     8 |
|    3 | Bihar  | 2008 | March |               51 |     6 |
|    4 | UP     | 2008 | May   |               25 |     7 |
+------+--------+------+-------+------------------+-------+

+------------+
| avg(death) |
+------------+
|     6.5000 |
+------------+

+--------+-------+
| state  | death |
+--------+-------+
| Kerala |    12 |
+--------+-------+

+--------+------------+
| state  | max(death) |
+--------+------------+
| Kerala |         12 |
+--------+------------+
+------+--------+------+-------+-----------------+-------+
| s_no | state  | year | month | no_of_infections | death |
+------+--------+------+-------+-----------------+-------+
|    4 | UP     | 2008 | May   |              25 |     7 |
|    1 | Kerala | 2004 | July  |              50 |    12 |
|    2 | Goa    | 2005 | June  |              58 |     8 |
|    3 | Bihar  | 2008 | March |              51 |     6 |
+------+--------+------+-------+-----------------+-------+
```

Query 8 (SET OPERATORS AND NESTED QUERIES)

```
create table arts (id int primary key, name varchar(20), event varchar(20),
grade varchar(20));

mysql> insert into arts values (1, 'Aarav', 'Drawing', 'A'), (2, 'Ashwin',
'Drawing', 'A+'), (3, 'Manas', 'Story Telling', 'A+');

select * from arts;

create table sports (id int primary key, name varchar(20), item varchar(20),
grade varchar(20));

insert into sports values (1, 'Aarav', 'Cricket', 'C+'), (3, 'Manas',
'Football', 'A+'), (4, 'Gokul', 'Badminton', 'A+');

select * from sports;

select id, name  from arts union select id, name  from sports;

select id, name from arts intersect select id, name from sports;

select id, name  from sports except select id, name from arts;

create table project (id int primary key, name varchar(20), p_item varchar(20),
expense float(7, 2));

insert into project values (1, 'Aarav', 'Ai art generator', 10000),(2, 'Ashwin',
'Drone', 90000),(3, 'Anantha', 'Ai assistant', 50000);

select * from project;

select * from project where expense = (select max(expense) from project);
```

Output

```
+----+--------+---------------+-------+
| id | name   | event         | grade |
+----+--------+---------------+-------+
|  1 | Aarav  | Drawing       | A     |
|  2 | Ashwin | Drawing       | A+    |
|  3 | Manas  | Story Telling | A+    |
+----+--------+---------------+-------+

+----+-------+-----------+-------+
| id | name  | item      | grade |
+----+-------+-----------+-------+
|  1 | Aarav | Cricket   | C+    |
|  3 | Manas | Football  | A+    |
|  4 | Gokul | Badminton | A+    |
+----+-------+-----------+-------+
+----+--------+
| id | name   |
+----+--------+
|  1 | Aarav  |
|  2 | Ashwin |
|  3 | Manas  |
|  4 | Gokul  |
+----+--------+

+----+-------+
| id | name  |
+----+-------+
|  1 | Aarav |
|  3 | Manas |
+----+-------+

+----+-------+
| id | name  |
+----+-------+
|  4 | Gokul |
+----+-------+

+----+---------+------------------+----------+
| id | name    | p_item           | expense  |
+----+---------+------------------+----------+
|  1 | Aarav   | Ai art generator | 10000.00 |
|  2 | Ashwin  | Drone            | 90000.00 |
|  3 | Anantha | Ai assistant     | 50000.00 |
+----+---------+------------------+----------+

+----+--------+--------+----------+
| id | name   | p_item | expense  |
+----+--------+--------+----------+
|  2 | Ashwin | Drone  | 90000.00 |
+----+--------+--------+----------+
```

Query 9 (VIEWS)

```
create table shop (id int primary key, item varchar(20), price float(7, 2),
quantity int, discount int);

insert into shop values(1, 'Pen', 10, 20, 1), (2, 'Pencil', 5, 100, 2), (3,
'Paper', 20, 50, 3), (4, 'Eraser', 3, 0, 1);

select * from shop;

create view item_price as select  item, price from shop;

select * from item_price;

create view quantity_not_zero as select item, quantity from shop where quantity
!= 0;

select * from quantity_not_zero;

create view discount as select item, price, discount from shop where discount >
3;

select * from discount;

drop view discount;
```

Output

```
+----+--------+-------+----------+----------+
| id | item   | price | quantity | discount |
+----+--------+-------+----------+----------+
|  1 | Pen    | 10.00 |       20 |        1 |
|  2 | Pencil |  5.00 |      100 |        2 |
|  3 | Paper  | 20.00 |       50 |        3 |
|  4 | Eraser |  3.00 |        0 |        1 |
+----+--------+-------+----------+----------+

+--------+-------+
| item   | price |
+--------+-------+
| Pen    | 10.00 |
| Pencil |  5.00 |
| Paper  | 20.00 |
| Eraser |  3.00 |
+--------+-------+
+--------+----------+
| item   | quantity |
+--------+----------+
| Pen    |       20 |
| Pencil |      100 |
| Paper  |       50 |
+--------+----------+
```

```
+-------+-------+----------+
| item  | price | discount |
+-------+-------+----------+
| Paper | 20.00 |        3 |
+-------+-------+----------+
```

Query 10 (JOIN QUERY)

```
create table customer (c_id int primary key, name varchar(20), phone long,
address varchar(20));

insert into customer values (1, 'Ashwin', 123456789, 'Padipurackal'), (2,
'Aarav', 123456789, 'Ranjith bhavan'), (3, 'Anandhu', 123456789, 'Anjana
bhavan');

select * from customer;

create table account (a_no int primary key, name varchar(20), bank_code int,
a_type varchar(20), balance float(10, 2));

insert into account values(1, 'Aarav', 10, 'Recurring', 10000), (2, 'Ashwin',
10, 'Savings', 90000), (3, 'Anandhu', 10, 'Current', 50000);

select * from account;

select c_id, customer.name, address, a_no from customer inner join account on
customer.name = account.name;

create table loan (l_id int primary key, name varchar(20), l_type varchar(20),
l_amount float(10,2));

insert into loan values (1, 'Ashwin', 'Personal', 20000), (2, 'Aarav', 'gold',
5000), (3, 'Anandhu', 'Home', '10000');

select * from loan;

create table installment (i_no int primary key, l_id int, name varchar(20),
total_amount float(10, 2));

insert into installment values (1, 1, 'Ashwin', 10000), (2, 2, 'Aarav', 2000),
(3, 3, 'Anandhu', 5000);

select * from installment;

select loan.l_id, l_type, total_amount from loan inner join installment on
loan.name = installment.name;
```

Output

```
+------+--------+-----------+----------------+
| c_id | name   | phone     | address        |
+------+--------+-----------+----------------+
|    1 | Ashwin | 123456789 | Padipurackal   |
|    2 | Aarav  | 123456789 | Ranjith bhavan |
|    3 | Anandhu | 123456789 | Anjana bhavan |
+------+--------+-----------+----------------+
+------+---------+-----------+-----------+----------+
| a_no | name    | bank_code | a_type    | balance  |
+------+---------+-----------+-----------+----------+
|    1 | Aarav   |        10 | Recurring | 10000.00 |
|    2 | Ashwin  |        10 | Savings   | 90000.00 |
|    3 | Anandhu |        10 | Current   | 50000.00 |
+------+---------+-----------+-----------+----------+

+------+---------+----------------+------+
| c_id | name    | address        | a_no |
+------+---------+----------------+------+
|    2 | Aarav   | Ranjith bhavan |    1 |
|    1 | Ashwin  | Padipurackal   |    2 |
|    3 | Anandhu | Anjana bhavan  |    3 |
+------+---------+----------------+------+


+------+---------+----------+----------+
| l_id | name    | l_type   | l_amount |
+------+---------+----------+----------+
|    1 | Ashwin  | Personal | 20000.00 |
|    2 | Aarav   | gold     |  5000.00 |
|    3 | Anandhu | Home     | 10000.00 |
+------+---------+----------+----------+

+------+------+---------+--------------+
| i_no | l_id | name    | total_amount |
+------+------+---------+--------------+
|    1 |    1 | Ashwin  |     10000.00 |
|    2 |    2 | Aarav   |      2000.00 |
|    3 |    3 | Anandhu |      5000.00 |
+------+------+---------+--------------+

+------+----------+--------------+
| l_id | l_type   | total_amount |
+------+----------+--------------+
|    1 | Personal |     10000.00 |
|    2 | gold     |      2000.00 |
|    3 | Home     |      5000.00 |
+------+----------+--------------+
```

Query 11 (STORED PROCEDURE)

```
create table customer (id int primary key, name varchar(20), city varchar(20),
pin int, ph long);

insert into customer values (1, 'Ashwin', 'pta', 689647, 123456789),(2, 'Aarav',
'Ranni', 690508, 123456789), (3, 'Anandhu', 'Adoor', 690504, 123456789);

select * from customer;

delimiter $$

create procedure d1()
    -> begin
    -> select name, city from customer;
    -> end $$

delimiter ;

call d1;

delimiter $$

create procedure d2(in c_city varchar(20))
    -> begin
    -> select * from customer where city = c_city;
    -> end $$

delimiter ;

call d2('Ranni');

delimiter $$

create procedure d3(in v_name varchar(20), out v_ph varchar(20))
    -> begin
    -> select ph into @phone from customer where name = v_name;
    -> end $$

delimiter ;

set @name = 'Anandhu';

call d3(@name, @phone);

select @phone;



call d3(@name, @phone);
```

Output

```
+----+---------+-------+--------+-----------+
| id | name    | city  | pin    | ph        |
+----+---------+-------+--------+-----------+
|  1 | Ashwin  | pta   | 689647 | 123456789 |
|  2 | Aarav   | Ranni | 690508 | 123456789 |
|  3 | Anandhu | Adoor | 690504 | 123456789 |
+----+---------+-------+--------+-----------+

+---------+-------+
| name    | city  |
+---------+-------+
| Ashwin  | pta   |
| Aarav   | Ranni |
| Anandhu | Adoor |
+---------+-------+

+----+-------+-------+--------+-----------+
| id | name  | city  | pin    | ph        |
+----+-------+-------+--------+-----------+
|  2 | Aarav | Ranni | 690508 | 123456789 |
+----+-------+-------+--------+-----------+

+-----------+
| @phone    |
+-----------+
| 123456789 |
+-----------+
```

Query 12 (SQL TRANSACTION)

```
create table account (a_no int primary key, c_no int, balance float(10, 2));

insert into account values (1, 1, 500), (2, 2, 2000), (3, 3, 4000);

select * from account;

start transaction;

update account set balance = balance + 1000 where a_no = 1;

savepoint A;

select * from account;

update account set balance = balance + 5000 where a_no = 2;

savepoint B;

select * from account;

rollback to savepoint A;

select * from account;

commit;
```

Output

```
+------+------+---------+
| a_no | c_no | balance |
+------+------+---------+
|    1 |    1 |  500.00 |
|    2 |    2 | 2000.00 |
|    3 |    3 | 4000.00 |
+------+------+---------+

+------+------+---------+
| a_no | c_no | balance |
+------+------+---------+
|    1 |    1 | 1500.00 |
|    2 |    2 | 2000.00 |
|    3 |    3 | 4000.00 |
+------+------+---------+
+------+------+---------+
| a_no | c_no | balance |
+------+------+---------+
|    1 |    1 | 1500.00 |
|    2 |    2 | 7000.00 |
|    3 |    3 | 4000.00 |
+------+------+---------+

+------+------+---------+
| a_no | c_no | balance |
+------+------+---------+
|    1 |    1 | 1500.00 |
|    2 |    2 | 2000.00 |
|    3 |    3 | 4000.00 |
+------+------+---------+
```

PL/SQL

Query 13 (A) (IF-THEN)

```
SQL> set serveroutput on

  1  declare
  2     a number := &x;
  3  begin
  4     if a = 0 then
  5             dbms_output.put_line('Number is zero: '||a);
  6     else
  7             dbms_output.put_line('Number not is zero: '||a);
  8     end if;
  9* end;
```

Output

```
Enter value for x: 1
old   2:        a number := &x;
new   2:        a number := 1;
Number not is zero: 1

PL/SQL procedure successfully completed.

SQL> /
Enter value for x: 0
old   2:        a number := &x;
new   2:        a number := 0;
Number is zero: 0

PL/SQL procedure successfully completed.
```

Query 13 (B) (IF-THEN-ELSE)

```
SQL> set serveroutput on

  1  declare
  2     a number := &x;
  3  begin
  4     if a mod 2 = 0 then
  5             dbms_output.put_line('Number is Even ');
  6     else
  7             dbms_output.put_line('Number id odd ');
  8     end if;
  9* end;
```

Output

```
Enter value for x: 2
old   2:        a number := &x;
new   2:        a number := 2;
Number is Even

PL/SQL procedure successfully completed.

SQL> /
Enter value for x: 3
old   2:        a number := &x;
new   2:        a number := 3;
Number id odd

PL/SQL procedure successfully completed.
```

Query 13 (C) (IF-THEN-ELSEIF)

```
SQL> set serveroutput on

  1  declare
  2     a number := &x;
  3     b number := &y;
  4     c number := &z;
  5  begin
  6     if ((a < b) and (b > c)) then
  7             dbms_output.put_line( b||' is the greatest');
  8     else if ((c > a) and (c > b)) then
  9             dbms_output.put_line( c||' is the greatest');
 10     else
 11             dbms_output.put_line( a||' is the greatest');
 12     end if;
 13  end if;
 14* end;
```

Output

```
Enter value for x: 1
old   2:        a number := &x;
new   2:        a number := 1;
Enter value for y: 2
old   3:        b number := &y;
new   3:        b number := 2;
Enter value for z: 3
old   4:        c number := &z;
new   4:        c number := 3;
3 is the greatest

PL/SQL procedure successfully completed.
```

Query 13 (D) (CASE)

```
SQL> set serveroutput on

  1  declare
  2     choice number := &c;
  3     a number := &x;
  4     b number := &y;
  5  begin
  6     case choice
  7             when 1 then
  8                     dbms_output.put_line('Area :  '||a*a);
  9             when 2 then
 10                     dbms_output.put_line('Area :  '||a*B);
 11             else
 12                     dbms_output.put_line('Invalid choice!!');
 13     end case;
 14* end;
```

Output

```
Enter value for c: 2
old   2:        choice number := &c;
new   2:        choice number := 2;
Enter value for x: 2
old   3:        a number := &x;
new   3:        a number := 2;
Enter value for y: 3
old   4:        b number := &y;
new   4:        b number := 3;
Area :  6
```

Query 13 (E) (WHILE)

```
SQL> set serveroutput on

  1  declare
  2     n number := &n;
  3     m number;
  4     s number := 0;
  5     r number;
  6     len number;
  7  begin
  8     m := n;
  9     len := length(to_char(n));
 10     while (n > 0) loop
 11             r := mod(n, 10);
 12             s := s + power(r, len);
 13             n := trunc(n/10);
 14     end loop;
 15     if (s = m) then
 16             dbms_output.put_line('Number is armstrong');
 17     else
 18             dbms_output.put_line('Number is not armstrong');
 19     end if;
 20* end;
```

Output

```
Enter value for n: 153
old   2:        n number := &n;
new   2:        n number := 153;
Number is armstrong

PL/SQL procedure successfully completed.


Enter value for n: 152
old   2:        n number := &n;
new   2:        n number := 152;
Number is not armstrong

PL/SQL procedure successfully completed.
```

Query 14 (TRIGGER)

```
create table employee (id number primary key, name varchar(20), age number, city
varchar(20), department varchar(20), desig varchar(20), salary float(10));
insert into employee values(1, 'Ashwin', 20, 'PTA', 'Creative', 'Lead', 100000);
insert into employee values(2, 'Anantha', 20, 'TVM', 'Tech', 'Lead', 100000);
insert into employee values(3, 'Manas', 21, 'PTA', 'Marketing', 'Lead', 90000);
select * from employee;
```

```
create or replace trigger salary_diff before insert or update or delete  on
employee for each row when (NEW.ID>0)
declare
   sal_diff number;
begin
   if inserting then
           dbms_output.put_line('New salary'||:NEW.salary);
   else if updating then
           sal_diff := :NEW.salary - :OLD.salary;
           dbms_output.put_line('Old salary :  '||:OLD.salary);
           dbms_output.put_line('New salary :  '||:NEW.salary);
           dbms_output.put_line('Salary difference :  '||sal_diff);
   else if deleting then
           dbms_output.put_line('Old salary :  '||:OLD.salary);
   end if;
   end if;
   end if;
end;
```

```
update employee set salary = 100000 where id = 3;
```

Output

```
        ID NAME                        AGE CITY
---------- -------------------- ---------- --------------------
DEPARTMENT           DESIG                    SALARY
-------------------- -------------------- ----------
         1 Ashwin                       20 PTA
Creative             Lead                     100000

         2 Anantha                      20 TVM
Tech                 Lead                     100000

         3 Manas                        21 PTA

Marketing            Lead                      90000



Old salary :  90000
New salary :  100000
Salary difference :  10000
```

Query 15 (CURSOR)

```
create table employee (id number primary key, name varchar(20), age number, city
varchar(20), department varchar(20), desig varchar(20), salary float(10));
insert into employee values(1, 'Ashwin', 20, 'PTA', 'Creative', 'Lead', 100000);
insert into employee values(2, 'Anantha', 20, 'TVM', 'Tech', 'Lead', 100000);
insert into employee values(3, 'Manas', 21, 'PTA', 'Marketing', 'Lead', 30000);
select * from employee;
```

```
declare
        cursor cur is select id, name, desig, salary from employee where
        salary > 40000;
        e_id employee.id%type;
        e_name employee.name%type;
        e_desig employee.desig%type;
        e_salary employee.salary%type;
     begin
        open cur;
        loop
        fetch cur into e_id, e_name, e_desig, e_salary;
                  EXIT when cur%notfound;
               dbms_output.put_line('ID :  '||e_id||'Name :  '||e_name||'
               Designation :  '||e_desig||' Salary :  '||e_salary);
        end loop;
        close cur;
 end;
```

Output

```
        ID NAME                        AGE CITY
---------- -------------------- ---------- --------------------
DEPARTMENT           DESIG                    SALARY
-------------------- -------------------- ----------
         1 Ashwin                       20 PTA
Creative             Lead                     100000

         2 Anantha                      20 TVM
Tech                 Lead                     100000

         3 Manas                        21 PTA
Marketing            Lead                      30000


ID :  1Name :  Ashwin Designation :  Lead Salary :  100000
ID :  2Name :  Anantha Designation :  Lead Salary :  100000
```

Query 16 (PROCEDURE)

```
create procedure gen_fibonacci (n in number) as
   term1 number := 0;
   term2 number := 1;
   temp number;
begin
   if (n < 1) then
           dbms_output.put_line('Enter atleast one');
           return;
   end if;
   dbms_output.put_line('Fibonacci :  ');
   for i in 1 .. n loop
           dbms_output.put_line(' '||term1);
           temp := term1 + term2;
           term1 := term2;
           term2 := temp;
   end loop;
end gen_fibonacci;
```

```
SQL> begin
  2  gen_fibonacci(10);
  3  end;
  4  /
```

Output

```
Fibonacci :
0
1
1
2
3
5
8
13
21
34
```

Query 17 (FUNCTION)

```
  1  create or replace function is_prime(num in number) return boolean is
  2         limit number:= num / 2;
  3      begin
  4         if (num < 2 ) then
  5                 return false;
  6         end if;
  7         for j in 2 .. limit loop
  8               if((num mod j ) = 0 )then
  9                       return false;
 10               end if;
 11       end loop;
 12       return true;
 13* end is_prime;
```

```
  1   create or replace function nth_prime (n in number) return number is
  2       i number := 0;
  3       num number := 1;
  4    begin
  5       while (i < n) loop
  6          num := num + 1;
  7               if(is_prime(num)) then
  8                       i := i + 1;
  9               end if;
 10       end loop;
 11       return num;
 12*  end;
```

```
  1  declare
  2     num number := &n;
  3     res number;
  4  begin
  5     res := nth_prime(num);
  6     dbms_output.put_line(num||'  th Prime number is :  '||res);
  7* end;
  8  /
```

Output

```
Enter value for n: 2
old   2:        num number := &n;
new   2:        num number := 2;
2  th Prime number is :  3


Enter value for n: 9
old   2:        num number := &n;
new   2:        num number := 9;
9  th Prime number is :  23
```

Query 18 (EXCEPTION HANDLING)

```
 1  declare
  2     numerator number := numerator;
  3     denominator number := denominator;
  4     result number;
  5  begin
  6     result := numerator / denominator;
  7     dbms_output.put_line('Result :  '||result);
  8  exception when ZERO_DIVIDE then
  9     dbms_output.put_line('Error: Division by zero');
 10* end;
```

Output

```
Enter value for numerator: 10
old   2:        numerator number := numerator;
new   2:        numerator number := 10;
Enter value for denominator: 0
old   3:        denominator number := denominator;
new   3:        denominator number := 0;
Error: Division by zero


Enter value for numerator: 10
old   2:        numerator number := numerator;
new   2:        numerator number := 10;
Enter value for denominator: 5
old   3:        denominator number := denominator;
new   3:        denominator number := 5;
Result :  2
```

Query 19 (USER DEFINED EXCEPTION)

```
create table employees (id number primary key, name varchar(20), address
varchar(20))
insert into employees values (1, 'Ashwin', 'Padipurackal')
insert into employees values (2, 'Aarav', 'Ranjith bhavan')
insert into employees values (3, 'Anandhu', 'Anjana bhavan')
select * from employees;
```

Output

```
  1  declare
  2     invalid_id exception;
  3     not_found exception;
  4     e_name varchar(20);
  5     e_address varchar(20);
  6     e_id number := &id;
  7     c number;
  8  begin
  9     if(e_id < 0) then raise invalid_id;
 10     else
 11             select count(*) into c from employees where id = e_id;
 12             if(c = 0) then raise not_found;
 13             else
 14                     select name, address into e_name, e_address from
                        employees where employees.id = e_id;
 15                     dbms_output.put_line('Name :  '||e_name||'  Address :
                        '||e_address);
 16             end if;
 17     end if;
 18  exception
 19     when not_found then dbms_output.put_line('Employee not found');
 20     when invalid_id then dbms_output.put_line('Invalid employee ID');
 21* end;
```

```
        ID NAME                 ADDRESS
---------- -------------------- --------------------
         1 Ashwin               Padipurackal
         2 Aarav                Ranjith bhavan
         3 Anandhu              Anjana bhavan
```

```
Enter value for id: 3
old   6:        e_id number := &id;
new   6:        e_id number := 3;
Name :  Anandhu  Address :  Anjana bhavan
```

```
Enter value for id: -1
old   6:        e_id number := &id;
new   6:        e_id number := -1;
Invalid employee ID
```

```
Enter value for id: 5
old   6:        e_id number := &id;
new   6:        e_id number := 5;
Employee not found
```