Date: 18/08/2025

Task 3.1: DML commands using clauses operators and functions in queries.

Aim:

To implement of DML commands using clauses, operators and functions in queries.

Data manupulation language (DML):

The data manipulation language (DML) is using to retrive, ingest and modify database information. These commands will be used by all database users during the routine operation of the database. Let's take a brief look at the basic look at the basic DML commands:

1. INSERT
2. UPDATE
3. DELETE

INSERT INTO

This is used to add records into a relation.

Syntax:

INSERT INTO (table_name) (field1, field2, ... fieldN)
VALUES (data1, data2 .... dataN);

Example:

SQL

INSERT INTO Patients VALUES(111, 'Arun', 'Cardiology', 'male');

Table after INSERT:

PatientID	Patient Name	Department	Gender
111	Arun	cardiology	male

UPDATE - SET - WHERE:

This is used to update the content of a record in a relation.

Syntax:

SQL:

UPDATE table_name SET field1=data WHERE condition;

Example:

SQL:

UPDATE patients SET PatientName='kumar' WHERE PatientID = 111;

Table after UPDATE:

Patient Id	Patient name	Department	Gender
111	kumar	cardiology	male

DELETE FROM:

This is used to delete all records of a relation but it retains the structure

Syntax:

SQL:

DELETE FROM table_name;

Example:

SQL:

DELETE FROM Appointments;

Appointments table after DELETE;

AppointmentID	PatientID	DoctorID	Appointment name	Optio

DELETE - FROM - WHERE:

This is used to delete specific records from a relation.

Syntax:

SQL:

DELETE FROM table_name WHERE condition;

Example:

SQL:

DELETE FROM Doctors WHERE DoctorID = 202;

Doctor Table after DELETE:

DoctorID	DoctorName	Department	Fees
201	Dr. Sharma	cardiology	1000
203	Dr. Ahmed	Neurology	900
204	Dr. Rajesh	orthopedic	500
205	Dr. Neha	Dermatology	800

TRUNCATE:

This removes all data permanently but keeps the table structure.

Syntax:

SQL:

TRUNCATE TABLE <table_name>;

Example:

SQL:

TRUNCATE TABLE Patients;

Patients Table after TRUNCATE:

PatientID	PatientName	Department	Gender

Simple Queries and output:

1. Retrieve patient names ending with letter 'n' and patient no between 111 and 115

Query:

SQL:

SELECT PatientName, Department, Gender
FROM Patients
WHERE PatientName LIKE '%n' AND
PatientID BETWEEN 111 AND 115.

Patient name	Department	Gender
Arun	Cardiology	male
Karan	orthopedics	male
Rohan	Dermatology	male

2. List doctors where consultation fees between 700 and 800

Query:

SQL

SELECT * FROM DOCTORS WHERE FEES BETWEEN 700 AND 800;

DOCTORID	DoctorName	departmentName	Fees
202	Dr.Priya	Pediatrics	700
205	Dr.Neha	Dermatology	800

3. Find the record with minimum appointment duration

Query:

SQL:

SELECT MIN(Duration) FROM Appointments;

MIN(Duration)

20

4. Find appointments with date >= '2023-02-07

Query:

SQL:

SELECT * FROM Appointments WHERE AppointmentDate = '2023-02-07'

AppointmentID	PatientID	DoctorID	AppointmentDate	Duration
302	112	203	2023-02-07	45
303	113	204	2023-02-09	20
304	114	202	2023-02-10	60
305	115	205	2023-02-12	25

5. List distinct PatientIDs

Query:

SQL:

SELECT DISTINCT PatientID FROM Patients;

PatientID
111
112
113
114
115

6. Combine PatientIDs from Patients and Appointments (UNION)

Query:

SQL:

SELECT PatientID FROM Patients

UNION

SELECT PatientID FROM Appointments;

output:

PatientID
111
102
113
114
118

7. Group patients based on gender and department

Query:

SQL:

```
SELECT Department, Gender, COUNT(*) AS
Total Patients
FROM Patients
GROUP BY Department, Gender;
```

Department	Gender	Total Patients
Cardiology	Male	1
Neurology	Female	1
Orthopedics	Male	1
Pediatrics	Female	1
Dermatology	Male	1

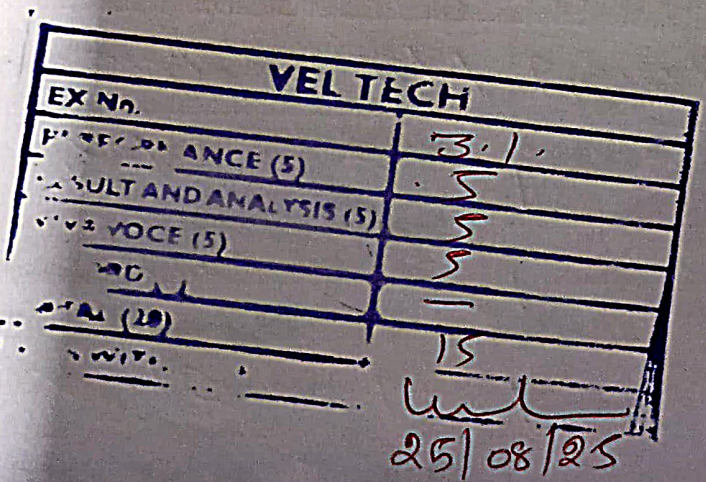
8. Find doctors and their department details using GROUP BY and ORDER BY

Query:

SQL:

```
SELECT doctorname, Department, COUNT(*) AS
Count
FROM Doctors
GROUP BY doctorname, Department
ORDER BY Doctor name;
```

Doctor name	Department	Count
Dr. Ahmed	Neurology	1
Dr. Neha	Dermatology	1
Dr. Priya	Pediatrics	1
Dr. Rajesh	Orthopedics	1
Dr. Sharma	Cardiology	1



RESULT:

The implementation of DML commands using clauses, generators and functions in queries executed successfully.

TASK 3.2: AGGREGATE FUNCTIONS
(Multi Row operations)

Aim:
To study and implement aggregate function (COUNT(), SUM(), AVG(), MIN(), MAX(), on a simple patient database

Procedure:
1. Create a table named students
2. Insert sample records
3. Write queries using aggregate functions
4. Observe and record the output.

COMMANDS WITH EXPLANATION:

Example Table: patients

PatientID	patientName	Department	BillAmount
101	Ayush	cardiology	2000
102	Sneha	neurology	3500
103	Karan	Orthopedics	1500
104	Meena	pediatrics	4000
105	Rohan	dermatology	

1) Count the total number of patients:
SQL:
SELECT COUNT(*) AS Total_patients
From patients;
Output:
Total_patients
5

2. Find the highest bill amount

SQL

```
SELECT MAX(Bill_Amount) AS Highest_Bill
FROM patients;
```

output:

Highest_Bill
4000

3. Find the average bill amount of patients

SQL:

```
SELECT AVG(Bill_Amount) AS Average_bill
FROM patients;
```

output:

Average_Bill
2700

4. Find the minimum bill amount among patients in Neurology department

SQL:

```
SELECT MIN(Bill_Amount) AS Min_Neuro_Bill
```

output:

Min_Neuro_Bill
3500

5. Find the total bill amount by each department

SQL:

```
SELECT department, SUM(Bill_Amount) AS
Total_Bill
FROM patients
GROUP BY department;
```

output:

Department	Total_Bill
cardiology	2000
Neurology	3500
orthopedics	1500
Pediatrics	4000
Dermatology	

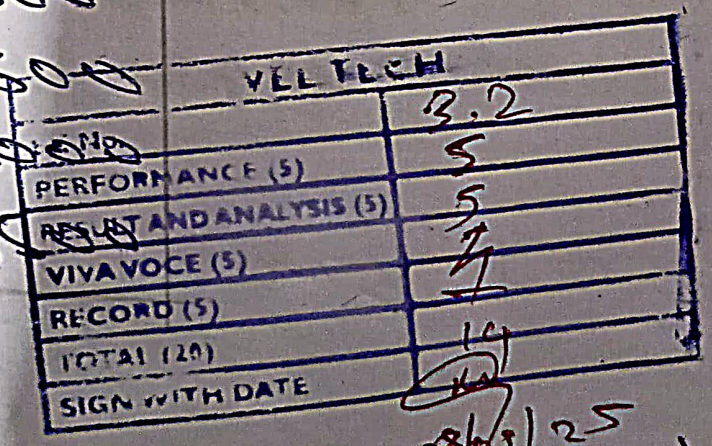
6, Find the average bill per department_ ordered by average descending

SQL:

```
SELECT department, AVG(Bill Amount)AS AVG_BILL
FROM Patients
GROUP BY department
ORDER BY AVG_Bill DESC;
```

output:

Department	AVG_Bill
Pediatrics	4000
Neurology	3500
Dermatology	2500
cardiology	2000
orthopedia	1500



RESULT:

The implementation of Aggregate functions are executed successfully.