1.

Create a db called company consist of the following tables.

1.Emp (eno,ename, job,hiredate,salary,commission,deptno,)

2.dept(deptno,deptname,location)

eno is primary key in emp

deptno is primary key in dept

SQL Queries:

```
CREATE DATABASE company;

USE company;

CREATE TABLE Emp(eno INT AUTO_INCREMENT, ename VARCHAR(30), job VARCHAR(20), hiredate DATE, salary INT, commission INT, deptno INT, PRIMARY KEY (eno));

CREATE TABLE dept(deptno INT, deptname VARCHAR(30), location VARCHAR(20), PRIMARY KEY (deptno));

INSERT INTO Emp(eno, ename, job, hiredate, salary, commission, deptno)
VALUES(1, 'Shriram', 'Manager', '1969-02-25', 80000, 7, 10),
(2, 'Krishna', 'Manager', '1999-05-27', 110000, 5, 20),
(3, 'Sanket', 'Clerk', '1979-06-14', 55000, 8, 30 ),
(4, 'Aniket', 'Salesman', '2001-08-07', 69000, 9, 20),
(5, 'Kisna', 'Analyst', '2000-06-12', 120000, 6, 10);

INSERT INTO dept (deptno, deptname, location)
VALUES(10, 'Dev', 'Tokyo'),
(20, 'Marketing', 'Nairobi'),
(30, 'Sales', 'Helsinki'),
```

(40, 'Hr', 'Denver');

Solve Queries by SQL :

1. List the maximum salary paid to salesman

SELECT MAX(salary) FROM Emp WHERE job = 'Salesman';

2. List name of emp whose name start with ‘I’

SELECT ename FROM Emp WHERE ename LIKE 'I%' ;

3. List details of emp who have joined before ’30-sept-81’

SELECT * FROM Emp WHERE hiredate<('1991-09-30');

4. List the emp details in the descending order of their basic salary

SELECT * FROM Emp ORDER BY salary DESC;

5. List of no. of emp & avg salary for emp in the dept no ‘20’

SELECT count(*), avg(salary) From Emp Where deptno = 20;

6. List the avg salary, minimum salary of the emp hiredatewise for dept no ‘10’.

//Doubt : SELECT AVG(salary),Min(salary) FROM Emp WHERE deptno = 10;

7. List emp name and its department

SELECT e.ename,d.deptname FROM Emp e, dept d WHERE e.deptno = d.deptno;

8. List total salary paid to each department

SELECT e.salary, d.deptname FROM Emp e, dept d WHERE e.deptno = d.deptno;

9. List details of employee working in ‘Dev’ department

SELECT * FROM Emp WHERE deptno IN

(SELECT deptno FROM dept WHERE deptname = 'Dev') ;

10. Update salary of all employees in deptno 10 by 5 %.

UPDATE Emp SET salary = (select salary+(select salary*.5))

WHERE deptno = 10;

6] The following tables form part of a database held in a relational DBMS:

Hotel (HotelNo, Name, City)

Room (RoomNo, HotelNo, Type, Price)

Booking (HotelNo, GuestNo, DateFrom, DateTo, RoomNo)

Guest (GuestNo, GuestName, GuestAddress)

Solve following queries by SQL

1. List full details of all hotels.

-> select * from hotel;

2. List full details of all hotels in London.

-> select * from hotel where city="London";

3. List all guests currently staying at the Grosvenor Hotel.

```
-> SELECT * FROM Guest
 WHERE guestNo =
 (SELECT guestNo FROM Booking
 WHERE dateFrom <= CURRENT_DATE AND
 dateTo >= CURRENT_DATE AND
 hotelNo =
 (SELECT hotelNo FROM Hotel
 WHERE hotelName = 'Grosvenor Hotel'));
```

4. List the names and addresses of all guests in London, alphabetically ordered by name.

```
-> SELECT guestName, guestAddress FROM Guest WHERE address LIKE '%London%'
 ORDER BY guestName;
```

5. List the bookings for which no date_to has been specified.

```
-> SELECT * FROM Booking WHERE dateTo IS NULL;
```

6. How many hotels are there?

```
-> SELECT COUNT(*) FROM Hotel;
```

7. List the rooms that are currently unoccupied at the Grosvenor Hotel.

```
-> SELECT * FROM room r
WHERE roomno NOT IN
(SELECT roomno FROM booking b, hotel h
WHERE (datefrom <= CURRENT_DATE AND
dateto >= CURRENT_DATE) AND
b.hotelno = h.hotelno AND hotelname = 'Grosvenor');
```

8. What is the lost income from unoccupied rooms at each hotel today?

```
-> SELECT hotelno, SUM(price) FROM room r
```

```
WHERE roomno NOT IN
(SELECT roomno FROM booking b, hotel h
WHERE (datefrom <= CURRENT_DATE AND
dateto >= CURRENT_DATE) AND
b.hotelno = h.hotelno)
GROUP BY hotelno;
```

9. Create index on one of the field and show is performance in query.

```
-> CREATE INDEX index_hotel ON Hotel (hotelno,Name,City);
```

10. Create one view on above database and query it.

```
-> CREATE VIEW [Hotel Detail] AS SELECT Name,City FROM Hotel WHERE hotelno = 101 ;
```

7] Consider the following database

Project(project_id,proj_name,chief_arch) , project_id is primary key

Employee(Emp_id,Emp_name) , Emp_id is primary key

Assigned-To(Project_id,Emp_id)

Find the SQL queries for the following:

1. Get the details of employees working on project C353

```
-> select A.empid, emp_name from A.Assigned_To A, Employee where project_id = 'C353' ;
```

2. Get employee number of employees working on project C353

```
->select emp_id from Assigned_To where projectid = 'C353';
```

3. Obtain details of employees working on Database project

```
-> select Emp_name, A. Emp_id from A. Assigned_To A, Employee where project_id in (select P.
project_id
from P. project where P. project_name = 'Database');
```

4. Get details of employees working on both C353 and C354

-> (select Emp_name, A. emp_id from Assigned_to A, Employee where A.Project_id = C354)

intersect

(select emp_name, A.empid from A.Assigned_To A, Employee where project_id = 'C354');

5. Get employee numbers of employees who do not work on project C453

-> (select emp_id from Employee)

minus

(select emp_id from assigned_to where project_id = 'C453');

8] Consider the following database

Employee(emp_no,name,skill,pay-rate) eno primary key

Position(posting_no,skill) posting_no primary key

Duty_allocation(posting_no,emp_no,day,shift)

Find the SQL queries for the following:

1. Get the duty allocation details for emp_no 123461 for the month of April 1986.

-> select posting_no., shift, day from Duty_allocation where emp_no = 123461 and

Day ≥ 19860401 and Day ≤ 19860430 ;

2. Find the shift details for Employee ‘xyz’

-> select posting_no., shift, day from Duty_allocation, Employee

where Duty allocation.emp_no. = Employee.emp_no and Name = 'XYZ';

3. Get employees whose rate of pay is more than or equal to the rate of pay of employee ‘xyz’

-> select S.name, S.pay_rate from Employee as S, Employee as T where S.pay_rate > T.pay_rate

and T.name = 'XYZ';

4. Get the names and pay rates of employees with emp_no less than 123460 whose rate of pay is

more than the rate of pay of at least one employee with emp_no greater than or equal to 123460.

-> select name, pay_rate from Employee where emp_no < 123460 and pay_rate > some

(select pay_rate from Employee where emp_no ≥ 123460);

5. Find the names of employees who are assigned to all positions that require a Chef's skill

-> select S.Name from Employee S where (select posting_no from Duty_allocation D where S.emp_no = D.emp_no)

contains

(select P.posting_no from position P where P.skill = 'Chef');

6.Find the employees with the lowest pay rate

-> select emp_no, Name, Pay_rate from Employee where pay_rate ≤ all

(select pay_rate from Employee)

7. Get the employee numbers of all employees working on at least two dates.

-> select emp_no from Duty_allocation group by emp_no having (count;*) > 1

8. Get a list of names of employees with the skill of Chef who are assigned a duty

-> select Name from Employee where emp_no in ((select emp_no from Employee where skill = 'Chef')

intersect

(select emp_no from Duty_allocation));

9. Get a list of employees not assigned a duty

-> (select emp_no from Employee)

minus

(select emp_no from Duty_allocation)

10. Get a count of different employees on each shift

-> select shift, count (distinct emp_no) from Duty_allocation group by shift;

9] Create the following tables. And Solve following queries by SQL

- Deposit (actno,cname,bname,amount,adate)
- Branch (bname,city)
- Customers (cname, city)
- Borrow(loanno,cname,bname, amount)

Add primary key and foreign key wherever applicable.

Tables :

1) create table deposit(actno varchar2(5),cname varchar2(18),bname varchar2(18),amount number(8,2),adate

date);

2) create table branch(bname varchar2(18),city varchar2(18));

3) create table customer(cname varchar2(18), city varchar2(18));

4) create table borrow(loanno varchar2(5),cname varchar2(18),bname varchar2(18),amount number(8,2));

DEPOSIT TABLE:

1) insert into deposit values('100','Anil','wakad',1000.00,'1-mar-95');

2) insert into deposit values('101','Sunil','Pimpri',5000.00,'4-jan-96');

3) insert into deposit values('102','Aniket','KAROLBAGH',3500.00,'17-nov-95');

4) insert into deposit values('104','Krishna','bahgya nagar',1200.00,'17-dec-95');

5) insert into deposit values('105','Sanket','nehru nagar',3000.00,'27-mar-96');

6) insert into deposit values('106','Anuj','shivaji nagar',2000.00,'31-mar-96');

7) insert into deposit values('107','Shriram','Karvenagar',1000.00,'5-sep-95');

8) insert into deposit values('108','Bhosle','chinchwad',5000.00,'2-jul-95');

9) insert into deposit values('109','Anuj','akhurdi',7000.00,'10-aug-95');

BRANCH TABLE:

1) insert into branch values('wakad','Pune');

2) insert into branch values('Pimpri','Nagpur');

3) insert into branch values('KAROLBAGH','Delhi');

4) insert into branch values('bahgya nagar','Delhi');

5) insert into branch values('nehru nagar','Nagpur');

6) insert into branch values('shivaji nagar','Solapur');

7) insert into branch values('Karvenagar','Bombay');

8) insert into branch values('chinchwad','Umarkhed');

9) insert into branch values('akhurdi,'Delhi');

10) insert into branch values('POWAI','Bombay');

CUSTOMER TABLE:

1) insert into customer values('Anil','Calcutta');

2) insert into customer values('Sunil','Delhi');

3) insert into customer values('Aniket','Pune');

4) insert into customer values('Krishna','Nanded');

5) insert into customer values('Sanket','Solapur');

6) insert into customer values('Anuj','Nagpur');

7) insert into customer values('Shriram','Surat');

8) insert into customer values('Bhosle','Bombay');

9) insert into customer values('Om','Bombay');

10) insert into customer values('Deva','Nashik');

BORROW TABLE:

1) insert into borrow values('201','Anil','Calcutta',1000);

2) insert into borrow values('206','Shriram','Nashik',5000);

3) insert into borrow values('311','Sunil','DHARAMPETH',3000);

4) insert into borrow values('321','Sanket','Solapur',2000);

5) insert into borrow values('375','Krishna','Nanded',8000);

6) insert into borrow values('481','Bhosle','Mumbai',3000);

Insert data into the above created tables.

1. Display names of depositors having amount greater than 4000.

-> select cname from deposit where amount>'4000';

2. Display account date of customers Anil

-> select adate from deposit where cname='Anil';

3. Display account no. and deposit amount of customers having account opened

between dates 1-12-96 and 1-5-97

-> select actno,amount from deposit where adate between '01-DEC-95' and '01-MAY-96';

4. Find the average account balance at the Perryridge branch.

-> select avg (balance) from account where branch-name = “Perryridge” ;

5. Find the names of all branches where the average account balance is more

than $1,200.

-> select branch-name, avg (balance) from account group by branch-name having avg (balance) > 1200

6. Delete depositors having deposit less than 5000

-> DELETE FROM DEPOSIT WHERE AMOUNT < 5000

7. Create a view on deposit table.

->CREATE VIEW [Customers Detail] AS SELECT cname,bname FROM deposit WHERE actno = 101 ;

10]

Insert data into the above created tables.

a. Display names of all branches located in city Bombay.

-> select bname from branch where city='Bombay';

b. Display account no. and amount of depositors.

-> select actno,amount from deposit;

c. Update the city of customers Anil from Pune to Mumbai

->

d. Find the number of depositors in the bank

-> select count (distinct customer-name)

from depositor

e. Calculate Min,Max amount of customers.

->

f. Create an index on deposit table

-> CREATE INDEX index_customer_branch ON deposit (cname,bname);

g. Create View on Borrow table.

-> CREATE VIEW [Customers Detail] AS SELECT cname, loanno , bname FROM borrow WHERE bname = 'ANDHERI';

11]

Insert data into the above created tables.

a. Display account date of customers Anil.

-> select adate from deposit where cname='Anil';

b. Modify the size of attribute of amount in deposit

-> alter table deposit modify amount number(10,2);

c. Display names of customers living in city pune.

-> select cname from customer where city='pune';

d. Display name of the city where branch KAROLBAGH is located.

-> select city from branch where bname='KAROLBAGH';

e. Find the number of tuples in the customer relation

-> select count (*) from customer;

f. Delete all the record of customers Sunil

->delete from deposit where cname='Sunil';

g. Create a view on deposit table.

CREATE VIEW [KAROLBAGH Branch] AS SELECT cname FROM deposit WHERE bname = 'KAROLBAGH';

15.a) Write an SQL code block these raise a user defined exception where business rule is voilated. BR for client_ master table specifies when

the value of bal_due field is less than 0 handle the exception.

*******Code*********

---

b) Write an SQL code block Borrow(Roll_no, Name, DateofIssue, NameofBook, Status) Fine(Roll_no,Date,Amt) Accept roll_no & name of book from user.

Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day. If no. of days>30, per day fine

will be Rs 50 per day & for days less than 30, Rs. 5 per day. After submitting the book, status will change from I to R. If condition of fine is true,

then details will be stored into fine table. Also handles the exception by named exception handler or user define exception handler.

*******Code*********

_

First Create Both the tables Borrower and Fine and Insert values in Borrower

```
Create procedure calcfine6(rno  int,bname varchar(20))
begin
declare fine int default 0;
declare days int default 0;
declare issuedate date;
declare namee varchar(20);
declare exit handler for SQLEXCEPTION select 'create table definition';
Select name  into namee from Borrower where Roll_no = rno;
Select DOI into issuedate from Borrower where Roll_no = rno  AND  NameofBook = bname;
Set days = datediff(now(),issuedate);
If(days > 15 && days < 30) then
Set fine  = days * 5;
End if;
If(days > 30) then
Set fine  = days * 50;
End if;
If(fine > 0)  then
Insert into finet values(rno,namee,fine);
End if;
Update Borrower set Status = 'R' where Roll_no = rno AND NameofBook = bname;
End
//_
```

****

---

16. Cursor (Any Two)

a) The bank manager has decided to activate all those accounts which were previously marked as inactive for performing no transaction in last 365 days. Write a

PL/SQ block (using implicit cursor) to update the status of account, display an approximate message based on the no. of rows affected by the update. (Use of %FOUND,

%NOTFOUND, %ROWCOUNT)

*******Code*********

```
create table bankcursor(acc_no number(10),status varchar(10));


insert into bankcursor values (12,'active');

insert into bankcursor values (13,'inactive');

insert into bankcursor values (14,'inactive');

insert into bankcursor values (15,'active');

insert into bankcursor values (16,'inactive');


Declare

Rows_affe number(10);

Begin

update bankcursor set status='active'where status='inactive';

Rows_affe=(SQL%rowcount);

dbms_output.put_line(Rows_affe || ' rows areaffected...');

END;

/
```

---

b)Organization has decided to increase the salary of employees by 10% of existing salary, who are having salary less than average salary of organization,

Whenever such salary updates takes place, a record for the same is maintained in the increment_salary table

*******Code*********

```
Create table employee(e_id int,salary int);

insert into employee values(1,10000);
insert into employee values(2,11000);
insert into employee values(3,12500);
insert into employee values(4,9000);
insert into employee values(5,10000);
insert into employee values(6,7000);
insert into employee values(7,12000);

create table increament_t(
e_id int,
salary int
);

Declare
Cursor c1 is select e_id,salary from employee where salary < (select avg(salary) from employee);
me_no employee.e_id%type;
msalary employee.salary%type;
Begin
open c1;
if(c1%isopen) then
loop
fetch c1 into me_no,msalary;
exit when c1%notfound;
if(c1%found) then
update employee set salary  = salary + (salary * 0.10) where e_id = me_no;
select salary into msalary from employee where e_id = me_no;
insert into increament_t values(me_no,msalary);
```

```
end if;

end loop;

end if;

end;
```

---

c)Write PL/SQL block using explicit cursor for following requirements: College has decided to mark all those students detained (D) who are having attendance

less than 75%. Whenever such update takes place, a record for the same is maintained in the D_Stud table. create table stud21(roll number(4), att number(4),

status varchar(1))

*******Code*********

```
create table stud1(roll_no number(5),attendance number(5),status varchar(7));


insert into stud1 values(8,76,'');

insert into stud1 values(2,74,'');

insert into stud1 values(3,77,'');

insert into stud1 values(4,73,'');

insert into stud1 values(5,78,'');

insert into stud1 values(6,72,'');


declare

cursor c1 is select   roll_no,attendance, from employee;

roll stud1.roll_no%type;

attend stud1.attendance%type;

begin

open c1;

if(c1%isopen) then

loop

fetch c1 into roll,attend;
```

```
exit when c1%notfound;
if(attend > 75) then
update stud1 set status = 'ND' where roll_no = roll;
else
update stud1 set status = 'D' where roll_no = roll;
insert into D_stud values(roll);
end if;
end loop;
end if;
end;
```

---

17. Cursor (Any Two)

b)Write a PL/SQL block of code using parameterized Cursor, that will merge the data available in the newly created table N_RollCall with the data available

in the table O_RollCall. If the data in the first table already exist in the second table then that data should be skipped. output:

*******Code*********

```
create table o_rollcall(
roll_no int,
name varchar(20)
);

create table n_rollcall(
roll_no int,
name varchar(20)
);

create procedure n4(IN rno1 int)
```

```
begin
declare rno2 int;
declare exit_cond boolean;
declare c1 cursor for select  roll_no from o_rollcall where roll_no>rno1;
open c1;
l1:loop
fetch c1 into rno2;
if not exists(select * from n_rollcall where roll_no = rno2) then
insert into n_rollcall select * from o_rollcall where roll_no = rno2;
end if;
if exit_cond then
close c1;
leave l1;
end if;
end loop l1;
end;
/
```

---

c)Write the PL/SQL block for following requirements using parameterized Cursor: Consider table EMP(e_no, d_no, Salary), department wise average salary

should be inserted into new table dept_salary(d_no, Avg_salary)

*******Code*********

```
create table EMP(e_no int, d_no int, Salary int);
create table dept_salary(d_no int, Avg_salary int);

CREATE or REPLACE PROCEDURE avgsal
IS
CURSOR department is
```

```
select d_no,AVG(salary) as avgsal from emp

group by d_no;

BEGIN

FOR data in department

LOOP

insert into dept_salary values(data.d_no,data.avgsal);

end LOOP;

end;
```

18.Trigger a) Write a update, delete trigger on clientmstr table. The System should keep track of the records that ARE BEING updated or deleted. The old value of updated or deleted records should be added in audit_trade table.

(separate implementation using both row and statement triggers).

*******Code*********

```
create or replace trigger storedata

before delete

on

clientmstr

for each row

begin

insert into  audit_trade values(old.c_id,c_name);

end;

/

create or replace trigger storedata

before update

on

clientmstr

for each row

begin

insert into  audit_trade values(old.c_id,c_name);
```

end;

/

---

b) Write a before trigger for Insert, update event considering following requirement: Emp(e_no, e_name, salary) I) Trigger action should be initiated

when salary is tried to be inserted is less than Rs. 50,000/- II) Trigger action should be initiated when salary is tried to be updated for value less

than Rs. 50,000/- Action should be rejection of update or Insert operation by displaying appropriate error message. Also the new values expected to be

nserted will be stored in new table Tracking(e_no, salary).

*******Code**********

```
create or replace trigger emp_sal_trigger
before insert or update on emp
for each row
begin
if:NEW.salary < 50000 then
RAISE_APPLICATION_ERROR(-20001,'MINIMUM SALARY SHOULD BE 50000');
end if;
end;
```
