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 departments [dept_narender]:

```
create table dept_narender(
  deptno     number(2,0),
  dname      varchar2(14),
  loc        varchar2(13),
  constraint pk_dept primary key (deptno)
)
```

```
Table created.
```

Creating table employees [emp_narender]:

```
create table emp_narender(
  empno    number(4,0),
  ename    varchar2(50),
  job      varchar2(9),
  mgr      number(4,0),
  hiredate date,
  sal      number(7,2),
  comm     number(7,2),
  deptno   number(2,0),
  constraint pk_emp primary key (empno),
  constraint fk_deptno foreign key (deptno) references dept_narender (deptno)
)
```

```
Table created.
```

Inserting Records in department table:

```
/* Inserting records */
insert into dept_narender (DEPTNO, DNAME, LOC) values(10, 'ACCOUNTING', 'NEW YORK')
insert into dept_narender (DEPTNO, DNAME, LOC) values(20, 'RESEARCH', 'DALLAS')
insert into dept_narender (DEPTNO, DNAME, LOC) values(30, 'SALES', 'CHICAGO')
insert into dept_narender (DEPTNO, DNAME, LOC) values(40, 'OPERATIONS', 'BOSTON')
```

Checking Description of department table:

```
desc dept_narender;
```

TABLE DEPT_NARENDER

Column	Null?	Type
DEPTNO	NOT NULL	NUMBER(2,0)
DNAME	-	VARCHAR2(14)
LOC	-	VARCHAR2(13)

Download CSV

3 rows selected.

Viewing all the records of the department table:

```
select * from dept_narender;
```

DEPTNO	DNAME	LOC
20	RESEARCH	DALLAS
40	OPERATIONS	BOSTON
10	ACCOUNTING	NEW YORK
30	SALES	CHICAGO

Download CSV

4 rows selected.

Inserting records in employee table:

```
/* Inserting records */
insert into emp_narender values(7839, 'Narender Keswani', 'PRESIDENT', null, to_date('10-11-1999','dd-mm-yyyy'),5000, null, 10)
insert into emp_narender values(7698, 'Neel Deskhmukh', 'MANAGER', 7839, to_date('1-5-1981','dd-mm-yyyy'), 2850, null, 30)
insert into emp_narender values(7782, 'Hassan Haque', 'MANAGER', 7839, to_date('9-6-1981','dd-mm-yyyy'), 2450, null, 10)
insert into emp_narender values(7566, 'Wilson Rao', 'MANAGER', 7839, to_date('2-4-1981','dd-mm-yyyy'), 2975, null, 20)
insert into emp_narender values(7788, 'Ritesh Yadav', 'ANALYST', 7566, to_date('13-JUL-87','dd-mm-rr') - 85,  3000, null, 20)
insert into emp_narender values(7902, 'Ronak Karia', 'ANALYST', 7566, to_date('3-12-1981','dd-mm-yyyy'), 3000, null, 20)
insert into emp_narender values(7369, 'Ashok Gawade', 'CLERK', 7902, to_date('17-12-1980','dd-mm-yyyy'),  800, null, 20)
insert into emp_narender values(7499, 'Pranot Gawade', 'SALESMAN', 7698, to_date('20-2-1981','dd-mm-yyyy'), 1600, 300, 30)
insert into emp_narender values(7521, 'Rajaram Gawade', 'SALESMAN', 7698, to_date('22-2-1981','dd-mm-yyyy'), 1250, 500, 30)
insert into emp_narender values(7654, 'Ganesh Keswani', 'SALESMAN', 7698, to_date('28-9-1981','dd-mm-yyyy'), 1250, 1400, 30)
insert into emp_narender values(7844, 'Avnish Khandewal', 'SALESMAN', 7698, to_date('8-9-1981','dd-mm-yyyy'), 1500, 0, 30)
insert into emp_narender values(7876, 'Kalpesh Khandewal', 'CLERK', 7788, to_date('13-JUL-87', 'dd-mm-rr') - 51, 1100, null, 20)
insert into emp_narender values(7900, 'Shernik Jain', 'CLERK', 7698, to_date('3-12-1981','dd-mm-yyyy'), 950, null, 30)
insert into emp_narender values(7934, 'Ujjal Ray', 'CLERK', 7782, to_date('23-1-1982','dd-mm-yyyy'), 1300, null, 10)
```

Viewing description of employee table:

```
47  desc emp_narender;
```

TABLE EMP_NARENDER

Column	Null?	Type
EMPNO	NOT NULL	NUMBER(4,0)
ENAME	-	VARCHAR2(50)
JOB	-	VARCHAR2(9)
MGR	-	NUMBER(4,0)
HIREDATE	-	DATE
SAL	-	NUMBER(7,2)
COMM	-	NUMBER(7,2)
DEPTNO	-	NUMBER(2,0)

Download CSV

8 rows selected.

Viewing all the records of the employee table:

```
select * from emp_narender;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	Narender Keswani	PRESIDENT	-	10-NOV-99	5000	-	10
7698	Neel Deskhmukh	MANAGER	7839	01-MAY-81	2850	-	30
7788	Ritesh Yadav	ANALYST	7566	19-APR-87	3000	-	20
7902	Ronak Karia	ANALYST	7566	03-DEC-81	3000	-	20
7369	Ashok Gawade	CLERK	7902	17-DEC-80	800	-	20
7521	Rajaram Gawade	SALESMAN	7698	22-FEB-81	1250	500	30
7900	Shernik Jain	CLERK	7698	03-DEC-81	950	-	30
7782	Hassan Haque	MANAGER	7839	09-JUN-81	2450	-	10
7876	Kalpesh Khandewal	CLERK	7788	23-MAY-87	1100	-	20
7934	Ujjal Ray	CLERK	7782	23-JAN-82	1300	-	10
7566	Wilson Rao	MANAGER	7839	02-APR-81	2975	-	20
7654	Ganesh Keswani	SALESMAN	7698	28-SEP-81	1250	1400	30
7499	Pranot Gawade	SALESMAN	7698	20-FEB-81	1600	300	30
7844	Avnish Khandewal	SALESMAN	7698	08-SEP-81	1500	0	30

Download CSV

14 rows selected.

Viewing combining records of the employee and department table based on the joins:

```
51 /* Viewing all Records */
52 select ename, dname, job, empno, hiredate, loc from emp_narender, dept_narender where emp_narender.deptno = dept_narender.deptno order by ename
```

ENAME	DNAME	JOB	EMPNO	HIREDATE	LOC
Ashok Gawade	RESEARCH	CLERK	7369	17-DEC-80	DALLAS
Avnish Khandewal	SALES	SALESMAN	7844	08-SEP-81	CHICAGO
Ganesh Keswani	SALES	SALESMAN	7654	28-SEP-81	CHICAGO
Hassan Haque	ACCOUNTING	MANAGER	7782	09-JUN-81	NEW YORK
Kalpesh Khandewal	RESEARCH	CLERK	7876	23-MAY-87	DALLAS
Narender Keswani	ACCOUNTING	PRESIDENT	7839	10-NOV-99	NEW YORK
Neel Deskhmukh	SALES	MANAGER	7698	01-MAY-81	CHICAGO
Pranot Gawade	SALES	SALESMAN	7499	20-FEB-81	CHICAGO
Rajaram Gawade	SALES	SALESMAN	7521	22-FEB-81	CHICAGO
Ritesh Yadav	RESEARCH	ANALYST	7788	19-APR-87	DALLAS
Ronak Karia	RESEARCH	ANALYST	7902	03-DEC-81	DALLAS
Shernik Jain	SALES	CLERK	7900	03-DEC-81	CHICAGO
Ujjal Ray	ACCOUNTING	CLERK	7934	23-JAN-82	NEW YORK
Wilson Rao	RESEARCH	MANAGER	7566	02-APR-81	DALLAS

Download CSV

14 rows selected.

A) ROLL -UP:

```
55  /* ROLL-UP */
56  select deptno, job, count(*), sum(sal) from emp_narender group by rollup(deptno, job);
57
58  select deptno, job, count(*), avg(sal) from emp_narender group by rollup(deptno, job);
59
60  select deptno, job, count(*), min(sal) from emp_narender group by rollup(deptno, job);
61
62  select deptno, job, count(*), max(sal) from emp_narender group by rollup(deptno, job);
```

EXAMPLE OF SUM:

DEPTNO	JOB	COUNT(*)	SUM(SAL)
10	CLERK	1	1300
10	MANAGER	1	2450
10	PRESIDENT	1	5000
10	-	3	8750
20	CLERK	2	1900
20	ANALYST	2	6000
20	MANAGER	1	2975
20	-	5	10875
30	CLERK	1	950
30	MANAGER	1	2850
30	SALESMAN	4	5600
30	-	6	9400
-	-	14	29025

Download CSV

13 rows selected.

EXAMPLE OF AVG:

DEPTNO	JOB	COUNT(*)	AVG(SAL)
10	CLERK	1	1300
10	MANAGER	1	2450
10	PRESIDENT	1	5000
10	-	3	2916.66666666666666666666666666666666666667
20	CLERK	2	950
20	ANALYST	2	3000
20	MANAGER	1	2975
20	-	5	2175
30	CLERK	1	950
30	MANAGER	1	2850
30	SALESMAN	4	1400
30	-	6	1566.66666666666666666666666666666666666667
-	-	14	2073.21428571428571428571428571428571428571

Download CSV

13 rows selected.

EXAMPLE OF MIN:

DEPTNO	JOB	COUNT(*)	MIN(SAL)
10	CLERK	1	1300
10	MANAGER	1	2450
10	PRESIDENT	1	5000
10	-	3	1300
20	CLERK	2	800
20	ANALYST	2	3000
20	MANAGER	1	2975
20	-	5	800
30	CLERK	1	950
30	MANAGER	1	2850
30	SALESMAN	4	1250
30	-	6	950
-	-	14	800

Download CSV

13 rows selected.

EXAMPLE OF MAX:

DEPTNO	JOB	COUNT(*)	MAX(SAL)
10	CLERK	1	1300
10	MANAGER	1	2450
10	PRESIDENT	1	5000
10	-	3	5000
20	CLERK	2	1100
20	ANALYST	2	3000
20	MANAGER	1	2975
20	-	5	3000
30	CLERK	1	950
30	MANAGER	1	2850
30	SALESMAN	4	1600
30	-	6	2850
-	-	14	5000

Download CSV

13 rows selected.

B) CUBE:

```
64  /* CUBE */
65  select deptno, job, count(*), sum(sal) from emp_narender group by cube(deptno, job);
66
67  select deptno, job, count(*), avg(sal) from emp_narender group by cube(deptno, job);
68
69  select deptno, job, count(*), min(sal) from emp_narender group by cube(deptno, job);
70
71  select deptno, job, count(*), max(sal) from emp_narender group by cube(deptno, job);
```

DEPTNO	JOB	COUNT(*)	SUM(SAL)
-	-	14	29025
-	CLERK	4	4150
-	ANALYST	2	6000
-	MANAGER	3	8275
-	SALESMAN	4	5600
-	PRESIDENT	1	5000
10	-	3	8750
10	CLERK	1	1300
10	MANAGER	1	2450
10	PRESIDENT	1	5000
20	-	5	10875
20	CLERK	2	1900
20	ANALYST	2	6000
20	MANAGER	1	2975
30	-	6	9400
30	CLERK	1	950
30	MANAGER	1	2850
30	SALESMAN	4	5600

Download CSV

18 rows selected.

DEPTNO	JOB	COUNT(*)	AVG(SAL)
-	-	14	2073.2142857142857142857142857142857143
-	CLERK	4	1037.5
-	ANALYST	2	3000
-	MANAGER	3	2758.3333333333333333333333333333333333
-	SALESMAN	4	1400
-	PRESIDENT	1	5000
10	-	3	2916.6666666666666666666666666666666667
10	CLERK	1	1300
10	MANAGER	1	2450
10	PRESIDENT	1	5000
20	-	5	2175
20	CLERK	2	950
20	ANALYST	2	3000
20	MANAGER	1	2975
30	-	6	1566.6666666666666666666666666666666667
30	CLERK	1	950
30	MANAGER	1	2850
30	SALESMAN	4	1400

Download CSV
18 rows selected.

DEPTNO	JOB	COUNT(*)	MIN(SAL)
-	-	14	800
-	CLERK	4	800
-	ANALYST	2	3000
-	MANAGER	3	2450
-	SALESMAN	4	1250
-	PRESIDENT	1	5000
10	-	3	1300
10	CLERK	1	1300
10	MANAGER	1	2450
10	PRESIDENT	1	5000
20	-	5	800
20	CLERK	2	800
20	ANALYST	2	3000
20	MANAGER	1	2975
30	-	6	950
30	CLERK	1	950
30	MANAGER	1	2850
30	SALESMAN	4	1250

Download CSV

18 rows selected.

DEPTNO	JOB	COUNT(*)	MAX(SAL)
-	-	14	5000
-	CLERK	4	1300
-	ANALYST	2	3000
-	MANAGER	3	2975
-	SALESMAN	4	1600
-	PRESIDENT	1	5000
10	-	3	5000
10	CLERK	1	1300
10	MANAGER	1	2450
10	PRESIDENT	1	5000
20	-	5	3000
20	CLERK	2	1100
20	ANALYST	2	3000
20	MANAGER	1	2975
30	-	6	2850
30	CLERK	1	950
30	MANAGER	1	2850
30	SALESMAN	4	1600

Download CSV

18 rows selected.

C) RANK:

```
/* RANK */
select empno, deptno, sal, comm, rank() over(partition by deptno order by sal)as Rank from emp_narender;
```

EMPNO	DEPTNO	SAL	COMM	RANK
7934	10	1300	-	1
7782	10	2450	-	2
7839	10	5000	-	3
7369	20	800	-	1
7876	20	1100	-	2
7566	20	2975	-	3
7902	20	3000	-	4
7788	20	3000	-	4
7900	30	950	-	1
7654	30	1250	1400	2
7521	30	1250	500	2
7844	30	1500	0	4
7499	30	1600	300	5
7698	30	2850	-	6

Download CSV

14 rows selected.

D) **DENSE-RANK:**

```
/* DENSE-RANK */
select empno, deptno, sal, comm, dense_rank() over(partition by deptno order by sal)as Rank from emp_narender;
```

EMPNO	DEPTNO	SAL	COMM	RANK
7934	10	1300	-	1
7782	10	2450	-	2
7839	10	5000	-	3
7369	20	800	-	1
7876	20	1100	-	2
7566	20	2975	-	3
7902	20	3000	-	4
7788	20	3000	-	4
7900	30	950	-	1
7654	30	1250	1400	2
7521	30	1250	500	2
7844	30	1500	0	3
7499	30	1600	300	4
7698	30	2850	-	5

Download CSV

14 rows selected.

E) LEAD:

```
/* LEAD */
select empno, hiredate, lead(hiredate,1) over(order by hiredate) as "next" from emp_narender;

select empno, hiredate, lead(hiredate,1) over(order by hiredate) as "next" from emp_narender where deptno=10;
```

EMPNO	HIREDATE	next
7369	17-DEC-80	20-FEB-81
7499	20-FEB-81	22-FEB-81
7521	22-FEB-81	02-APR-81
7566	02-APR-81	01-MAY-81
7698	01-MAY-81	09-JUN-81
7782	09-JUN-81	08-SEP-81
7844	08-SEP-81	28-SEP-81
7654	28-SEP-81	03-DEC-81
7900	03-DEC-81	03-DEC-81
7902	03-DEC-81	23-JAN-82
7934	23-JAN-82	19-APR-87
7788	19-APR-87	23-MAY-87
7876	23-MAY-87	10-NOV-99
7839	10-NOV-99	-

Download CSV

14 rows selected.

EMPNO	HIREDATE	next
7782	09-JUN-81	23-JAN-82
7934	23-JAN-82	10-NOV-99
7839	10-NOV-99	-

Download CSV

3 rows selected.

F) LAG:

```
/* LAG */
select empno, hiredate, lag(hiredate,1) over(order by hiredate) as "previous" from emp_narender;

select empno, hiredate, lag(hiredate,1) over(order by hiredate) as "previous" from emp_narender where deptno=10;
```

EMPNO	HIREDATE	previous
7369	17-DEC-80	-
7499	20-FEB-81	17-DEC-80
7521	22-FEB-81	20-FEB-81
7566	02-APR-81	22-FEB-81
7698	01-MAY-81	02-APR-81
7782	09-JUN-81	01-MAY-81
7844	08-SEP-81	09-JUN-81
7654	28-SEP-81	08-SEP-81
7900	03-DEC-81	28-SEP-81
7902	03-DEC-81	03-DEC-81
7934	23-JAN-82	03-DEC-81
7788	19-APR-87	23-JAN-82
7876	23-MAY-87	19-APR-87
7839	10-NOV-99	23-MAY-87

Download CSV

14 rows selected.

EMPNO	HIREDATE	previous
7782	09-JUN-81	-
7934	23-JAN-82	09-JUN-81
7839	10-NOV-99	23-JAN-82

Download CSV

3 rows selected.

G) **FIRST:**

```
/* FIRST */
select deptno, sal, max(sal) keep(DENSE_RANK FIRST ORDER BY sal desc) over(PARTITION BY deptno)"max" from emp_narender;
```

DEPTNO	SAL	max
10	2450	5000
10	1300	5000
10	5000	5000
20	2975	3000
20	1100	3000
20	800	3000
20	3000	3000
20	3000	3000
30	1250	2850
30	2850	2850
30	950	2850
30	1250	2850
30	1500	2850
30	1600	2850

Download CSV

14 rows selected.

H) LAST:

```
/* LAST */
select deptno, sal, min(sal) keep(DENSE_RANK LAST ORDER BY sal desc) over(PARTITION BY deptno)"min" from emp_narender;
```

DEPTNO	SAL	min
10	2450	1300
10	1300	1300
10	5000	1300
20	2975	800
20	1100	800
20	800	800
20	3000	800
20	3000	800
30	1250	950
30	2850	950
30	950	950
30	1250	950
30	1500	950
30	1600	950

Download CSV

14 rows selected.

CONCLUSION:

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