Green University of Bangladesh
Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering
Semester: (Fall, Year:2024), B.Sc. in CSE (Day)

Lab Report NO: 04
Course Title: Database System Lab
Course Code: CSE 210 Section:231(D1)

Lab Experiment Name: Querying and Filtering data in MYSQL Table (Extended) & Implementation of MYSQL Aggregate Function

Student Details

	Name	ID
1.	Promod Chandra Das	231002005

Lab Date : 18.11.2024
Submission Date : 24.11.2024
Course Teacher's Name : Fatema-Tuj- Johora

Lab Report Status

Marks: …… **Signature:.....**
Comments:..... **Date:.....**

❖ TITLE OF THE LAB REPORT EXPERIMENT

Querying and Filtering data in MYSQL Table (Extended) & Implementation of MYSQL Aggregate Function

❖ OBJECTIVES

1. Understanding Querying and Filtering Data

- Learn how to fetch data from a MySQL table using SELECT queries.
- Understand how to filter data with conditions using the WHERE clause.
- Explore the use of logical operators (AND, OR, NOT) for complex filtering.
- Use comparison operators (e.g., =, <, >, <=, >=, !=) to refine queries.
- Incorporate pattern matching with LIKE and wildcards for advanced searches.
- Leverage sorting with ORDER BY and limiting results with LIMIT.

2. Applying Aggregate Functions

- **Common Aggregate Functions:**
  - COUNT(): Count the number of rows.
  - SUM(): Calculate the total sum of a numeric column.
  - AVG(): Find the average value of a numeric column.
  - MIN(): Retrieve the minimum value.
  - MAX(): Retrieve the maximum value.

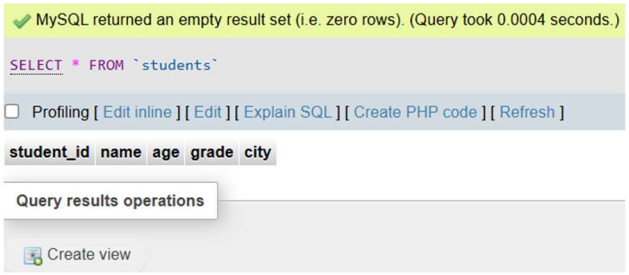
- Apply aggregate functions with basic queries.
- Understand the use of GROUP BY for grouping data by specific columns.
- Use HAVING to filter grouped data after applying aggregate functions.

❖ IMPLEMENTATION

- **Step 1: Input Multiple Data**

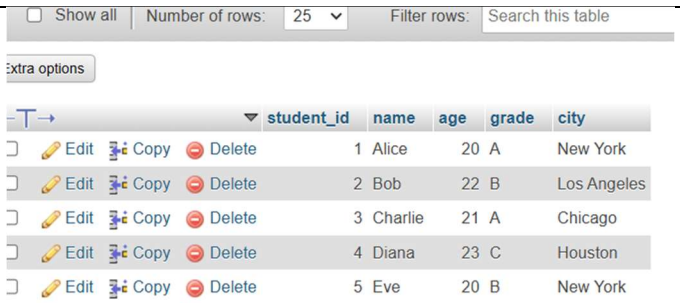
Example Table: students

```
 CREATE TABLE students (
student_id INT PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(100),
age INT,
grade VARCHAR(10),
city VARCHAR(100)
);
```



Insert Multiple Rows:

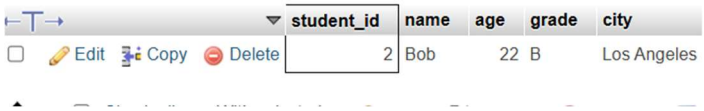
```
INSERT INTO students (name, age, grade, city)
VALUES
('Alice', 20, 'A', 'New York'),
('Bob', 22, 'B', 'Los Angeles'),
('Charlie', 21, 'A', 'Chicago'),
('Diana', 23, 'C', 'Houston'),
 ('Eve', 20, 'B', 'New York');
```



- **Step 2: Queries**

Query with Primary Key:

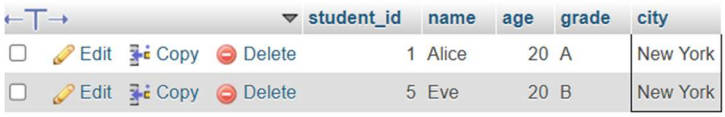
```
SELECT * FROM students WHERE student_id = 2;
```



	student_id	name	age	grade	city
Edit Copy Delete	2	Bob	22	B	Los Angeles

Query with Condition:

```
SELECT * FROM students WHERE city = 'New York';
```



	student_id	name	age	grade	city
Edit Copy Delete	1	Alice	20	A	New York
Edit Copy Delete	5	Eve	20	B	New York

Retrieve students from New York:

```
SELECT * FROM students WHERE city = 'New York';
```

	student_id	name	age	grade	city
Edit Copy Delete	2	Bob	22	B	Los Angeles
Edit Copy Delete	4	Diana	23	C	Houston

Query with Comparison Operation:

Find students older than 21:

```
SELECT * FROM students WHERE age > 21;
```

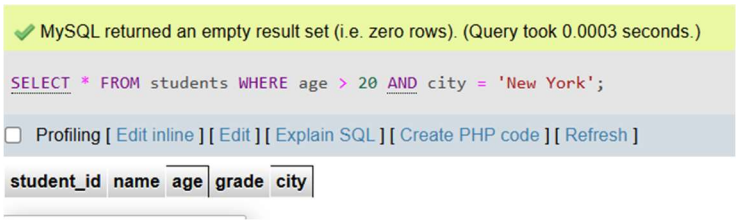
	student_id	name	age	grade	city
Edit Copy Delete	2	Bob	22	B	Los Angeles
Edit Copy Delete	4	Diana	23	C	Houston

- **Step 3: Queries with Logical and Comparison Operators**

Using AND, OR, NOT:

-- AND: Students older than 20 and in New York

```
SELECT * FROM students WHERE age > 20 AND city = 'New York';
```



-- OR: Students in New York or Chicago

```
SELECT * FROM students WHERE city = 'New York' OR city = 'Chicago';
```

	student_id	name	age	grade	city
Edit Copy Delete	1	Alice	20	A	New York
Edit Copy Delete	3	Charlie	21	A	Chicago
Edit Copy Delete	5	Eve	20	B	New York

-- NOT: Students not from New York

```
SELECT * FROM students WHERE NOT city = 'New York';
```

	student_id	name	age	grade	city
Edit Copy Delete	2	Bob	22	B	Los Angeles
Edit Copy Delete	3	Charlie	21	A	Chicago
Edit Copy Delete	4	Diana	23	C	Houston

Using ORDER BY:

-- Ascending order by age

```
SELECT * FROM students ORDER BY age ASC;
```

	student_id	name	age ▲ 1	grade	city
Edit Copy Delete	1	Alice	20	A	New York
Edit Copy Delete	5	Eve	20	B	New York
Edit Copy Delete	3	Charlie	21	A	Chicago
Edit Copy Delete	2	Bob	22	B	Los Angeles
Edit Copy Delete	4	Diana	23	C	Houston

-- Descending order by name

```
SELECT * FROM students ORDER BY name DESC;
```

	student_id	name	age	grade	city
Edit Copy Delete	5	Eve	20	B	New York
Edit Copy Delete	4	Diana	23	C	Houston
Edit Copy Delete	3	Charlie	21	A	Chicago
Edit Copy Delete	2	Bob	22	B	Los Angeles
Edit Copy Delete	1	Alice	20	A	New York

Using BETWEEN and NOT BETWEEN:

-- Students aged between 20 and 22

```
SELECT * FROM students WHERE age BETWEEN 20 AND 22;
```

	student_id	name	age	grade	city
Edit Copy Delete	1	Alice	20	A	New York
Edit Copy Delete	2	Bob	22	B	Los Angeles
Edit Copy Delete	3	Charlie	21	A	Chicago
Edit Copy Delete	5	Eve	20	B	New York

-- Students not aged between 20 and 22

```
SELECT * FROM students WHERE age NOT BETWEEN 20 AND 22;
```

	student_id	name	age	grade	city
Edit Copy Delete	4	Diana	23	C	Houston

Using IN and NOT IN:

-- Students from specific cities

```
SELECT * FROM students WHERE city IN ('New York', 'Chicago');
```

	student_id	name	age	grade	city
Edit Copy Delete	1	Alice	20	A	New York
Edit Copy Delete	3	Charlie	21	A	Chicago
Edit Copy Delete	5	Eve	20	B	New York

-- Students not from specific cities

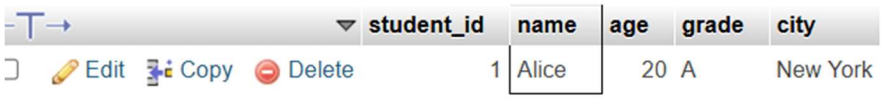
```
SELECT * FROM students WHERE city NOT IN ('New York', 'Chicago');
```

	student_id	name	age	grade	city
Edit Copy Delete	2	Bob	22	B	Los Angeles
Edit Copy Delete	4	Diana	23	C	Houston

Using LIKE (Pattern Matching):

-- Students whose names start with 'A'

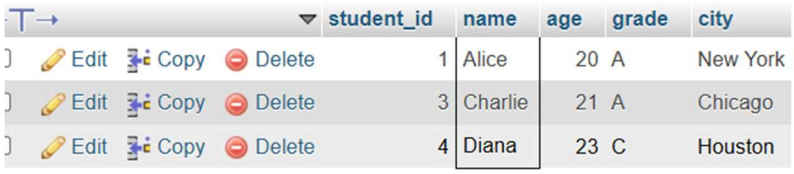
```
SELECT * FROM students WHERE name LIKE 'A%';
```



	student_id	name	age	grade	city
Edit Copy Delete	1	Alice	20	A	New York

-- Students whose names contain 'i'

```
SELECT * FROM students WHERE name LIKE '%i%';
```



	student_id	name	age	grade	city
Edit Copy Delete	1	Alice	20	A	New York
Edit Copy Delete	3	Charlie	21	A	Chicago
Edit Copy Delete	4	Diana	23	C	Houston

❖ Implementation of MYSQL Aggregate Function

1. Create the product_order_info Table

```
CREATE TABLE product_order_info (
   product_no INT(11) NOT NULL AUTO_INCREMENT,
   product_name VARCHAR(255) NOT NULL,
   product_type ENUM('electronics', 'stationary', 'food', 'beverage') DEFAULT NULL,
   product_price FLOAT(10,2) NOT NULL,
   product_quantity SMALLINT NOT NULL,
   order_date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
   PRIMARY KEY (product_no)
);
```

---

2. Insert Multiple Rows

```
INSERT INTO product_order_info (product_no, product_name, product_type, product_price,
    product_quantity)
VALUES
(101, 'Laptop', 'electronics', 67000, 1),
(NULL, 'Mobile', 'electronics', 23500, 1),
(NULL, 'Watch', 'electronics', 8650, 2),
(NULL, 'Butter', 'stationary', 50, 5),
(NULL, 'Coca-cola', 'beverage', 35, 2),
(NULL, 'Seven-Up', 'beverage', 55, 1);
```

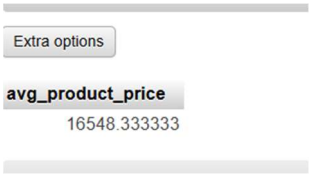
	product_no	product_name	product_type	product_price	product_quantity	order_date
Edit Copy Delete	101	Laptop	electronics	67000.00	1	2024-11-20 01:05:13
Edit Copy Delete	102	Mobile	electronics	23500.00	1	2024-11-20 01:05:13
Edit Copy Delete	103	Watch	electronics	8650.00	2	2024-11-20 01:05:13
Edit Copy Delete	104	Butter	stationary	50.00	5	2024-11-20 01:05:13
Edit Copy Delete	105	Coca-cola	beverage	35.00	2	2024-11-20 01:05:13
Edit Copy Delete	106	Seven-Up	beverage	55.00	1	2024-11-20 01:05:13

---

3. Queries to Retrieve Data

3.1 Average Product Price

```
SELECT AVG(product_price) AS avg_product_price FROM product_order_info;
```



3.2 Total Rows in the Table

```
SELECT COUNT(product_no) AS total_order FROM product_order_info;
```

total_order
6

3.3 Count Rows Grouped by Product Type

```
SELECT product_type, COUNT(*) AS total_products
FROM product_order_info
GROUP BY product_type;
```

product_type	total_products
electronics	3
stationary	1
beverage	2

3.4 Count Rows for Specific Product Type (electronics)

```
SELECT COUNT(*) AS total_electronics
FROM product_order_info
WHERE product_type = 'electronics';
```

total_electronics
3

3.5 Total Sales per Product

```
SELECT
```

```
    product_no,
    product_name,
    product_price,
    product_quantity,
    (product_price * product_quantity) AS total_per_product
FROM product_order_info;
```

	product_no	product_name	product_price	product_quantity	total_per_product
Edit Copy Delete	101	Laptop	67000.00	1	67000.00
Edit Copy Delete	102	Mobile	23500.00	1	23500.00
Edit Copy Delete	103	Watch	8650.00	2	17300.00
Edit Copy Delete	104	Butter	50.00	5	250.00
Edit Copy Delete	105	Coca-cola	35.00	2	70.00
Edit Copy Delete	106	Seven-Up	55.00	1	55.00

3.6 Maximum Product Price

```
SELECT MAX(product_price) AS max_price FROM product_order_info;
```

max_price
67000.00

3.7 Minimum Product Price

```
SELECT MIN(product_price) AS min_price FROM product_order_info;
```

---

nin_price
35.00

4. String and Numeric Functions

4.1 MySQL LENGTH() Function

```
SELECT product_no, product_name, LENGTH(product_name) AS name_length FROM
    product_order_info;
```

	product_no	product_name	name_length
Edit Copy Delete	101	Laptop	6
Edit Copy Delete	102	Mobile	6
Edit Copy Delete	103	Watch	5
Edit Copy Delete	104	Butter	6
Edit Copy Delete	105	Coca-cola	9
Edit Copy Delete	106	Seven-Up	8

4.2 UCASE() Function

```
SELECT product_no, product_name, UCASE(product_name) AS uppercase_name FROM
    product_order_info;
```

	product_no	product_name	uppercase_name
Edit Copy Delete	101	Laptop	LAPTOP
Edit Copy Delete	102	Mobile	MOBILE
Edit Copy Delete	103	Watch	WATCH
Edit Copy Delete	104	Butter	BUTTER
Edit Copy Delete	105	Coca-cola	COCA-COLA
Edit Copy Delete	106	Seven-Up	SEVEN-UP

4.3 LCASE() Function

```
SELECT product_no, product_name, LCASE(product_name) AS lowercase_name FROM
    product_order_info;
```

	product_no	product_name	lowercase_name
Edit Copy Delete	101	Laptop	laptop
Edit Copy Delete	102	Mobile	mobile
Edit Copy Delete	103	Watch	watch
Edit Copy Delete	104	Butter	butter
Edit Copy Delete	105	Coca-cola	coca-cola
Edit Copy Delete	106	Seven-Up	seven-up

4.4 FLOOR() Function

```
SELECT product_no, product_price, FLOOR(product_price) AS floor_price FROM product_order_info;
```

	product_no	product_price	floor_price
Edit Copy Delete	101	67000.00	67000
Edit Copy Delete	102	23500.00	23500
Edit Copy Delete	103	8650.00	8650
Edit Copy Delete	104	50.00	50
Edit Copy Delete	105	35.00	35
Edit Copy Delete	106	55.00	55

4.5 CEIL() Function

```
SELECT product_no, product_price, CEIL(product_price) AS ceil_price FROM product_order_info;
```

	product_no	product_price	ceil_price
Edit Copy Delete	101	67000.00	67000
Edit Copy Delete	102	23500.00	23500
Edit Copy Delete	103	8650.00	8650
Edit Copy Delete	104	50.00	50
Edit Copy Delete	105	35.00	35
Edit Copy Delete	106	55.00	55

4.6 ROUND() Function

```
SELECT product_no, product_price, ROUND(product_price, 1) AS rounded_price FROM
    product_order_info;
```

	product_no	product_price	rounded_price
Edit Copy Delete	101	67000.00	67000.0
Edit Copy Delete	102	23500.00	23500.0
Edit Copy Delete	103	8650.00	8650.0
Edit Copy Delete	104	50.00	50.0
Edit Copy Delete	105	35.00	35.0
Edit Copy Delete	106	55.00	55.0

4.7 MID() Function

```
SELECT product_no, product_name, MID(product_name, 1, 3) AS partial_name FROM
    product_order_info;
```

	product_no	product_name	partial_name
Edit Copy Delete	101	Laptop	Lap
Edit Copy Delete	102	Mobile	Mob
Edit Copy Delete	103	Watch	Wat
Edit Copy Delete	104	Butter	But
Edit Copy Delete	105	Coca-cola	Coc
Edit Copy Delete	106	Seven-Up	Sev

4.8 CONCAT() Function

```
SELECT product_no, CONCAT(product_name, ' (', product_type, ')') AS product_details FROM
    product_order_info;
```

	product_no	product_details
Edit Copy Delete	101	Laptop (electronics)
Edit Copy Delete	102	Mobile (electronics)
Edit Copy Delete	103	Watch (electronics)
Edit Copy Delete	104	Butter (stationary)
Edit Copy Delete	105	Coca-cola (beverage)
Edit Copy Delete	106	Seven-Up (beverage)

. Sorting and Grouping

5.1 Sorting Data by Price

```
SELECT * FROM product_order_info ORDER BY product_price DESC;
```

	product_no	product_name	product_type	product_price 1	product_quantity	order_date
Edit Copy Delete	101	Laptop	electronics	67000.00	1	2024-11-20 01:05:13
Edit Copy Delete	102	Mobile	electronics	23500.00	1	2024-11-20 01:05:13
Edit Copy Delete	103	Watch	electronics	8650.00	2	2024-11-20 01:05:13
Edit Copy Delete	106	Seven-Up	beverage	55.00	1	2024-11-20 01:05:13
Edit Copy Delete	104	Butter	stationary	50.00	5	2024-11-20 01:05:13
Edit Copy Delete	105	Coca-cola	beverage	35.00	2	2024-11-20 01:05:13

5.2 Grouping by Product Type

```
SELECT product_type, COUNT(*) AS total_products FROM product_order_info GROUP BY
    product_type;
```

product_type	total_products
electronics	3
stationary	1
beverage	2

❖ **DISCUSSION:**

Querying in MySQL involves retrieving specific data using SELECT, while filtering refines results with WHERE, logical operators (AND, OR), and conditions (BETWEEN, LIKE). Aggregate functions like SUM, AVG, MAX, MIN, and COUNT perform calculations on data groups, often combined with GROUP BY for grouped analysis.