**Case Study: Usage in E-commerce**

#Scenario

An e-commerce company wants to optimize its operations using MySQL. The company focuses on:

1. Tracking customer behavior.

2. Managing inventory.

3. Analyzing sales performance.

#Step-by-Step Implementation

Step 1: Database Design

#Objective

Create a schema that represents the core operations of an e-commerce business.

#Schema

-Customers Table:

Stores customer details like names, email, and addresses.

CREATE TABLE customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100),

email VARCHAR(100) UNIQUE,

address TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-Products Table:

Contains product details, including name, price, and stock quantity.

CREATE TABLE products (

product\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100),

price DECIMAL(10, 2),

stock\_quantity INT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-Orders Table:

Tracks orders placed by customers.

sh

-Order\_Items Table:

Tracks the products and quantities within each order.

CREATE TABLE order\_items (

order\_item\_id INT AUTO\_INCREMENT PRIMARY KEY,

order\_id INT,

product\_id INT,

quantity INT,

price DECIMAL(10, 2),

FOREIGN KEY (order\_id) REFERENCES orders(order\_id),

FOREIGN KEY (product\_id) REFERENCES products(product\_id)

);

Step 2: Data Population

#Objective

Populate tables with sample data for analysis.

# Queries

- Insert sample customers:

INSERT INTO customers (name, email, address)

VALUES

('Alice Smith', 'alice@example.com', '123 Maple St'),

('Bob Jones', 'bob@example.com', '456 Oak St');

- Insert sample products:

INSERT INTO products (name, price, stock\_quantity)

VALUES

('Laptop', 1000.00, 50),

('Smartphone', 500.00, 100),

('Tablet', 300.00, 80);

- Insert sample orders and order items:

INSERT INTO orders (customer\_id, total\_amount)

VALUES (1, 1500.00), (2, 800.00);

INSERT INTO order\_items (order\_id, product\_id, quantity, price)

VALUES

(1, 1, 1, 1000.00),

(1, 2, 1, 500.00),

(2, 3, 2, 800.00);

Step 3: Querying Data

#Objective

Generate actionable insights.

#Examples

1.Top Customers by Spending:

SELECT c.name, SUM(o.total\_amount) AS total\_spent

FROM customers c

JOIN orders o ON c.customer\_id = o.customer\_id

GROUP BY c.name

ORDER BY total\_spent DESC;

+-------------+-------------+

| name | total\_spent |

+-------------+-------------+

| Alice Amith | 1500.00 |

| Bob Jones | 800.00 |

+-------------+-------------+

2.Low Stock Alert:

Identify products with stock below 10 units.

SELECT name, stock\_quantity

FROM products

WHERE stock\_quantity < 10;

Empty set (0.00 sec) -- as stock\_quantity not have below 10

3.Sales Summary by Product:

Calculate total sales for each product.

SELECT p.name, SUM(oi.quantity) AS total\_sold, SUM(oi.price \* oi.quantity) AS total\_revenue

FROM products p

JOIN order\_items oi ON p.product\_id = oi.product\_id

GROUP BY p.name

ORDER BY total\_revenue DESC;

+------------+------------+---------------+

| name | total\_sold | total\_revenue |

+------------+------------+---------------+

| Tablet | 2 | 1600.00 |

| laptop | 1 | 1000.00 |

| Smartphone | 1 | 500.00 |

+------------+------------+---------------+

4.Customer Purchase History:

Retrieve all orders and items purchased by a specific customer.

SELECT o.order\_id, p.name AS product\_name, oi.quantity, oi.price

FROM orders o

JOIN order\_items oi ON o.order\_id = oi.order\_id

JOIN products p ON oi.product\_id = p.product\_id

WHERE o.customer\_id = 1;

+----------+--------------+----------+---------+

| order\_id | product\_name | quantity | price |

+----------+--------------+----------+---------+

| 1 | laptop | 1 | 1000.00 |

| 1 | Smartphone | 1 | 500.00 |

+----------+--------------+----------+---------+

Step 4: Performance Optimization

#Objective

Optimize queries and ensure scalability.

#Actions

1.Indexes:

Create indexes on frequently queried columns:

CREATE INDEX idx\_customer\_email ON customers(email);

CREATE INDEX idx\_order\_date ON orders(order\_date);

2.Partitioning:

Partition large tables (e.g., `orders`) by date for faster querying:

PARTITION BY RANGE (YEAR(order\_date)) (

PARTITION p2023 VALUES LESS THAN (2024),

PARTITION p2024 VALUES LESS THAN (2025)

);

mysql> partition by range(year(order\_date))(partition p2023 values less than(2024),partition p2024 values less then(2024));

ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'partition by range(year(order\_date))(partition p2023 values less than(2024),part' at line 1

mysql> PARTITION BY RANGE (YEAR(ORDER\_DATE))(

-> PARTITION p2023 VALUES LESS THAN (2024),

-> PARTITION p2024 VALUES LESS THEN (2025)

-> );

ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'PARTITION BY RANGE (YEAR(ORDER\_DATE))(

PARTITION p2023 VALUES LESS THAN (2024),

' at line 1

Step 5: Advanced Analytics

#Objective

Leverage complex queries for strategic decisions.

#Examples

1.Average Order Value (AOV):

SELECT AVG(total\_amount) AS avg\_order\_value

FROM orders;

+-----------------+

| AVG\_ORDER\_VALUE |

+-----------------+

| 1150.000000 |

+-----------------+

1 row in set (0.01 sec)

2.Most Popular Product Categories:H

If a `category` column exists in `products`:

SELECT category, SUM(oi.quantity) AS total\_sold

FROM products p

JOIN order\_items oi ON p.product\_id = oi.product\_id

GROUP BY category

ORDER BY total\_sold DESC;

(Category else name )

------------+------------+

| name | total\_sold |

+------------+------------+

| Tablet | 2 |

| laptop | 1 |

| Smartphone | 1 |

+------------+------------+

3 rows in set (0.01 sec)

3.Monthly Sales Trends:

SELECT MONTH(order\_date) AS month, SUM(total\_amount) AS monthly\_sales

FROM orders

GROUP BY month

ORDER BY month;

-------+---------------+

| month | monthly\_sales |

+-------+---------------+

| 2 | 2300.00 |

Step 6: Reporting and Visualization

#Objective

Export data for visualization in tools like Tableau or Power BI.

#Actions

- Use `SELECT INTO OUTFILE` to export query results:

SELECT \* FROM orders

INTO OUTFILE '/var/lib/my-files/orders.csv'

FIELDS TERMINATED BY ',' ENCLOSED BY '"'

LINES TERMINATED BY '\n';

Outcomes

- Improved inventory management by identifying low-stock items.

- Enhanced customer satisfaction through targeted marketing.

- Increased profitability by identifying best-selling products and trends.

Here's how to implement the detailed steps for using MySQL queries and integrating the results into visualization tools like Tableau or Power BI:

1. Implementing Queries in MySQL

Step 1: Setting Up MySQL

1.Install MySQL Server:

- Download and install MySQL Server and MySQL Workbench from the [official MySQL website](https://dev.mysql.com/).

- Configure the root user with a password.

2.Create and Populate the Database:

- Use MySQL Workbench or command-line tools to create the database schema and populate it with the SQL scripts provided in the previous steps.

3.Verify the Data:

- Run `SELECT` queries to ensure data is correctly inserted:

SELECT \* FROM customers;

SELECT \* FROM products;

SELECT \* FROM orders;

SELECT \* FROM order\_items;

mysql> SELECT \* FROM customers;

+-------------+-------------+-------------------+--------------+---------------------+

| customer\_id | name | email | address | created\_at |

+-------------+-------------+-------------------+--------------+---------------------+

| 1 | Alice Amith | Alice@example.com | 123 Maple st | 2025-02-03 14:50:40 |

| 2 | Bob Jones | Bob@example.com | 456 oak St | 2025-02-03 14:50:40 |

+-------------+-------------+-------------------+--------------+---------------------+

2 rows in set (0.00 sec)

mysql> SELECT \* FROM products;

+------------+------------+---------+----------------+---------------------+

| product\_id | name | price | stock\_quantity | create\_at |

+------------+------------+---------+----------------+---------------------+

| 1 | laptop | 1000.00 | 50 | 2025-02-03 14:56:51 |

| 2 | Smartphone | 500.00 | 100 | 2025-02-03 14:56:51 |

| 3 | Tablet | 300.00 | 80 | 2025-02-03 14:56:51 |

+------------+------------+---------+----------------+---------------------+

3 rows in set (0.00 sec)

mysql> SELECT \* FROM orders;

+----------+-------------+---------------------+--------------+

| order\_id | customer\_id | order\_date | total\_amount |

+----------+-------------+---------------------+--------------+

| 1 | 1 | 2025-02-03 14:58:03 | 1500.00 |

| 2 | 2 | 2025-02-03 14:58:03 | 800.00 |

+----------+-------------+---------------------+--------------+

2 rows in set (0.00 sec)

mysql> SELECT \* FROM order\_items;

+---------------+----------+------------+----------+---------+

| order\_item\_id | order\_id | product\_id | quantity | price |

+---------------+----------+------------+----------+---------+

| 1 | 1 | 1 | 1 | 1000.00 |

| 2 | 1 | 2 | 1 | 500.00 |

| 3 | 2 | 3 | 2 | 800.00 |

+---------------+----------+------------+----------+---------+

3 rows in set (0.00 sec)

2. Running Advanced Queries

Once the data is verified, run the advanced SQL queries for analytics, such as:

-Monthly Sales Trends:

SELECT MONTH(order\_date) AS month, SUM(total\_amount) AS monthly\_sales

FROM orders

GROUP BY month

ORDER BY month;

+-------+---------------+

| month | monthly\_sales |

+-------+---------------+

| 2 | 2300.00 |

+-------+---------------+

-Export Data for Visualization:

Use the `SELECT INTO OUTFILE` statement to export data as a CSV file:

SELECT o.order\_id, c.name AS customer\_name, SUM(oi.quantity \* oi.price) AS order\_total

FROM orders o

JOIN customers c ON o.customer\_id = c.customer\_id

JOIN order\_items oi ON o.order\_id = oi.order\_id

GROUP BY o.order\_id

INTO OUTFILE '/var/lib/mysql-files/orders\_summary.csv'

FIELDS TERMINATED BY ',' ENCLOSED BY '"'

LINES TERMINATED BY '\n';

3. Preparing Data for Visualization

Step 1: Export Data from MySQL

- Save query results as CSV files using the `SELECT INTO OUTFILE` method.

- Ensure the MySQL user has the necessary permissions to write to the file system (`OUTFILE` privileges).

Step 2: Clean and Organize Data

- Use tools like Excel or Python pandas to preprocess the CSV file if necessary.

- Ensure column names are descriptive and consistent.

4. Integrating with Tableau

1.Install Tableau:

- Download and install Tableau Desktop from [Tableau's official website](https://www.tableau.com/).

2.Connect Tableau to MySQL:

- Open Tableau Desktop.

- SelectConnect > To a Server > MySQL.

- Provide connection details (host, port, database name, username, and password).

3.Import Data:

- Choose the tables or run custom SQL queries within Tableau to fetch data.

4.Create Visualizations:

- Use Tableau's drag-and-drop interface to create dashboards.

- Example: Use a bar chart to displaymonthly sales trends.

5. Integrating with Power BI

1.Install Power BI:

- Download and install Power BI Desktop from [Power BI's official website](https://powerbi.microsoft.com/).

2.Connect Power BI to MySQL:

- Open Power BI.

- Go toHome > Get Data > More > MySQL Database.

- Enter the connection details.

3.Load Data:

- Import tables or use direct SQL queries.

- Example: Load the `orders` and `order\_items` tables.

4.Design Reports:

- Use visuals like pie charts and line graphs for customer segmentation and sales trends.

6. Automating Updates

For regularly updating reports:

-Schedule Queries: Use MySQL event scheduler to periodically run queries and export updated data.

-Refresh in Visualization Tools:

- Tableau: Use Tableau Server to schedule refreshes.

- Power BI: Publish reports to Power BI Service and set up automatic data refresh.

7. Example Use Case Workflow

Objective: Visualize monthly sales trends.

1. Run the SQL query:

SELECT MONTH(order\_date) AS month, SUM(total\_amount) AS monthly\_sales

FROM orders

GROUP BY month

ORDER BY month;

2. Export the result to a CSV file:

INTO OUTFILE '/var/lib/mysql-files/monthly\_sales.csv';

3. Load the CSV into Tableau or Power BI.

4. Create a line chart to show monthly sales trends over time.