PRACTICAL-2

Aim: Implementation of Analytical queries like RollUp, Cube, First, Last, Lead, Lag, Rank, Dense Rank, etc.

Analytical Queries :-

Analytic functions compute an aggregate value based on a group of rows. They differ from aggregate functions in that they return multiple rows for each group. The group of rows is called a window and is defined by the analytic_clause. For each row, a sliding window of rows is defined. The window determines the range of rows used to perform the calculations for the current row. Window sizes can be based on either a physical number of rows or a logical interval such as time.

Analytic functions are the last set of operations performed in a query except for the final ORDER BY clause. All joins and all WHERE, GROUP BY, and HAVING clauses are completed before the analytic functions are processed. Therefore, analytic functions can appear only in the select list or ORDER BY clause.

Analytic functions are commonly used to compute cumulative, moving, centered, and reporting aggregates.

ROLLUP :-

ROLLUP enables a SELECT statement to calculate multiple levels of subtotals across a specified group of dimensions. It also calculates a grand total. ROLLUP is a simple extension to the GROUP BY clause, so its syntax is extremely easy to use. The ROLLUP extension is highly efficient, adding minimal overhead to a query.

Syntax :-

```
SELECT ... GROUP BY
ROLLUP(grouping_column_reference_list)
```

CUBE :-

CUBE enables a SELECT statement to calculate subtotals for all possible combinations of a group of dimensions. It also calculates a grand total. This is the set of information typically needed for all cross-tabular reports, so CUBE can calculate a cross-tabular report with a single SELECT statement. Like ROLLUP, CUBE is a simple extension to the GROUP BY clause, and its syntax is also easy to learn.

Syntax :-

```
SELECT ...  GROUP BY
CUBE (grouping_column_reference_list)
```

FIRST :-

The FIRST functions can be used to return the first value from an ordered sequence. Say we want to display the salary of each employee, along with the highest within their department we may use something like.

Syntax :-

```
Function( ) KEEP (DENSE_RANK FIRST ORDER BY <expr>) OVER
(<partitioning_clause>)
```

LAST :-

The LAST functions can be used to return the last value from an ordered sequence. Say we want to display the salary of each employee, along with the lowest within their department we may use something like.

Syntax :-

```
Function( ) KEEP (DENSE_RANK LAST ORDER BY <expr>) OVER
(<partitioning_clause>)
```

LEAD :-

The LEAD function is used to return data from rows further down result set.

Syntax :-

```
LEAD { ( value_expr [, offset [, default]] ) [ { RESPECT | IGNORE } NULLS ]          |
( value_expr [ { RESPECT | IGNORE } NULLS ] [, offset [, default]] )
 }
 OVER ([ query_partition_clause ] order_by_clause)
```

LAG :- The LAG function is used to access data from a previous row.

Syntax :-

```
LAG { ( value_expr [, offset [, default]]) [ { RESPECT | IGNORE } NULLS ] |          (
value_expr [ { RESPECT | IGNORE } NULLS ] [, offset [, default]] )
}
OVER ([ query_partition_clause ] order_by_clause)
```

RANK :-

Let's assume we want to assign a sequential order, or rank, to people within a department based on salary, we might use the RANK function like this. The basic description for the RANK analytic function is shown below. The analytic clause is described in more detail here.

Syntax :-

RANK() OVER ([query_partition_clause] order_by_clause)

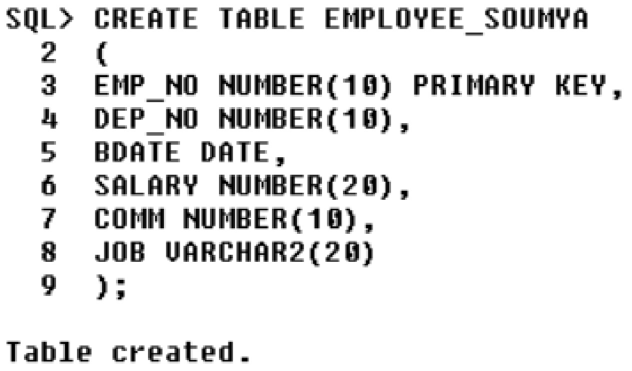
DENSE RANK :-

The DENSE_RANK function acts like the RANK function except that it assigns consecutive ranks, so this is not like olympic medaling. The basic description for the DENSE_RANK analytic function is shown below. The analytic clause is described in more detail here.

Syntax :-

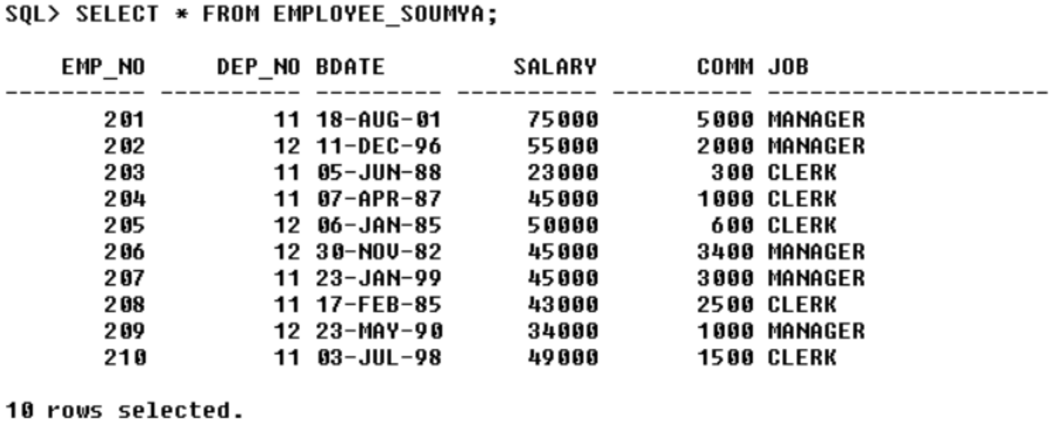
DENSE_RANK() OVER([query_partition_clause] order_by_clause)

Creating table Employee



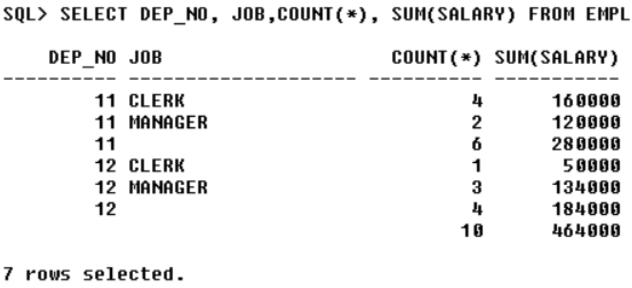
```
SQL> CREATE TABLE EMPLOYEE_SOUMYA
  2  (
  3  EMP_NO NUMBER(10) PRIMARY KEY,
  4  DEP_NO NUMBER(10),
  5  BDATE DATE,
  6  SALARY NUMBER(20),
  7  COMM NUMBER(10),
  8  JOB VARCHAR2(20)
  9  );

Table created.
```

Viewing all records of the table:



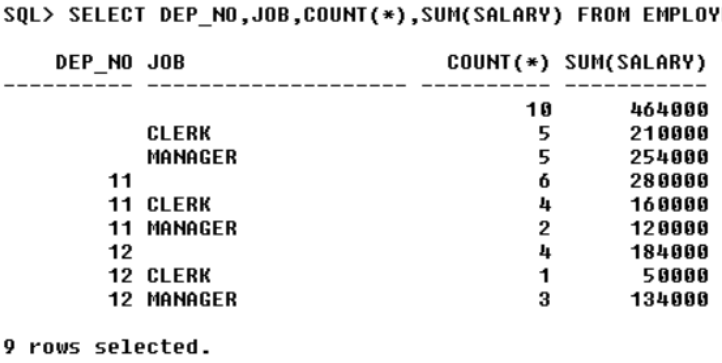
```
SQL> SELECT * FROM EMPLOYEE_SOUMYA;

    EMP_NO     DEP_NO BDATE         SALARY       COMM JOB
---------- ---------- --------- ---------- ---------- --------------------
       201         11 18-AUG-01      75000       5000 MANAGER
       202         12 11-DEC-96      55000       2000 MANAGER
       203         11 05-JUN-88      23000        300 CLERK
       204         11 07-APR-87      45000       1000 CLERK
       205         12 06-JAN-85      50000        600 CLERK
       206         12 30-NOV-82      45000       3400 MANAGER
       207         11 23-JAN-99      45000       3000 MANAGER
       208         11 17-FEB-85      43000       2500 CLERK
       209         12 23-MAY-90      34000       1000 MANAGER
       210         11 03-JUL-98      49000       1500 CLERK

10 rows selected.
```

ROLL-UP:



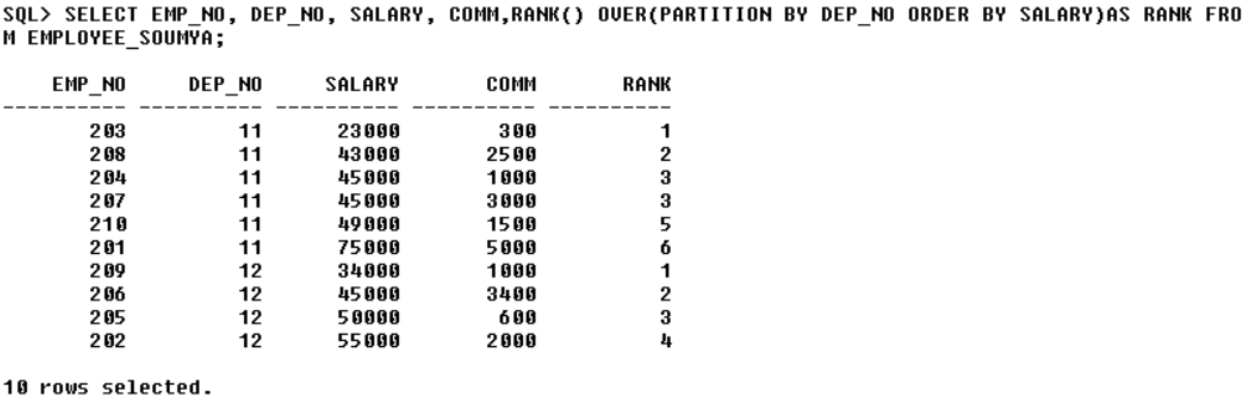
```
SQL> SELECT DEP_NO, JOB,COUNT(*), SUM(SALARY) FROM EMPLOYEE_SOUMYA GROUP BY ROLLUP(DEP_NO,JOB);

    DEP_NO JOB                    COUNT(*) SUM(SALARY)
---------- -------------------- ---------- -----------
        11 CLERK                         4      160000
        11 MANAGER                       2      120000
        11                               6      280000
        12 CLERK                         1       50000
        12 MANAGER                       3      134000
        12                               4      184000
                                        10      464000

7 rows selected.
```

CUBE:



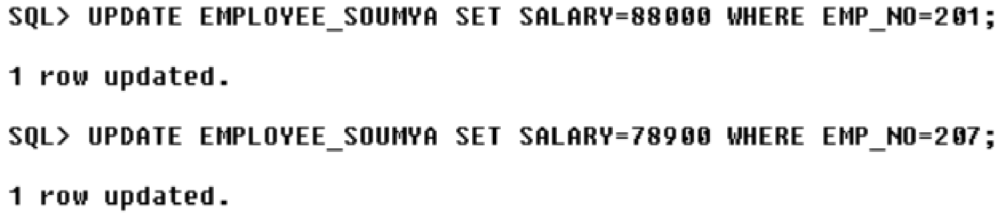
```
SQL> SELECT DEP_NO,JOB,COUNT(*),SUM(SALARY) FROM EMPLOYEE_SOUMYA GROUP BY CUBE (DEP_NO,JOB);

    DEP_NO JOB                    COUNT(*) SUM(SALARY)
---------- -------------------- ---------- -----------
                                        10      464000
           CLERK                         5      210000
           MANAGER                       5      254000
        11                               6      280000
        11 CLERK                         4      160000
        11 MANAGER                       2      120000
        12                               4      184000
        12 CLERK                         1       50000
        12 MANAGER                       3      134000

9 rows selected.
```

RANK:



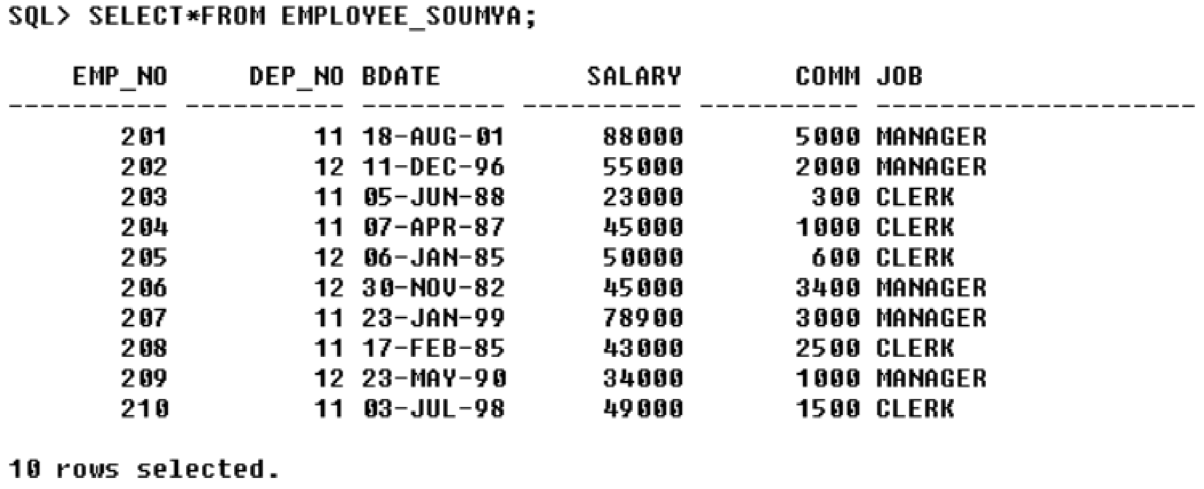
```
SQL> SELECT EMP_NO, DEP_NO, SALARY, COMM,RANK() OVER(PARTITION BY DEP_NO ORDER BY SALARY)AS RANK FRO
M EMPLOYEE_SOUMYA;

    EMP_NO     DEP_NO     SALARY       COMM       RANK
---------- ---------- ---------- ---------- ----------
       203         11      23000        300          1
       208         11      43000       2500          2
       204         11      45000       1000          3
       207         11      45000       3000          3
       210         11      49000       1500          5
       201         11      75000       5000          6
       209         12      34000       1000          1
       206         12      45000       3400          2
       205         12      50000        600          3
       202         12      55000       2000          4

10 rows selected.
```

UPDATE:

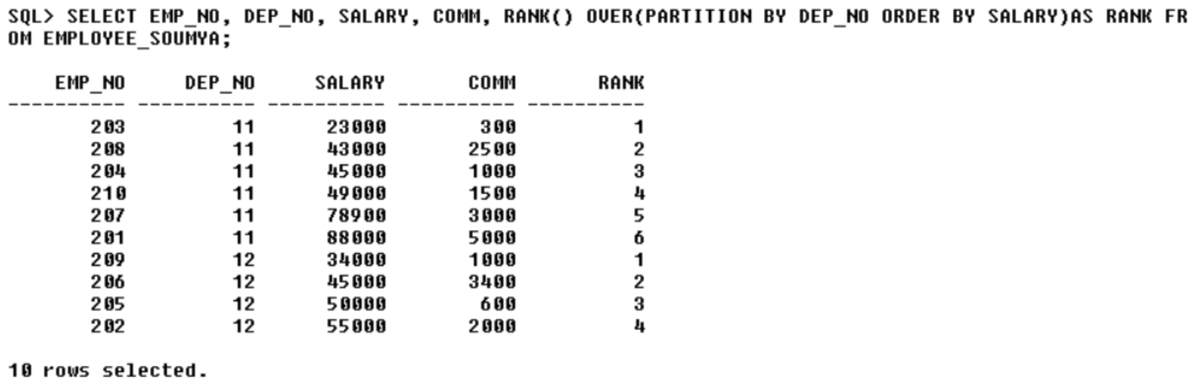


```
SQL> UPDATE EMPLOYEE_SOUMYA SET SALARY=88000 WHERE EMP_NO=201;

1 row updated.

SQL> UPDATE EMPLOYEE_SOUMYA SET SALARY=78900 WHERE EMP_NO=207;

1 row updated.
```

VIEWING ALL RECORDS OF THE TABLE:

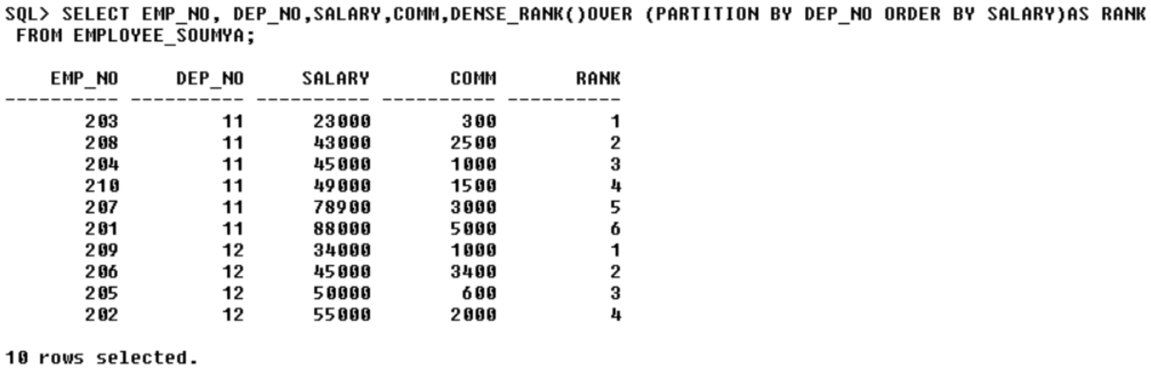


```
SQL> SELECT*FROM EMPLOYEE_SOUMYA;

    EMP_NO     DEP_NO BDATE         SALARY       COMM JOB
---------- ---------- --------- ---------- ---------- --------------------
       201         11 18-AUG-01      88000       5000 MANAGER
       202         12 11-DEC-96      55000       2000 MANAGER
       203         11 05-JUN-88      23000        300 CLERK
       204         11 07-APR-87      45000       1000 CLERK
       205         12 06-JAN-85      50000        600 CLERK
       206         12 30-NOV-82      45000       3400 MANAGER
       207         11 23-JAN-99      78900       3000 MANAGER
       208         11 17-FEB-85      43000       2500 CLERK
       209         12 23-MAY-90      34000       1000 MANAGER
       210         11 03-JUL-98      49000       1500 CLERK

10 rows selected.
```



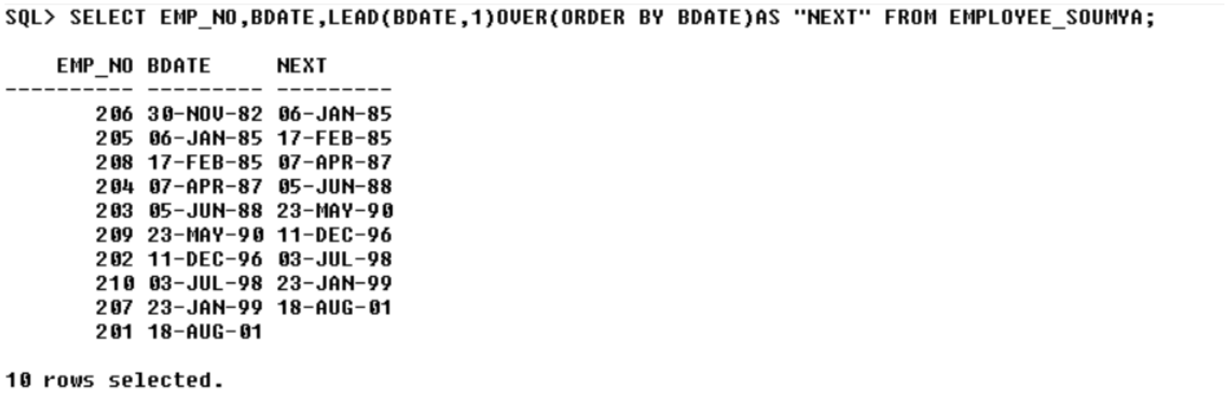
```
SQL> SELECT EMP_NO, DEP_NO, SALARY, COMM, RANK() OVER(PARTITION BY DEP_NO ORDER BY SALARY)AS RANK FR
OM EMPLOYEE_SOUMYA;

    EMP_NO     DEP_NO     SALARY       COMM       RANK
---------- ---------- ---------- ---------- ----------
       203         11      23000        300          1
       208         11      43000       2500          2
       204         11      45000       1000          3
       210         11      49000       1500          4
       207         11      78900       3000          5
       201         11      88000       5000          6
       209         12      34000       1000          1
       206         12      45000       3400          2
       205         12      50000        600          3
       202         12      55000       2000          4

10 rows selected.
```

DENSE RANK:

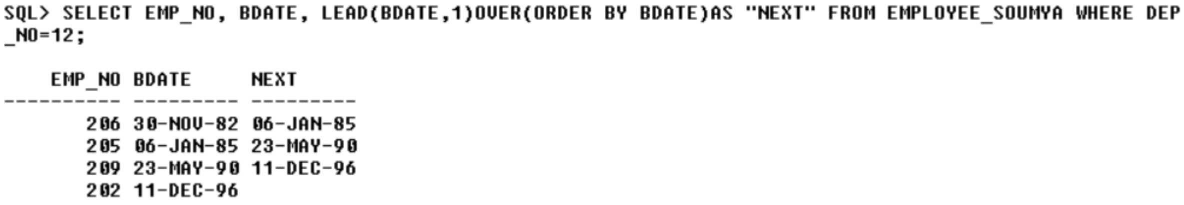


```
SQL> SELECT EMP_NO, DEP_NO,SALARY,COMM,DENSE_RANK()OVER (PARTITION BY DEP_NO ORDER BY SALARY)AS RANK
 FROM EMPLOYEE_SOUMYA;

    EMP_NO     DEP_NO     SALARY       COMM       RANK
---------- ---------- ---------- ---------- ----------
       203         11      23000        300          1
       208         11      43000       2500          2
       204         11      45000       1000          3
       210         11      49000       1500          4
       207         11      78900       3000          5
       201         11      88000       5000          6
       209         12      34000       1000          1
       206         12      45000       3400          2
       205         12      50000        600          3
       202         12      55000       2000          4

10 rows selected.
```

LEAD:

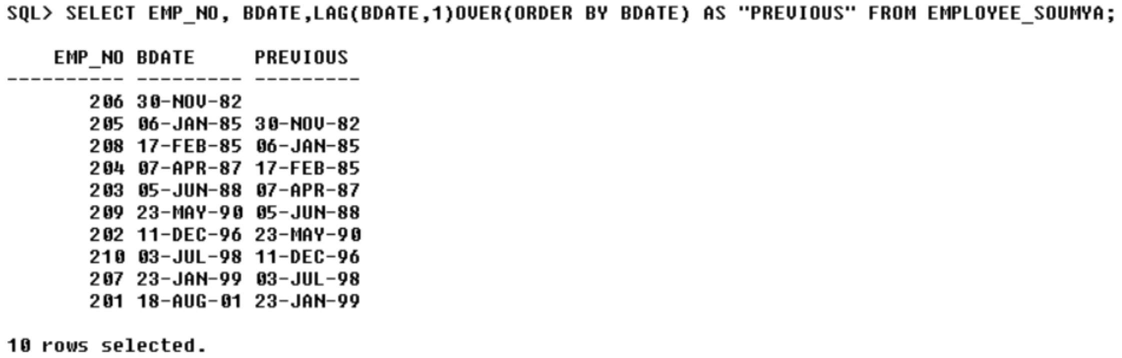


```
SQL> SELECT EMP_NO,BDATE,LEAD(BDATE,1)OVER(ORDER BY BDATE)AS "NEXT" FROM EMPLOYEE_SOUMYA;

    EMP_NO BDATE     NEXT
---------- --------- ---------
       206 30-NOV-82 06-JAN-85
       205 06-JAN-85 17-FEB-85
       208 17-FEB-85 07-APR-87
       204 07-APR-87 05-JUN-88
       203 05-JUN-88 23-MAY-90
       209 23-MAY-90 11-DEC-96
       202 11-DEC-96 03-JUL-98
       210 03-JUL-98 23-JAN-99
       207 23-JAN-99 18-AUG-01
       201 18-AUG-01

10 rows selected.
```

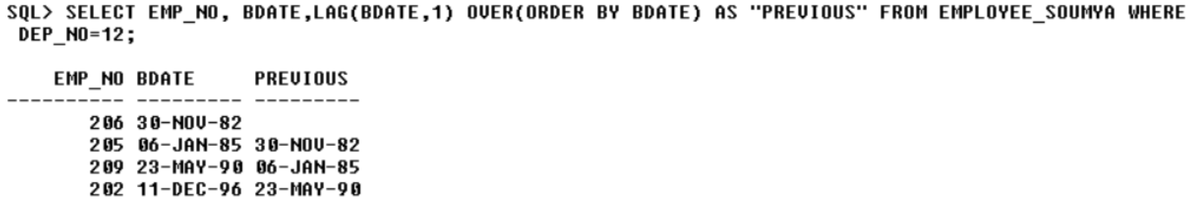


```
SQL> SELECT EMP_NO, BDATE, LEAD(BDATE,1)OVER(ORDER BY BDATE)AS "NEXT" FROM EMPLOYEE_SOUMYA WHERE DEP
_NO=12;

    EMP_NO BDATE     NEXT
---------- --------- ---------
       206 30-NOV-82 06-JAN-85
       205 06-JAN-85 23-MAY-90
       209 23-MAY-90 11-DEC-96
       202 11-DEC-96
```

LAG:

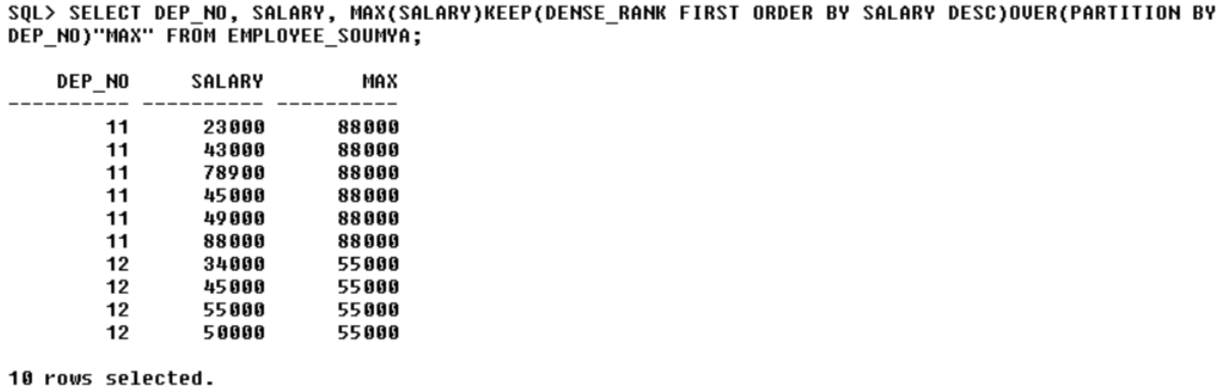


```
SQL> SELECT EMP_NO, BDATE,LAG(BDATE,1)OVER(ORDER BY BDATE) AS "PREVIOUS" FROM EMPLOYEE_SOUMYA;

    EMP_NO BDATE     PREVIOUS
---------- --------- ---------
       206 30-NOV-82
       205 06-JAN-85 30-NOV-82
       208 17-FEB-85 06-JAN-85
       204 07-APR-87 17-FEB-85
       203 05-JUN-88 07-APR-87
       209 23-MAY-90 05-JUN-88
       202 11-DEC-96 23-MAY-90
       210 03-JUL-98 11-DEC-96
       207 23-JAN-99 03-JUL-98
       201 18-AUG-01 23-JAN-99

10 rows selected.
```



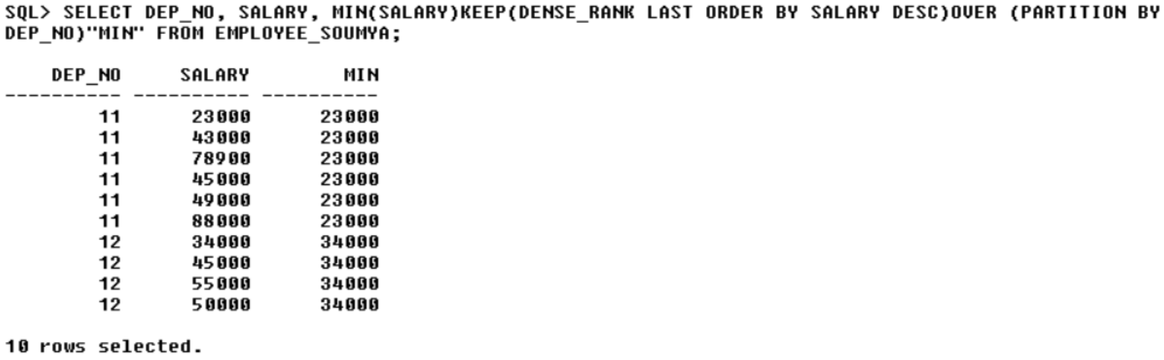
```
SQL> SELECT EMP_NO, BDATE,LAG(BDATE,1) OVER(ORDER BY BDATE) AS "PREVIOUS" FROM EMPLOYEE_SOUMYA WHERE
 DEP_NO=12;

    EMP_NO BDATE     PREVIOUS
---------- --------- ---------
       206 30-NOV-82
       205 06-JAN-85 30-NOV-82
       209 23-MAY-90 06-JAN-85
       202 11-DEC-96 23-MAY-90
```

FIRST:



```
SQL> SELECT DEP_NO, SALARY, MAX(SALARY)KEEP(DENSE_RANK FIRST ORDER BY SALARY DESC)OVER(PARTITION BY
DEP_NO)"MAX" FROM EMPLOYEE_SOUMYA;

    DEP_NO     SALARY        MAX
---------- ---------- ----------
        11      23000      88000
        11      43000      88000
        11      78900      88000
        11      45000      88000
        11      49000      88000
        11      88000      88000
        12      34000      55000
        12      45000      55000
        12      55000      55000
        12      50000      55000

10 rows selected.
```

LAST:



```
SQL> SELECT DEP_NO, SALARY, MIN(SALARY)KEEP(DENSE_RANK LAST ORDER BY SALARY DESC)OVER (PARTITION BY
DEP_NO)"MIN" FROM EMPLOYEE_SOUMYA;

    DEP_NO     SALARY        MIN
---------- ---------- ----------
        11      23000      23000
        11      43000      23000
        11      78900      23000
        11      45000      23000
        11      49000      23000
        11      88000      23000
        12      34000      34000
        12      45000      34000
        12      55000      34000
        12      50000      34000

10 rows selected.
```

CONCLUSION:

I have learned the implementation of Analytical queries like RollUp, Cube, First , Last, Lead, Lag, Rank, Dense Rank , etc.