

# Online Retail Sales Database Project Report:

---

## Introduction:

The Online Retail Sales Database is designed to manage e-commerce operations efficiently. It maintains records of customers, products, orders, and payments, ensuring smooth tracking of sales, inventory, and customer transactions. The database is normalized to reduce redundancy and improve data integrity.

## Abstract:

This project demonstrates a fully functional **relational database system** for an e-commerce platform. It includes the creation of a normalized SQL schema, sample data insertion, and the execution of queries and views to generate meaningful reports such as customer order history, total sales, and best-selling products. The design supports scalability and can be extended to include categories, staff users, and delivery management.

## Tools Used:

Database: MySQL

Schema Design: dbdiagram.io for ER modeling

Query Execution: MySQL Workbench

## Steps Involved in Building the Project

### Phase 1: Database & Schema Creation

- Create database ECOMMERCE.
- Define tables: Customers, Products, Orders, Order\_Items, Payments.
- Apply primary and foreign key constraints for referential integrity.

### Phase 2: Sample Data Insertion

- Insert 5–10 rows in each table for realistic testing.
- Include diverse customers, product catalog, orders, and payments.



### Phase 3: Queries & Views

- Simple Queries: List all products, all customers, and pending orders.
- JOIN Queries: Total spending per customer, best-selling products, sales by date.
- Views: Daily Sales Report, Customer Order History.

## Phase 4: Reports

- Generate visual reports using query outputs (e.g., daily\_sales.png, best\_selling.png).
- Use views for recurring report generation.

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
-- views for report
-- Daily Sales Report
CREATE VIEW Daily_Sales AS
SELECT DATE(o.order_date) AS sale_date, SUM(p.amount) AS total_sales
FROM Orders o
JOIN Payments p ON o.order_id = p.order_id
GROUP BY sale_date;
SELECT*FROM Daily_Sales;
```

The Result Grid shows the output of the query:

sale_date	total_sales
2025-08-01	55000.00
2025-08-02	140000.00
2025-08-03	25000.00
2025-08-04	13500.00
2025-08-05	2400.00
2025-08-06	35000.00
2025-08-07	60000.00
2025-08-08	80000.00
2025-08-09	15000.00
2025-08-10	18000.00

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
-- Customer Order History
CREATE VIEW Customer_Order_History AS
SELECT c.name, o.order_id, o.order_date, o.status, p.amount, p.method
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
JOIN Payments p ON o.order_id = p.order_id;
SELECT*FROM Customer_Order_History;
```

The Result Grid shows the output of the query:

	name	order_id	order_date	status	amount	method
▶	Amit Sharma	1	2025-08-01 10:30:00	Completed	55000.00	Credit Card
	Priya Verma	2	2025-08-02 14:00:00	Pending	140000.00	UPI
	Ravi Kumar	3	2025-08-03 16:20:00	Shipped	25000.00	Net Banking
	Sneha Reddy	4	2025-08-04 09:15:00	Completed	13500.00	Debit Card
	Anil Mehta	5	2025-08-05 11:45:00	Cancelled	2400.00	Cash
	Sunita Joshi	6	2025-08-06 13:50:00	Pending	35000.00	Credit Card
	Rahul Singh	7	2025-08-07 17:30:00	Shipped	60000.00	UPI
	Divya Iyer	8	2025-08-08 12:10:00	Completed	80000.00	Net Banking
	Karan Malhotra	9	2025-08-09 18:25:00	Pending	15000.00	Debit Card
	Neha Gupta	10	2025-08-10 15:05:00	Completed	18000.00	Credit Card

## Conclusion

The project successfully models an e-commerce platform with a \*normalized database schema\*, ensuring efficient data storage, retrieval, and reporting. The system supports operational needs such as order tracking, payment management, and sales analysis. With optional enhancements, it can evolve into a complete retail management system.