**In Built Functions: -** DATE FUNCTION, NUMERICAL FUNCTIONS, CHARACTER FUNCTIONS, CONVERSION FUNCTION

**CHARACTER/STRING FUNCTIONS:**

SQL> select upper('Geethanjali') from dual;

SQL> select upper('hello’) from dual;

SQL> select lower('HELLO') from dual;

SQL> select initcap(‘hello geethanjali') from dual;

SQL> select ltrim(' hello') from dual;

SQL> select rtrim('hai ')from dual;

SQL> select concat('CVR',' GROUPS')from dual;

SQL> select length('CVR COLLEGE’)from dual;

SQL> select **substr('GEETHANJALI'**, 6,6)from dual;

SQL> select translate('geetha','ha','a')from dual;

 **DATE & TIME FUNCTIONS**

SQL> select sysdate from dual;

SQL> select round(sysdate)from dual;

SQL> select add\_months(sysdate,3)from dual;

SQL> select last\_day(sysdate)from dual;

SQL> select sysdate+20 from dual;

SQL> select next\_day(sysdate,'thursday')from dual;

**NUMERIC FUNCTIONS**

SQL> select round(19.889)from dual;

SQL> select ceil(26.10)from dual;

SQL> select floor(34.56)from dual;

SQL> select trunc(17.6643)from dual;

SQL> select sign(-258)from dual;

SQL> select abs(-60)from dual;

Aggregate Functions:

1. MIN()

Syntax: SELECT MIN(column\_name)  
FROM table\_name  
WHERE condition;

1. MAX()
2. AVG()
3. Count()
4. Sum()

**WEEK 3: Querying**: NESTED QUERIES AND JOIN QUERIES and JOINS

In **nested queries**, a **query** is written inside a **query**. The result of inner **query** is used in execution of outer **query**.

**Example1**: Display all employee names and salary whose salary is greater than minimum salary of the company and job title starts with ‗S„.

**Solution:**

1. Use select from..
   1. Use like operator to match job and in select clause to get the result.

Query: SQL> select ename,sal from emp where sal>(select min(sal) from emp where job like 'A%');

**Example2:** Issue a query to find all the employees who work in the same job as Swetha.

**Query:**

SQL> select \* from emp;

SQL> select ename from emp where job=(select job from emp where ename='Swetha');

**Example3**: Issue a query to display information about employees who earn more than any employee in dept 101.

**Query**: SQL> select \* from emp where sal>(select max(sal) from emp where empno=101);

SQL> select \* from emp;

**Example4**: Display the employee details, departments that the departments are same in both the emp and dept.

**Solution**: 1. Use select from.

2. Use equi join in select clause to get the result.

**Query**: SQL> select \* from emp,dept where emp.deptno=dept.deptno;

**Example5**: Display the employee details, departments that the departments are not same in both the emp and dept.

**Solution:** 1.Use select from.

2. Use non equi join in select clause to get the result.

**Query**: SQL> select \* from emp,dept where emp.deptno!=dept.deptno;

SET OPERATORS:

UNION:

* UNION will be used to combine the result of two select statements.
* Duplicate rows will be eliminated from the results obtained after performing the UNION operation.

EXAMPLE:

Write a query to perform union all operation between the table t\_employees and the table t2\_employees.

**SELECT** \***FROM** t\_employees **UNION** **SELECT** \***FROM** t2\_employees;

Here, in a single query, we have written two SELECT queries. The first SELECT query will fetch the records from the t\_employees table and perform a UNION operation with the records fetched by the second SELECT query from the t2\_employees table.

Since we have performed union operation between both the tables, so only the records from the first and second table are displayed except for the duplicate records.

UNION ALL

* This operator combines all the records from both the queries.
* Duplicate rows will not be eliminated from the results obtained after performing the UNION ALL operation.

**Example 1:**

* Write a query to perform union all operation between the table t\_employees and the table t2\_employees

**SELECT** \***FROM** t\_employees **UNION** ALL **SELECT** \***FROM** t2\_employees;

Here, in a single query, we have written two SELECT queries. The first SELECT query will fetch the records from the t\_employees table and perform UNION ALL operation with the records fetched by the second SELECT query from the t2\_employees table.

Since we have performed union all operation between both the tables, so all the records from the first and second table are displayed, including the duplicate records.

INTERSECT

* It is used to combine two SELECT statements, but it only returns the records which are common from both SELECT statements.

**Example 1:**

Write a query to perform intersect operation between the table t\_employees and the table t2\_employees.

**SELECT** \***FROM** t\_employees **INTERSECT** **SELECT** \***FROM** t2\_employees;

Here, in a single query, we have written two SELECT queries. The first SELECT query will fetch the records from the t\_employees table and perform INTERSECT operation with the records fetched by the second SELECT query from the t2\_employees table

Since we have performed intersect operation between both the tables, so only the common records from both the tables are displayed.

MINUS

* It displays the rows which are present in the first query but absent in the second query with no duplicates.

**Example 1:**

Write a query to perform a minus operation between the table t\_employees and the table t2\_employees.

**SELECT** \***FROM** t\_employees MINUS **SELECT** \***FROM** t2\_employees;

Here, in a single query, we have written two SELECT queries. The first SELECT query will fetch the records from the t\_employees table and perform MINUS operation with the records fetched by the second SELECT query from the t2\_employees table.

Since we have performed Minus operation between both the tables, so only the unmatched records from both the tables are displayed.