# PRACTICAL - 2

**Aim:** Implementation of Analytical queries like Roll_Up, CUBE, First, Last, Lead, Rank etc.

## Theory:

Analytic functions in Oracle can be defined as functions similar to aggregate functions (Aggregate functions is used to group several rows of data into a single row) as it works on subset of rows and is used to calculate aggregate value based on a group of rows but in case of aggregate functions the number of rows returned by the query is reduced whereas in case of aggregate function the number of rows returned by the query is not reduced after execution.

**Roll Up:**

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.

**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.

**Lead:**

LEAD() function, as the name suggests, fetches the value of a specific column from the next row and returns the fetched value in the current row.

**Rank:**

The RANK() function is an analytic function that calculates the rank of a value in a set of values. This function returns the same rank for the rows with the same values. It adds the number of tied rows to the tied rank to calculate the next rank. Therefore, the ranks may not be consecutive numbers.

**Dense Rank:**

The DENSE_RANK() is an analytic function that calculates the rank of a row in an ordered set of rows. The returned rank is an integer starting from 1.

Unlike the RANK() function, the DENSE_RANK() function returns rank values as consecutive integers. It does not skip rank in case of ties. Rows with the same values for the rank criteria will receive the same rank values.

**First:**

The FIRST analytic function can be used to return the first value from an ordered sequence.

**Last:**

The LAST analytic function can be used to return the last value from an ordered sequence.

```
Enter user-name: user15b@sem1
Enter password:

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
```

**Creating Table student_siddhartha**

```
SQL> create table student_siddhartha(
  2  roll_no NUMBER(5) PRIMARY KEY,
  3  division VARCHAR2(10),
  4  name VARCHAR2(30),
  5  birthday DATE,
  6  salary NUMBER(7),
  7  address VARCHAR2(20));
Table created.
```

**Inserting Values into table student_siddhartha**

```
insert into student_siddhartha
values(1,'A','adam',to_date('26012000','DDMMYYYY'),3700,'chembur');

1 row created.
```

```
SQL> insert into student_siddhartha
values(2,'B','alex',to_date('16082002','DDMMYYYY'),4300,'chembur');

1 row created.

SQL> insert into student_siddhartha
values(3,'A','mitchel',to_date('02102001','DDMMYYYY'),4300,'worli');

1 row created.

SQL> insert into student_siddhartha
values(4,'A','roy',to_date('25012000','DDMMYYYY'),4100,'badlapur');

1 row created.

SQL> insert into student_siddhartha
values(5,'B','joe',to_date('19122001','DDMMYYYY'),2900,'chembur');

1 row created.

SQL> insert into student_siddhartha
values(6,'A','james',to_date('07072003','DDMMYYYY'),3500,'worli');

1 row created.
```

**Displaying all the records of table student_siddhartha**

```
SQL> select * from student_siddhartha;

   ROLL_NO DIVISION   NAME                           BIRTHDAY      SALARY ADDRESS
---------- ---------- ------------------------------ --------- ---------- --------------------
         1 A          adam                           26-JAN-00       3700 chembur
         2 B          alex                           16-AUG-02       4300 chembur
         3 A          mitchel                        02-OCT-01       4300 worli
         4 A          roy                            25-JAN-00       4100 badlapur
         5 B          joe                            19-DEC-01       2900 chembur
         6 A          james                          07-JUL-03       3500 worli

6 rows selected.
```

## Roll UP

**Code:**

SQL> select division,address,count(*),sum(salary) from student_siddhartha group by rollup(address,division);

**Output:**

```
DIVISION   ADDRESS              COUNT(*) SUM(SALARY)
---------- -------------------- ---------- -----------
A          worli                         2        7800
           worli                         2        7800
A          chembur                       1        3700
B          chembur                       2        7200
           chembur                       3       10900
A          badlapur                      1        4100
           badlapur                      1        4100
                                         6       22800

8 rows selected.
```

## Cube

**Code:**

SQL> select division,address,count(*),sum(salary) from student_siddhartha group by cube(address,division);

**Output:**

```
DIVISION   ADDRESS              COUNT(*) SUM(SALARY)
---------- -------------------- ---------- -----------
                                        6       22800
A                                       4       15600
B                                       2        7200
           worli                        2        7800
A          worli                        2        7800
           chembur                      3       10900
A          chembur                      1        3700
B          chembur                      2        7200
           badlapur                     1        4100
A          badlapur                     1        4100

10 rows selected.
```

## Rank

**Code:**

SQL> select roll_no,division,salary,address,rank() over(partition by division order by salary)as Rank from student_siddhartha;

**Output:**

| ROLL_NO | DIVISION | SALARY | ADDRESS | RANK |
|---|---|---|---|---|
| 6 | A | 3500 | worli | 1 |
| 1 | A | 3700 | chembur | 2 |
| 4 | A | 4100 | badlapur | 3 |
| 3 | A | 4300 | worli | 4 |
| 5 | B | 2900 | chembur | 1 |
| 2 | B | 4300 | chembur | 2 |

6 rows selected.

## DenseRank

**Code:**

SQL> select roll_no,division,salary,address, dense_rank() over(partition by division order by salary)as Rank from student_siddhartha;

**Output:**

| ROLL_NO | DIVISION | SALARY | ADDRESS | RANK |
|---|---|---|---|---|
| 6 | A | 3500 | worli | 1 |
| 1 | A | 3700 | chembur | 2 |
| 4 | A | 4100 | badlapur | 3 |
| 3 | A | 4300 | worli | 4 |
| 5 | B | 2900 | chembur | 1 |
| 2 | B | 4300 | chembur | 2 |

6 rows selected.

## Lead

**Code:**

SQL> select roll_no,birthday,lead(birthday,1) over (order by birthday) as "next" from student_siddhartha;

**Output:**

```
   ROLL_NO BIRTHDAY  next
---------- --------- ---------
         4 25-JAN-00 26-JAN-00
         1 26-JAN-00 02-OCT-01
         3 02-OCT-01 19-DEC-01
         5 19-DEC-01 16-AUG-02
         2 16-AUG-02 07-JUL-03
         6 07-JUL-03

6 rows selected.
```

## Lag

**Code:**

SQL> select roll_no,birthday,lag(birthday,1)over(order by birthday)as "Previous" from student_siddhartha;

**Output:**

```
   ROLL_NO BIRTHDAY  Previous
---------- --------- ---------
         4 25-JAN-00
         1 26-JAN-00 25-JAN-00
         3 02-OCT-01 26-JAN-00
         5 19-DEC-01 02-OCT-01
         2 16-AUG-02 19-DEC-01
         6 07-JUL-03 16-AUG-02

6 rows selected.
```

## Code:

SQL> select roll_no,birthday,lag(birthday,1)over(order by birthday)as "Previous" from student_siddhartha where division='B';

## Output:

| ROLL_NO | BIRTHDAY | Previous |
|---|---|---|
| 5 | 19-DEC-01 | |
| 2 | 16-AUG-02 | 19-DEC-01 |

**First**

## Code:

select division,salary,max(salary)keep(DENSE_RANK FIRST ORDER BY salary desc)over (PARTITION BY division)"max" from student_siddhartha;

## Output:

| DIVISION | SALARY | max |
|---|---|---|
| A | 3700 | 4300 |
| A | 4300 | 4300 |
| A | 4100 | 4300 |
| A | 3500 | 4300 |
| B | 2900 | 4300 |
| B | 4300 | 4300 |

6 rows selected.

**Last**

## Code:

```
SQL> select division,salary,min(salary)keep(DENSE_RANK LAST ORDER BY salary
desc)over(PARTITION BY division)"min" from student_siddhartha;
```

## Output:

```
DIVISION       SALARY        min
---------- ---------- ----------
A                3700       3500
A                4300       3500
A                4100       3500
A                3500       3500
B                2900       2900
B                4300       2900

6 rows selected.
```

**Conclusion:** I have successfully learned and implemented different Analytical Queries.