Online Retail Sales Database Report

# 1. Objective

Design and implement a normalized SQL database for an online retail platform to manage customers, products, orders, and payments efficiently.

# 2. Tools Used

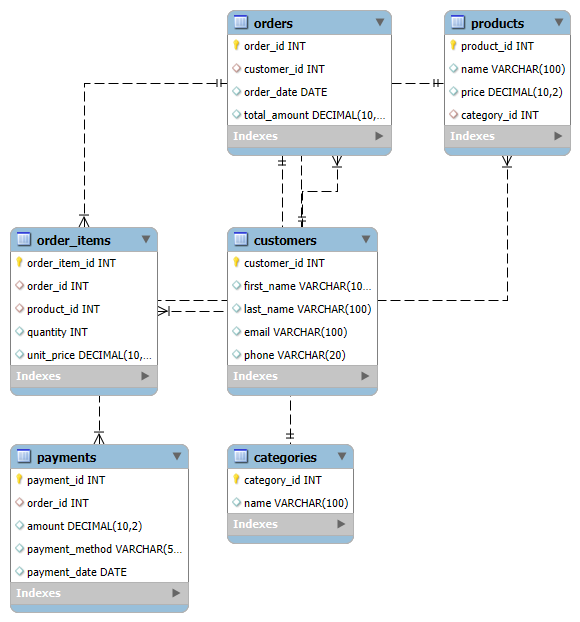
- MySQL Workbench  
- dbdiagram.io

# 3. Entities and Relationships

Entities: Customers, Categories, Products, Orders, Order\_Items, Payments  
  
Relationships:  
- Customer → Orders (1:M)  
- Order → Order\_Items (1:M)  
- Order\_Item → Product (M:1)  
- Order → Payments (1:M)  
- Product → Category (M:1)

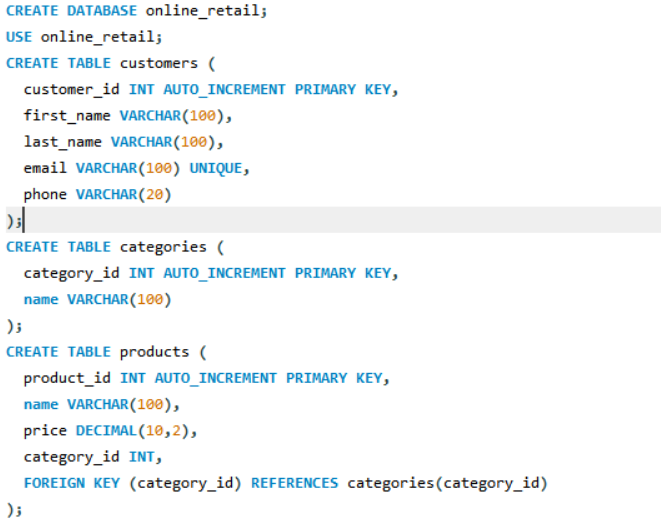
# 4. ER Diagram

Insert screenshot of ER diagram here  
Figure 1: ER Diagram of Online Retail Database



# 5. SQL Schema (DDL)

Insert screenshot of SQL table creation here  
Figure 2: SQL tables created in MySQL Workbench



# 

# 

# 6. Sample Data (DML)

Insert screenshot of sample data insertion here

INSERT INTO customers (first\_name, last\_name, email, phone)

VALUES ('Asha', 'Sharma', 'asha@example.com', '9876543210'),

('John', 'Doe', 'john@example.com', '9999999999');

INSERT INTO categories (name) VALUES ('Books'), ('Electronics');

INSERT INTO products (name, price, category\_id)

VALUES ('Database Design Book', 400.00, 1),

('Smartphone X', 25000.00, 2);

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

VALUES (1, '2025-10-01', 400.00),

(2, '2025-10-02', 25000.00);

INSERT INTO order\_items (order\_id, product\_id, quantity, unit\_price)

VALUES (1, 1, 1, 400.00),

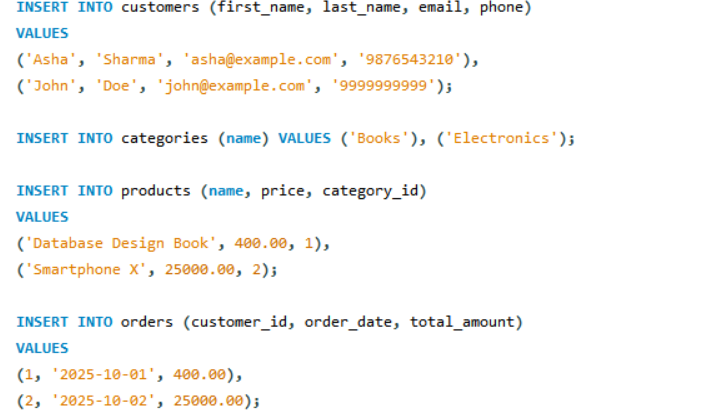
(2, 2, 1, 25000.00);

INSERT INTO payments (order\_id, amount, payment\_method, payment\_date)

VALUES (1, 400.00, 'Credit Card', '2025-10-01'),

(2, 25000.00, 'Net Banking', '2025-10-02');

Figure 3: Sample data inserted into tables



# 

# 7. Queries and Reports

# 7.1 View Customer Orders

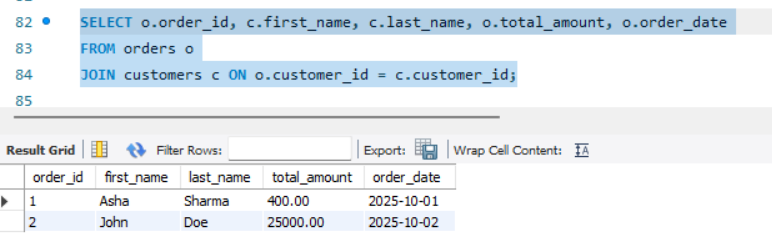
Insert screenshot of query result here

SELECT o.order\_id, c.first\_name, c.last\_name, o.total\_amount, o.order\_date

FROM orders o

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

Figure 4: Customer orders with total amount



# 7.2 Order Details

Insert screenshot of query result here

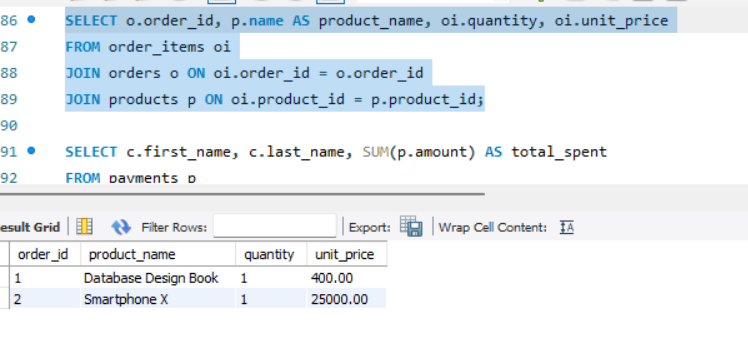
SELECT o.order\_id, p.name AS product\_name, oi.quantity, oi.unit\_price

FROM order\_items oi

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

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

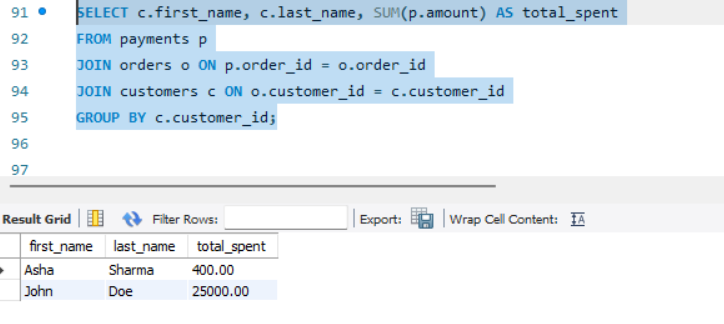
Figure 5: Products ordered with quantity and price



# 7.3 Total Amount Spent by Each Customer

Insert screenshot of query result here  
  
SELECT c.first\_name, c.last\_name, SUM(p.amount) AS total\_spent  
FROM payments p  
JOIN orders o ON p.order\_id = o.order\_id  
JOIN customers c ON o.customer\_id = c.customer\_id  
GROUP BY c.customer\_id;

Figure 6: Total spending per customer



# 8. Conclusion

The database is normalized to 3NF, supports relational integrity, and allows efficient sales reporting. Screenshots confirm the SQL scripts and query results work correctly.