QUERIES AND NORMAL-TESTING IN MYSQL

Group 5

Nidhi Gupta - 055006

Harshit Badgamia – 055011

1. Project Information

This project focuses on creating an ecommerce database with dummy data to run queries to assess database functionality and conducting stress testing to check if the database is in 1st Normal form(1NF) or not. The queries include data insertion, retrieval, updates, deletions, aggregations and complex joins to simulate real-world operations. The database details are in the ERD report.

Tables in our database are:

- Product
- Category
- Inventory
- Warehouse
- Customer
- Address
- Order
- OrderItem
- Payment
- Shipping
- Coupon
- Discount
- Review

Stress testing will be conducted to check if the database is in 1NF or not.

2. Project Objectives

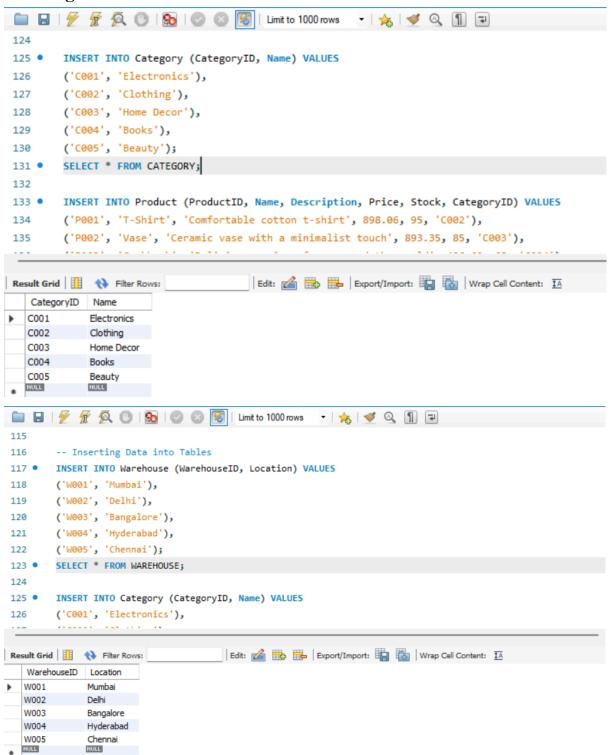
- Database Design and Implementation: Create a relational database for an e-commerce system with well-defined tables, relationships, and constraints to ensure data integrity and consistency.
- Data Insertion and Realism: Populate the database with realistic and consistent data, including product names, addresses, and customer information, while maintaining proper foreign key relationships.
- Data Retrieval and Analysis: Execute complex SQL queries to perform data retrieval, aggregation, and analysis, generating insights into sales, customer behavior, and inventory management.
- **Normalization and Integrity Check:** Verify that the database adheres to the First Normal Form (1NF) to eliminate redundancy and maintain atomicity of data.

3. SQL Queries

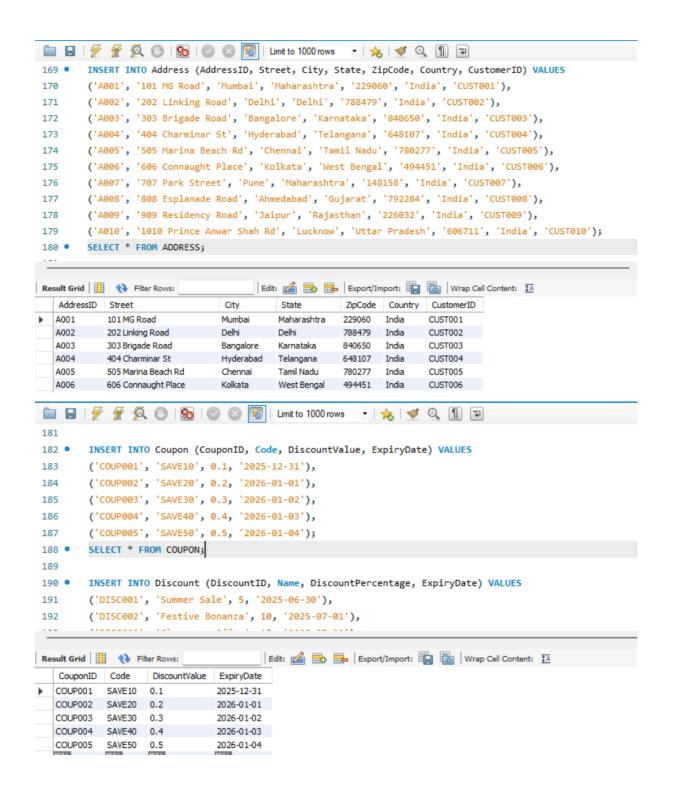
The performed queries can be categorized into Create, Retrieve, Update, Delete (CRUD operations) and Aggregation.

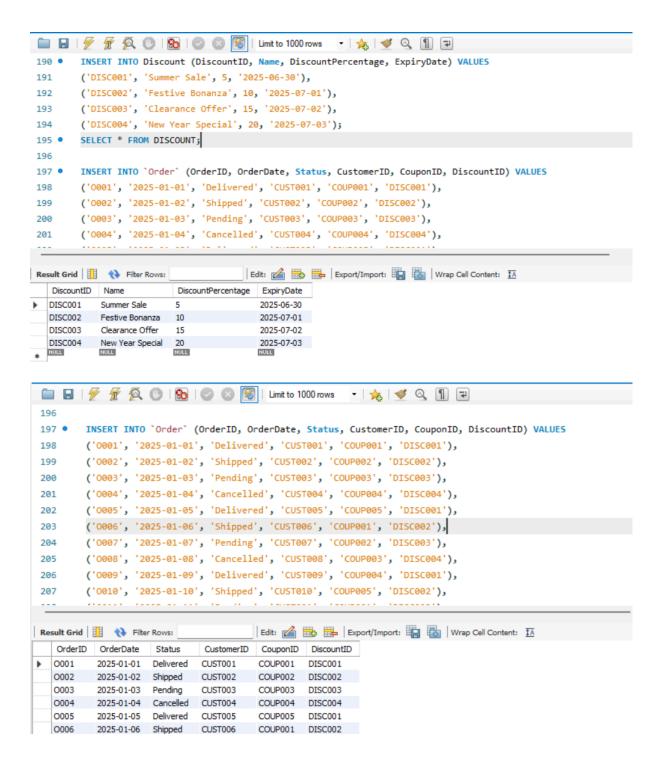
Create (Data Insertion)

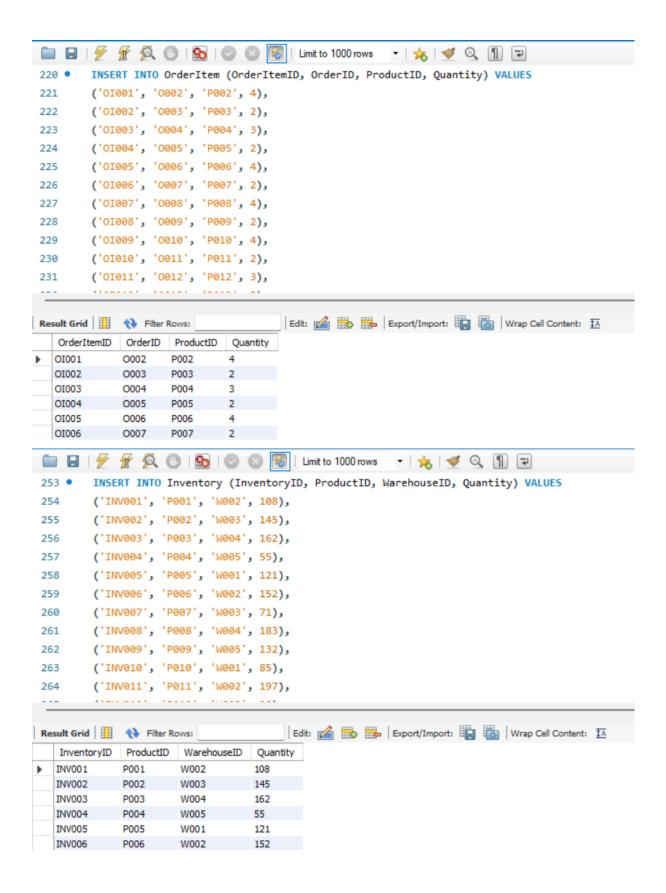
A. Inserting data into tables

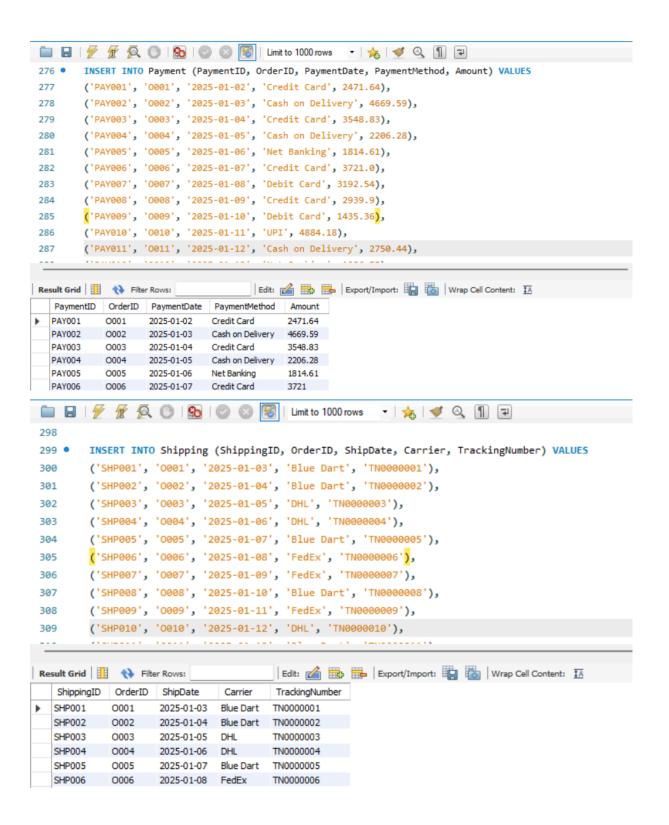


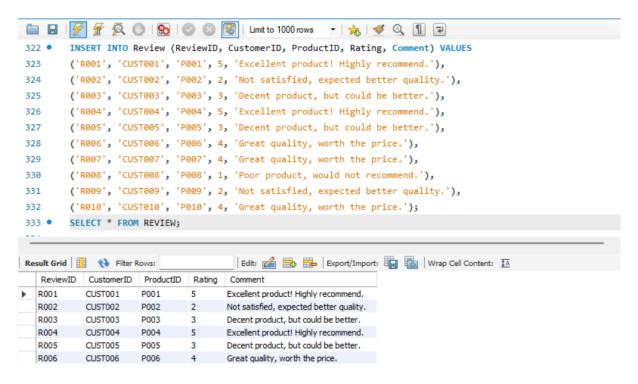








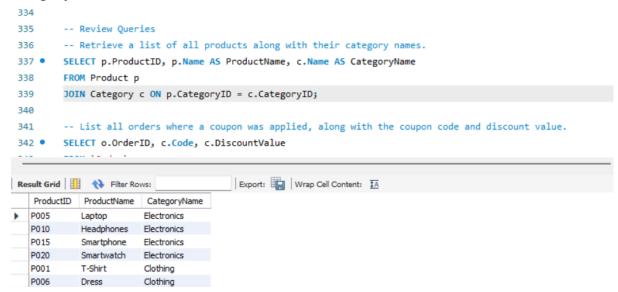




Retrieve

A. Retrieve all Products

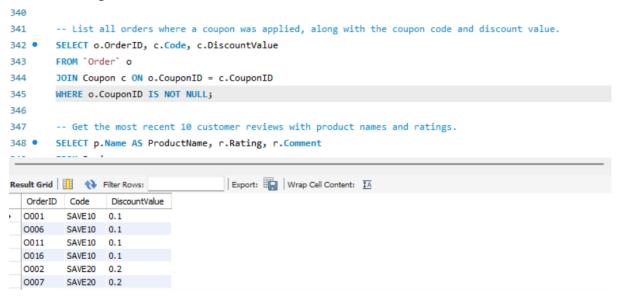
Problem Statement: Retrieve a list of all products along with their category names.



B. Orders with Applied Coupons

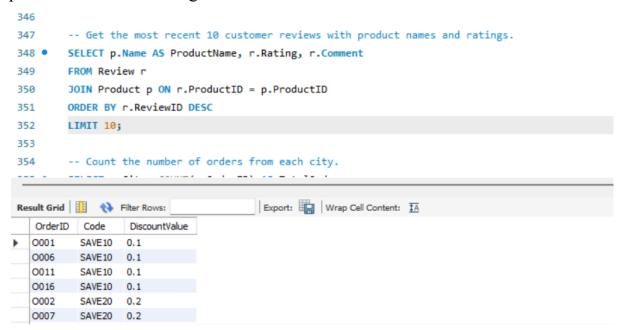
Problem Statement: List all orders where a coupon was applied, along

with the coupon code and discount value.



C. Recent Customer Reviews

Problem Statement: Get the most recent 10 customer reviews with product names and ratings.



Aggregation

A. Top 5 Best-Selling Products

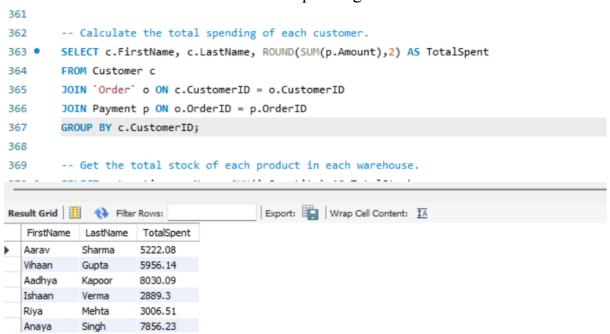
Problem Statement: Find the top 5 products based on the total quantity

```
sold.
353
354
         -- Find the top 5 products based on the total quantity sold.
355 •
         SELECT p.Name, SUM(oi.Quantity) AS TotalSold
         FROM OrderItem oi
356
         JOIN Product p ON oi.ProductID = p.ProductID
357
        GROUP BY p.Name
358
         ORDER BY TotalSold DESC
359
         LIMIT 5;
360
361
         -- Calculate the total spending of each customer.
         SELECT c.FirstName, c.LastName, SUM(p.Amount) AS TotalSpent
Export: Wrap Cell Content: TA Fetch rows:
   Name
             TotalSold
   T-Shirt
             11
   Vase
            11
   Textbook
   Moisturizer 8
```

B. Customer Spending

Dress

Problem Statement: Calculate the total spending of each customer.



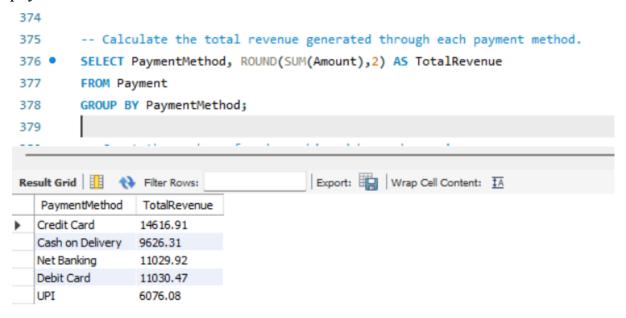
C. Average Rating of Each Product

Problem Statement: Calculate the average rating for each product.

```
-- Calculate the average rating for each product.
        SELECT p.Name, AVG(r.Rating) AS AverageRating
371
        FROM Product p
        JOIN Review r ON p.ProductID = r.ProductID
372
373
        GROUP BY p.Name;
Export: Wrap Cell Content: IA
  Name
              AverageRating
  T-Shirt
             5.0000
             3.0000
  Vase
  Cookbook
             3.0000
  Moisturizer
             5.0000
  Laptop
             3.0000
  Dress
             4.0000
```

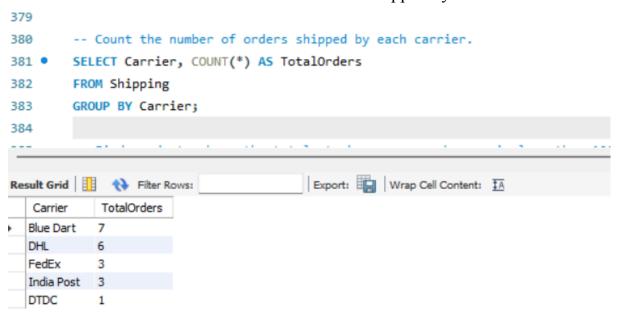
D. Revenue by Payment Method

Problem Statement: Calculate the total revenue generated through each payment method.



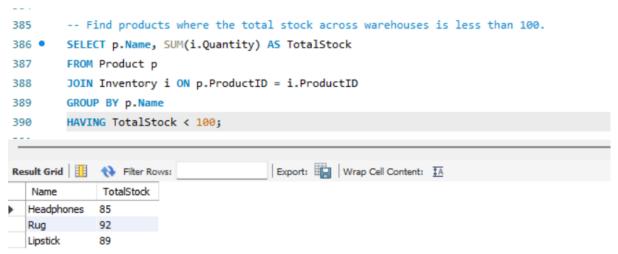
E. Orders Shipped by Each Carrier

Problem Statement: Count the number of orders shipped by each carrier



F. Products Below Reorder Level

Problem Statement: Find products where the total stock across warehouses is less than 100



G. Customer with Maximum Orders

Problem Statement: Find the customer who placed the maximum number of orders.

```
391
392
        -- Find the customer who placed the maximum number of orders.
        SELECT c.FirstName, c.LastName, COUNT(o.OrderID) AS TotalOrders
393 •
394
        FROM Customer c
        JOIN 'Order' o ON c.CustomerID = o.CustomerID
395
396
        GROUP BY c.CustomerID
        ORDER BY TotalOrders DESC
397
        LIMIT 1;
398
399
                                                                                □◊
                                       Export: Wrap Cell Content: TA Fetch rows:
FirstName
           LastName TotalOrders
 Aarav
           Sharma
```

H. Average Order Value

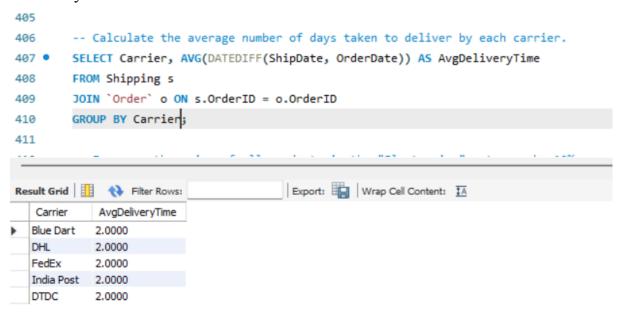
Problem Statement: Calculate the average payment amount for all completed orders.

```
399
         -- Calculate the average payment amount for all completed orders.
400
         SELECT ROUND(AVG(p.Amount),2) AS AverageOrderValue
401 •
         FROM Payment p
402
         JOIN 'Order' o ON p.OrderID = o.OrderID
403
         WHERE o.Status = 'Delivered';
494
405
Result Grid Filter Rows:
                                         Export: Wrap Cell Content: TA
   AverageOrderValue
2424.84
```

I. Average Delivery Time by each Carrier

Problem Statement: Calculate the average number of days taken to

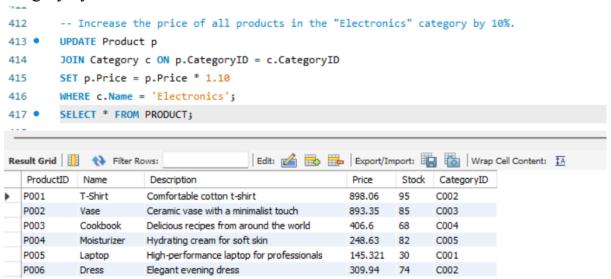
deliver by each carrier.



Update

A. Increase Price

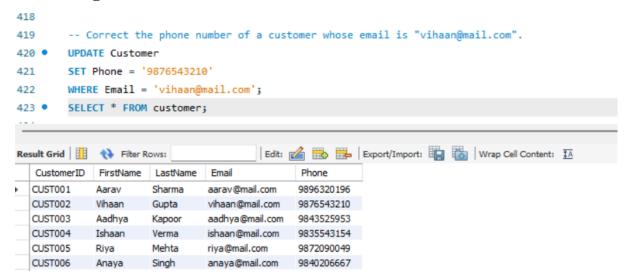
Problem Statement: Increase the price of all products in the "Electronics" category by 10%



B. Correct Phone Number

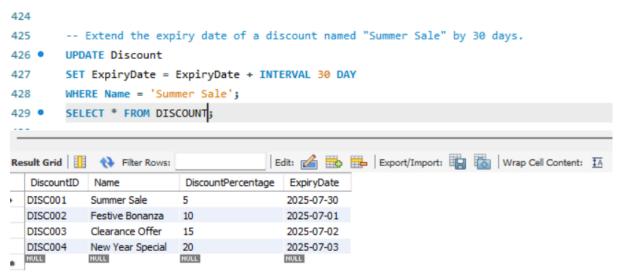
Problem Statement: Correct the phone number of a customer whose email

is "vihaan@mail.com".



C. Extend Discount Expiry Date

Problem Statement: Extend the expiry date of a discount named "Summer Sale" by 30 days.



Delete

A. Delete Poor Reviews

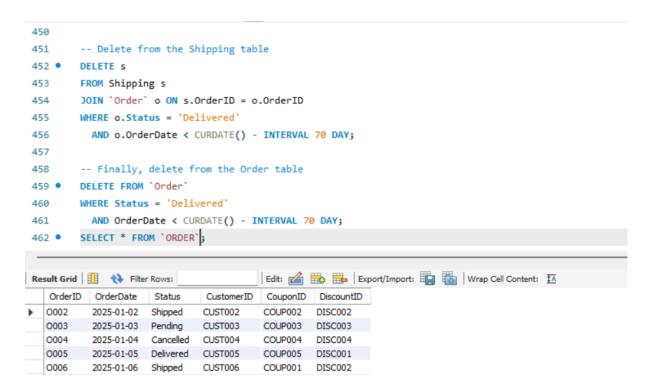
Problem Statement: Delete all reviews with a rating of 1.

```
431
        -- Delete all reviews with a rating of 1.
432 •
        DELETE FROM Review
        WHERE Rating = 1;
        SELECT * FROM REVIEW;
434
435
                                         | Edit: 🚄 📆 🖶 | Export/Import: 🏣 👸 | Wrap Cell Content: 🖽
ReviewID CustomerID ProductID Rating Comment
        CUST004 P004 5 Excellent product! Highly recommend.
  R005
           CUST005
                      P005
                                3
                                       Decent product, but could be better.
                      P006 4
           CUST006
  R006
                                    Great quality, worth the price.
  R007
           CUST007
                      P007
                                       Great quality, worth the price.
  R009
           CUST009
                      P009
                                2
                                       Not satisfied, expected better quality.
  R010
           CUST010
                      P010
                                       Great quality, worth the price.
```

B. Remove old completed orders

Problem Statement: Remove all completed orders that are older than 70 Days.

```
-- Remove all completed orders that are older than 70 DAYS.
436
        -- Delete from the child table (OrderItem) first
437
        DELETE oi
438
        FROM OrderItem oi
439
440
        JOIN 'Order' o ON oi.OrderID = o.OrderID
        WHERE o.Status = 'Delivered'
441
          AND o.OrderDate < CURDATE() - INTERVAL 70 DAY;
442
443
        -- Delete from the Payment table
444
445
        DELETE p
446
        FROM Payment p
        JOIN `Order` o ON p.OrderID = o.OrderID
447
        WHERE o.Status = 'Delivered'
448
449
          AND o.OrderDate < CURDATE() - INTERVAL 70 DAY;
450
```



Note: In order to delete an entry from the order table, first we need to remove entries associated with that specific OrderID Foreign Key in child tables that are OrderItem, Payment and Shipping.

4. Stress Testing(Normal Testing)

A. Atomicity Check

B. Check for Duplicate Values

```
-- Check for duplicate rows in each table
472
473 •
      SELECT 'Warehouse' AS TableName, COUNT(*) AS Total_Rows, COUNT(DISTINCT CONCAT_NS(',', WarehouseID, Location)) AS Unique_Rows
475
       UNION ALL
       SELECT 'Category', COUNT(*), COUNT(DISTINCT CONCAT WS(',', CategoryID, Name))
476
       FROM Category
477
       SELECT 'Product', COUNT(*), COUNT(DISTINCT CONCAT_WS(',', ProductID, Name, Description, Price, Stock, CategoryID))
      FROM Product
480
       UNTON ALL
Export: Wrap Cell Content: IA
  TableName
            Total_Rows Unique_Rows
Warehouse
  Category 5 5
  Product
  Inventory 20 20
  Address 10 10
  Coupon
  Discount 4 4
  Order
  OrderItem 29 29
  Shipping 19
```

C. Check for Inconsistent Data Types

5. Observation and Findings

Data Integrity Checks:

- All tables were tested for duplicate rows using a one-table approach.
- The results confirmed that **total row counts matched unique row counts** for every table, indicating the **absence of duplicate records**.

Normalization Check (1NF):

- The database follows the **First Normal Form (1NF)** principles:
 - Each table has a **primary key** that uniquely identifies each record.
 - All columns contain atomic values with no repeating groups or arrays.

 Columns contain data of a single type and hold only one value per field.

Data Consistency and Referential Integrity:

- Foreign key relationships were tested and verified to maintain **referential integrity**.
- **Update and delete operations** were performed safely without violating foreign key constraints.
- Proper cascading of changes was observed, ensuring that child records were correctly updated or deleted when the parent record was modified or removed.

CRUD Operations:

- The database was tested for **Create**, **Read**, **Update**, **and Delete** operations, including aggregation queries.
- All operations executed successfully without causing errors or data inconsistencies.

6. Conclusion

The database has been thoroughly analyzed and tested to ensure adherence to First Normal Form (1NF). All data is organized with atomic values, and no repeating groups or redundancies were detected. The referential integrity and data consistency are maintained across all tables, ensuring a reliable and efficient data structure.

The database is well-structured and normalized to 1NF, meeting all the requirements for data integrity and consistency.