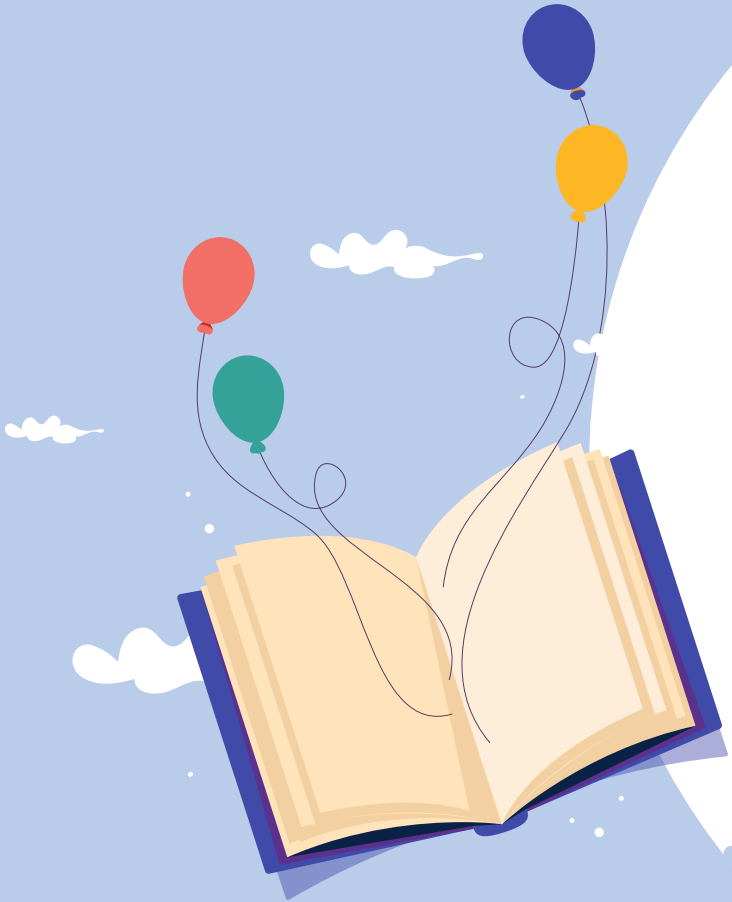
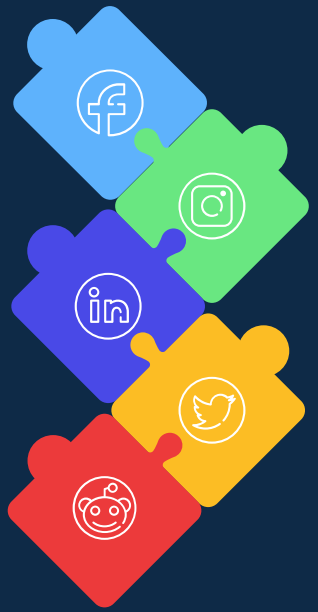


OracleDatabase

For coding starter

INDEX

- Database concept
- **Practice SQL(2)**
- DML
- PL/SQL



FUNCTION

Here are the key characteristics of an Oracle Database function:

1. **Name**: A function is given a unique name within the database schema.
2. **Input Parameters**: Functions can take input parameters, which are values passed to the function for processing.
3. **Return Value**: Unlike procedures that don't have a return value, functions return a single value. This value can be of any data type such as number, string, date, or even a complex user-defined type.
4. **Logic**: Inside the function's body, you can write PL/SQL code to perform calculations, queries, or any other operations required to compute the return value.
5. **Deterministic vs. Non-deterministic**: Functions can be classified as deterministic or non-deterministic. A deterministic function, when given the same input, will always produce the same output. This is crucial for optimizations within the database.
6. **Usage**: Functions can be used in SQL queries, PL/SQL blocks, or any other context where an expression can be used. For example, you could use a function to calculate the total sales of a product or determine the age of a person based on their birthdate.
7. **Security**: Functions can have access to the database's data, but their permissions can be controlled to restrict what data they can access and modify.

MULTIPLE-ROW & GROUP FUNCTION

- You can import and use basic calculation functions.

```
SELECT SUM(SAL)
  FROM EMP;
```

SUM(SAL)
24925

```
SELECT COUNT(*)
  FROM EMP
 WHERE DEPTNO = 30;
```

COUNT(*)
6

```
SELECT MAX(SAL)
  FROM EMP
 WHERE DEPTNO = 10;
```

MAX(SAL)
5000

```
SELECT MIN(SAL)
  FROM EMP
 WHERE DEPTNO = 10;
```

MIN(SAL)
1300

```
SELECT AVG(SAL)
  FROM EMP
 WHERE DEPTNO = 30;
```

AVG(SAL)
1566.6666666666666666

MULTIPLE-ROW & GROUP FUNCTION

- A GROUP BY clause, part of a *SelectExpression*, groups a result into subsets that have matching values for one or more columns. In each group, no two rows have the same value for the grouping column or columns. NULLs are considered equivalent for grouping purposes.

```
SELECT AVG(SAL), '10' AS DEPTNO FROM EMP WHERE DEPTNO = 10
UNION ALL
SELECT AVG(SAL), '20' AS DEPTNO FROM EMP WHERE DEPTNO = 20
UNION ALL
SELECT AVG(SAL), '30' AS DEPTNO FROM EMP WHERE DEPTNO = 30;
```

```
SELECT AVG(SAL), DEPTNO
  FROM EMP
GROUP BY DEPTNO;
```

AVG(SAL)	DEPTNO
1566.6666666666666666666666666666666666667	30
2258.3333333333333333333333333333333333333	20
2916.6666666666666666666666666666666666667	10

MULTIPLE-ROW & GROUP FUNCTION

- The HAVING clause restricts a grouped table, specifying a search condition (much like a WHERE clause) that can refer only to grouping columns or aggregates from the current scope.

```
SELECT DEPTNO, JOB, AVG(SAL)
  FROM EMP
GROUP BY DEPTNO, JOB
HAVING AVG(SAL) >= 2000
ORDER BY DEPTNO, JOB;
```

DEPTNO	JOB	AVG(SAL)
10	MANAGER	2450
10	PRESIDENT	5000
20	ANALYST	3000
20	MANAGER	2975
30	MANAGER	2850

MULTIPLE-ROW & GROUP FUNCTION

- -

```
SELECT DEPTNO, JOB, AVG(SAL)
  FROM EMP
 WHERE AVG(SAL) >= 2000
GROUP BY DEPTNO, JOB
ORDER BY DEPTNO, JOB;
```

Type	Message
Error (1 item)	
Execution	ORA-00934: group function is not allowed here

```
SELECT DEPTNO, JOB, AVG(SAL)
  FROM EMP
 WHERE SAL <= 3000
GROUP BY DEPTNO, JOB
HAVING AVG(SAL) >= 2000
ORDER BY DEPTNO, JOB;
```

DEPTNO	JOB	AVG(SAL)
10	MANAGER	2450
20	ANALYST	3000
20	MANAGER	2975
30	MANAGER	2850

MULTIPLE-ROW & GROUP FUNCTION

- **ROLLUP** and CUBE are simple extensions to the SELECT statement's GROUP BY clause.

```
SELECT DEPTNO, JOB, COUNT(*), MAX(SAL), SUM(SAL), AVG(SAL)
  FROM EMP
GROUP BY ROLLUP(DEPTNO, JOB);
```

DEPTNO	JOB	COUNT(*)	MAX(SAL)	SUM(SAL)	AVG(SAL)
10	CLERK	1	1300	1300	1300
10	MANAGER	1	2450	2450	2450
10	PRESIDENT	1	5000	5000	5000
10		3	5000	8750	2916.66666666666666666666666666666666666667
20	CLERK	1	800	800	800
20	ANALYST	1	3000	3000	3000
20	MANAGER	1	2975	2975	2975
20		3	3000	6775	2258.33333333333333333333333333333333333333
30	CLERK	1	950	950	950
30	MANAGER	1	2850	2850	2850
30	SALESMAN	4	1600	5600	1400
30		6	2850	9400	1566.66666666666666666666666666666666666667
		12	5000	24925	2077.08333333333333333333333333333333333333

MULTIPLE-ROW & GROUP FUNCTION

- ROLLUP and **CUBE** are simple extensions to the SELECT statement's GROUP BY clause.

```
SELECT DEPTNO, JOB, COUNT(*), MAX(SAL), SUM(SAL), AVG(SAL)
  FROM EMP
GROUP BY CUBE(DEPTNO, JOB)
ORDER BY DEPTNO, JOB;
```

DEPTNO	JOB	COUNT(*)	MAX(SAL)	SUM(SAL)	AVG(SAL)
10	CLERK	1	1300	1300	1300
10	MANAGER	1	2450	2450	2450
10	PRESIDENT	1	5000	5000	5000
10		3	5000	8750	2916.6666666666666666666666666666666666666667
20	ANALYST	1	3000	3000	3000
20	CLERK	1	800	800	800
20	MANAGER	1	2975	2975	2975
20		3	3000	6775	2258.3333333333333333333333333333333333333
30	CLERK	1	950	950	950
30	MANAGER	1	2850	2850	2850
30	SALESMAN	4	1600	5600	1400
30		6	2850	9400	1566.6666666666666666666666666666666666667
	ANALYST	1	3000	3000	3000
	CLERK	3	1300	3050	1016.6666666666666666666666666666666666667
	MANAGER	3	2975	8275	2758.3333333333333333333333333333333333333
	PRESIDENT	1	5000	5000	5000
	SALESMAN	4	1600	5600	1400
		12	5000	24925	2077.0833333333333333333333333333333333333

MULTIPLE-ROW & GROUP FUNCTION

- GROUPING SETS syntax lets you define multiple groupings in the same query. GROUP BY computes all the groupings specified and combines them with UNION ALL. For example, consider the following statement:

```
SELECT DEPTNO, JOB, COUNT(*)
  FROM EMP
GROUP BY GROUPING SETS(DEPTNO, JOB)
ORDER BY DEPTNO, JOB;
```

DEPTNO	JOB	COUNT(*)
10		3
20		3
30		6
	ANALYST	1
	CLERK	3
	MANAGER	3
	PRESIDENT	1
	SALESMAN	4

MULTIPLE-ROW & GROUP FUNCTION

- GROUPING distinguishes superaggregate rows from regular grouped rows.
- GROUP BY extensions such as ROLLUP and CUBE produce superaggregate rows where the set of all values is represented by null.

```
SELECT DEPTNO, JOB, COUNT(*), MAX(SAL), SUM(SAL), AVG(SAL),
       GROUPING(DEPTNO),
       GROUPING(JOB)
  FROM EMP
GROUP BY CUBE(DEPTNO, JOB)
ORDER BY DEPTNO, JOB;
```

DEPTNO	JOB	COUNT(*)	MAX(SAL)	SUM(SAL)	AVG(SAL)	GROUPING(DEPTNO)	GROUPING(JOB)
10	CLERK	1	1300	1300	1300	0	0
10	MANAGER	1	2450	2450	2450	0	0
10	PRESIDENT	1	5000	5000	5000	0	0
10		3	5000	8750	2916.6666666666666666666666666666666666667	0	1
20	ANALYST	1	3000	3000	3000	0	0
20	CLERK	1	800	800	800	0	0
20	MANAGER	1	2975	2975	2975	0	0
20		3	3000	6775	2258.3333333333333333333333333333333333333	0	1
30	CLERK	1	950	950	950	0	0
30	MANAGER	1	2850	2850	2850	0	0
30	SALESMAN	4	1600	5600	1400	0	0
30		6	2850	9400	1566.6666666666666666666666666666666666667	0	1
	ANALYST	1	3000	3000	3000	1	0
	CLERK	3	1300	3050	1016.6666666666666666666666666666666666667	1	0
	MANAGER	3	2975	8275	2758.3333333333333333333333333333333333333	1	0
	PRESIDENT	1	5000	5000	5000	1	0
	SALESMAN	4	1600	5600	1400	1	0
		12	5000	24925	2077.0833333333333333333333333333333333333	1	1

MULTIPLE-ROW & GROUP FUNCTION

- For a specified measure, LISTAGG orders data within each group specified in the ORDER BY clause and then concatenates the values of the measure column.

```
SELECT DEPTNO,
       LISTAGG(ENAME, ', ')
       WITHIN GROUP(ORDER BY SAL DESC) AS ENAMES
  FROM EMP
GROUP BY DEPTNO;
```

DEPTNO	ENAMES
10	KING, CLARK, MILLER
20	FORD, JONES, SMITH
30	BLAKE, ALLEN, TURNER, MARTIN, WARD, JAMES

MULTIPLE-ROW & GROUP FUNCTION

This article shows how to use the new PIVOT and UNPIVOT operators in 11g, as well as giving a pre-11g solution to the same problems.

```
SELECT *
  FROM(SELECT DEPTNO, JOB, SAL
         FROM EMP)
PIVOT(MAX(SAL)
      FOR DEPTNO IN (10, 20, 30)
      )
ORDER BY JOB;
```

JOB	10	20	30
ANALYST		3000	
CLERK	1300	800	950
MANAGER	2450	2975	2850
PRESIDENT	5000		
SALESMAN			1600

MULTIPLE-ROW & GROUP FUNCTION

This article shows how to use the new PIVOT and UNPIVOT operators in 11g, as well as giving a pre-11g solution to the same problems.

```sql
SELECT *
  FROM(SELECT DEPTNO,
              MAX(DECODE(JOB, 'CLERK' , SAL)) AS "CLERK",
              MAX(DECODE(JOB, 'SALESMAN' , SAL)) AS "SALESMAN",
              MAX(DECODE(JOB, 'PRESIDENT', SAL)) AS "PRESIDENT",
              MAX(DECODE(JOB, 'MANAGER' , SAL)) AS "MANAGER",
              MAX(DECODE(JOB, 'ANALYST' , SAL)) AS "ANALYST"
         FROM EMP
       GROUP BY DEPTNO
       ORDER BY DEPTNO)
UNPIVOT(
   SAL FOR JOB IN (CLERK, SALESMAN, PRESIDENT, MANAGER,ANALYST))
ORDER BY DEPTNO, JOB;
```

DEPTNO	JOB	SAL
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
20	ANALYST	3000
20	CLERK	800
20	MANAGER	2975
30	CLERK	950
30	MANAGER	2850
30	SALESMAN	1600