

Module 02: - OLAP with Oracle: Analytical Queries

Practical No.2

Aim: - Implementation of Analytical queries like ROLL_UP, CUBE, RANK, DENSE_RANK, LEAD, LAG, FIRST and LAST.

Objective: - To learn analytical queries.

Theory:

Analytical functions: - 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.

Syntax: -

analytic_function([arguments])OVER(analytic_clause)

The analytic_clause breaks down into the following optional elements.

[query_partition_clause][order_by_clause]

Implementation: -

ROLL_UP: - The ROLLUP is an extension of the GROUP BY clause. The ROLLUP calculates multiple levels of subtotals across a group of columns (or dimensions) along with the grand total. the ROLLUP extension produces group subtotals from right to left and a grand total. If "n" is the number of columns listed in the ROLLUP, there will be n+1 levels of subtotals.

Step 1: Create table employee with fields emp_no, dep_no, emp_name, dob, salary, comm, job.

Query: -

```
CREATE TABLE employee8
(
emp_no NUMBER(10),
dep_no NUMBER(10),
emp_name VARCHAR(25),
dob DATE,
salary NUMBER(10),
comm NUMBER(10),
job VARCHAR(25)
);
```

Output: -

```
SQL> CREATE TABLE employee6
2 (
3 emp_no NUMBER(10),
4 dep_no NUMBER(10),
5 emp_name VARCHAR(25),
6 dob DATE,
7 salary NUMBER(10),
8 comm NUMBER(10),
9 job VARCHAR(25)
10 );

Table created.

SQL> |
```

Step 2: - Insert data into the above table**Query:** -

```
insert into employee8
VALUES(101,10,'TEJAS',TO_DATE('12/01/82','DD/MM/YYYY'),22000,1000,'CLERK');
insert into employee8
VALUES(102,10,'SACHIN',TO_DATE('12/02/83','DD/MM/YYYY'),52000,2000,'CLERK');
insert into employee8
VALUES(103,10,'MANISH',TO_DATE('12/03/84','DD/MM/YYYY'),25000,5000,'CLERK');
insert into employee8
VALUES(104,20,'VANCHITA',TO_DATE('12/04/82','DD/MM/YYYY'),35000,5000,'MANAGER');
insert into employee8
VALUES(105,20,'SWARUPA',TO_DATE('12/05/86','DD/MM/YYYY'),45000,6000,'MANAGER');
insert into employee8
VALUES(106,20,'ASHWINI',TO_DATE('12/06/87','DD/MM/YYYY'),26000,2000,'MANAGER');
insert into employee8
VALUES(107,10,'SUDIP',TO_DATE('12/07/88','DD/MM/YYYY'),55000,6000,'MANAGER');
insert into employee8
VALUES(108,20,'AKSHAY',TO_DATE('12/08/89','DD/MM/YYYY'),35000,6000,'CLERK');
```

Output: -

```
SQL> insert into employee7 VALUES(101,10,'TEJAS',TO_DATE('12/01/82','DD/MM/YYYY'),22000,1000,'CLERK');
1 row created.
SQL> insert into employee7 VALUES(102,10,'SACHIN',TO_DATE('12/02/83','DD/MM/YYYY'),52000,2000,'CLERK');
1 row created.
SQL> insert into employee7 VALUES(103,10,'MANISH',TO_DATE('12/03/84','DD/MM/YYYY'),25000,5000,'CLERK');
1 row created.
SQL> insert into employee7 VALUES(104,20,'VANCHITA',TO_DATE('12/04/82','DD/MM/YYYY'),35000,5000,'MANAGER');
1 row created.
SQL> insert into employee7 VALUES(105,20,'SMARUPA',TO_DATE('12/05/86','DD/MM/YYYY'),45000,6000,'MANAGER');
1 row created.
SQL> insert into employee7 VALUES(106,20,'ASHWINI',TO_DATE('12/06/87','DD/MM/YYYY'),26000,2000,'MANAGER');
1 row created.
SQL> insert into employee7 VALUES(107,10,'SUDIP',TO_DATE('12/07/88','DD/MM/YYYY'),55000,6000,'MANAGER');
1 row created.
SQL> insert into employee7 VALUES(108,20,'AKSHAY',TO_DATE('12/08/89','DD/MM/YYYY'),35000,6000,'CLERK');
1 row created.
SQL> |
```

Step 3: - Display the data from employee7

Query: -

select * from employee8;

Output: -

```
SQL> select * from employee8;
EMP_NO  DEP_NO  EMP_NAME  DOB  SALARY  COMM
-----
JOB
101      10      TEJAS      12-JAN-82  22000  1000
CLERK
102      10      SACHIN     12-FEB-83  52000  2000
CLERK
103      10      MANISH     12-MAR-84  25000  5000
CLERK
104      20      VANCHITA  12-APR-82  35000  5000
MANAGER
105      20      SMARUPA   12-MAY-86  45000  6000
MANAGER
106      20      ASHWINI   12-JUN-87  26000  2000
MANAGER
107      10      SUDIP     12-JUL-88  55000  6000
MANAGER
108      20      AKSHAY    12-AUG-89  35000  6000
CLERK
8 rows selected.
SQL> |
```

Q. Display dep_no, job, job count, sum of salary group them up using a roll up function in the order of dep_no, job.

Query: -

select dep_no,job,count(*),sum(salary) from employee7 group by rollup(dep_no,job);

Output: -

```
SQL> select dep_no,job,count(*),sum(salary) from employee7 group by rollup(dep_no,job);
DEP_NO JOB  COUNT(*)  SUM(SALARY)
-----
10 CLERK  3          99000
10 MANAGER 1          55000
10      4          154000
20 CLERK  1          35000
20 MANAGER 3          106000
20      4          141000
20      8          295000
7 rows selected.
SQL> |
```

CUBE: - In addition to the subtotals generated by the ROLLUP extension, the CUBE extension will generate subtotals for all combinations of the dimensions specified. If "n" is the number of columns listed in the CUBE, there will be 2ⁿ subtotal combinations.

Q. Display dep_no,job,job count, sum of salary group them up using a cube function in the order of dep_no,job.

Query: -

Select dep_no,job,count(*),sum(salary) from employee8 group by cube(dep_no,job);

Output: -

```
SQL Plus
9 rows selected.
SQL> Select dep_no,job,count(*),sum(salary) from employee8 group by cube(dep_no,job);
DEP_NO JOB  COUNT(*)  SUM(SALARY)
-----
8          295000
CLERK      4          134000
MANAGER    4          161000
10         4          154000
10 CLERK    3          99000
10 MANAGER 1          55000
20         4          141000
20 CLERK    1          35000
20 MANAGER 3          106000
9 rows selected.
SQL> |
```

Q. Display emp_no, emp_name and salary from employee table and give numbers to each row.

Query: -

select rownum, emp_no, emp_name, salary from employee8;

Output: -

```
SQL Plus
20 MANAGER 3 106000
9 rows selected.
SQL> select rownum, emp_no, emp_name, salary from employee8;
ROWNUM  EMP_NO  EMP_NAME  SALARY
-----
1        101     TEJAS      22000
2        102     SACHIN     52000
3        103     MANISH     25000
4        104     VANCHITA  35000
5        105     SMARUPA   45000
6        106     ASHWINI   26000
7        107     SUDIP     55000
8        108     AKSHAY    35000
8 rows selected.
SQL> |
```

Q. Display emp_no, emp_name and salary from employee table and give numbers to each row.

Query:-

Select rownum, emp_no, emp_name, salary from employee8;

Output:-

```
SQL> Select rownum, emp_no, emp_name, salary from employee8;
```

ROWNUM	EMP_NO	EMP_NAME	SALARY
1	101	TEJAS	22000
2	102	SACHIN	52000
3	103	MANISH	25000
4	104	VANCHITA	35000
5	105	SWARUPA	45000
6	106	ASHWINI	26000
7	107	SUDIP	55000
8	108	AKSHAY	35000

8 rows selected.

Query:-

select rownum, emp_no, emp_name, salary from employee8 order by salary ;

Output: -

```
SQL> select rownum, emp_no, emp_name, salary from employee8 order by salary ;
```

ROWNUM	EMP_NO	EMP_NAME	SALARY
1	101	TEJAS	22000
3	103	MANISH	25000
6	106	ASHWINI	26000
8	108	AKSHAY	35000
4	104	VANCHITA	35000
5	105	SWARUPA	45000
2	102	SACHIN	52000
7	107	SUDIP	55000

8 rows selected.

Q. Display emp_no, emp_name and descending order of salary from employee table and give numbers to each row.

Query:-

select rownum, emp_no, emp_name, salary from employee8 order by salary desc;

Output:-

```
SQL> select rownum, emp_no, emp_name, salary from employee8 order by salary desc;
```

ROWNUM	EMP_NO	EMP_NAME	SALARY
7	107	SUDIP	55000
2	102	SACHIN	52000
5	105	SWARUPA	45000
8	108	AKSHAY	35000
4	104	VANCHITA	35000
6	106	ASHWINI	26000
3	103	MANISH	25000
1	101	TEJAS	22000

8 rows selected.

ROW_NUMBER():- It is an analytical function and unlike NTILE this function assigns a unique sequential number to each row of the result set.

Q. Use Row_number() analytical function to give numbering according to salary.

Query:-

select row_number() over(order by salary), emp_no, emp_name, salary from employee8 order by salary desc;

Output:-

```
SQL> select row_number() over(order by salary), emp_no, emp_name, salary from employee8 order by salary desc;
```

ROW_NUMBER() OVER(ORDER BY SALARY)	EMP_NO	EMP_NAME	SALARY
8	107	SUDIP	55000
7	102	SACHIN	52000
6	105	SWARUPA	45000
4	108	AKSHAY	35000
5	104	VANCHITA	35000
3	106	ASHWINI	26000
2	103	MANISH	25000
1	101	TEJAS	22000

8 rows selected.

RANK():- The RANK() function is an analytic function that calculates the rank of a value in a set of values. The RANK() function returns the same rank for the rows with the same values.

Q. Display eno, ename and salary from employee table and rank them according to ascending order of salary.

Query:-

select emp_no, emp_name, salary, rank() over(order by salary) from employee8 order by salary;

Output:-

```
SQL> select emp_no, emp_name, salary, rank() over(order by salary) from employee8 order by salary;
```

EMP_NO	EMP_NAME	SALARY	RANK() OVER(ORDER BY SALARY)
101	TEJAS	22000	1
103	MANISH	25000	2
106	ASHWINI	26000	3
108	AKSHAY	35000	4
104	VANCHITA	35000	4
105	SWARUPA	45000	6
102	SACHIN	52000	7
107	SUDIP	55000	8

8 rows selected.

DENSE_RANK():- The DENSE_RANK() function is an analytic function that calculates the rank of a value in a set of values. The DENSE_RANK() function returns the same rank for the rows with the same values. DENSE_RANK() does not have any gap in rankings

Q. Display eno, name, salary from employee table and rank them according to ascending order of salary using dense_rank()

Query:-

select emp_no, emp_name, dense_rank() over(order by salary) from employee8 order by salary;

Output:-

```
SQL> select emp_no, emp_name, dense_rank() over(order by salary) from employee8 order by salary;
```

EMP_NO	EMP_NAME	DENSE_RANK()OVER(ORDERBYSALARY)
101	TEJAS	1
103	MANISH	2
106	ASHWINI	3
108	AKSHAY	4
104	VANCHITA	4
105	SWARUPA	5
102	SACHIN	6
107	SUDIP	7

8 rows selected.

Q. Display three highest salaried person.

Query:-

select * from (select dense_rank() over(order by salary desc) top, emp_name, emp_no, salary from employee8) where top <= 3;

Output:-

```
SQL> select * from (select dense_rank() over(order by salary desc) top, emp_name, emp_no, salary from employee8) where top <= 3;
```

TOP	EMP_NAME	EMP_NO	SALARY
1	SUDIP	107	55000
2	SACHIN	102	52000
3	SWARUPA	105	45000

Q. Partition by department_number

Query:-

select emp_no, dep_no, salary, comm, rank() over(partition by dep_no order by salary) as Rank from employee8;

Output:-

```
SQL> select emp_no, dep_no, salary, comm, rank() over(partition by dep_no order by salary) as Rank from employee8;
```

EMP_NO	DEP_NO	SALARY	COMM	RANK
101	10	22000	1000	1
103	10	25000	5000	2
102	10	52000	2000	3
107	10	55000	6000	4
106	20	26000	2000	1
108	20	35000	6000	2
104	20	35000	5000	2
105	20	45000	6000	4

8 rows selected.

Q. Update employee salary

Query:-

update employee8 set salary=33000 where emp_no=101;
update employee8 set salary=44000 where emp_no=102;

Output:-

```
SQL> update employee8 set salary=33000 where emp_no=101;
```

1 row updated.

```
SQL>
```

```
SQL> update employee8 set salary=44000 where emp_no=102;
```

1 row updated.

```
SQL> select * from employee8;
```

EMP_NO	DEP_NO	EMP_NAME	DOB	SALARY	COMM
101	10	TEJAS	12-JAN-82	33000	1000
102	10	SACHIN	12-FEB-83	44000	2000
103	10	MANISH	12-MAR-84	25000	5000

Q. Inserting null value in commission

Query:-

insert into employee8 values(101,10,'TEJAS',TO_DATE('12/01/82','DD/MM/YYYY'),22000,null,'CLERK');

Q. Replace null values of commission by 1000

Query:-

Select emp_no, emp_name, salary, nvl(comm,1000) new_comm from employee8 order by comm desc;

Output:-

```
SQL> select emp_no, emp_name, salary, nvl(comm,1000) new_comm from employee8 order by comm desc;
```

EMP_NO	EMP_NAME	SALARY	NEW_COMM
101	TEJAS	22000	1000
105	SWARUPA	45000	6000
108	AKSHAY	35000	6000
107	SUDIP	55000	6000
104	VANCHITA	35000	5000
103	MANISH	25000	5000
106	ASHWINI	26000	2000
102	SACHIN	44000	2000
101	TEJAS	33000	1000

9 rows selected.

Q. Display details of employee and give ranking only for employee in dept_no 10

Query:-

```
select dense_rank()over(partition by dep_no order by salary)Rank,dep_no,emp_name,salary from employee8 where dep_no=10;
```

Output:-

```
SQL> select dense_rank()over(partition by dep_no order by salary)Rank,dep_no,emp_name,salary from employee8 where dep_no=10;
```

RANK	DEP_NO	EMP_NAME	SALARY
1	10	TEJAS	22000
2	10	MANISH	25000
3	10	TEJAS	33000
4	10	SACHIN	44000
5	10	SUDIP	55000

Q. Display name, job & salary and Rank the salary job wise

Query:-

```
select dense_rank()over(partition by job order by salary)Rank,job,emp_name,salary from employee8 order by job;
```

Output:-

```
SQL> select dense_rank()over(partition by job order by salary)Rank,job,emp_name,salary from employee8 order by job;
```

RANK	JOB	EMP_NAME	SALARY
1	CLERK	TEJAS	22000
2	CLERK	MANISH	25000
3	CLERK	TEJAS	33000
4	CLERK	AKSHAY	35000
5	CLERK	SACHIN	44000
1	MANAGER	ASHWINI	26000
2	MANAGER	VANCHITA	35000
3	MANAGER	SWARUPA	45000
4	MANAGER	SUDIP	55000

9 rows selected.

Q. Display information of employee & rank them for employee working as manager

Query:-

```
select dense_rank()over(partition by job order by salary)Rank,job,emp_name,salary from employee8 where job='MANAGER';
```

Output:-

```
SQL> select dense_rank()over(partition by job order by salary)Rank,job,emp_name,salary from employee8 where job='MANAGER';
```

RANK	JOB	EMP_NAME	SALARY
1	MANAGER	ASHWINI	26000
2	MANAGER	VANCHITA	35000
3	MANAGER	SWARUPA	45000
4	MANAGER	SUDIP	55000

Q. Display first five records of employee in descending order of salary

Query:-

```
select*from(select emp_no, emp_name, salary, dense_rank()over(order by salary desc)rank from employee8) where rank<=5;
```

Output:-

```
SQL> select*from(select emp_no, emp_name, salary, dense_rank()over(order by salary desc)rank from employee8) where rank<=5;
```

EMP_NO	EMP_NAME	SALARY	RANK
107	SUDIP	55000	1
105	SWARUPA	45000	2
102	SACHIN	44000	3
104	VANCHITA	35000	4
108	AKSHAY	35000	4
101	TEJAS	33000	5

6 rows selected.

LEAD() and LAG():- The LEAD function is used to access data from SUBSEQUENT rows along with data from the current row. The LAG function is used to access data from PREVIOUS rows along with data from the current row

Syntax:- LEAD(expression,[offset],[default]) over ([query_partition_clause] order_by_clause)

Q. Display Employee details using Lead() analytical function.

Query:-

```
select emp_no,dob,lead(dob,1)over(order by dob)as "next" from employee8;
```

Output:-

```
SQL> select emp_no,dob,lead(dob,1)over(order by dob)as "next" from employee8;
```

EMP_NO	DOB	next
101	12-JAN-82	12-JAN-82
101	12-JAN-82	12-APR-82
104	12-APR-82	12-FEB-83
102	12-FEB-83	12-MAR-84
103	12-MAR-84	12-MAY-86
105	12-MAY-86	12-JUN-87
106	12-JUN-87	12-JUL-88
107	12-JUL-88	12-AUG-89
108	12-AUG-89	

9 rows selected.

Q. Display Employee details using Lag() analytical function.

Query:-

```
select emp_no,dob,lag(dob,1)over(order by dob)as "previous" from employee8;
```

Output:-

```
SQL> select emp_no,dob,lag(dob,1)over(order by dob)as "previous" from employee8;

EMP_NO DOB          previous
-----
101 12-JAN-82
101 12-JAN-82 12-JAN-82
104 12-APR-82 12-JAN-82
102 12-FEB-83 12-APR-82
103 12-MAR-84 12-FEB-83
105 12-MAY-86 12-MAR-84
106 12-JUN-87 12-MAY-86
107 12-JUL-88 12-JUN-87
108 12-AUG-89 12-JUL-88

9 rows selected.
```

FIRST and LAST :- FIRST is an analytic function as the name suggests is used to provide the value of the first row in an ordered set of rows. LAST is also an analytical function which is used to get the value of the last row in an ordered set of rows.

Query:-

```
select dep_no, salary, max(salary) keep
(DENSE_RANK FIRST ORDER BY salary
desc)over(PARTITION BY dep_no)"max" from
employee8;
```

Output:-

```
SQL> select dep_no, salary, max(salary) keep (DENSE_RANK FIRST ORDER BY salary desc)over(PARTITION BY dep_no)"max" from employee8;

DEP_NO SALARY    max
-----
10      25000    55000
10      55000    55000
10      44000    55000
10      33000    55000
10      22000    55000
20      35000    45000
20      26000    45000
20      35000    45000
20      45000    45000

9 rows selected.
```

Query:-

```
select dep_no,salary,min(salary)keep(DENSE_RANK
LAST ORDER BY salary desc)over(PARTITION BY
dep_no)"min" from employee8;
```

Output:-

```
SQL> select dep_no,salary,min(salary)keep(DENSE_RANK LAST ORDER BY salary desc)over(PARTITION BY dep_no)"min" from employee8;

DEP_NO SALARY    min
-----
10      25000    22000
10      55000    22000
10      44000    22000
10      33000    22000
10      22000    22000
20      35000    26000
20      26000    26000
20      35000    26000
20      45000    26000

9 rows selected.
```

Windowing Functions: -

The ROWS PRECEDING and ROWS FOLLOWING functions are used to specify rows before and after the current row in a window function:

- **ROWS PRECEDING:** Specifies a fixed number of rows before the current row
- **ROWS FOLLOWING:** Specifies a fixed number of rows after the current row

Q. Display empno,name,salary,sum of salary dept wise and display the salary of current and previous 2 records.

Output:

```
SQL> select empno,ename,sal,deptno,sum(sal) over (partition by deptno order by deptno rows 2 PRECEDING)total from emp1 order by deptno;
```

EMPNO	ENAME	SAL	DEPTNO	TOTAL
105	kalpita	35000	1	35000
104	suman	95000	1	130000
103	kajal	65000	1	195000
102	julie	55000	1	215000
101	john	25000	1	145000
108	neha	87000	2	87000
107	sonal	65000	2	152000
106	darshana	25000	2	177000
110	priya	20000	3	20000
109	ankita	25000	3	45000
111	sheela	25000	3	70000

11 rows selected.

Q. Display empno,name,salary,sum of 3 earlier rows and 1 next row dept wise.

Output:

```
SQL> select empno,ename,sal,deptno,sum(sal) over (partition by deptno order by d
eptno rows BETWEEN 3 PRECEDING AND 1 FOLLOWING)total from emp1 order by deptno;
```

EMPNO	ENAME	SAL	DEPTNO	TOTAL
105	kalpita	35000	1	130000
104	suman	95000	1	195000
103	kajal	65000	1	250000
102	julie	55000	1	275000
101	john	25000	1	240000
108	neha	87000	2	152000
107	sonal	65000	2	177000
106	darshana	25000	2	177000
110	priya	20000	3	45000
109	ankita	25000	3	70000
111	sheela	25000	3	70000

11 rows selected.

Q. Display empno,ename,deptno,sal and sum of salary for 1 earlier row and 1 next row.

Output:

```
SQL> select empno,ename,sal,deptno,sum(sal) over (partition by deptno order by d
eptno rows BETWEEN 1 PRECEDING AND 1 FOLLOWING)total from emp1 order by deptno;
```

EMPNO	ENAME	SAL	DEPTNO	TOTAL
105	kalpita	35000	1	130000
104	suman	95000	1	195000
103	kajal	65000	1	215000
102	julie	55000	1	145000
101	john	25000	1	80000
108	neha	87000	2	152000
107	sonal	65000	2	177000
106	darshana	25000	2	90000
110	priya	20000	3	45000
109	ankita	25000	3	70000
111	sheela	25000	3	50000

11 rows selected.

Q.Display empno,ename,deptno,sal and sum of salary for 1 following and 3 following dept wise.

Output:

```
SQL> select empno,ename,sal,deptno,sum(sal) over (partition by deptno order by d
eptno rows BETWEEN 1 FOLLOWING AND 3 FOLLOWING)total from emp1 order by deptno;
```

EMPNO	ENAME	SAL	DEPTNO	TOTAL
105	kalpita	35000	1	215000
104	suman	95000	1	145000
103	kajal	65000	1	80000
102	julie	55000	1	25000
101	john	25000	1	
108	neha	87000	2	90000
107	sonal	65000	2	25000
106	darshana	25000	2	
110	priya	20000	3	50000
109	ankita	25000	3	25000
111	sheela	25000	3	

11 rows selected.

Q. Display empno,ename,deptno,sal and sum of salary for 3 preceding and 1 preceding row dept wise.

Output:

```
SQL> select empno,ename,sal,deptno,sum(sal) over (partition by deptno order by d
eptno rows BETWEEN 3 PRECEDING AND 1 PRECEDING)total from emp1 order by deptno;
```

EMPNO	ENAME	SAL	DEPTNO	TOTAL
105	kalpita	35000	1	35000
104	suman	95000	1	130000
103	kajal	65000	1	195000
102	julie	55000	1	215000
101	john	25000	1	275000
108	neha	87000	2	240000
107	sonal	65000	2	152000
106	darshana	25000	2	177000
110	priya	20000	3	87000
109	ankita	25000	3	152000
111	sheela	25000	3	20000

11 rows selected.

Q.Display empno,ename,deptno,sal and sum of salary for all preceding and current row dept wise.

Output:

```
SQL> select empno,ename,sal,deptno,sum(sal) over (partition by deptno order by d
eptno rows BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)total from emp1 order by
deptno;
```

EMPNO	ENAME	SAL	DEPTNO	TOTAL
105	kalpita	35000	1	35000
104	suman	95000	1	130000
103	kajal	65000	1	195000
102	julie	55000	1	250000
101	john	25000	1	275000
108	neha	87000	2	240000
107	sonal	65000	2	152000
106	darshana	25000	2	177000
110	priya	20000	3	20000
109	ankita	25000	3	45000
111	sheela	25000	3	70000

11 rows selected.

Q. Write a query to find out information of employee who were hired first in every department.

Output:

```
SQL> select deptno,ename,sal,min(hire) keep(dense_rank FIRST order by hire) over
(partition by deptno) from empl;
```

DEPTNO	ENAME	SAL	MIN(HIRE)KEEP(DENSE_
1	kalpita	35000	01-aug-2014
1	suman	95000	01-aug-2014
1	kajal	65000	01-aug-2014
1	julie	55000	01-aug-2014
1	john	25000	01-aug-2014
2	neha	87000	01-aug-2014
2	sonal	65000	01-aug-2014
2	darshana	25000	01-aug-2014
3	priya	20000	01-aug-2014
3	ankita	25000	01-aug-2014
3	sheela	25000	01-aug-2014

11 rows selected.

Q. write a query which returns salary from previous row. Give colname as sal_previous.

Output:

```
SQL> select empno,ename,job,sal,lag(sal,1,0) over (order by sal) as sal_prev fro
m empl order by sal;
```

EMPNO	ENAME	JOB	SAL	SAL_PREV
110	priya	programmer	20000	0
111	sheela	programmer	25000	20000
101	john	manager	25000	25000
109	ankita	manager	25000	25000
106	darshana	designer	25000	25000
105	kalpita	manager	35000	25000
102	julie	designer	55000	35000
103	kajal	manager	65000	55000
107	sonal	manager	65000	65000
108	neha	programmer	87000	65000
104	suman	designer	95000	87000

11 rows selected.

Q. calculate difference between salary of current row with previous row.

Output:

```
SQL> select empno,ename,sal,lag(sal,1,0) over (order by sal) as sal_prev,sal-lag
(sal,1,0) over (order by sal) as diff from empl;
```

EMPNO	ENAME	SAL	SAL_PREV	DIFF
110	priya	20000	0	20000
111	sheela	25000	20000	5000
101	john	25000	25000	0
109	ankita	25000	25000	0
106	darshana	25000	25000	0
105	kalpita	35000	25000	10000
102	julie	55000	35000	20000
103	kajal	65000	55000	10000
107	sonal	65000	65000	0
108	neha	87000	65000	22000
104	suman	95000	87000	8000

11 rows selected.

Name: Arif Shaikh

Q. write a query which returns the sal from next row. Give colname as sal_next.

Output:

```
SQL> select empno,ename,job,sal,lead(sal,1,0) over (order by sal) as sal_next fr
om empl order by sal;
```

EMPNO	ENAME	JOB	SAL	SAL_NEXT
110	priya	programmer	20000	25000
111	sheela	programmer	25000	25000
101	john	manager	25000	25000
109	ankita	manager	25000	25000
106	darshana	designer	25000	35000
105	kalpita	manager	35000	55000
102	julie	designer	55000	65000
103	kajal	manager	65000	65000
107	sonal	manager	65000	87000
108	neha	programmer	87000	95000
104	suman	designer	95000	0

11 rows selected.

Q. Calculate the difference between current sal and its following salary.

Output:

```
SQL> select empno,ename,sal,lag(sal,1,0) over (order by sal) as sal_prev,sal-lea
d(sal,1,0) over (order by sal) as diff from empl;
```

EMPNO	ENAME	SAL	SAL_PREV	DIFF
110	priya	20000	0	-5000
111	sheela	25000	20000	0
101	john	25000	25000	0
109	ankita	25000	25000	0
106	darshana	25000	25000	-10000
105	kalpita	35000	25000	-20000
102	julie	55000	35000	-10000
103	kajal	65000	55000	0
107	sonal	65000	65000	-22000
108	neha	87000	65000	-8000
104	suman	95000	87000	95000

11 rows selected.

Q. Write a query which returns the difference of salary of current and previous row department wise.

Output:

```
SQL> select empno,ename,sal,lag(sal,1,0) over (partition by deptno order by sal)
as sal_prev,sal-lag(sal,1,0) over (partition by deptno order by sal) as diff from empl;
```

EMPNO	ENAME	SAL	SAL_PREV	DIFF
110	priya	20000	0	20000
101	john	25000	0	5000
106	darshana	25000	0	0
111	sheela	25000	20000	0
109	ankita	25000	25000	0
105	kalpita	35000	25000	10000
102	julie	55000	35000	20000
107	sonal	65000	25000	10000
103	kajal	65000	55000	0
108	neha	87000	65000	22000
104	suman	95000	65000	8000

11 rows selected.

FYMCA Sem – I RollNo:39

Q. create table sales

Query: -

```
create table sales
(time number(4),
Region varchar2(7) check(region
in('central','east','west')),
department varchar2(10) check(department
in('pensales','booksales')),
profit number(5));
```

Insert 10 records in sales table.

```
insert into sales values(2000,'central','pensales',5000);
insert into sales values(2001,'central','pensales',6000);
insert into sales values(2002,'east','booksales',5500);
insert into sales values(2000,'west','pensales',5000);
insert into sales
values(2003,'central','booksales',5000);
insert into sales values(2001,'east','pensales',5500);
insert into sales values(2002,'west','booksales',6000);
insert into sales values(2000,'central','pensales',7000);
insert into sales
values(2002,'central','booksales',2000);
insert into sales values(2000,'east','pensales',2000);
insert into sales values(2000,'east','booksales',2000);
insert into sales values(2000,'east','pensales',12000);
```

Display data of sales.

Output:

```
SQL> select * from sales;

-----
TIME REGION DEPARTMENT PROFIT
-----
2000 central pensales 5000
2001 central pensales 6000
2002 east booksales 5500
2000 west pensales 5000
2003 central booksales 5000
2001 east pensales 5500
2002 west booksales 6000
2000 central pensales 7000
2002 central booksales 2000
2001 west booksales 11000
2000 west booksales 11000

TIME REGION DEPARTMENT PROFIT
-----
2000 east pensales 2000
2000 east booksales 2000
2000 east pensales 12000

14 rows selected.
```

Q. Display the time ,region, department and sum of profit from the sales table and group them in the order of time, region and department.

Output:

```
SQL> select time, region, department, sum(profit) as PROFIT from sales group by time, region, department;
```

TIME	REGION	DEPARTMENT	PROFIT
2003	central	booksales	5000
2001	east	pensales	5500
2000	west	booksales	11000
2002	west	booksales	6000
2002	central	booksales	2000
2001	west	booksales	11000
2000	central	pensales	12000
2001	central	pensales	6000
2000	west	pensales	5000
2000	east	booksales	2000
2000	east	pensales	14000
2002	east	booksales	5500

12 rows selected.

Q. Display the time ,region, department and sum of profit from the sales table and group them up using a roll up function in the order of time, region and department.

Output:

```
SQL> select time,region,department,sum(profit) as PROFIT from sales group by ROLLUP(time,region,department);
```

TIME	REGION	DEPARTMENT	PROFIT
2000	west	pensales	5000
2000	west	booksales	11000
2000	west		16000
2000	central	pensales	12000
2000	central		12000
2000		pensales	28000
2001	east		5500
2001	east	pensales	5500
2001	west	booksales	11000
2001	west		11000
2001	central	pensales	6000
2001	central		6000
2001			22500
2002	east	booksales	5500
2002	east		5500
2002	west	booksales	6000
2002	west		6000
2002	central	booksales	2000
2002	central		2000
2002			13500
2003	central	booksales	5000
2003	central		5000
2003			69000

24 rows selected.

Q. Perform a partial rollup on the sales table by grouping the time first and the region and department together.

Output:

```
SQL> select time,region,department,sum(profit) as PROFIT from sales GROUP BY time, ROLLUP(region,department);
```

TIME	REGION	DEPARTMENT	PROFIT
2000	east	pensales	2000
2000	east		2000
2000	west	pensales	5000
2000	west	booksales	11000
2000	west		16000
2000	central	pensales	12000
2000	central		12000
2000			30000
2001	east	pensales	5500
2001	east		5500
2001	west	booksales	11000
2001			
2001	west		11000
2001	central	pensales	6000
2001	central		6000
2001			22500
2002	east	booksales	5500
2002	east		5500
2002	west	booksales	6000
2002	west		6000
2002	central	booksales	2000
2002	central		2000
2002			13500
2003	central	booksales	5000
2003	central		5000
2003			5000

25 rows selected.

```
SQL> select time,region,department,sum(profit) as PROFIT from sales GROUP BY time, ROLLUP(region,department);
```

Q. Using a grouping clause display time, region and department sum of profit and rollup by time, region and department.

Output:

```
SQL> select time,region,department,sum(profit) as PROFIT,GROUPING(time) as GT,GROUPING(region)as GR,GROUPING(department) as GD from sales GROUP BY ROLLUP(time,region,department);
```

TIME	REGION	DEPARTMENT	PROFIT	GT	GR	GD
2000	east	pensales	14000	0	0	0
2000	east	booksales	2000	0	0	0
2000	east		16000	0	0	1
2000	west	pensales	5000	0	0	0
2000	west	booksales	11000	0	0	0
2000	west		16000	0	0	1
2000	central	pensales	12000	0	0	0
2000	central		12000	0	0	1
2000			44000	0	1	1
2001	east	pensales	5500	0	0	0
2001	east		5500	0	0	1
2001	west	booksales	11000	0	0	0
2001	west		11000	0	0	1
2001	central	pensales	6000	0	0	0
2001	central		6000	0	0	1
2001			22500	0	1	1
2002	east	booksales	5500	0	0	0
2002	east		5500	0	0	1
2002	west	booksales	6000	0	0	0
2002	west		6000	0	0	1
2002	central	booksales	2000	0	0	0
2002	central		2000	0	0	1
2002			13500	0	1	1
2003	central	booksales	5000	0	0	0
2003	central		5000	0	0	1
2003			5000	0	1	1
2003			85000	1	1	1

27 rows selected.

Q. Display the time, region and department from sales table and group them and cube only for time, region and department.

Output:

```
SQL> select time,nvl(region,0) REGION,nvl(department,0)DEPT,sum(profit) as PROFIT from sales GROUP BY CUBE(time,region,department);
```

TIME	REGION	DEPT	PROFIT
0	0		85000
0		pensales	42500
0		booksales	42500
east	0		27000
east		pensales	19500
east		booksales	7500
west	0		33000
west		pensales	5000
west		booksales	28000
central	0		25000
central		pensales	18000

TIME	REGION	DEPT	PROFIT
central		booksales	7000
2000	0	0	44000
2000	0	pensales	31000
2000	0	booksales	13000
2000	east	0	16000
2000	east	pensales	14000
2000	east	booksales	2000
2000	west	0	16000
2000	west	pensales	5000
2000	west	booksales	11000
2000	central	0	12000

TIME	REGION	DEPT	PROFIT
2000	central	pensales	12000
2001	0	0	22500
2001	0	pensales	11500
2001	0	booksales	11000
2001	east	0	5500
2001	east	pensales	5500
2001	west	0	11000
2001	west	booksales	11000
2001	central	0	6000
2001	central	pensales	6000

2002	0	0	13500
------	---	---	-------

TIME	REGION	DEPT	PROFIT
2002	0	booksales	13500
2002	east	0	5500
2002	east	booksales	5500
2002	west	0	6000
2002	west	booksales	6000
2002	central	0	2000
2002	central	booksales	2000
2003	0	0	5000
2003	0	booksales	5000
2003	central	0	5000
2003	central	booksales	5000

44 rows selected.

Q. Display time, region and department from sales table and group them by time and cube only for region and department.

Output:

SQL> select time,region,department,sum(profit) as PROFIT from sales GROUP BY time,CUBE(region,department);

TIME	REGION	DEPARTMENT	PROFIT

2000			44000
2000		pensales	31000
2000		booksales	13000
2000	east		16000
2000	east	pensales	14000
2000	east	booksales	2000
2000	west		16000
2000	west	pensales	5000
2000	west	booksales	11000
2000	central		12000
2000	central	pensales	12000

TIME	REGION	DEPARTMENT	PROFIT

2001			22500
2001		pensales	11500
2001		booksales	11000
2001	east		5500
2001	east	pensales	5500
2001	west		11000
2001	west	booksales	11000
2001	central		6000
2001	central	pensales	6000
2002			13500
2002		booksales	13500

TIME	REGION	DEPARTMENT	PROFIT

2002	east		5500
2002	east	booksales	5500
2002	west		6000
2002	west	booksales	6000
2002	central		2000
2002	central	booksales	2000
2003			5000
2003		booksales	5000
2003	central		5000
2003	central	booksales	5000

Q. Display time, region and department from sales table and apply grouping sets on them.

Output:

9 rows selected.

SQL> SELECT REGION, DEPARTMENT, TIME,SUM(PROFIT) AS PROFIT FROM SALES GROUP BY grouping sets(region, department,time);

REGION	DEPARTMENT	TIME	PROFIT

west			33000
central			25000
east			27000
	booksales		42500
	pensales		42500
		2001	22500
		2000	44000
		2003	5000
		2002	13500

9 rows selected.