PROJECT – "ONLINE_SHOPPING_SYSTEM 2" USING MYSQL

INTRODUCTION TO THE ONLINE SHOPPING SYSTEM

The **Online Shopping System** is a database-driven project designed to simulate the core functionalities of a modern e-commerce platform. This system facilitates seamless interaction between customers and sellers by enabling users to browse products, place orders, and manage transactions efficiently.

The project utilizes **MySQL** as the backend database to store, manage, and retrieve data related to users, products, orders, payments, and inventory. Through the implementation of **Entity-Relationship (ER) diagrams**, table creation, and complex SQL queries, the system ensures data integrity, relational consistency, and efficient data retrieval.

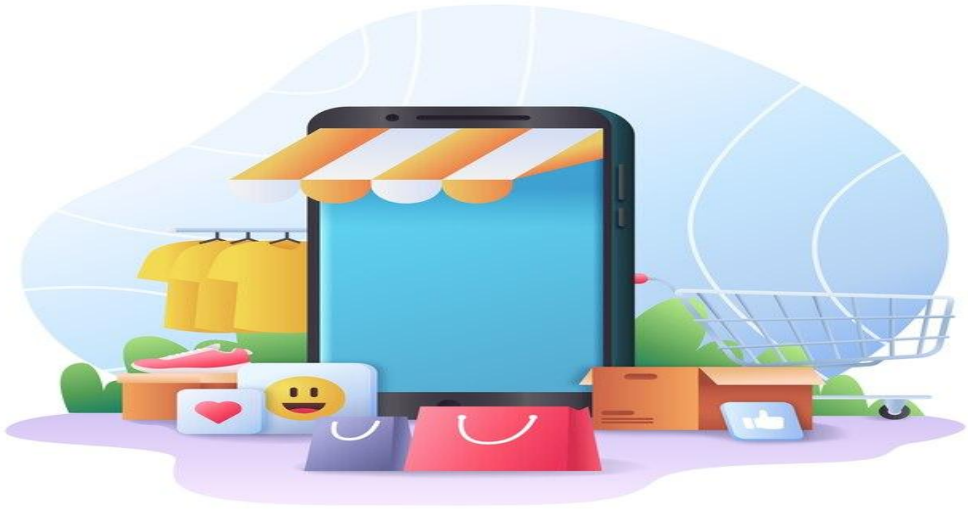
This project aims to demonstrate a practical understanding of database concepts such as **Primary Key, foreign key constraints, indexing, joins, and aggregate functions**. It also highlights the use of SQL for **data manipulation, querying, and reporting**, thereby providing a strong foundation in relational database management.

Key features of an Online Shopping System include:

1. **Product Catalogues**: A wide range of products displayed with detailed descriptions, images, prices, and availability.
2. **User Accounts**: Customers can create personal accounts to track their orders, save payment methods, and receive recommendations based on previous purchases.

PROJECT – "ONLINE_SHOPPING_SYSTEM 2" USING MYSQL

3. **Shopping Cart**: A virtual cart where customers can add items before proceeding to checkout.
4. **Payment Gateway Integration**: Secure processing of online payments through various methods like credit/debit cards, digital wallets, and net banking.
5. **Order Management**: Customers can view their order history, track the status of current orders, and manage returns or exchanges.
6. **Customer Reviews and Ratings**: Users can read reviews and ratings from other customers to help make informed purchasing decisions.
7. **Search and Filtering**: Customers can search for specific items or filter products based on categories, prices, brands, etc.



DESCRIPTION OF AN ER DIAGRAM

-- Project - "Online Shopping System"

-- The Project shows ER Diagram of system, Tables, Inserting data & Queries are solved by using MySQL

-- ER Diagram is drawn in Paint app

-- The ER diagram for the Online Shopping System consists of the following entities:

-- 1. Entities:

-- Customers: Represents the customers in the system.

/*Attributes:

customer_id (Primary Key)

first_name

last_name

email

phone*/

-- Products: Represents the products available for sale in the system.

/*Attributes:

product_id (Primary Key)

product_name

price

DESCRIPTION OF AN ER DIAGRAM

category*/

-- Orders: Represents the orders placed by customers.

/*Attributes:

order_id (Primary Key)

customer_id (Foreign Key referencing Customers)

order_date

total_amount*/

-- OrderItems: Represents the items within an order, linking products to orders.

/*Attributes:

order_item_id (Primary Key)

order_id (Foreign Key referencing Orders)

product_id (Foreign Key referencing Products)

quantity

price*/

-- Payments: Represents payments made for orders.

/*Attributes:

payment_id (Primary Key)

DESCRIPTION OF AN ER DIAGRAM

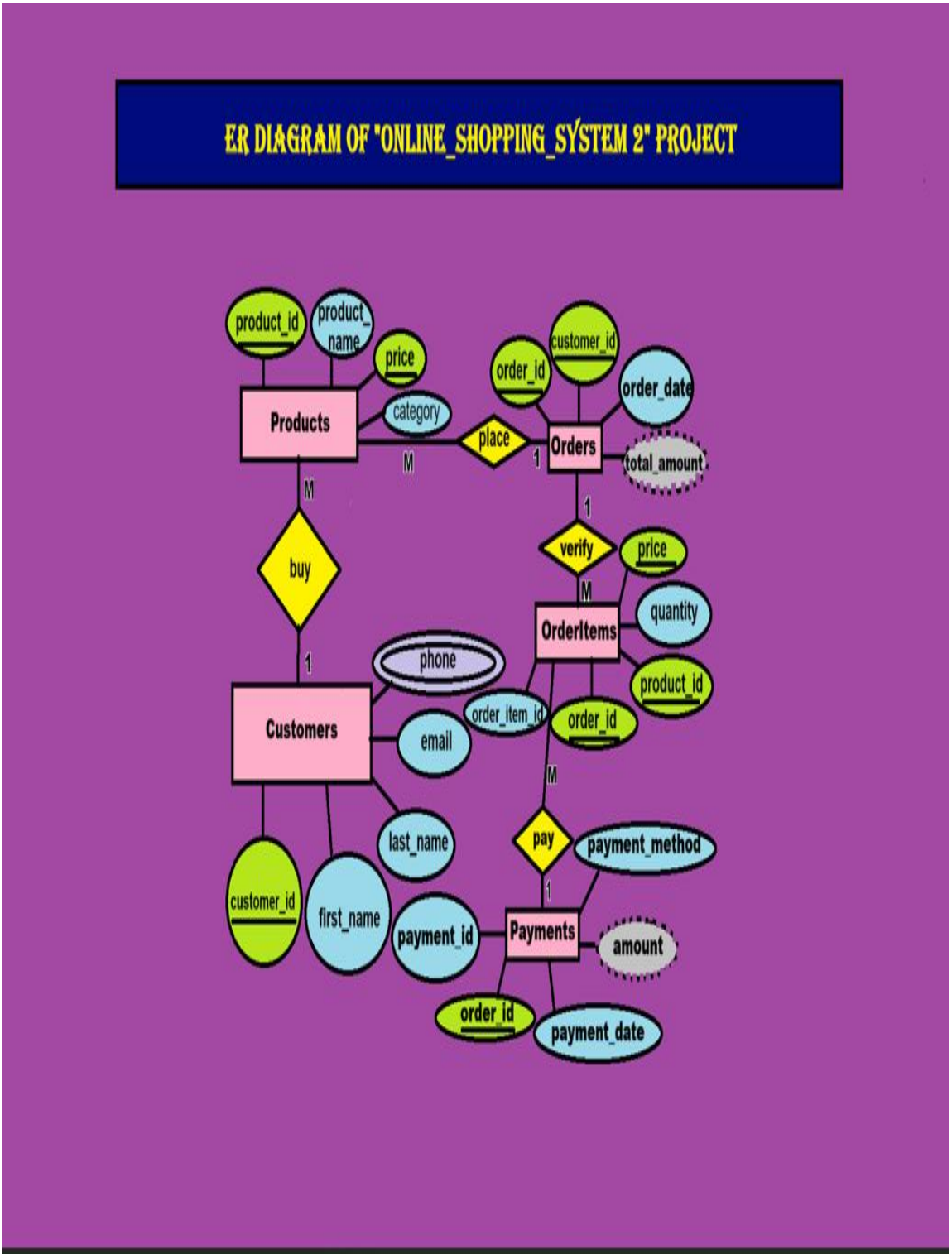
order_id (Foreign Key referencing Orders)

payment_date

amount

payment_method*/

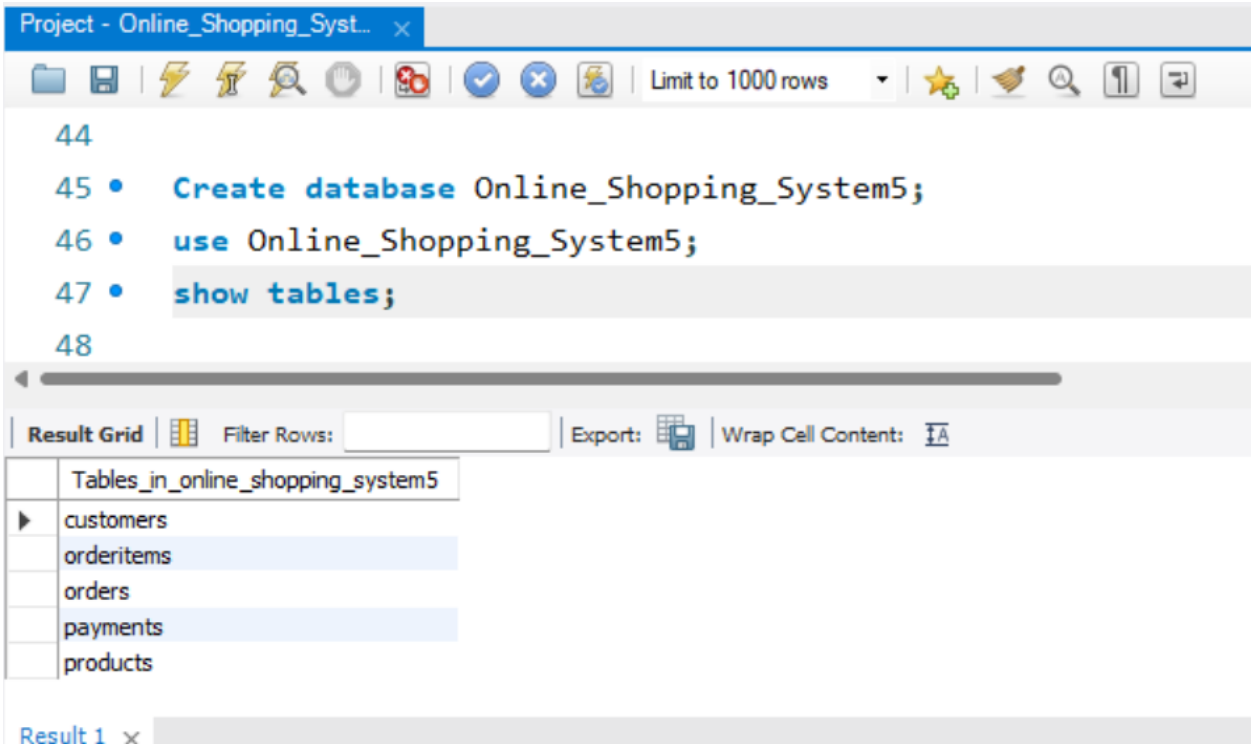
ER DIAGRAM OF AN "ONLINE SHOPPING SYSTEM" PROJECT

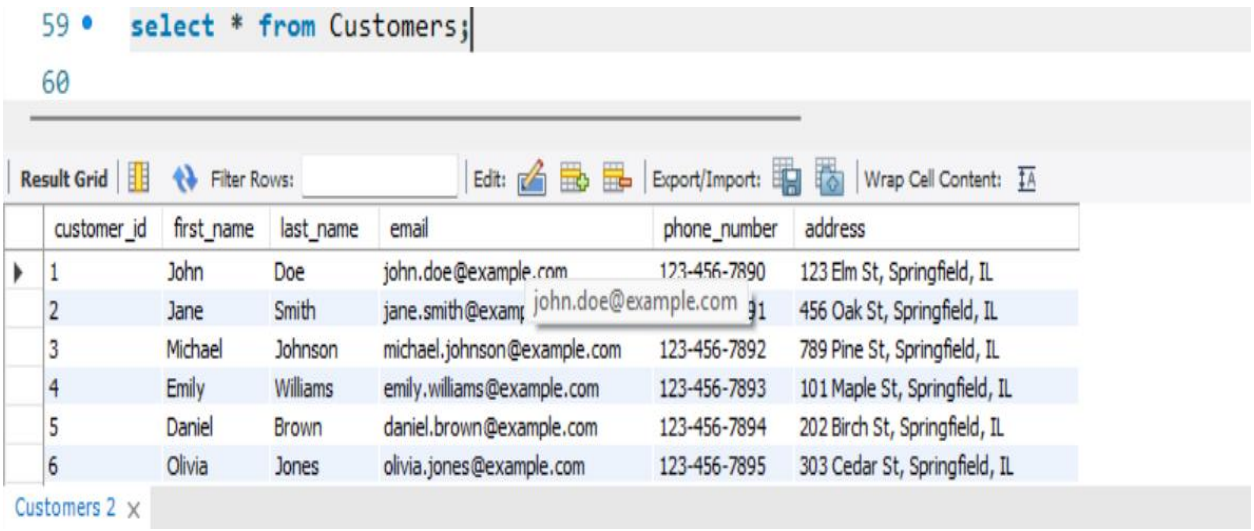


SQL QUERY

/*SQL Queries

Here are 24 sample SQL queries that can use to extract information from this system.*/





SQL QUERY

```
69 •  select * from Products;
70
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

product_id	product_name	description	price	stock_quantity
1	Laptop	15-inch laptop with Intel i7 processor	999.99	50
2	Smartphone	5G smartphone with 128GB storage	699.99	100
3	Headphones	Noise-canceling wireless headphones	199.99	150
4	Smartwatch	Fitness tracker with heart rate monitor	149.99	80
5	Tablet	10-inch Android tablet with stylus	249.99	70
6	Bluetooth Speaker	Portable Bluetooth speaker with bass boost	59.99	120

Products 3 X

```
80 •  select * from Orders;
81
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

order_id	customer_id	order_date	total_amount
1	1	2024-12-01	1099.98
2	2	2024-12-02	1199.99
3	3	2024-12-02	899.97
4	4	2024-12-03	499.99
5	5	2024-12-04	799.99
6	6	2024-12-05	399.98

Orders 5 X

SQL QUERY

```
93 •  select * from OrderItems;
94
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

order_item_id	order_id	product_id	quantity	price
1	1	1	1	999.99
2	1	2	1	99.99
3	2	3	2	99.99
4	2	5	1	249.99
5	3	4	1	149.99
6	3	7	1	89.99

OrderItems 6 x

```
105 •  select * from Payments;
106
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

payment_id	order_id	payment_date	amount	payment_method
1	1	2024-12-01	1099.98	Credit Card
2	2	2024-12-02	1199.99	PayPal
3	3	2024-12-02	899.97	Debit Card
4	4	2024-12-03	499.99	Credit Card
5	5	2024-12-04	799.99	Credit Card
6	6	2024-12-05	399.98	PayPal

Payments 7 x

SQL QUERY

```
371    -- 1. Simple SELECT Query
372 •  SELECT * FROM Customers;
373
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customer_id	first_name	last_name	email	phone_number	address
1	John	Doe	john.doe@example.com	123-456-7890	123 Elm St, Springfield, IL
2	Jane	Smith	jane.smith@example.com	123-456-7891	456 Oak St, Springfield, IL
3	Michael	Johnson	michael.johnson@example.com	123-456-7892	789 Pine St, Springfield, IL
4	Emily	Williams	emily.williams@example.com	123-456-7893	101 Maple St, Springfield, IL
5	Daniel	Brown	daniel.brown@example.com	123-456-7894	202 Birch St, Springfield, IL

Customers 2

```
374    -- 2. SELECT Specific Columns
375 •  SELECT customer_id, first_name, last_name FROM Customers;
376
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customer_id	first_name	last_name
1	John	Doe
2	Jane	Smith
3	Michael	Johnson
4	Emily	Williams
5	Daniel	Brown
6	Olivia	Jones

Customers 9

SQL QUERY

```
377    -- 3. WHERE Clause for Filtering
378 •  SELECT * FROM Products WHERE price > 100;
379
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

product_id	product_name	description	price	stock_quantity
1	Laptop	15-inch laptop with Intel i7 processor	999.99	50
2	Smartphone	5G smartphone with 128GB storage	699.99	100
3	Headphones	Noise-canceling wireless headphones	199.99	150
4	Smartwatch	Fitness tracker with heart rate monitor	149.99	80
5	Tablet	10-inch Android tablet with stylus	249.99	70
10	Monitor	24-inch Full HD LED monitor	129.99	40

Products 10 ×

```
380    -- 4. AND/OR Conditions in WHERE
381 •  SELECT * FROM Orders WHERE total_amount > 500 AND order_date > '2024-12-01';
382
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

order_id	customer_id	order_date	total_amount
2	2	2024-12-02	1199.99
3	3	2024-12-02	899.97
5	5	2024-12-04	799.99
7	7	2024-12-06	599.98
10	10	2024-12-09	1299.99
12	12	2024-12-10	999.99

Orders 11 ×

SQL QUERY

```
383    -- 5. ORDER BY Clause
384 •  SELECT * FROM Products ORDER BY price DESC;
385
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

product_id	product_name	description	price	stock_quantity
1	Laptop	15-inch laptop with Intel i7 processor	999.99	50
39	Refrigerator	Energy-efficient fridge with dual freezer compar...	799.99	15
27	Drone	Quadcopter drone with 4K camera	799.99	30
2	Smartphone	5G smartphone with 128GB storage	699.99	100
25	Camera	DSLR camera with 24MP sensor	549.99	40
40	Washing Machine	Front-load washing machine with 5kg capacity	399.99	25

Products 12 ×

```
386    -- 6. DISTINCT Keyword
387 •  SELECT DISTINCT payment_method FROM Payments;
388
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

payment_method
Credit Card
PayPal
Debit Card

Payments 13 ×

SQL QUERY

```
389    -- 7. COUNT Function
390 •  SELECT COUNT(*) FROM Orders WHERE order_date = '2024-12-01';
391
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	COUNT(*)
▶	1

Result 14 x

```
392    -- 8. SUM Function
393 •  SELECT SUM(total_amount) FROM Orders;
394
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	SUM(total_amount)
▶	20779.44

Result 15 x

SQL QUERY

```
395    -- 9. GROUP BY Clause
396 •  SELECT customer_id, COUNT(order_id) FROM Orders GROUP BY customer_id;
397
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	COUNT(order_id)
1	1
2	1
3	1
4	1
5	1
6	1

Result 16

```
398    -- 10. JOIN - INNER JOIN
399 •  SELECT o.order_id, c.first_name, c.last_name
400    FROM Orders o
401    INNER JOIN Customers c ON o.customer_id = c.customer_id;
402
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	first_name	last_name
1	John	Doe
2	Jane	Smith
3	Michael	Johnson
4	Emily	Williams
5	Daniel	Brown
6	Olivia	Jones

Result 17

SQL QUERY

```
403     -- 11. LEFT JOIN
404 •   SELECT o.order_id, c.first_name, c.last_name
405     FROM Orders o
406     LEFT JOIN Customers c ON o.customer_id = c.customer_id;
407
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	first_name	last_name
1	John	Doe
2	Jane	Smith
3	Michael	Johnson
4	Emily	Williams
5	Daniel	Brown
6	Olivia	Jones

Result 18

```
408     -- 12. RIGHT JOIN
409 •   SELECT o.order_id, c.first_name, c.last_name
410     FROM Orders o
411     RIGHT JOIN Customers c ON o.customer id = c.customer id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	first_name	last_name
1	John	Doe
2	Jane	Smith
3	Michael	Johnson
4	Emily	Williams
5	Daniel	Brown
6	Olivia	Jones

Result 20

SQL QUERY

```
413    -- 13. SELECT with LIKE
414 •  SELECT * FROM Products WHERE product_name LIKE '%phone%';
415
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

product_id	product_name	description	price	stock_quantity
2	Smartphone	5G smartphone with 128GB storage	699.99	100
3	Headphones	Noise-canceling wireless headphones	199.99	150
23	Phone Case	Durable silicone phone case for iPhone	19.99	200
NULL	NULL	NULL	NULL	NULL

Products 21 ×

```
416    -- 14. SELECT with IN
417 •  SELECT * FROM Products WHERE product_id IN (1, 2, 5, 8);
418
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

product_id	product_name	description	price	stock_quantity
1	Laptop	15-inch laptop with Intel i7 processor	999.99	50
2	Smartphone	5G smartphone with 128GB storage	699.99	100
5	Tablet	10-inch Android tablet with stylus	249.99	70
8	Mouse	Wireless optical mouse	19.99	300
NULL	NULL	NULL	NULL	NULL

Products 22 ×

SQL QUERY

```
419    -- 15. UPDATE Statement
420 •  UPDATE Customers SET email = 'newemail@example.com' WHERE customer_id = 3;
421
422    -- 16. LIMIT Clause
423 •  SELECT * FROM Orders LIMIT 5;
424
425    -- 17. BETWEEN Operator
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
42	00:31:23	SELECT order_id, CASE WHEN total_amount > 1000 THEN 'High' ...	50 row(s) returned	0.000 sec / 0.000 sec
43	00:32:01	SELECT o.order_id, c.first_name, p.product_name FROM Orders o JOIN Customer...	50 row(s) returned	0.000 sec / 0.000 sec
44	00:32:31	SELECT product_name FROM Products WHERE price < 100 UNION SELECT pro...	29 row(s) returned	0.000 sec / 0.000 sec
45	00:32:50	SELECT customer_id, first_name, last_name FROM Customers WHERE EXISTS (S...	50 row(s) returned	0.000 sec / 0.000 sec
46	00:33:35	UPDATE Customers SET email = 'newemail@example.com' WHERE customer_id = 3	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0	0.000 sec

```
422    -- 16. LIMIT Clause
423 •  SELECT * FROM Orders LIMIT 5;
424
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

order_id	customer_id	order_date	total_amount
1	1	2024-12-01	1099.98
2	2	2024-12-02	1199.99
3	3	2024-12-02	899.97
4	4	2024-12-03	499.99
5	5	2024-12-04	799.99
NULL	NULL	NULL	NULL

Orders 23 X

SQL QUERY

```
425    -- 17. BETWEEN Operator
426 •  SELECT * FROM Orders WHERE order_date BETWEEN '2024-12-01' AND '2024-12-15';
427
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

order_id	customer_id	order_date	total_amount
1	1	2024-12-01	1099.98
2	2	2024-12-02	1199.99
3	3	2024-12-02	899.9 (899.97)
4	4	2024-12-03	499.99
5	5	2024-12-04	799.99
6	6	2024-12-05	399.98

Orders 24 X

```
428    -- 18. Subquery
429 •  SELECT first_name, last_name FROM Customers
430    WHERE customer_id IN (SELECT customer_id FROM Orders WHERE total_amount > 500);
431
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

first_name	last_name
John	Doe
Jane	Smith
Michael	Johnson
Daniel	Brown
William	Garcia
Charlotte	Davis

Customers 26 X

SQL QUERY

```
-- 19. SELECT with NOT IN
SELECT * FROM Products WHERE product_id NOT IN (1, 2, 3, 4);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

product_id	product_name	description	price	stock_quantity
5	Tablet	10-inch Android tablet with stylus	249.99	70
6	Bluetooth Speaker	Portable Bluetooth speaker with bass boost	59.99	120
7	Keyboard	Mechanical keyboard with RGB lighting	89.99	200
8	Mouse	Wireless optical mouse	19.99	300
9	Webcam	1080p HD webcam with built-in microphone	79.99	60
10	Monitor	24-inch Full HD LED monitor	129.99	40

Products 27 x

```
-- 20. Using NULL with IS NULL
SELECT * FROM Customers WHERE email IS NULL;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customer_id	first_name	last_name	email	phone_number	address
NULL	NULL	NULL	NULL	NULL	NULL

Customers 28 x

SQL QUERY

```sql
SELECT order_id,
       CASE
           WHEN total_amount > 1000 THEN 'High'
           WHEN total_amount BETWEEN 500 AND 1000 THEN 'Medium'
           ELSE 'Low'
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	order_value
1	High
2	High
3	Medium
4	Low
5	Medium
6	Low

Result 29

```sql
-- 22. Alias for Tables
SELECT o.order_id, c.first_name, p.product_name
FROM Orders o
JOIN Customers c ON o.customer id = c.customer id
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	first_name	product_name
1	John	Laptop
1	John	Smartphone
2	Jane	Headphones
2	Jane	Tablet
3	Michael	Smartwatch
3	Michael	Keyboard

Result 30

SQL QUERY

```
454     -- 23. UNION
455 •   SELECT product_name FROM Products WHERE price < 100
456     UNION
457     SELECT product_name FROM Products WHERE price > 1000;
458
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_name
Bluetooth Speaker
Keyboard
Mouse
Webcam
Printer
Charger

Result 31

```
459     -- 24. EXISTS Subquery
460 •   SELECT customer_id, first_name, last_name
461     FROM Customers
462     WHERE EXISTS (SELECT * FROM Orders WHERE Orders.customer_id = Customers.customer_id);
463
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customer_id	first_name	last_name
1	John	Doe
2	Jane	Smith
3	Michael	Johnson
4	Emily	Williams
5	Daniel	Brown
6	Olivia	Jones

Customers 32

INSIGHTS OF "ONLINE SHOPPING SYSTEM" PROJECT

The **Online Shopping System** project provided valuable hands-on experience in designing and managing a relational database for a real-world scenario. Below are the key insights gained from the project:

🧩 1. Database Design & Normalization

- Learned how to design a well-structured **Entity-Relationship (ER) diagram** to represent the relationships between key entities like Customers, Products, Orders, Payments, and OrderItems.
- Applied **normalization techniques** to eliminate data redundancy and ensure data consistency across the database.

🛠️ 2. Table Creation & Schema Implementation

- Created multiple relational tables using **MySQL DDL (Data Definition Language)** with appropriate primary and foreign keys.
- Defined **constraints** (e.g., NOT NULL, UNIQUE, CHECK) to ensure data integrity.

🔍 3. Data Manipulation & Querying

- Inserted sample records to simulate a working e-commerce environment using **INSERT** statements.

INSIGHTS OF "ONLINE SHOPPING SYSTEM" PROJECT

- Wrote and executed various **DML queries** to perform tasks such as:
    - Retrieving customer order history
    - Fetching product details by category
    - Calculating total sales and revenue
    - Checking stock availability

🔁 4. Use of Joins and Subqueries

- Mastered the use of **INNER JOIN, LEFT JOIN, RIGHT JOIN**, and **self-joins** to combine and retrieve data from multiple related tables.
- Used **subqueries** and **nested queries** to solve complex business queries like:
    - Finding the most popular products
    - Identifying top customers by purchase value
    - Generating monthly sales reports

📊 5. Aggregate Functions & Grouping

- Utilized **aggregate functions** such as SUM(), AVG(), COUNT(), MIN() and MAX() with **GROUP BY** and **HAVING** clauses for business insights.

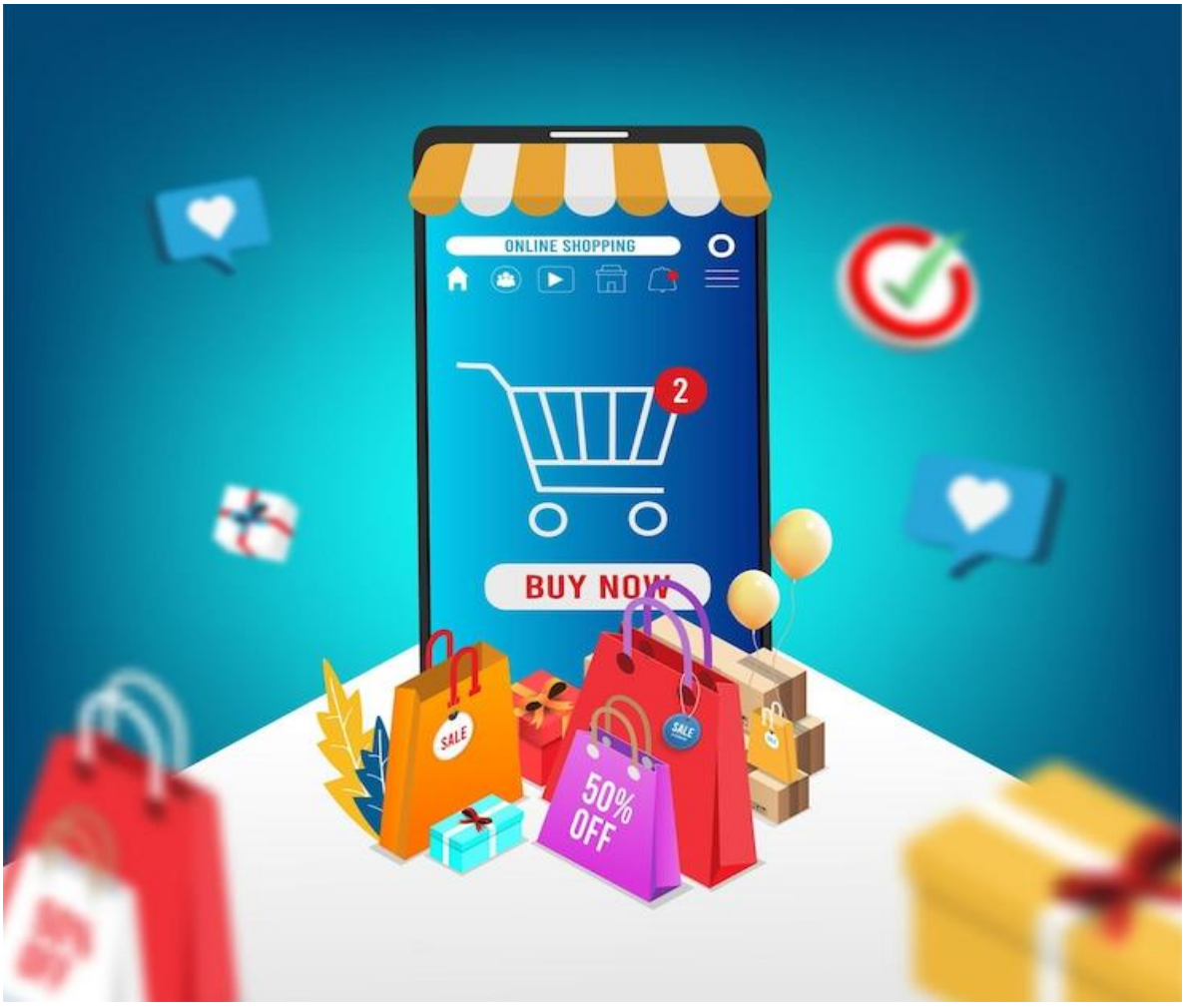
⚙️ 6. Practical Understanding of Business Logic

INSIGHTS OF "ONLINE SHOPPING SYSTEM" PROJECT

- Understood how different entities interact in a typical online shopping scenario.
- Learned to translate **real-world business logic** into structured SQL queries.

🔒 7. Data Security & Access Control (Optional)

- Explored basic concepts of **user privileges** and **access control** in MySQL (if implemented).



FUTURE_SCOPE OF "ONLINE SHOPPING SYSTEM" PROJECT

📦 1. Inventory & Logistics Management

- Introduce advanced inventory tracking with **automatic stock updates**.
- Add modules for **shipment tracking, delivery status**, and **logistics provider integration**.

📈 2. Data Analytics and Reporting

- Build **dashboard features** for admin to visualize sales trends, customer behavior, and product performance using **Power BI, Tableau, or custom reporting tools**.
- Add support for **predictive analytics** using historical data.

💳 3. Payment Gateway Integration

- Simulate or integrate real-time **payment gateways** (e.g., Razorpay, Stripe, PayPal) for handling transactions securely.

🤖 4. Machine Learning Features

- Recommend products to users based on their browsing and purchasing history using **recommendation systems**.
- Detect fraudulent transactions or suspicious activities using **ML models**.


CONCLUSION OF "ONLINE SHOPPING SYSTEM" PROJECT

The "**Online Shopping System"** project successfully demonstrates the application of **relational database concepts** using **MySQL** to manage and streamline the core functionalities of an e-commerce platform. From designing a normalized database schema to implementing complex SQL queries for real-world scenarios, this project showcases the importance of structured data management in building reliable, scalable, and efficient systems.

Through this project, essential skills in **database design, query optimization, data manipulation, and integrity enforcement** were developed. The system lays the groundwork for further enhancements, such as frontend integration, advanced analytics, and cloud deployment, making it a scalable solution for real-world use.

Overall, the project reflects a solid understanding of MySQL and its practical applications in developing database-driven systems that align with modern business requirements.


OPEN
THANK YOU
For Shopping With Us
Subscribe Now


THANK
YOU