Online Shopping Marketplace – End-to-End SQL & Data Analytics Project

Project Overview

This project simulates a **real-world online shopping marketplace** and demonstrates how a data analyst designs, builds, and analyzes a relational database to answer **business-critical questions**.

The focus of the project is on:

- Designing a **normalized relational database** using MySQL
- Implementing **data integrity rules** (primary keys, foreign keys, triggers)
- Writing **advanced SQL queries** for business analytics
- Using **window functions, set operations, and stored procedures**
- Translating raw data into **actionable business insights**

This project is designed to reflect the type of work expected from a **Junior Data Analyst / BI Analyst / SQL Analyst** in an e-commerce or marketplace environment.

Business Problem

An online shopping marketplace wants to better understand:

- Who are **most valuable customers**
- Which **vendors and products** generate the most revenue
- Whether **customer reviews impact sales**
- How to rank products and categories based on performance
- How to calculate **profitability** efficiently over time

To support these goals, a scalable and analytics-ready database system is required.

Tools & Technologies

- **Database:** MySQL (RDBMS)
- **Query Language:** SQL (Joins, Aggregations, Window Functions, Set Operations)
- **Data Generation:** Python (for synthetic product data)
- **Data Import:** Excel → MySQL
- **Concepts:** Normalization (3NF), ACID, CAP Theorem, Query Optimization

Database Design

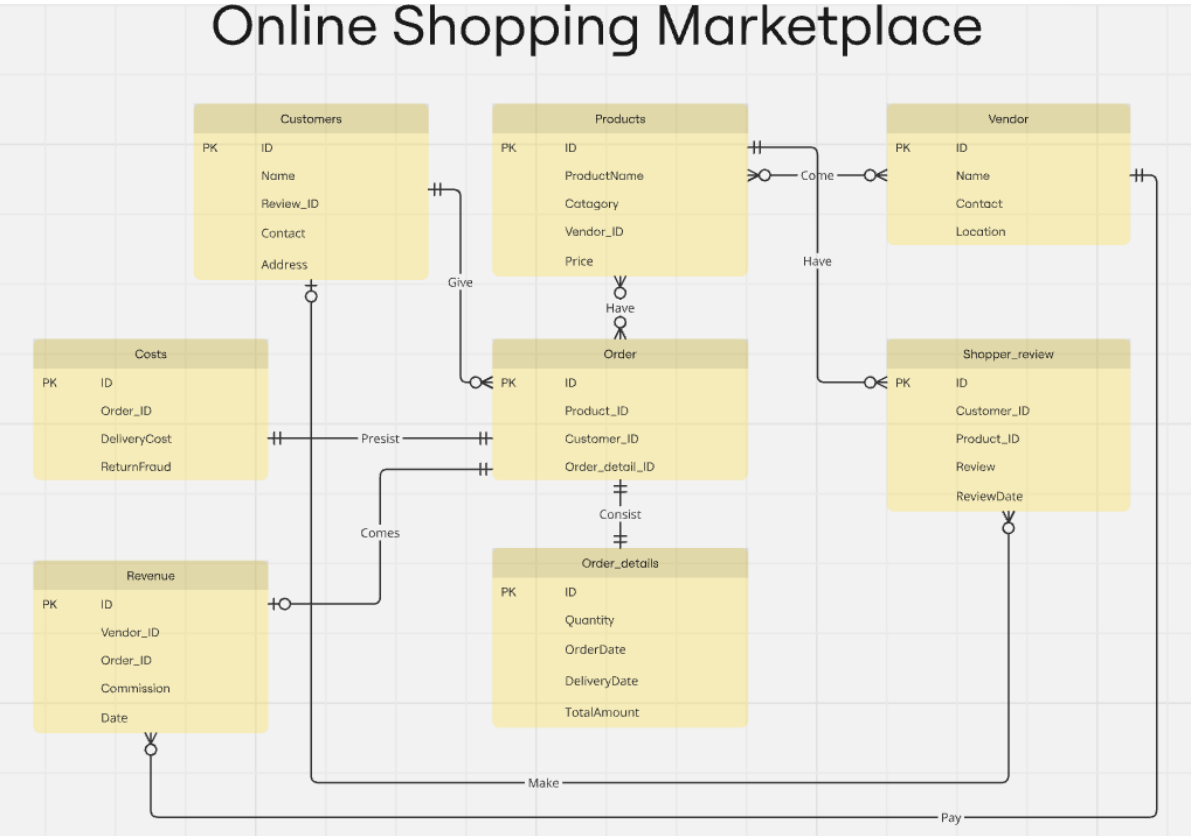
Entity Relationship Model

The database was designed using an **Entity Relationship (ER) Diagram** to ensure clarity, scalability, and proper normalization.

Core Entities:

- Customers
- Products
- Vendors
- Orders
- Order Details
- Shopper Reviews
- Costs
- Revenue

Figure 1: ER Diagram for the database



The schema was normalized to **Third Normal Form (3NF)** to reduce redundancy and improve query performance. Relationships were enforced using **foreign key constraints**.

Database Schema Implementation

Each entity was implemented as a MySQL table with:

- Appropriate data types
- Primary keys
- Foreign key constraints

Below we will see the tables:

Figure 2: Product table

Column Name	Datatype
ID	INT
ProductName	VARCHAR(100)
Catagory	VARCHAR(45)
Vendor_ID	INT
Price	INT

Foreign Key Name	Referenced Table
Vendor_ID	`online_shopping_marketplace`.`vendor`

Figure 3: Customers Table

Column Name	Datatype
ID	INT
Name	VARCHAR(100)
Review_ID	INT
Contact	VARCHAR(100)
Address	VARCHAR(100)

Foreign Key Name	Referenced Table
Review_ID	`online_shopping_marketplace`.`shopper_revi...

Figure 4: Vendor Table

Column Name	Datatype
ID	INT
Name	VARCHAR(45)
Contact	VARCHAR(100)
Location	VARCHAR(45)

Figure 5: Orders Table

Column Name	Datatype
ID	INT
Product_ID	INT
Customer_ID	INT
Order_detail_ID	INT

Foreign Key Name	Referenced Table
Customer_ID	`online_shopping_marketplace`.`customers`
Order_detail_ID	`online_shopping_marketplace`.`order_details`
product_ID	`online_shopping_marketplace`.`products`

Figure 6: Order details Table

Column Name	Datatype
ID	INT
Quantity	INT
OrderDate	DATE
DeliveryDate	DATE
TotalAmount	INT

Figure 7: Shopper review Table

Column Name	Datatype
ID	INT
Customers_ID	INT
Products_ID	INT
Review	INT
ReviewDate	DATE

Foreign Key Name	Referenced Table
Customers_ID	`online_shopping_marketplace`.`customers`
Products_ID	`online_shopping_marketplace`.`products`

Figure 8: Costs Table

Column Name	Datatype
ID	INT
Order_ID	INT
DeliveryCost	FLOAT
ReturnFraud	INT

Foreign Key Name	Referenced Table
Order_ID	`online_shopping_marketplace`.`orders`

Figure 9: Revenue Table

Column Name	Datatype
ID	INT
Vendor_ID	INT
Orders_ID	INT
Commission	INT
Date	DATE

Foreign Key Name	Referenced Table
Orders_ID	`online_shopping_marketplace`.`orders`
Vendors_ID	`online_shopping_marketplace`.`vendor`

Data Integrity & Automation

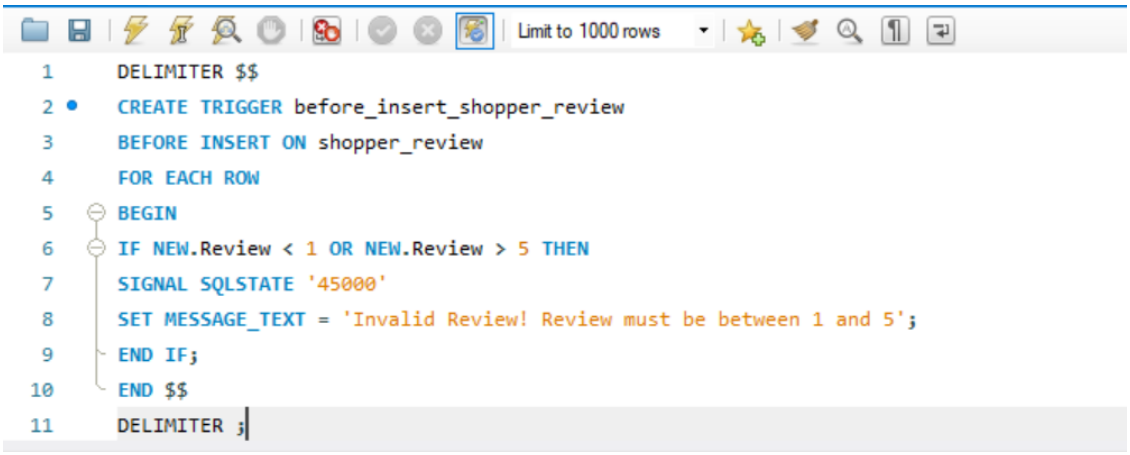
Trigger: Review Validation

A database trigger was implemented on the **Shopper Reviews** table to enforce valid rating values.

Business Rule:

- Product ratings must be between **1 and 5**
- Invalid ratings are rejected automatically

Figure 10: Trigger SQL Query

- 

This ensures high data quality and prevents invalid user input.

Data Ingestion

Synthetic Data Generation

- Python was used to generate realistic product data (names, categories, prices, vendors)
- Data was inserted into MySQL using a database connector

Figure 11: Populating Products table using Python

```
import mysql.connector
import random


conn = mysql.connector.connect(
    host="127.0.0.1",
    user="root",
    password="Faisalkhandahri#007",
    database="online_shopping_marketplace"
)

cursor = conn.cursor()

products = [
        ("Mobile", "Technology"),
        ("Laptop", "Technology"),
        ("Desk", "Furniture"),
        ("Chair", "Furniture"),
        ("Computer", "Technology"),
        ("Lamp", "Furniture"),
        ("Sofa", "Furniture"),
        ("Dishwasher", "Kitchen_Equipment"),
        ("Oven", "Kitchen_Equipment")
]

vendor_id = [f"{i:03d}" for i in range(1, 101)]

sql = "INSERT INTO Products (ID, ProductName, Catagory, Vendor_ID, Price) VALUES (%s, %s, %s, %s, %s)"

data = [
       (i + 1,  # ID (starting from 1 to 1000)
       *random.choice(products),
       random.choice(vendor_id),
       round(random.uniform(50, 2000), 2))
       for i in range(1000)
]

cursor.executemany(sql, data)  # Insert multiple rows efficiently
conn.commit()
```

Figure 12: Result of the Python code for Products Table

Result Grid | Filter Rows: | Edit:

ID	ProductName	Catagory	Vendor_ID	Price
1	Desk	Furniture	27	805.2
2	Oven	Kitchen_Equipment	9	90.39
3	Computer	Technology	54	1224.53
4	Desk	Furniture	84	1262.1
5	Chair	Furniture	24	1230.66
6	Laptop	Technology	14	1287.74

Bulk Import

- Remaining tables were populated using Excel-to-MySQL imports
- Furthermore, we successfully populated the rest of the 7 tables of our database by importing excel files.

Business Analytics Using SQL

1. Customer Purchasing Behavior

Objective: Identify the **Top 5 highest-spending customers**

Techniques Used:

- Multi-table joins
- Aggregations (SUM)
- Sorting & limiting results

Figure 13: Analysis of Customer Purchasing Behavior/ Top 5 Customers

```
1  SELECT Customers.ID AS Customer_ID, Customers.Name,
2  SUM(Order_details.TotalAmount) AS Total_Spending
3  FROM Customers
4  JOIN Orders ON Customers.ID = Orders.Customer_ID
5  JOIN Order_details ON Orders.Order_detail_ID = Order_details.ID
6  GROUP BY Customers.ID, Customers.Name
7  ORDER BY Total_Spending DESC
8  LIMIT 5;
```

Customer_ID	Name	Total_Spending
34	Juline Mougel	10
42	Alexandros Scrancher	10
56	Nessy Mundwell	10
44	Saxe Hince	10
90	Teresa Sucre	10

Insight: A small group of customers contributes disproportionately to total revenue.

2. Most Profitable Vendor

Objective: Determine which vendor generates the highest revenue

Techniques Used:

- Aggregations
- Multi-level joins
- ORDER BY + LIMIT optimization

Figure 14: Most Profitable Vendor

```
1 •  SELECT Vendor.ID AS Vendor_ID, Vendor.Name AS Vendor_Name,
2    SUM(TotalAmount) AS Total_Revenue
3    FROM Vendor
4    JOIN Products ON Products.Vendor_ID = Vendor.ID
5    JOIN Orders ON Orders.Product_ID = Products.ID
6    JOIN order_details ON Orders.order_detail_ID = Order_Details.ID
7    GROUP BY Vendor.ID, Vendor.Name
8    ORDER BY Total_Revenue DESC
9    LIMIT 1;
```

	Vendor_ID	Vendor_Name	Total_Revenue
▶	71	Mycat	92

Insight: Vendor performance can be ranked to support vendor partnerships and commission strategies.

3. Top 3 Best-Selling Products

Objective: Identify products with the highest sales volume

Techniques Used:

- SUM on quantities
- Product-level grouping

Figure 15: Top 3 Best-selling Products

```
1 •  SELECT ProductName, SUM(Quantity) AS Total_Sold
2    FROM Products
3    JOIN Orders ON Orders.Product_ID = Products.ID
4    JOIN order_details ON Orders.order_detail_ID = Order_Details.ID
5    GROUP BY ProductName
6    ORDER BY Total_Sold DESC
7    LIMIT 3;
```

ProductName	Total_Sold
Desk	723
Computer	716
Dishwasher	690

4. Revenue vs Customer Ratings

Objective: Analyze whether higher ratings lead to higher revenue

Products were grouped into:

- **High-rated (4–5 stars)**
- **Low-rated (1–3 stars)**

Techniques Used:

- CASE statements
- Averages
- Multi-table joins

Figure 16: Average Revenue base on Customers Reviews

```
SELECT
CASE
WHEN Review >= 4 THEN 'High Rated'
ELSE 'Low Rated'
END AS Rating_Category,
AVG(TotalAmount) AS Avg_Revenue
FROM Shopper_Review
JOIN products ON shopper_review.Products_ID = products.ID
JOIN Orders  ON Orders.Product_ID = Products.ID
JOIN order_details ON Orders.order_detail_ID = Order_Details.ID
GROUP BY Rating_Category;
```

Rating_Category	Avg_Revenue
Low Rated	4.8759
High Rated	5.0595

Insight: Higher-rated products tend to generate higher average revenue.

5. Product Ranking Using Window Functions

Objective: Rank products by total revenue while displaying average ratings

Techniques Used:

- Window functions (RANK() OVER())
- Aggregations without collapsing rows

Figure 17: Ranking Products based on Revenue and Review

```
1  SELECT
2      ProductName,
3      SUM(TotalAmount) AS Total_Revenue,
4      AVG(Review) AS Avg_Review,
5      RANK() OVER (ORDER BY SUM(TotalAmount) DESC) AS Revenue_Rank
6      FROM Products
7      JOIN Orders ON orders.Product_ID = products.ID
8      JOIN Order_details  ON orders.order_detail_ID = order_details.ID
9      JOIN Shopper_Review ON shopper_review.Products_ID = products.ID
10     GROUP BY ProductName;
```

ProductName	Total_Revenue	Avg_Review	Revenue_Rank
Dishwasher	674	3.2016	1
Lamp	653	3.4609	2
Desk	648	3.0923	3
Computer	575	2.8607	4
Oven	566	2.9107	5
Sofa	498	3.1132	6
Laptop	496	3.0000	7
Mobile	485	2.9158	8
Chair	358	2.9136	9

6. Best-Selling Category (Set Operations)

Objective: Identify the top-performing product category

Techniques Used:

- UNION set operation
- Aggregation by category

Figure 18: Best Selling Category

```sql
SELECT Catagory, SUM(TotalAmount) AS Total_Revenue
FROM Products
JOIN Orders ON orders.Product_ID = Products.ID
JOIN order_details ON orders.order_detail_ID = order_details.ID
GROUP BY Catagory

UNION

SELECT 'Top Category' AS Catagory, MAX(Total_Revenue)
FROM (
SELECT Catagory, SUM(TotalAmount) AS Total_Revenue
FROM Products
JOIN Orders ON orders.Product_ID = Products.ID
JOIN order_details ON orders.order_detail_ID = order_details.ID
GROUP BY Catagory
) AS RevenueData
ORDER BY Total_Revenue DESC
LIMIT 1;
```

Catagory	Total_Revenue
Furniture	2157

7. Stored Procedure for Profit Analysis

A stored procedure was created to calculate:

- Total Revenue
- Total Cost (delivery + fraud)
- Total Profit

This allows analysts or stakeholders to retrieve profitability metrics for a given date with a single command.

Figure 19: Store Procedure Query/ Results

```
DELIMITER $$
CREATE PROCEDURE monthly_report(IN report_month VARCHAR(7))
BEGIN
DECLARE total_revenue DECIMAL(10,2);
DECLARE total_cost DECIMAL(10,2);
DECLARE total_profit DECIMAL(10,2);

SELECT SUM(r.Commission)
INTO total_revenue
FROM Revenue r
JOIN Orders o ON r.Orders_ID = o.ID
WHERE DATE_FORMAT(r.Date, '%Y-%m') = report_month;

SELECT SUM(c.DeliveryCost + c.ReturnFraud)
INTO total_cost
FROM Cost c
JOIN Orders o ON c.Order_ID = o.ID
JOIN Order_details od ON o.Order_detail_ID = od.ID
WHERE DATE_FORMAT(od.DeliveryDate, '%Y-%m') = report_month;

SET total_profit = total_revenue - total_cost;

SELECT report_month AS Month,
total_revenue AS Total_Revenue,
total_cost AS Total_Cost,
total_profit AS Total_Profit;

END $$
DELIMITER ;
```

```
1 •  CALL monthly_report('2012-11');
```

Month	Total_Revenue	Total_Cost	Total_Profit
2012-11	154.00	19.78	134.22

Benefit: Improved reusability, performance, and maintainability.

Business Value:

- Customers receive relevant product suggestions
- Vendors optimize inventory and promotions

Key Skills Demonstrated

- Relational Database Design (3NF)
- Advanced SQL Analytics
- Data Integrity & Automation
- Business-Oriented Problem Solving
- Performance & Scalability Awareness

Conclusion

This project demonstrates the complete lifecycle of a **data analytics solution**—from database design to business insight generation.

It reflects real-world analytical work in e-commerce.