

Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology
(Deemed to be University Estd. u/s 3 of UGC Act, 1956) School of Computing

B.Tech. – Information



Technology



VTR UGE2021- (CBCS)

Academic Year: 2025–2026

SUMMER SEMESTER - SS2526

Course Code : 10211IT201

Course Name : Database t System concepts

Slot No : S12L5

DBMS TASK - 3 REPORT

Submitted by:

VTUNO	REGISTER NUMBER	STUDENT NAME
VTU29010	24UEIT0040	P.Manikanta

ABSTRACT

The purpose of this task is to demonstrate the use of various SQL clauses, operators, and built-in functions for data manipulation and retrieval.

It involves using aggregate functions, date/time functions, and comparison/logical operators to query relational data effectively.

By applying these concepts on sample tables, users learn how to perform analytical and conditional data retrieval for better database management and reporting.

AIM:-

Using Clauses, Operators and Functions in queries Perform the query processing on databases for different retrieval results of queries using DML, DRL operations using aggregate, date, string, indent functions, set clauses and operators.

SQL Aggregate Functions

- MIN() - returns the smallest value within the selected column
- MAX() - returns the largest value within the selected column
- COUNT() - returns the number of rows in a set
- SUM() - returns the total sum of a numerical column
- AVG() - returns the average value of a numerical column

Step 1:

Execute the below statement to create an employee table:

```
CREATE TABLE Employee (
    Name VARCHAR2(45) NOT NULL,
    Occupation VARCHAR2(35) NOT NULL,
    Working_Date DATE,
    Working_Hours NUMBER(3)
);
```

SQL> DESC EMPLOYEE

Name	Null?	Type
NAME	NOT NULL	VARCHAR2(45)
OCCUPATION	NOT NULL	VARCHAR2(35)
WORKING_DATE		DATE
WORKING_HOURS		NUMBER(3)

Step 2 : Execute the below statement to insert the records

INSERT INTO employee VALUES

('Robin', 'Scientist', '2020-10-04', 12), ('Warner', 'Engineer', '2020-10-04', 10), ('Peter', 'Actor', '2020-10-04', 13), ('Marco', 'Doctor', '2020-10-04', 14), ('Brayden', 'Teacher', '2020-10-04', 12), ('Antonio', 'Business', '2020-10-04', 11);

SQL> SELECT*FROM EMPLOYEE;

NAME	OCCUPATION	WORKING_DATE	WORKING_HOURS
Robin			
Scientist		04-OCT-20	12
Warner			
Engineer		04-OCT-20	10

Peter
Actor 04-OCT-20 13

Marco
Doctor 04-OCT-20 14

Brayden
Teacher 04-OCT-20 12

Antonio
Business 04-OCT-20 11

1. Count() Function

```
SELECT COUNT(name) FROM employee;
```

2. Sum() Function

```
SELECT SUM(working_hours) AS "Total working hours" FROM employee;
```

3. AVG() Function

```
SELECT AVG(working_hours) AS "Average working hours" FROM employee;
```

4. MIN() Function

```
SELECT MIN(working_hours) AS Minimum_working_hours FROM employee;
```

5. MAX() Function

```
SELECT MAX(working_hours) AS Maximum_working_hours FROM employee;
```

6. FIRST() Function

```
SELECT working_date FROM employee LIMIT 1;
```

SQL Date and Time Functions

1. NOW()

Returns the current date and time.

Query:

```
SELECT NOW();
```

2. CURDATE()

Returns the current date.

Query:

```
SELECT CURDATE();
```

3. CURTIME()

Returns the current time.

Query:

```
SELECT CURTIME();
```

4. DATE()

Query:

```
SELECT Name, DATE(BirthTime) AS BirthDate FROM Test;
```

5. EXTRACT()

Query1:

```
SELECT Name, Extract(DAY FROM BirthTime) AS BirthDay FROM Test; Query2:
```

```
SELECT Name, Extract(YEAR FROM BirthTime) AS BirthYear FROM Test;
```

Query3:

```
SELECT Name, Extract(SECOND FROM BirthTime) AS BirthSecond FROM Test;
```

6. DATE_ADD()

Query 1:

```
SELECT Name, DATE_ADD(BirthTime, INTERVAL 1 YEAR) AS BirthTimeModified  
FROM Test;
```

Query 2:

```
SELECT Name, DATE_ADD(BirthTime, INTERVAL 30 DAY) AS BirthDayModified  
FROM Test;
```

7. DATEDIFF()

```
SELECT DATEDIFF(month,'2017-01-13','2017-01-03') AS DateDiff
```

SQL OPERATORS

SQL OPERATORS

1. AND Operator

Query

```
SELECT * FROM employee WHERE emp_city = 'Allahabad' AND emp_country =  
'India';
```

2. IN Operator

Query

```
SELECT * FROM employee WHERE emp_city IN ('Allahabad', 'Patna');
```

3. NOT Operator

Query

```
SELECT * FROM employee WHERE emp_city NOT LIKE 'A%';
```

4. OR Operator

Query

```
SELECT * FROM employee WHERE emp_city = 'Varanasi' OR emp_country = 'India';
```

5. LIKE Operator

Query

```
SELECT * FROM employee WHERE emp_city LIKE 'P%';
```

6. BETWEEN Operator

Query

```
SELECT * FROM employee WHERE emp_id BETWEEN 101 AND 104;
```

SQL Comparison Operators

1. = (Equal to) Operator

```
SELECT * FROM customers WHERE age = 20;
```

2. != (Not equal to) Operator SELECT * FROM customers

```
WHERE age != 20;
```

3. > (Greater than) Operator

```
SELECT * FROM customers WHERE age > 20;
```

4. !> (Not greater than) Operator

```
SELECT * FROM customers WHERE age !> 20;
```

5. < (Less than) Operator

```
SELECT * FROM customers WHERE age < 20;
```

6. !< (Not less than) Operator

```
SELECT * FROM customers WHERE age !< 20;
```

7. >= (Greater than or equal to) Operator

```
SELECT * FROM customers WHERE age >= 20;
```

8. <= (Less than or equal to) Operator

```
SELECT * FROM customers WHERE age <= 20;
```

9. \neq (Not equal to) Operator

```
SELECT * FROM customers WHERE age  $\neq$  20;
```

RESULT:- the queries are executed successfully