Exp No: 5 ***COMPLEX SQL QUERIES - Date,Strings,Joins,Subquery***

Aim:

To execute the given Queries and to perform different Join operation for the application chosen .

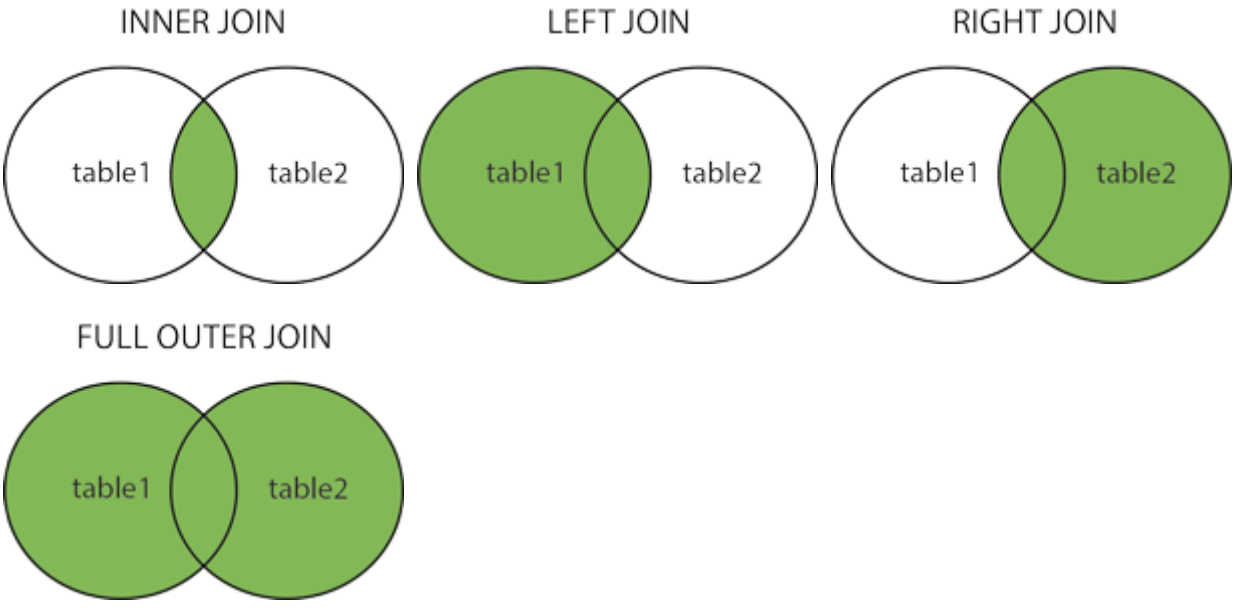
Description:

SQL JOIN

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Different Types of SQL JOINs

- (INNER) JOIN: Returns records that have matching values in both tables
- LEFT (OUTER) JOIN: Return all records from the left table, and the matched records from the right table
- RIGHT (OUTER) JOIN: Return all records from the right table, and the matched records from the left table
- FULL (OUTER) JOIN: Return all records when there is a match in either left or right table




```
SQL> select * from customer;

     US_ID US_NAME    DOB
---------- ---------- ----------
        11 Nivas      03.03.2005
        12 Parvatha   05.04.2004
        13 Vaishu     15.09.2003
        14 Raju       21.10.1999
        20 Geeta      28.02.2003
```

```
SQL> select * from workout112;

    EX_ID EX_NAME                WATERIN
--------- ------------------ ----------
        1 Plank                       5
        3 Inverted row                5
        4 Goblet Squat                4
        2 Squiz                       5
```

```
SQL> select * from ExPlan;

     EX_ID      US_ID       SETS       REPS BREAK
---------- ---------- ---------- ---------- -----
         1         11          3          3 60s
         2         12          3         12 60s
         3         13          3         45 60s
         4         14          4         12 60s
         5         15          3         45 60s
```

INNER JOIN:

Syntax:

SELECT *column_name(s)*FROM *table1* INNER JOIN *table2*

ON *table1.column_name* =

table2.column_name;

```
SQL> select ExPlan.Ex_Id,Customer.Us_Name from ExPlan inner join Customer on ExPlan.Us_Id=Customer.Us_Id;

     EX_ID US_NAME
---------- ----------
         1 Nivas
         2 Parvatha
         3 Vaishu
         4 Raju
```

```
SQL> select ExPlan.Ex_Id,Customer.Us_Name from Customer inner join ExPlan on ExPlan.Us_Id=Customer.Us_Id;

     EX_ID US_NAME
---------- ----------
         1 Nivas
         2 Parvatha
         3 Vaishu
         4 Raju
```

```
SQL> select ExPlan.Ex_Id,Customer.Us_Name from Customer inner join ExPlan on ExPlan.Us_Id=Customer.Us_Id order by Us_Name;

     EX_ID US_NAME
---------- ----------
         1 Nivas
         2 Parvatha
         4 Raju
         3 Vaishu
```

SQL LEFT JOIN Keyword

The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.

Syntax:

SELECT *column_name(s)* FROM *table1* LEFT JOIN *table2* ON *table1.column_name* = *table2.column_name*;

```
SQL> select ExPlan.Ex_Id,Customer.Us_Name from ExPlan left join Customer on ExPlan.Us_Id=Customer.Us_Id;

     EX_ID US_NAME
---------- ----------
         1 Nivas
         2 Parvatha
         3 Vaishu
         4 Raju
         5

SQL> select ExPlan.Ex_Id,Customer.Us_Name from Customer left join ExPlan on ExPlan.Us_Id=Customer.Us_Id;

     EX_ID US_NAME
---------- ----------
         1 Nivas
         2 Parvatha
         3 Vaishu
         4 Raju
```

SQL RIGHT JOIN Keyword

The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.

Syntax

SELECT *column_name(s)* FROM *table1* RIGHT JOIN *table2*
ON *table1.column_name* =
table2.column_name;

```
SQL> select ExPlan.Ex_Id,Customer.Us_Name from ExPlan right join Customer on ExPlan.Us_Id=Customer.Us_Id;

     EX_ID US_NAME
---------- ----------
         1 Nivas
         2 Parvatha
         3 Vaishu
         4 Raju
           Geeta

SQL> select ExPlan.Ex_Id,Customer.Us_Name from Customer right join ExPlan on ExPlan.Us_Id=Customer.Us_Id;

     EX_ID US_NAME
---------- ----------
         1 Nivas
         2 Parvatha
         3 Vaishu
         4 Raju
         5

SQL> select ExPlan.Ex_Id,Customer.Us_Name from Customer right join ExPlan on ExPlan.Us_Id=Customer.Us_Id order by Us_Name;

     EX_ID US_NAME
---------- ----------
         1 Nivas
         2 Parvatha
         4 Raju
         3 Vaishu
         5
```

SQL FULL OUTER JOIN

The FULL OUTER JOIN keyword return all records when there is a match in either left (table1) or right (table2) table records.

SYNTAX

SELECT *column_name(s)* FROM *table1* FULL OUTER JOIN *table2*
ON *table1.column_name* = *table2.column_name*;

```
SQL> select ExPlan.Ex_Id,Customer.Us_Name,ExPlan.Sets from Customer full outer join ExPlan on ExPlan.Us_Id=Customer.Us_Id;

     EX_ID US_NAME          SETS
---------- ---------- ----------
         1 Nivas               3
         2 Parvatha            3
         3 Vaishu              3
         4 Raju                4
         5                     3
           Geeta

6 rows selected.

SQL> select ExPlan.Ex_Id,Customer.Us_Name,ExPlan.Sets from Customer full outer join ExPlan on ExPlan.Us_Id=Customer.Us_Id order by Us_Name;

     EX_ID US_NAME          SETS
---------- ---------- ----------
           Geeta
         1 Nivas               3
         2 Parvatha            3
         4 Raju                4
         3 Vaishu              3
         5                     3

6 rows selected.
```

SQL Self JOIN

A self-JOIN is a regular join, but the table is joined with itself.

Syntax

SELECT *column_name(s)* FROM *table1 T1, table1 T2* WHERE *condition*;

```
SQL> select A.Us_Name as Us1,B.Us_Name as Us2, A.DOB from Customer A,Customer B where A.Us_Id>B.Us_Id;

US1        US2        DOB
---------- ---------- ----------
Parvatha   Nivas      05.04.2004
Vaishu     Nivas      15.09.2003
Vaishu     Parvatha   15.09.2003
Raju       Nivas      21.10.1999
Raju       Parvatha   21.10.1999
Raju       Vaishu     21.10.1999
Geeta      Nivas      28.02.2003
Geeta      Parvatha   28.02.2003
Geeta      Vaishu     28.02.2003
Geeta      Raju       28.02.2003

10 rows selected.
```

```
SQL> select A.Us_Name as Us1,B.Us_Name as Us2, A.DOB from Customer A,Customer B where A.Us_Id>=B.Us_Id and A.Us_Name=B.Us_Name order by A.Us_Id;

US1        US2        DOB
---------- ---------- ----------
Nivas      Nivas      03.03.2005
Parvatha   Parvatha   05.04.2004
Vaishu     Vaishu     15.09.2003
Raju       Raju       21.10.1999
Geeta      Geeta      28.02.2003
```

SQL UNION Operator

The UNION operator is used to combine the result-set of two or more SELECT statements.

- Each SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in each SELECT statement must also be in the same order

Syntax

SELECT *column_name(s)* FROM *table1* UNION SELECT *column_name(s)* FROM *table2*;

```
SQL> select Us_Id from Customer Union select Us_Id from ExPlan order by Us_Id;

     US_ID
----------
        11
        12
        13
        14
        15
        20

6 rows selected.

SQL> select Us_Id from Customer Union select Ex_Id from ExPlan order by Us_Id;

     US_ID
----------
         1
         2
         3
         4
         5
        11
        12
        13
        14
        20

10 rows selected.
```

Union all: The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL:

Syntax

SELECT *column_name(s)* FROM *table1* UNION ALL SELECT *column_name(s)* FROM *table2*;

```
SQL> select Us_Id from Customer Union all select Ex_Id from ExPlan order by Us_Id;

     US_ID
----------
         1
         2
         3
         4
         5
        11
        12
        13
        14
        20
```

SQL Aliases

SQL aliases are used to give a table, or a column in a table, a temporary name.Aliases are often used to make column names more readable.An alias only exists for the duration of the query.

Alias Column Syntax

SELECT *column_name* AS *alias_name* FROM *table_name;*

```
SQL> select Us_Name as Usname from Customer
  2  ;

USNAME
----------
Nivas
Parvatha
Vaishu
Raju
Geeta
```

Alias Table Syntax

SELECT *column_name(s)* FROM *table_name* AS *alias_name;*

QUERIES:

Customer Table:

customer_id cust_name city grade salesman_id

3002 Nick Rimando New York 100 5001

3005 Graham Zusi California 200 5002

3001 Brad Guzan London 100 5005

3004 Fabian Johns Paris 300 5006

3007 Brad Davis New York 200 5001

3009 Geoff Camero Berlin 100 5003

3008 Julian Green London 300 5002

3003 Jozy Altidor Moncow 200 5007

Orders Table:

ord_no purch_amt ord_date customer_id salesman_id

70001 150.5 2012-10-05 3005 5002

70009 270.65 2012-09-10 3001 5005

70002 65.26 2012-10-05 3002 5001

70004 110.5 2012-08-17 3009 5003

70007 948.5 2012-09-10 3005 5002

70005 2400.6 2012-07-27 3007 5001

70008 5760 2012-09-10 3002 5001

70010 1983.43 2012-10-10 3004 5006

70003 2480.4 2012-10-10 3009 5003

70012 250.45 2012-06-27 3008 5002

70011 75.29 2012-08-17 3003 5007

70013 3045.6 2012-04-25 3002 5001

Salesman Table:

salesman_id name city commission

5001 James Hoog New York 0.15

5002 Nail Knite Paris 0.13

5005 Pit Alex London 0.11

5006 Mc Lyon Paris 0.14

5003 Lauson Hen San Jose 0.12

5007 Paul Adam Rome 0.13

```
SQL> select * from customerr;

CUST CNAME CITY        GRADE SALE
---- ----- ------ ---------- ----
3002 nick  newyok         100 5001
3005 graha califo         200 5002
3001 brad  london         100 5005
3004 johns paris          300 5006
3007 davis newyok         200 5001
3009 geoff berlin         100 5003
3008 julia london         300 5002
3003 jozy  moncow         200 5007

8 rows selected.
```

```
SQL> select * from salesman;

SALE NAME   S_CITY COMM
---- ------ ------ ----
5001 james  newyok 0.15
5007 paul   rome   0.13
5002 nail   paris  0.13
5003 lauson Sanjos 0.12
5005 pit    london 0.11
5006 mclyon paris  0.14

6 rows selected.
```

```
SQL> select * from orders;

ORD_N  PURCH_AMT ORD_DATE            CUST SALE
----- ---------- ------------------- ---- ----
70001        151 2012-09-10          3001 5005
70009        271 2012-10-05          3002 5001
70002       65.3 2012-08-17          3009 5003
70004        111 2012-09-10          3005 5002
70007        949 2012-07-27          3007 5001
70005       2400 2012-10-05          3005 5002
70008       5760 2012-09-10          3002 5001
70010       1980 2012-10-10          3004 5006
70003       2480 2012-10-10          3009 5003
70012        250 2012-06-27          3008 5002
70013       3050 2012-04-25          3002 5001

11 rows selected.
```

Boolean and Relational operators

1. Write a query to display all customers with a grade above 100.

```
SQL> select * from customerr where grade>100;

CUST CNAME CITY        GRADE SALE
---- ----- ------ ---------- ----
3005 graha califo         200 5002
3004 johns paris          300 5006
3007 davis newyok         200 5001
3008 julia london         300 5002
3003 jozy  moncow         200 5007
```

2. Write a query statement to display all customers in New York who have a grade value above 100

```
SQL> select * from customerr where grade>100 and city='newyok';

CUST CNAME CITY        GRADE SALE
---- ----- ------ ---------- ----
3007 davis newyok        200 5001
```

3. Write a SQL statement to display all customers, who are either belongs to the city New York or had a grade above 100.

```
SQL> select * from customerr where grade>100 or city='newyok';

CUST CNAME CITY        GRADE SALE
---- ----- ------ ---------- ----
3002 nick  newyok        100 5001
3005 graha califo        200 5002
3004 johns paris         300 5006
3007 davis newyok        200 5001
3008 julia london        300 5002
3003 jozy  moncow        200 5007
```

4. Write a SQL statement to display all the customers, who are either belongs to the city New York or not had a grade above 100.

```
SQL> select * from customerr where grade<=100 or city='newyok';

CUST CNAME CITY        GRADE SALE
---- ----- ------ ---------- ----
3002 nick  newyok        100 5001
3001 brad  london        100 5005
3007 davis newyok        200 5001
3009 geoff berlin        100 5003
```

5. Write a SQL query to display those customers who are neither belongs to the city New York nor grade value is more than 100.

```
SQL> select * from customerr where grade<=100 or city!='newyok';

CUST CNAME CITY        GRADE SALE
---- ----- ------ ---------- ----
3002 nick  newyok        100 5001
3005 graha califo        200 5002
3001 brad  london        100 5005
3004 johns paris         300 5006
3009 geoff berlin        100 5003
3008 julia london        300 5002
3003 jozy  moncow        200 5007
```

6. Write a SQL statement to display either those orders which is not issued on date 2012-09-10 and issued by the salesman whose ID is 505 and below or those orders which purchase amount is 1000.00 and below.

ORD_N	PURCH_AMT	ORD_DATE	CUST	SALE
70001	151	2012-09-10	3001	5005
70009	271	2012-10-05	3002	5001
70002	65.3	2012-08-17	3009	5003
70004	111	2012-09-10	3005	5002
70007	949	2012-07-27	3007	5001
70005	2400	2012-10-05	3005	5002
70003	2480	2012-10-10	3009	5003
70012	250	2012-06-27	3008	5002
70013	3050	2012-04-25	3002	5001

7. Write a SQL statement to display salesman_id, name, city and commission who gets the commission within the range more than 0.10% and less than 0.12%.

```
SQL> select * from salesman where commission > '0.10' and commission < '0.12';

SALE NAME   S_CITY COMM
---- ------ ------ ----
5005 pit    london 0.11
```

8. Write a SQL statement to display all information where purchase amount less than a specified amount or reverse order date greater than or equal to a specified date and customer id less than a specified number

```
SQL> select * from orders where(purch_amt<2000 or ord_date>='2012-09-10' and cust_id<300
4);

ORD_N  PURCH_AMT ORD_DATE            CUST SALE
----- ---------- ------------------- ---- ----
70001        151 2012-09-10          3001 5005
70009        271 2012-10-05          3002 5001
70002       65.3 2012-08-17          3009 5003
70004        111 2012-09-10          3005 5002
70007        949 2012-07-27          3007 5001
70008       5760 2012-09-10          3002 5001
70010       1980 2012-10-10          3004 5006
70012        250 2012-06-27          3008 5002

8 rows selected.
```

9. Write a SQL query to display all orders where purchase amount less than a specified amount or reverse orders in a specified date and customer ID less than a specified number.

```
SQL> select * from orders where(purch_amt<2000 or (ord_date='2012-09-10' and cust_id<300
4));

ORD_N  PURCH_AMT ORD_DATE            CUST SALE
----- ---------- ------------------- ---- ----
70001        151 2012-09-10          3001 5005
70009        271 2012-10-05          3002 5001
70002       65.3 2012-08-17          3009 5003
70004        111 2012-09-10          3005 5002
70007        949 2012-07-27          3007 5001
70008       5760 2012-09-10          3002 5001
70010       1980 2012-10-10          3004 5006
70012        250 2012-06-27          3008 5002

8 rows selected.
```

10. Display all reverse orders datew where order dates equal to a specified date or customer id greater than a specified number and purchase amount less than a specified amount

```
SQL> select ord_date from orders where(ord_date='2012-09-10' or cust_id>3004 and purch_a
mt<1000);

ORD_DATE
--------------------
2012-09-10
2012-08-17
2012-09-10
2012-07-27
2012-09-10
2012-06-27

6 rows selected.
```

11. Write a SQL query to display order number, purchase amount,forthose order which exceeds the 50% of target value of 6000.

```
SQL> select ord_no,purch_amt from orders where(purch_amt>((50/100)*6000));

ORD_N  PURCH_AMT
----- ----------
70008       5760
70013       3050
```

Aggregate Functions and Group by

Create a table employee with the following fields:

EMPLOYEE_ID | FIRST_NAME | LAST_NAME | EMAIL | PHONE_NUMBER | HIRE_DATE | JOB_ID

```
SQL> select * from Employee;

     EM_ID FN         LN         HIRE_DATE  JOB_I     SALARY     MAN_ID DP
---------- ---------- ---------- ---------- ----- ---------- ---------- --
         1 Aravin     R          12.12.2012 A1         24000        100 90
         2 Nivas      Renga      1.10.2012  A2         17000        101 50
         3 Geeta      Renga      03.1.2011  B1          9000        101 60
         4 Renga      Nathan     03.1.2022  A1          6000        103 60
         5 Viji       G          13.7.2023  C1         48000        103 90
```

1. Write a query to list the number of jobs available in the employees table.

```
SQL> select count(distinct Job_Id) from Employee;

COUNT(DISTINCTJOB_ID)
---------------------
                    4
```

2. Write a query to get the total salaries payable to employees.

```
SQL> select sum(salary) from Employee;

SUM(SALARY)
-----------
     104000
```

3. Write a query to get the minimum salary from employees table

```
SQL> select min(salary) from Employee;

MIN(SALARY)
-----------
       6000
```

4. Write a query to get the maximum salary of an employee working as a Programmer.

```
SQL> select max(salary) from Employee;

MAX(SALARY)
-----------
      48000
```

5. Write a query to get the average salary and number of employees working the department 90.

```
SQL> select Dp,avg(salary),count(*) from Employee where Dp='90' group by Dp order by avg(salary);

DP AVG(SALARY)   COUNT(*)
-- ----------- ----------
90      36000          2
```

6. Write a query to get the highest, lowest, sum, and average salary of all employees

```
SQL> select max(salary),min(salary),avg(salary),sum(salary) from
Employee;

MAX(SALARY) MIN(SALARY) AVG(SALARY) SUM(SALARY)
----------- ----------- ----------- -----------
      48000        6000       20800      104000
```

7. Write a query to get the number of employees with the same job.

```
SQL> select Job_Id, count(*) from Employee group by Job_Id;

JOB_I   COUNT(*)
----- ----------
A2             1
A1             2
B1             1
C1             1
```

8. Write a query to get the difference between the highest and lowest salaries

```
SQL> select max(salary)-min(salary) difference from Employee;

DIFFERENCE
----------
     42000
```

9. Write a query to find the manager ID and the salary of the lowest-paid employee for that manager.

```
SQL> select Man_Id,min(salary) from Employee where Man_Id is not
null group by Man_Id order by min(salary) desc;

    MAN_ID MIN(SALARY)
---------- ------------
       100        24000
       101         9000
       103         6000
```

10. Write a query to get the department ID and the total salary payable in each department

```
SQL> select Dp,sum(salary) from Employee where Dp is not null group by Dp order by sum(salary);

DP SUM(SALARY)
-- -----------
60       15000
50       17000
90       72000
```

```
SQL> select * from Employee;

     EM_ID FN         LN           SALARY PH         EMAIL
---------- ---------- ---------- ---------- ---------- ---------------
         1 Aravin     Mac             24000 9999456790 ar@example.com
         2 Nivas      Renga           17000 9126784212 ni@example.com
         3 Geeta      Renga            9000 9143424212 ge@example.com
         4 Renga      Nathan           6000 6743210986 re@example.com
         5 Viji       G               48000 6543290876 vi@example.com
```

```
SQL> select * from Employee;

     EM_ID FN         LN           SALARY STREET     LOCATION
---------- ---------- ---------- ---------- ---------- ----------
         1 Aravin     Mac             24000 Abbey      California
         2 Nivas      Renga           17000 Pendlee    US
         3 Geeta      Renga            9000 Pitt       California
         4 Renga      Nathan           6000 Bal        Europe
         5 Viji       G               48000 Churh      India
```

String functions:

1. Write a query to update the portion of the phone_number in the employees table, within the phone number the substring '124' will be replaced by '999'.

```
SQL> update Employee set Ph=replace(Ph,'123','999')
  2  where Ph like '%123%';

1 row updated.

SQL> select * from Employee;

     EM_ID FN         LN           SALARY PH         EMAIL
---------- ---------- ---------- ---------- ---------- ---------------
         1 Aravin     R               24000 9999456790 ar
         2 Nivas      Renga           17000 9126784212 ni
         3 Geeta      Renga            9000 9143424212 ge
         4 Renga      Nathan           6000 6743210986 re
         5 Viji       G               48000 6543290876 vi
```

2. Write a query to get the details of the employees where the length of the first name greater than or equal to 8

```
SQL> select * from Employee where length(FN)>=5;

     EM_ID FN         LN           SALARY PH         EMAIL
---------- ---------- ---------- ---------- ---------- ---------------
         1 Aravin     R               24000 9999456790 ar
         2 Nivas      Renga           17000 9126784212 ni
         3 Geeta      Renga            9000 9143424212 ge
         4 Renga      Nathan           6000 6743210986 re
```

3. Write a query to display leading zeros before maximum and minimum salary.

```
SQL> select * from Employee;

     EM_ID FN         LN            MAX_SAL    MIN_SAL
---------- ---------- ---------- ---------- ----------
         1 Aravin     Mac             24000      10000
         2 Nivas      Renga           17000       9000
         3 Geeta      Renga            9000       5000
         4 Renga      Nathan           6000       3000
         5 Viji       G               48000      33000
```

```
SQL> select Em_Id, lpad(Max_sal,7,'0') ,lpad(Min_sal,7,'0') from Employee;

    EM_ID LPAD(MAX_SAL,7,'0')         LPAD(MIN_SAL,7,'0')
--------- ------------------------- -------------------------
        1 0024000                   0010000
        2 0017000                   0009000
        3 0009000                   0005000
        4 0006000                   0003000
        5 0048000                   0033000
```

4. Write a query to append '@example.com' to email field.

```
SQL> update Employee set Email=concat(Email,'@example.com');

5 rows updated.

SQL> select * from Employee;

     EM_ID FN         LN            SALARY PH         EMAIL
---------- ---------- ---------- ---------- ---------- ---------------
         1 Aravin     R               24000 9999456790 ar@example.com
         2 Nivas      Renga           17000 9126784212 ni@example.com
         3 Geeta      Renga            9000 9143424212 ge@example.com
         4 Renga      Nathan           6000 6743210986 re@example.com
         5 Viji       G               48000 6543290876 vi@example.com
```

5. Write a query to get the employee id, email id (discard the last three characters)

```
SQL> select Em_Id, reverse(substr(reverse(Email),4)) as email from Employee;

     EM_ID EMAIL
---------- -----------------------------------------------
         1 ar@example.
         2 ni@example.
         3 ge@example.
         4 re@example.
         5 vi@example.
```

6. Write a query to extract the last 4 character of phone numbers.

```
SQL> select Ph, substr(ph,7) from Employee;

PH         SUBSTR(PH,7)
---------- ----------------
9999456790 6790
9126784212 4212
9143424212 4212
6743210986 0986
6543290876 0876
```

7. Write a query to get the locations that have minimum street length

```
SQL> select location from Employee where length(Street)<=(select min(length(St
reet)) from Employee);

LOCATION
----------
Europe
```

8. Write a query to display the length of first name for employees where last name contain character 'c' after 2nd position.

```
SQL> select FN,length(FN),LN from Employee where instr(LN,'c')>2;

FN         LENGTH(FN) LN
---------- ---------- ----------
Aravin              6 Mac
```

9. Write a query that displays the first name and the length of the first name for all employees whose name starts with the letters 'A', 'J' or 'M'. Give each column an appropriate label. Sort the results by the employees' first names.

```
SQL> select FN,length(FN) from Employee
  2  where FN like 'A%'
  3  or FN like 'M%'
  4  or FN like 'J%'
  5  order by FN;

FN         LENGTH(FN)
---------- ----------
Aravin              6
```

10. Write a query to display the first name and salary for all employees. Format the salary to be 10 characters long, left-padded with the $ symbol. Label the column SALARY.

```
SQL> select FN,
  2  LPAD(salary,10,'$') SALARY
  3  from Employee;

FN         SALARY
---------- ----------------------------------------
Aravin     $$$$$24000
Nivas      $$$$$17000
Geeta      $$$$$$9000
Renga      $$$$$$6000
Viji       $$$$$48000
```

Date functions :

1. Write a query to display the first day of the month (in datetime format) three months before the current month.

```
SQL> SELECT TRUNC(ADD_MONTHS(TRUNC(SYSDATE, 'MM'), -3)) AS first_day_of_month FROM DUAL;

FIRST_DAY
---------
01-NOV-23
```

2. Write a query to get the distinct Mondays from hire_date in employees tables.

```
SQL> SELECT DISTINCT NEXT_DAY(TO_DATE(Hire_Date, 'YYYY-MM-DD') - 7, 'MONDAY')
AS monday_date
  2  FROM Employee;

MONDAY_DA
---------
29-JAN-24
10-JUN-24
18-NOV-24
06-MAY-24
04-MAR-24
```

3. Write a query to get the last day of the current year

```
SQL> SELECT TRUNC(ADD_MONTHS(TRUNC(SYSDATE, 'YEAR'), 12) - 1) AS last_day_of_year FROM DUAL;

LAST_DAY_
---------
31-DEC-24
```

4. Write a query to get the current date in the following format.

Sample date : 2014-09-04

Output : September 4, 2014

```
SQL> SELECT TO_CHAR(SYSDATE, 'Month DD, YYYY') AS formatted_date FROM DUAL;

FORMATTED_DATE
------------------------------------------
February  23, 2024
```

5. Write a query to get the current date in the following format. Thursday September 2014

```
SQL> SELECT TO_CHAR(SYSDATE, 'Day Month YYYY') AS formatted_date FROM DUAL;

FORMATTED_DATE
------------------------------------------------------------------------------
Friday    February  2024
```

6. Write a query to get the first name and hire date from employees table where hire date between '1987-06-01' and '1987-07-30'

```
SQL> SELECT FN, Hire_Date
  2  FROM Employee
  3  WHERE  Hire_Date BETWEEN '2024-03-01' AND '2024-07-30';

FN         HIRE_DATE
---------- ----------
Nivas      2024-05-12
Geeta      2024-03-07
Viji       2024-06-15
```

Result:

Thus the given Queries were executed and Join operations were performed for the application chosen