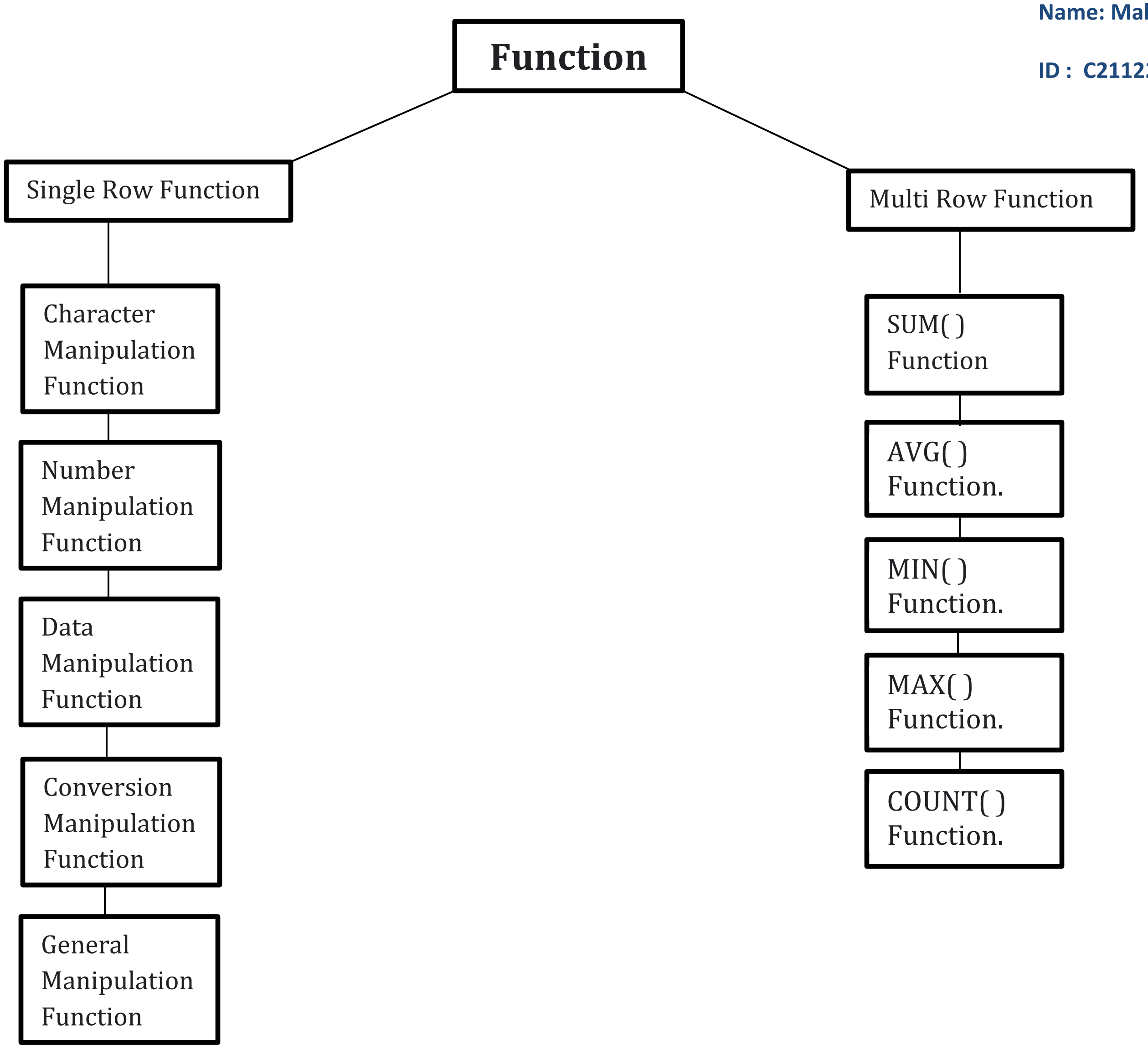
Name: Maksura Nuha

ID : C211224


Function
Single Row Function
Character Manipulation Function
Number Manipulation Function
Data Manipulation Function
Conversion Manipulation Function
General Manipulation Function
Multi Row Function
SUM( ) Function
AVG( ) Function.
MIN( ) Function.
MAX( ) Function.
COUNT( ) Function.

- **Single Row functions** - Single row functions are the one who work on single row and return one output per row.

- **Multiple Row functions** - Multiple row functions work upon group of rows and return one result for the complete set of rows. We have to use Group By for multi row functions.

Character Manipulation Function(Case)

- **Use of Lower( ) :** //converts into Lower case

Select first_name, lower (first_name)

from employees

- **Use of Upper( ) :** //converts into Upper case

Select first_name, upper (first_name)

from employees

❖ **Use of Initcap( ) :** //converts into first letter of each word in uppercase, all other letters in lowercase

Select first_name, initcap (first_name)

from employees

▪ **Search Using Character Manipulation Function (Lower( ) , Upper( ) , Initcap( ))**

Search employees, where first name starts with 'david' :

Select first_name, lower (first_name) //You can use Upper & Initcap instead of lower

from employees

where lower(first_name)= lower('david')

Character Manipulation Function(Character)

- **Use of Length( ) :**

Select first_name, Length (first_name) //represent length of 1st name

from employees

- **Use of INSTR( ) :**

Select first_name, INSTR(first_name, 'a') //find the substring 'a' in employees 1st name

from employees

- **Use of LPAD( ) :** //LPAD(*string, length, lpad_string*)

The LPAD() function left-pads a string with another string, to a certain length.

Select first_name, LPAD(salary, 10, '#') //represent salary in 10 digits using '#'

from employees

where lower(first_name)=lower(:input_value)

- **Use of Concat( ) :**

Select first_name, Concat(first_name, last_name) //combines first & last name

from employees

- Find first name & name of employees who have 'an' in their name :

Select first_name, Concat(first_name, last_name)

from employees

where first_name LIKE '%an' or last_name LIKE '%an'

- **Use of Substr( ) :**

Select first_name, last_name, job_id//seperates string to search using position

from employees

where SUBSTR(job_id, 4)='REP' //here 'REP' string, is in 4th position as 'SA_REP'

- **Use of RPAD( ) :** //RPAD(*string, length, rpad_string*)

The RPAD() function right-pads a string with another string, to a certain length.

SELECT RPAD(First_Name, 10, 'a')

FROM employees

- **Use of TRIM( ) :**

```
1. SELECT TRIM('@$ ' FROM '    @Whatever$    ')
2. SELECT TRIM('dav')
from employees
```

- **Use of REPLACE( ) :**

SELECT REPLACE('david', 'd', 'v')

from employees

Number Function

1	ABS() Returns the absolute value of numeric expression.
2	ACOS() Returns the arccosine of numeric expression. Returns NULL if the value is not in the range -1 to 1.
3	ASIN() Returns the arcsine of numeric expression. Returns NULL if value is not in the range -1 to 1
4	ATAN() Returns the arctangent of numeric expression.
5	ATAN2() Returns the arctangent of the two variables passed to it.
6	BIT_AND() Returns the bitwise AND all the bits in expression.
7	BIT_COUNT() Returns the string representation of the binary value passed to it.
8	BIT_OR() Returns the bitwise OR of all the bits in the passed expression.
9	CEIL() Returns the smallest integer value that is not less than passed numeric expression
10	CEILING() Returns the smallest integer value that is not less than passed numeric expression

11	CONV() Convert numeric expression from one base to another.
12	COS() Returns the cosine of passed numeric expression. The numeric expression should be expressed in radians.
13	COT() Returns the cotangent of passed numeric expression.
14	DEGREES() Returns numeric expression converted from radians to degrees.
15	EXP() Returns the base of the natural logarithm (e) raised to the power of passed numeric expression.
16	FLOOR() Returns the largest integer value that is not greater than passed numeric expression.
17	FORMAT() Returns a numeric expression rounded to a number of decimal places.
18	GREATEST() Returns the largest value of the input expressions.
19	INTERVAL() Takes multiple expressions exp1, exp2 and exp3 so on.. and returns 0 if exp1 is less than exp2, returns 1 if exp1 is less than exp3 and so on.
20	LEAST() Returns the minimum-valued input when given two or more.
21	LOG() Returns the natural logarithm of the passed numeric expression.
22	LOG10()

	Returns the base-10 logarithm of the passed numeric expression.
23	MOD() Returns the remainder of one expression by diving by another expression.
24	OCT() Returns the string representation of the octal value of the passed numeric expression. Returns NULL if passed value is NULL.
25	PI() Returns the value of pi
26	POW() Returns the value of one expression raised to the power of another expression
27	POWER() Returns the value of one expression raised to the power of another expression
28	RADIANS() Returns the value of passed expression converted from degrees to radians.
29	ROUND() Returns numeric expression rounded to an integer. Can be used to round an expression to a number of decimal points
30	SIN() Returns the sine of numeric expression given in radians.
31	SQRT() Returns the non-negative square root of numeric expression.
32	STD() Returns the standard deviation of the numeric expression.
33	STDDEV() Returns the standard deviation of the numeric expression.

34	TAN() Returns the tangent of numeric expression expressed in radians.
35	TRUNCATE() Returns numeric exp1 truncated to exp2 decimal places. If exp2 is 0, then the result will have no decimal point.

- **Use of ROUND( ) :**

Select ROUND(23.83921, 2) //it will show output 23.84

From DUAL //DUAL is a dummy table, where raw values are inserted

Date Functions

SYS_DATE :

Select employee_id, hire_date, sysdate //shows system date

From employees

ROUND and TRUNC functions are used to round and truncates the date value.

MONTHS_BETWEEN( ):

Select employee_id, hire_date,sys_date, round(MONTHS_BETWEEN (sys_date,hire_date)/12)

From employees

ADD_MONTHS( ) :

Select employee_id, hire_date,sys_date, add_months(sys_date,3)

From employees

NEXT_DAY( ):

Select employee_id, hire_date,sys_date, next_day(sys_date,sunday)

From employees

LAST_DAY( ):

Select employee_id, hire_date,sys_date, last_day(sys_date,sunday)

From employees

Conversion Functions

Use of TO_CHAR():

Represent date in 'DD/MM/YYYY' format -

```
Select employee_id, hire_date, sysdate, TO_CHAR(sysdate,'DD/MM/YYYY')

From employees
```

Represent Hire Date & Salary in '$99999.99' format -

```
Select employee_id, hire_date, TO_CHAR (salary, '$99999.99')

From employees
```

Use of TO_NUMBER():

1. SELECT TO_NUMBER('224.21', '9G999D99')

FROM DUAL

2. SELECT TO_NUMBER('2254.21', '9999.99')

FROM DUAL;

Use of TO_DATE():

```
SELECT TO_DATE('January 15, 1989', 'Month dd, YYYY')

FROM DUAL;
```

General Functions

Use of NVL(): Syntax : NVL(C_N, Value) , C_N is column name //2 expressions

```
SELECT  first_name, NVL(JOB_ID, 'n/a')

FROM employees;
```

Use of NVL2( ): Syntax : NVL2(string1, value_if_not_null, value_if_null) //3 expressions

1. Show Employee who gets salary with commissions & without commissions :

SELECT employee_id,commission_pct,salary, NVL2(commission_pct, salary, salary+salary*commission_pct)

FROM employees;

Use of COALESCE( ): Syntax : COALESCE (expr1, expr2, ... expr_n) //n>3 expressions

SELECT first_name, last_name, COALESCE (Salary, commission_pct) AS SALARY_COMMISSION

FROM employees

Use of DECODE( ):

SELECT first_name, salary, DECODE (hire_date, sysdate,'NEW JOINEE','EMPLOYEE')

FROM employees;

Use of CASE : //When....Then works as If .... Else

```
1. SELECT first_name, CASE     WHEN salary < 200 THEN 'GRADE 1'

                              WHEN salary > 200 AND salary < 5000 THEN 'GRADE 2'

                              ELSE 'GRADE 3'

                         END CASE

FROM employees
```

```
2. Select employee_id , job_id ,
      CASE job_id WHEN 'IT_PROG' THEN salary+salary*0.1
                  WHEN 'SA_REP' THEN salary+salary*0.15
       ELSE salary END
   from Employees
```

Multi Row Function

Types of Multi-Row Functions :

- Maximum(Max)
- Minimum(MIN)

- Average(Avg)
- Sum
- Count

The **Group By** is used to group data based on the same value in a specific column. The **ORDER BY** sorts the result and shows it in ascending or descending order.

- SELECT job_id, sum(salary), round(avg(salary)), max(salary), min(salary), count(job_id)

  FROM employees

  group by job_id

The GROUP BY statement is often used with aggregate functions (COUNT( ), MAX( ), MIN( ), SUM( ), AVG( )) to group the result-set by one or more columns.

Join

In DBMS, a join statement is mainly used to combine two tables based on a specified common field between them.

2 types of join;

1. Inner Join – Natural & Equi
2. Outer Join – Left, Right & Full

Inner Join

A join that can be used to return all the values that have matching values in both the tables.

Syntax:

```
SELECT table1.column1, table1.column2, table2.column1,....
FROM table1
INNER JOIN table2
ON table1.matching_column = table2.matching_column;
```

Natural Join :

```
Select *
from Employees natural join Departments
```

Equi Join :

Select *

from Employees e ,Departments d

where e.department_id=d.department_id

Outer Join

Outer Join is a join that can be used to return the records in both the tables whether it has matching records in both the tables or not.

Right Join Syntax:

SELECT table1.column1, table1.column2, table2.column1,....

FROM table1

RIGHT JOIN table2

ON table1.matching_column = table2.matching_column;

- Select *

  from Employees e right outer join Departments d

  on e.department_id=d.department_id

Left Join Syntax:

SELECT table1.column1, table1.column2, table2.column1,....

FROM table1

LEFT JOIN table2

ON table1.matching_column = table2.matching_column;

- Select *
  from Employees e left outer join Departments d
  on e.department_id=d.department_id

Full Join Syntax:

SELECT table1.column1, table1.column2, table2.column1,....

FROM table1

FULL JOIN table2

ON table1.matching_column = table2.matching_column;

-
  ```
  Select *
  from Employees e full outer join Departments d
  on e.department_id=d.department_id
  ```

Sub Query

A subquery is a SQL query nested inside a larger query.

- Occurs in :
  - A SELECT clause
  - A FROM clause
  - A WHERE clause

1.
   ```
   SELECT  *
   FROM Employees
   WHERE department_id = (SELECT department_id
   FROM employees
   WHERE first_name='Lex') and  first_name!='Lex'
   ```

2.
   ```
   SELECT  a.studentid, a.name, b.total_marks
     FROM student a, marks b
   WHERE a.studentid = b.studentid AND b.total_marks > (SELECT  total_marks
                              FROM marks
                                WHERE student_id = 'V002' )
   ```

3.
   ```
   SELECT  *
   FROM Employees
      WHERE department_id  IN (SELECT department_id
                        FROM Employees
                         WHERE  first_name='David')
   ```