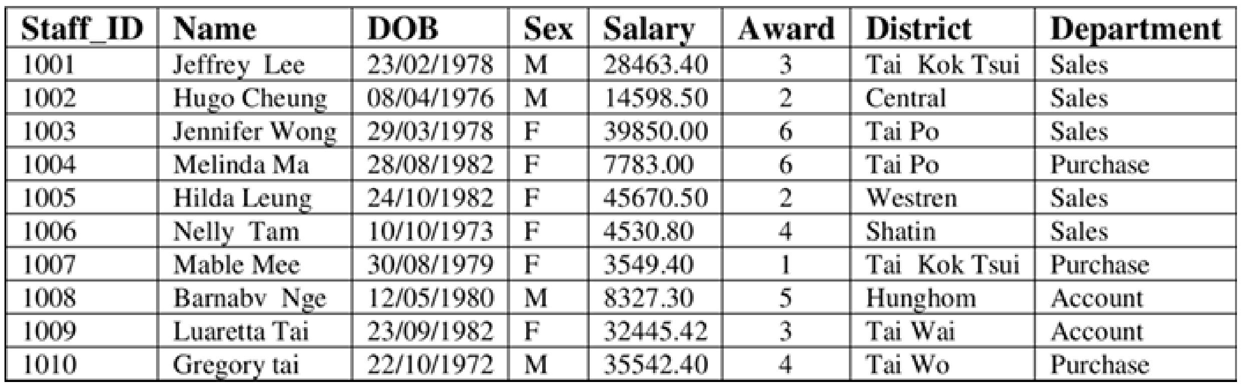
---SIMPLE---

1) Create the following table and write the queries



Staff_ID	Name	DOB	Sex	Salary	Award	District	Department
1001	Jeffrey Lee	23/02/1978	M	28463.40	3	Tai Kok Tsui	Sales
1002	Hugo Cheung	08/04/1976	M	14598.50	2	Central	Sales
1003	Jennifer Wong	29/03/1978	F	39850.00	6	Tai Po	Sales
1004	Melinda Ma	28/08/1982	F	7783.00	6	Tai Po	Purchase
1005	Hilda Leung	24/10/1982	F	45670.50	2	Westren	Sales
1006	Nelly Tam	10/10/1973	F	4530.80	4	Shatin	Sales
1007	Mable Mee	30/08/1979	F	3549.40	1	Tai Kok Tsui	Purchase
1008	Barnabv Nge	12/05/1980	M	8327.30	5	Hunghom	Account
1009	Luaretta Tai	23/09/1982	F	32445.42	3	Tai Wai	Account
1010	Gregory tai	22/10/1972	M	35542.40	4	Tai Wo	Purchase

-- Create the table

```
CREATE TABLE Staff (
    Staff_ID INT PRIMARY KEY,
    Name VARCHAR(50),
    DOB DATE,
    Sex CHAR(1),
    Salary DECIMAL(10,2),
    Award INT,
    District VARCHAR(50),
    Department VARCHAR(50)
);
--------------------------------------------------------------------------------------------------------------------
-
```

-- Insert statements

```
INSERT INTO Staff (Staff_ID, Name, DOB, Sex, Salary, Award, District, Department) VALUES
(1001, 'Jeffrey Lee', TO_DATE('1978-02-23', 'YYYY-MM-DD'), 'M', 28463.40, 3, 'Tai Kok Tsui',
'Sales');
INSERT INTO Staff (Staff_ID, Name, DOB, Sex, Salary, Award, District, Department) VALUES
(1002, 'Hugo Cheung', TO_DATE('1976-04-08', 'YYYY-MM-DD'), 'M', 14598.50, 2, 'Central', 'Sales');
INSERT INTO Staff (Staff_ID, Name, DOB, Sex, Salary, Award, District, Department) VALUES
(1003, 'Jennifer Wong', TO_DATE('1978-03-29', 'YYYY-MM-DD'), 'F', 39850.00, 6, 'Tai Po', 'Sales');
INSERT INTO Staff (Staff_ID, Name, DOB, Sex, Salary, Award, District, Department) VALUES
(1004, 'Melinda Ma', TO_DATE('1982-08-28', 'YYYY-MM-DD'), 'F', 7783.00, 6, 'Tai Po', 'Purchase');
INSERT INTO Staff (Staff_ID, Name, DOB, Sex, Salary, Award, District, Department) VALUES
(1005, 'Hilda Leung', TO_DATE('1982-10-24', 'YYYY-MM-DD'), 'F', 45670.50, 2, 'Western', 'Sales');
INSERT INTO Staff (Staff_ID, Name, DOB, Sex, Salary, Award, District, Department) VALUES
(1006, 'Nelly Tam', TO_DATE('1973-10-10', 'YYYY-MM-DD'), 'F', 4530.80, 4, 'Shatin', 'Sales');
INSERT INTO Staff (Staff_ID, Name, DOB, Sex, Salary, Award, District, Department) VALUES
(1007, 'Mable Mee', TO_DATE('1979-08-30', 'YYYY-MM-DD'), 'F', 3549.40, 1, 'Tai Kok Tsui',
'Purchase');
INSERT INTO Staff (Staff_ID, Name, DOB, Sex, Salary, Award, District, Department) VALUES
(1008, 'Barnaby Nge', TO_DATE('1980-12-05', 'YYYY-MM-DD'), 'M', 8327.30, 5, 'Hunghom',
'Account');
INSERT INTO Staff (Staff_ID, Name, DOB, Sex, Salary, Award, District, Department) VALUES
(1009, 'Lauretta Tai', TO_DATE('1982-09-23', 'YYYY-MM-DD'), 'F', 32445.42, 3, 'Tai Wai', 'Account');
```

```
INSERT INTO Staff (Staff_ID, Name, DOB, Sex, Salary, Award, District, Department) VALUES
(1010, 'Gregory Tai', TO_DATE('1972-10-22', 'YYYY-MM-DD'), 'M', 35542.40, 4, 'Tai Wo', 'Purchase');
```

-

a) Write a SQL statement to print a list of districts that consists of a single word.The list should not consist of repeating items and is arranged in descending alphabetical order.

```
SELECT DISTINCT district
FROM Staff
WHERE district NOT LIKE '% %'
ORDER BY district DESC;
```

-

b) Write a SQL statement to show those staff born in the months between September and December. Display the dates 10 days before these dates of birth so that manager has enough time to prepare present for the staff. Arrange dates from nearest to furthest.

```
SELECT name, dob,
dob - INTERVAL '10' DAY AS prep_date
FROM Staff
WHERE EXTRACT(MONTH FROM dob) BETWEEN 9 AND 12
ORDER BY prep_date ASC;
```

-

c) Write a SQL statement to display employees who joined in January month.

```
SELECT name, dob
FROM Staff
WHERE EXTRACT(MONTH FROM dob) = 1;
```

-

d) Add primary key constraint and not null constraint to the employee table.

```
ALTER TABLE employee
MODIFY (name NOT NULL);
ALTER TABLE employee
MODIFY (sex NOT NULL);
ALTER TABLE employee
MODIFY (district NOT NULL);
```

2) Consider the following schemas Sailors(sid: integer, sname: string, rating: integer, age: real); Boats(bid: integer, bname: string, color: string); Reserves(sid: integer, bid: integer, day: date). Answer the following Queries using Aggregate functions, GROUP BY, and HAVING clauses.

a) who is the youngest sailor

```
select sname from sailors where age=(select min(age) from sailors) ;
```

-

b) Find the name of the sailor who have maximum rating

```
select sname from sailors where rating=(select max(rating) from sailors);
```

-

c) What is the total rating of all Sailors.

```
select sum(rating) as total_rating from sailors;
```

----

d) How many sailors are there with a rating above 9.
select count(*) as num_of_sailors from sailors where rating>9;

-
e) Display the number of sailors in each country, sorted from high to low.
SELECT country, COUNT(*) as no_of_sailors
FROM Sailors
GROUP BY country
ORDER BY no_of_sailors DESC;

-
f) Display the number of boats in each country, sorted from low to high.
SELECT country, COUNT(*) as no_of_boats
FROM Boats
GROUP BY country
ORDER BY no_of_boats ASC;

3) Consider the following schemas Sailors(sid: integer, sname: string, rating: integer, age: real); Boats(bid: integer, bname: string, color: string); Reserves(sid: integer, bid: integer, day: date). Answer the following Queries using Aggregate functions, GROUP BY, and HAVING clauses.
a) who is the oldest sailor
select sname from sailors where age=(select max(age) from sailors);

-
b) Find the name of the sailor who has the minimum rating
select sname from sailors where rating=(select min(rating) from sailors);

-
c) What is the average rating of all Sailors
select avg(rating) as average_rating from sailors;

-
d) how many sailors are there with a rating above 7.
select count(*) as no_of_sailors from sailors
where rating>7 ;

-
e) . Display the number of sailors in each rating, sorted from high to low.
select rating ,count(*) as no_of_sailors
from sailors
GROUP BY rating
ORDER BY no_of_sailors desc;

-
f) Display the number of boats in each boatid, sorted from low to high.
SELECT bid, COUNT(*) as no_of_boats
FROM Boat
GROUP BY bid
ORDER BY no_of_boats ASC;

4) Consider the following schemas Sailors(sid: integer, sname: string, rating: integer, age: real); Boats(bid: integer, bname: string, color: string); Reserves(sid: integer, bid: integer, day: date). Answer the following Queries using Aggregate functions, GROUP BY, and HAVING clauses.

a) Find the Sid's of sailors who have reserved a red or a green boat.

```
select s.sname from sailors s ,reserve r,boat b where s.sid=r.sid AND r.bid=b.bid AND
b.bcolour='red'
UNION
select s.sname from sailors s ,reserve r,boat b where s.sid=r.sid AND r.bid=b.bid AND
b.bcolour='green';
```

-

b)Find the names of sailors who have reserved a red and a green boat.

```
select  s.sname from sailors s ,reserve r,boat b where s.sid=r.sid AND r.bid=b.bid AND
b.bcolour='red'
intersect
select  s.sname from sailors s ,reserve r,boat b where s.sid=r.sid AND r.bid=b.bid AND
b.bcolour='green';
```

-

c) Find the names of sailors who have reserved all boats.

```
SELECT sname
FROM Sailors s
WHERE NOT EXISTS (
    SELECT b.bid
    FROM Boat b
    WHERE NOT EXISTS (
        SELECT r.bid
        FROM Reserve r
        WHERE r.sid = s.sid AND r.bid = b.bid
    )
);
```

-

d)Find the names of sailors who have reserved boat number 103 using correlated nested query.

```
select s.sname from sailors s where EXISTS(select * from reserves r where r.bid=103 AND
r.sid = s.sid);
```

5) Consider the following schemas Sailors(sid: integer, sname: string, rating: integer, age: real); Boats(bid: integer, bname: string, color: string); Reserves(sid: integer, bid: integer, day: date). Answer the following Queries.

a) Find the names of sailors who have reserved a red but not green boats.

```
select  s.sid from sailors s ,reserve r,boat b where s.sid=r.sid AND r.bid=b.bid AND
b.bcolour='red' and b.bcolour!='green';
```

-

b) Find all sids of sailors who have a rating of 10 or reserved boat 104.

```
select s.sid from sailors s where s.rating=10
```

union
select r.sid from reserve r where r.bid=104 ;

-
c) Find the names of sailors who have reserved boat 103 using independent nested query
select s.sname from sailors s
where sid in (select r.sid from reserve r where r.bid=103);

-
d) Find the names of sailors who have reserved a red boat.
select s.sname from sailors s where sid in(select r.sid from reserve r,boat b where
r.bid=b.bid and b.bcolour='red');

-
e) Find the names of sailors who have not reserved a red boat
select s.sname from sailors s where sid not in(select r.sid from reserve r,boat b where
r.bid=b.bid and b.bcolour='red');

6) write PL/SQL code for insert trigger, delete trigger, and update trigger using the passenger database. Passenger (Passport_ id INTEGER PRIMARY KEY, Name VARCHAR (50) Not NULL, Age Integer Not NULL, Sex Char, Address VARCHAR (50) Not NULL);
a) Write an Insert Trigger to check if the Passport_id is exactly six digits or not.

```
 CREATE TABLE Passenger(Passport_id INT PRIMARY KEY,
   Name VARCHAR(50) NOT NULL,
   Age INT NOT NULL,
   Sex VARCHAR(10),
   Addr VARCHAR(10) NOT NULL);
```

```
 CREATE OR REPLACE TRIGGER checkid
 BEFORE
 INSERT ON Passenger
 FOR EACH ROW
 BEGIN
 IF
 LENGTH(:new.Passport_id) != 6 THEN
 RAISE_APPLICATION_ERROR(-20001, 'Passport id has to be 6
 digits minimum.');
 END IF;
 END;
```

-
b) Write a trigger on passenger to display messages ‘1 Record is inserted’, ‘1 record is deleted’, ‘1 record is updated’ when insertion, deletion and updating are done on passenger respectively.

```
CREATE OR REPLACE TRIGGER displayIns
 AFTER
 INSERT ON Passenger
 BEGIN
 dbms_output.put_line('1 Record is inserted');
 END;
```

```
------------------------------------------------------------------------------------------------------------------------
-
CREATE OR REPLACE TRIGGER displayUpd
 AFTER
 UPDATE ON Passenger
 BEGIN
 dbms_output.put_line('1 Record is updated');
 END;
------------------------------------------------------------------------------------------------------------------------
-
 CREATE OR REPLACE TRIGGER displayDel
 AFTER
 DELETE ON Passenger
 BEGIN
 dbms_output.put_line('1 Record is deleted');
 END;
------------------------------------------------------------------------------------------------------------------------
-
 INSERT INTO Passenger VALUES(123456, 'aaa', 12, 'M', 'Sec');
 INSERT INTO Passenger VALUES(12345, 'bbb', 13, 'M', 'Secb');
 INSERT INTO Passenger VALUES(123457, 'ccc', 18, 'F', 'Hyd');
 UPDATE Passenger SET Age=18 WHERE Passport_id = 123456;
 DELETE FROM Passenger WHERE Age=18;
```

7) Consider the following schemas . Sailors(sid: integer, sname: string, rating: integer, age: real); Boats(bid: integer, bname: string, color: string); Reserves(sid: integer, bid: integer, day: date). Answer the following i)Add constraint primary key, foreign key and not null to the reserves table ii)Insert values into all tables and use commit. iii)Find sailors whose rating is better than some sailor called 'Horatio'. iv)Find the names of sailors who have reserved a red and a green boat using intersect. .v) Find the sailors with the highest rating using all operator.

i) Add constraint primary key, foreign key, and not null to the reserves table:

```
ALTER TABLE Reserves
ADD CONSTRAINT pk_reserves PRIMARY KEY (sid, bid, day);

ALTER TABLE Reserves
ADD CONSTRAINT fk_reserves_sailors FOREIGN KEY (sid) REFERENCES Sailors(sid);

ALTER TABLE Reserves
ADD CONSTRAINT fk_reserves_boats FOREIGN KEY (bid) REFERENCES Boats(bid);

ALTER TABLE Reserves
MODIFY sid NOT NULL,
MODIFY bid NOT NULL,
MODIFY day NOT NULL;
------------------------------------------------------------------------------------------------------------------------
-
```

ii) Insert values into all tables and use commit:

```
-- Insert values into Sailors table
INSERT INTO Sailors (sid, sname, rating, age) VALUES (1, 'John', 8, 25);
INSERT INTO Sailors (sid, sname, rating, age) VALUES (2, 'Horatio', 7, 30);
INSERT INTO Sailors (sid, sname, rating, age) VALUES (3, 'Alice', 9, 22);

-- Insert values into Boats table
INSERT INTO Boats (bid, bname, color) VALUES (101, 'Boaty', 'red');
INSERT INTO Boats (bid, bname, color) VALUES (102, 'Sailor', 'green');
INSERT INTO Boats (bid, bname, color) VALUES (103, 'Wave', 'blue');

-- Insert values into Reserves table
INSERT INTO Reserves (sid, bid, day) VALUES (1, 101, '2024-12-01');
INSERT INTO Reserves (sid, bid, day) VALUES (2, 102, '2024-12-02');
INSERT INTO Reserves (sid, bid, day) VALUES (3, 103, '2024-12-03');

-- Commit the changes
COMMIT;
```

-

iii) Find sailors whose rating is better than some sailor called 'Horatio':

```
SELECT sname

FROM Sailors

WHERE rating > (SELECT rating FROM Sailors WHERE sname = 'Horatio');
```

-

iv) Find the names of sailors who have reserved a red and a green boat using intersect:

```
SELECT sname
FROM Sailors s
WHERE s.sid IN (
   SELECT r.sid
   FROM Reserves r
   JOIN Boats b ON r.bid = b.bid
   WHERE b.color = 'red'
)
INTERSECT
SELECT sname
FROM Sailors s
WHERE s.sid IN (
   SELECT r.sid
   FROM Reserves r
   JOIN Boats b ON r.bid = b.bid
   WHERE b.color = 'green'
);
```

-

v) Find the sailors with the highest rating using all operator:

```
SELECT sname
FROM Sailors
```

```
WHERE rating = (SELECT MAX(rating) FROM Sailors);
```

---COMPLEX---

8. Write a PL/SQL function which returns number of sailors for a given rating level.
Creation of Function

```
CREATE OR REPLACE FUNCTION sailcount
(low IN Sailors.Rating%TYPE, high IN Sailors.Rating %TYPE)
RETURN INT IS cnt INT;
BEGIN
SELECT COUNT(*) INTO cnt FROM Sailors WHERE Rating BETWEEN low AND high;
RETURN cnt;
END sailcount;
```

Execution of Function

```
DECLARE
L INT;
H INT;
cnt INT;
BEGIN
L := :L;
H := :H;
cnt:= sailcount(L,H);
dbms_output.put_line(cnt);
END;
```

----

9. write a pl/SQL trigger which will calculate the total marks and percentage of students after insert/update the details of a student in database.
Creation of Table:

```
CREATE TABLE Stu (Rollno INT PRIMARY KEY, Name VARCHAR(10), M1 INT, M2 INT, M3
INT, TOTAL INT, PERCENT FLOAT);
```

Creation of Trigger:

```
CREATE OR REPLACE TRIGGER stutotal
BEFORE
INSERT ON Stu
FOR EACH ROW
BEGIN
:new.TOTAL := :new.M1+:new.M2+:new.M3;
:new.PERCENT := :new.TOTAL/3;
END;
```

Insertion of Records:

```
INSERT INTO Stu(Rollno, Name, M1, M2,M3) values(503, 'AAA', 20, 30, 80);
INSERT INTO Stu(Rollno, Name, M1, M2,M3) values(502, 'AAA', 100, 90, 60);
INSERT INTO Stu(Rollno, Name, M1, M2,M3) values(501, 'AAA', 50, 40, 40);
```

Displaying of Table:

```
SELECT * FROM Stu;
```

10.write a PL/SQL procedure to check whether the given number is prime or not.
Creation of Procedure

```
CREATE OR REPLACE PROCEDURE isprime
(A IN INT) IS
FLAG INT := 0;
BEGIN
```

```
FOR COUNTER IN 2..A-1 LOOP
IF A MOD COUNTER = 0 THEN
  dbms_output.put_line('Not Prime');
  FLAG:=1; EXIT;
END IF;
END LOOP;
IF FLAG = 0 THEN
dbms_output.put_line('Prime');
END IF;
END isprime;
```

Execution of Procedure

```
DECLARE
A INT;
BEGIN
A:= :A;
isprime(A);
END;
```

11.write a PL/SQL function which returns average age of sailors for a given rating level.

Creation of Function

```
CREATE OR REPLACE FUNCTION avgage
(low IN Sailors.Rating%TYPE, high IN Sailors.Rating %TYPE)
RETURN INT IS
cnt INT;
BEGIN
SELECT AVG(Age) INTO cnt FROM Sailors WHERE Rating BETWEEN low AND high;
RETURN cnt;
END avgage;
```

Execution of Function

```
DECLARE
L INT;
H INT;
cnt INT;
BEGIN L := :L;
H := :H;
cnt:= avgage(L,H);
dbms_output.put_line(cnt);
END;
```

12.Write a PL/SQL block using implicit cursor that will display message, the salaries of all the employees in the 'employee' table are updated. If none of the employee's salary are updated, we get a message 'None of the salaries were updated'. Else we get a message like for example, 'Salaries for 1000 employees are updated' if there are 1000 rows in 'employee'table.

```
DECLARE
   cnt NUMBER(10) := 0;
BEGIN
   UPDATE emp SET sal = sal* 1.1;
   cnt:= SQL%ROWCOUNT;
   IF cnt = 0 THEN
      DBMS_OUTPUT.PUT_LINE('None of the salaries were updated.');
   ELSE
```

```
          DBMS_OUTPUT.PUT_LINE('Salaries for ' || cnt || ' employees are updated.');
      END IF;
end;
```
----

13. Write a Cursor to find an employee with the given job and dept no.

```
declare
dno number(10):= :Dept_No;
j varchar2(32):= :JOB;
cursor c5 is select * from emp where job=j and
deptno=dno;
rec emp%rowtype;
begin open c5;
dbms_output.put_line('....LIST OF EMPLOYEES ----');
loop
fetch c5 into rec;
exit when c5%notfound;
dbms_output.put_line( 'Emp name='||rec.ename||' JOB=' || rec.job || ' Dept No= '||rec.deptno);
end loop;
close c5;
end;
```
----

14. Write PL/SQL a function which returns week day of a given date .

Creation of Function

```
CREATE OR REPLACE FUNCTION weekday
(date IN DATE) RETURN VARCHAR IS
day VARCHAR(10);
BEGIN
day := TO_CHAR(date, 'DAY');
RETURN day;
END;
```

Execution of Function

```
DECLARE
A DATE;
day VARCHAR(10);
BEGIN A:= :A;
day := weekday(A);
dbms_output.put_line('Weekday: '||day);
END;
```

--ADDITIONAL—

15. Write a Cursor to Display the employee names and their salary for the given department number.

```
DECLARE
CURSOR c1 IS SELECT * FROM emp WHERE deptno = :D_NO;
rec emp%rowtype;
BEGIN
OPEN c1;
LOOP
fetch c1 into rec;
exit when c1%notfound;
dbms_output.put_line('Dept No: ' ||rec.deptno || 'Employee Name: '||rec.ename|| ' Employee Salary:
'||rec.sal);
END LOOP;
CLOSE c1;
END;
```

---

16. write a procedure to find the lucky number of a given birth date .

Creation of Procedure:

```
CREATE OR REPLACE PROCEDURE GetLuckyNumber (
   p_birth_date IN DATE
) AS
   v_lucky_number NUMBER := 0;
BEGIN
   FOR i IN 1 .. LENGTH(TO_CHAR(p_birth_date, 'DDMMYYYY')) LOOP
      v_lucky_number := v_lucky_number +
         TO_NUMBER(SUBSTR(TO_CHAR(p_birth_date, 'DDMMYYYY'), i, 1));
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('Lucky Number for ' || TO_CHAR(p_birth_date, 'DD-MM-YYYY') || '
is: ' || v_lucky_number);
END GetLuckyNumber;
```

Executing the procedure

```
DECLARE
   v_birth_date DATE := TO_DATE('19901225', 'YYYYMMDD'); -- Example birthdate
BEGIN
   GetLuckyNumber(v_birth_date);
END;`
```

---

17. Write a trigger that keeps backup of deleted records of emp_trig table. Deleted records of emp_trigger inserted in emp_backup table.

Creation of Tables:

```
CREATE TABLE Emp(Eid INT PRIMARY KEY, Ename VARCHAR(10), Sal INT);
CREATE TABLE Backup(Eid INT PRIMARY KEY, Ename VARCHAR(10), Sal INT);
```

Creation of Trigger:

```
CREATE OR REPLACE TRIGGER bckup
BEFORE
DELETE ON Emp
FOR EACH ROW
```

```
BEGIN
INSERT INTO Backup VALUES(:old.Eid, :old.Ename, :old.Sal);
END;
```
Insertion of Records:
```
INSERT INTO Emp VALUES(101, 'aaa', 10000);
INSERT INTO Emp VALUES(102, 'bbb', 20000);
INSERT INTO Emp VALUES(103, 'ccc', 30000);
INSERT INTO Emp VALUES(104, 'ddd', 12000);
INSERT INTO Emp VALUES(105, 'eee', 40000);
```
Displaying of Table – Before Deletion:
```
SELECT * FROM Emp;
SELECT * FROM Backup;
```
Deletion of Records:
```
DELETE FROM Emp WHERE Eid=101;
DELETE FROM Emp WHERE Eid=102;
```
Displaying of Table - Before Deletion:
```
SELECT * FROM Emp;
SELECT * FROM Backup;
```
----

18. write a Cursor to display the list of employees who are working as a manager or Analyst.
```
declare
cursor c4 is select * from emp where job in('MANAGER','ANALYST');
rec emp%rowtype;
begin
open c4;
dbms_output.put_line('....LIST OF EMPLOYEES WHO ARE MANAGERS OR ANALYSTS');
loop
fetch c4 into rec;
exit when c4%notfound;
dbms_output.put_line('Emp name= '||rec.ename||' JOB='|| rec.job);
end loop;
close c4;
end;
```
----

19. write a procedure to check whether the given number is palindrome or not
Creation of procedure:
```
CREATE OR REPLACE PROCEDURE check_palindrome (num IN NUMBER, is_palindrome OUT
BOOLEAN) IS
   original_num VARCHAR2(100);
   reversed_num VARCHAR2(100) := '';
BEGIN
   original_num := TO_CHAR(num);
   FOR i IN REVERSE 1..LENGTH(original_num) LOOP
      reversed_num := reversed_num || SUBSTR(original_num, i, 1);
   END LOOP;
   IF original_num = reversed_num THEN
      is_palindrome := TRUE;
   ELSE
      is_palindrome := FALSE;
   END IF;
END;
```

Execution:

```
DECLARE
   is_pal BOOLEAN;
BEGIN
   check_palindrome(12321, is_pal);
   IF is_pal THEN
      DBMS_OUTPUT.PUT_LINE('The number is a palindrome.');
   ELSE
      DBMS_OUTPUT.PUT_LINE('The number is not a palindrome.');
   END IF;
END;
```

20.Write a Cursor to find an employee with the given job and dept no.

```
declare
dno number(10):= :Dept_No;
j varchar2(32):= :JOB;
cursor c5 is select * from emp where job=j and
deptno=dno;
rec emp%rowtype;
begin open c5;
dbms_output.put_line('....LIST OF EMPLOYEES ----');
loop
fetch c5 into rec;
exit when c5%notfound;
dbms_output.put_line( 'Emp name='||rec.ename||' JOB=' || rec.job || ' Dept No= '||rec.deptno);
end loop;
close c5;
end;
```
