A.Rollup, Partial Rollup, Cube, Partial Cube

Q.1 . Create a table Sales with the attribute dept_id, deptname, year_of_sales,region and profit.

```
SQL>  create table sales33
  2     (
  3     dept_id number,
  4     dept_name varchar(10),
  5     year_of_sales number,
  6     region varchar(10),
  7     profit number
  8     );

Table created.
```

Perform the rollup operation on this table.

1. **Display year wise total profit.**

select year_of_sales, sum(profit) from sales33 group by rollup(year_of_sales);

2. **Display year wise total profit of each region.**

select year_of_sales, region, sum(profit) from sales33 group by rollup(year_of_sales, region);

3. **Display year wise, region wise and department wise total profit for the department "IT"**

select year_of_sales, region, dept_name, sum(profit) from sales33 group by rollup(year_of_sales, region, dept_name) having dept_name = 'IT';

4. **Display year wise total profit of each department.**

select year_of_sales, dept_name, sum(profit) from sales33 group by rollup(year_of_sales, dept_name);

5. **Display region wise total profit of each department.**

select region, dept_name, sum(profit) from sales33 group by rollup(region, dept_name);

6. **Display region wise total profit if total profit >1500**

select year_of_sales, region, dept_name, sum(profit) from sales33 group by rollup(year_of_sales, region, dept_name) having sum(profit)>1500;

7. **Display region wise total profit.**

select region, sum(profit) from sales33 group by rollup(region);

8. **Display department wise total profit.**

select dept_name, sum(profit) from sales33 group by rollup(dept_name);

Q.2 Apply partial rollup on same table.

1. **Display region wise total profit of each department by partially rolling the year**

select year_of_sales, region, dept_name, sum(profit) from sales33 group by year_of_sales, rollup(region, dept_name);

Q.3 Implement Cube operation on the same table.

1. **Display year, region and dept wise total profit using cube function.**

select year_of_sales, region, dept_name, sum(profit) from sales33 group by cube(year_of_sales, region, dept_name);

2. **Display region and dept wise total profit using year_of_sales as partial cube dimension.**

select year_of_sales, region, dept_name, sum(profit) from sales33 group by year_of_sales, cube(region, dept_name);

B. Rank and Dense Rank

Q1. Create a table student with attribute roll_num, name, subject, marks.

```
SQL> create table student33(
  2  roll_num number primary key,
  3  s_name varchar(20),
  4  subject varchar(20),
  5  marks number
  6  );

Table created.
```

1. **Display content of table.**

select * from students33;

2. **Assign sequence order for the student for the same subject based on their marks.**

select roll_num, s_name, subject, marks, rank() over(partition by subject order by marks) as rank from student33;

3. **Assign sequential order for the student for the same subject based on their marks in descending order**

select roll_num, s_name, subject, marks, rank() over(partition by subject order by marks desc) as rank from student33;

4. **Assign sequential order using dense rank function.**

select roll_num, s_name, subject, marks, dense_rank() over(partition by subject order by marks) as rank

from student33;

C. First and Last

Q.2) Find the first and the last.

1. **Display the lowest marks of each subject.**

select roll_num, s_name, subject, marks, min(marks) keep (dense_rank FIRST order by marks) over (partition by subject) as lowest

from student33 order by subject, marks;

2. **Display the highest marks of each subject.**

select roll_num, s_name, subject, marks, min(marks) keep (dense_rank LAST order by marks) over (partition by subject) as highest

from student33 order by subject, marks;

D. LEAD and LAG

Q3. Create a table employee with attribute empid, name, deptid,deptname, salary and joining date.

```
SQL> create table employee33 (
  2  emp_id number primary key,
  3  e_name varchar(20),
  4  dept_id number,
  5  dept_name varchar(20),
  6  salary number,
  7  join_date date
  8  );

Table created.
```

1. **Display contetnts of the table.**

Select * from employee33;

2. **Display the joining details of the entire employee joined just after the joining date of each employee in sales department.**

select emp_id, e_name, join_date, LEAD(join_date, 1) over (order by join_date) as next_join

from employee33 where dept_name = 'HR';

3. **Display joining date of all employee joined just before the joining date of employee.**

SELECT emp_id, e_name, join_date, LAG(join_date, 1) OVER (ORDER BY join_date) AS last_join

FROM employee33

WHERE dept_name = 'HR'

ORDER BY join_date;

4. **For each employee in employee table display the salary of the employee joined just before.**

SELECT emp_id, e_name, join_date, salary,

LAG(salary, 1) OVER (ORDER BY join_date) AS next_join

FROM employee33

WHERE dept_name = 'HR' ORDER BY join_date;

E. Windowing functions

Q.4 Create a table employee with attribute emp_no , emp_name , dept_name and salary.

1. **Display emp_no,emp_name,dept_name,salary and dept wise sum of salary of current and previous two records.**

```
SELECT emp_id, e_name, dept_name, salary,
    SUM(salary) OVER (PARTITION BY dept_name ORDER BY dept_name ROWS 2 PRECEDING) AS total
FROM employee33
ORDER BY dept_name;
```

2. **Display emp_no,emp_name,dept_name,salary and sum of salary for 3 earlier row and 1 next row dept wise.**

```
SELECT emp_id, e_name, dept_name, salary,
    SUM(salary) OVER (
        PARTITION BY dept_name
        ORDER BY salary
        ROWS BETWEEN 3 PRECEDING AND 1 FOLLOWING
    ) AS total
FROM employee33
ORDER BY dept_name, salary;
```

3. **Display emp_no.emp_name,dept_name,salary and sum of salary 3 preceding row and 1 preceding row dept wise**

```
SELECT emp_id, e_name, dept_name, salary,
    SUM(salary) OVER (
        PARTITION BY dept_name
        ORDER BY salary
        ROWS BETWEEN 3 PRECEDING AND 1 PRECEDING
    ) AS total
FROM employee33
ORDER BY dept_name, salary;
```

4. **Display emp_no.emp_name,dept_name,salary and sum of salary 1 following and 3 following row dept wise.**

```
SELECT emp_id, e_name, dept_name, salary,
    SUM(salary) OVER (
        PARTITION BY dept_name
        ORDER BY salary
        ROWS BETWEEN 1 FOLLOWING AND 3 FOLLOWING
    ) AS total
FROM employee33
ORDER BY dept_name, salary;
```