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

```sql
INSERT INTO Category (CategoryID, Name) VALUES
('C001', 'Electronics'),
('C002', 'Clothing'),
('C003', 'Home Decor'),
('C004', 'Books'),
('C005', 'Beauty');
SELECT * FROM CATEGORY;

INSERT INTO Product (ProductID, Name, Description, Price, Stock, CategoryID) VALUES
('P001', 'T-Shirt', 'Comfortable cotton t-shirt', 898.06, 95, 'C002'),
('P002', 'Vase', 'Ceramic vase with a minimalist touch', 893.35, 85, 'C003'),
```

CategoryID	Name
C001	Electronics
C002	Clothing
C003	Home Decor
C004	Books
C005	Beauty
NULL	NULL

```sql
-- Inserting Data into Tables
INSERT INTO Warehouse (WarehouseID, Location) VALUES
('W001', 'Mumbai'),
('W002', 'Delhi'),
('W003', 'Bangalore'),
('W004', 'Hyderabad'),
('W005', 'Chennai');
SELECT * FROM WAREHOUSE;

INSERT INTO Category (CategoryID, Name) VALUES
('C001', 'Electronics'),
```

WarehouseID	Location
W001	Mumbai
W002	Delhi
W003	Bangalore
W004	Hyderabad
W005	Chennai
NULL	NULL

```sql
INSERT INTO Product (ProductID, Name, Description, Price, Stock, CategoryID) VALUES
('P001', 'T-Shirt', 'Comfortable cotton t-shirt', 898.06, 95, 'C002'),
('P002', 'Vase', 'Ceramic vase with a minimalist touch', 893.35, 85, 'C003'),
('P003', 'Cookbook', 'Delicious recipes from around the world', 406.60, 68, 'C004'),
('P004', 'Moisturizer', 'Hydrating cream for soft skin', 248.63, 82, 'C005'),
('P005', 'Laptop', 'High-performance laptop for professionals', 132.11, 30, 'C001'),
('P006', 'Dress', 'Elegant evening dress', 309.94, 74, 'C002'),
('P007', 'Vase', 'Ceramic vase with a minimalist touch', 534.04, 12, 'C003'),
('P008', 'Textbook', 'Comprehensive material for academic studies', 982.40, 69, 'C004'),
('P009', 'Perfume', 'Long-lasting fragrance for daily use', 621.31, 28, 'C005'),
('P010', 'Headphones', 'Noise-cancelling over-ear headphones', 840.63, 70, 'C001'),
('P011', 'T-Shirt', 'Comfortable cotton t-shirt', 833.93, 69, 'C002'),
```

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

ProductID	Name	Description	Price	Stock	CategoryID
P001	T-Shirt	Comfortable cotton t-shirt	898.06	95	C002
P002	Vase	Ceramic vase with a minimalist touch	893.35	85	C003
P003	Cookbook	Delicious recipes from around the world	406.6	68	C004
P004	Moisturizer	Hydrating cream for soft skin	248.63	82	C005
P005	Laptop	High-performance laptop for professionals	132.11	30	C001
P006	Dress	Elegant evening dress	309.94	74	C002

```sql
SELECT * FROM PRODUCT;

INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone) VALUES
('CUST001', 'Aarav', 'Sharma', 'aarav@mail.com', '9896320196'),
('CUST002', 'Vihaan', 'Gupta', 'vihaan@mail.com', '9839905657'),
('CUST003', 'Aadhya', 'Kapoor', 'aadhya@mail.com', '9843525953'),
('CUST004', 'Ishaan', 'Verma', 'ishaan@mail.com', '9835543154'),
('CUST005', 'Riya', 'Mehta', 'riya@mail.com', '9872090049'),
('CUST006', 'Anaya', 'Singh', 'anaya@mail.com', '9840206667'),
('CUST007', 'Arjun', 'Rao', 'arjun@mail.com', '9835841261'),
('CUST008', 'Diya', 'Patel', 'diya@mail.com', '9894174252'),
('CUST009', 'Kabir', 'Chawla', 'kabir@mail.com', '9815388887'),
```

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

CustomerID	FirstName	LastName	Email	Phone
CUST001	Aarav	Sharma	aarav@mail.com	9896320196
CUST002	Vihaan	Gupta	vihaan@mail.com	9839905657
CUST003	Aadhya	Kapoor	aadhya@mail.com	9843525953
CUST004	Ishaan	Verma	ishaan@mail.com	9835543154
CUST005	Riya	Mehta	riya@mail.com	9872090049
CUST006	Anaya	Singh	anaya@mail.com	9840206667

```sql
INSERT INTO Address (AddressID, Street, City, State, ZipCode, Country, CustomerID) VALUES
('A001', '101 MG Road', 'Mumbai', 'Maharashtra', '229060', 'India', 'CUST001'),
('A002', '202 Linking Road', 'Delhi', 'Delhi', '788479', 'India', 'CUST002'),
('A003', '303 Brigade Road', 'Bangalore', 'Karnataka', '840650', 'India', 'CUST003'),
('A004', '404 Charminar St', 'Hyderabad', 'Telangana', '648107', 'India', 'CUST004'),
('A005', '505 Marina Beach Rd', 'Chennai', 'Tamil Nadu', '780277', 'India', 'CUST005'),
('A006', '606 Connaught Place', 'Kolkata', 'West Bengal', '494451', 'India', 'CUST006'),
('A007', '707 Park Street', 'Pune', 'Maharashtra', '148158', 'India', 'CUST007'),
('A008', '808 Esplanade Road', 'Ahmedabad', 'Gujarat', '792284', 'India', 'CUST008'),
('A009', '909 Residency Road', 'Jaipur', 'Rajasthan', '226032', 'India', 'CUST009'),
('A010', '1010 Prince Anwar Shah Rd', 'Lucknow', 'Uttar Pradesh', '606711', 'India', 'CUST010');
SELECT * FROM ADDRESS;
```

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

AddressID	Street	City	State	ZipCode	Country	CustomerID
A001	101 MG Road	Mumbai	Maharashtra	229060	India	CUST001
A002	202 Linking Road	Delhi	Delhi	788479	India	CUST002
A003	303 Brigade Road	Bangalore	Karnataka	840650	India	CUST003
A004	404 Charminar St	Hyderabad	Telangana	648107	India	CUST004
A005	505 Marina Beach Rd	Chennai	Tamil Nadu	780277	India	CUST005
A006	606 Connaught Place	Kolkata	West Bengal	494451	India	CUST006

```sql

INSERT INTO Coupon (CouponID, Code, DiscountValue, ExpiryDate) VALUES
('COUP001', 'SAVE10', 0.1, '2025-12-31'),
('COUP002', 'SAVE20', 0.2, '2026-01-01'),
('COUP003', 'SAVE30', 0.3, '2026-01-02'),
('COUP004', 'SAVE40', 0.4, '2026-01-03'),
('COUP005', 'SAVE50', 0.5, '2026-01-04');
SELECT * FROM COUPON;

INSERT INTO Discount (DiscountID, Name, DiscountPercentage, ExpiryDate) VALUES
('DISC001', 'Summer Sale', 5, '2025-06-30'),
('DISC002', 'Festive Bonanza', 10, '2025-07-01'),
```

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

CouponID	Code	DiscountValue	ExpiryDate
COUP001	SAVE10	0.1	2025-12-31
COUP002	SAVE20	0.2	2026-01-01
COUP003	SAVE30	0.3	2026-01-02
COUP004	SAVE40	0.4	2026-01-03
COUP005	SAVE50	0.5	2026-01-04

```sql
INSERT INTO Discount (DiscountID, Name, DiscountPercentage, ExpiryDate) VALUES
('DISC001', 'Summer Sale', 5, '2025-06-30'),
('DISC002', 'Festive Bonanza', 10, '2025-07-01'),
('DISC003', 'Clearance Offer', 15, '2025-07-02'),
('DISC004', 'New Year Special', 20, '2025-07-03');
SELECT * FROM DISCOUNT;

INSERT INTO `Order` (OrderID, OrderDate, Status, CustomerID, CouponID, DiscountID) VALUES
('O001', '2025-01-01', 'Delivered', 'CUST001', 'COUP001', 'DISC001'),
('O002', '2025-01-02', 'Shipped', 'CUST002', 'COUP002', 'DISC002'),
('O003', '2025-01-03', 'Pending', 'CUST003', 'COUP003', 'DISC003'),
('O004', '2025-01-04', 'Cancelled', 'CUST004', 'COUP004', 'DISC004'),
```

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

DiscountID	Name	DiscountPercentage	ExpiryDate
DISC001	Summer Sale	5	2025-06-30
DISC002	Festive Bonanza	10	2025-07-01
DISC003	Clearance Offer	15	2025-07-02
DISC004	New Year Special	20	2025-07-03
NULL	NULL	NULL	NULL

```sql
INSERT INTO `Order` (OrderID, OrderDate, Status, CustomerID, CouponID, DiscountID) VALUES
('O001', '2025-01-01', 'Delivered', 'CUST001', 'COUP001', 'DISC001'),
('O002', '2025-01-02', 'Shipped', 'CUST002', 'COUP002', 'DISC002'),
('O003', '2025-01-03', 'Pending', 'CUST003', 'COUP003', 'DISC003'),
('O004', '2025-01-04', 'Cancelled', 'CUST004', 'COUP004', 'DISC004'),
('O005', '2025-01-05', 'Delivered', 'CUST005', 'COUP005', 'DISC001'),
('O006', '2025-01-06', 'Shipped', 'CUST006', 'COUP001', 'DISC002'),
('O007', '2025-01-07', 'Pending', 'CUST007', 'COUP002', 'DISC003'),
('O008', '2025-01-08', 'Cancelled', 'CUST008', 'COUP003', 'DISC004'),
('O009', '2025-01-09', 'Delivered', 'CUST009', 'COUP004', 'DISC001'),
('O010', '2025-01-10', 'Shipped', 'CUST010', 'COUP005', 'DISC002'),
```

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

OrderID	OrderDate	Status	CustomerID	CouponID	DiscountID
O001	2025-01-01	Delivered	CUST001	COUP001	DISC001
O002	2025-01-02	Shipped	CUST002	COUP002	DISC002
O003	2025-01-03	Pending	CUST003	COUP003	DISC003
O004	2025-01-04	Cancelled	CUST004	COUP004	DISC004
O005	2025-01-05	Delivered	CUST005	COUP005	DISC001
O006	2025-01-06	Shipped	CUST006	COUP001	DISC002

```sql
INSERT INTO OrderItem (OrderItemID, OrderID, ProductID, Quantity) VALUES
('OI001', 'O002', 'P002', 4),
('OI002', 'O003', 'P003', 2),
('OI003', 'O004', 'P004', 3),
('OI004', 'O005', 'P005', 2),
('OI005', 'O006', 'P006', 4),
('OI006', 'O007', 'P007', 2),
('OI007', 'O008', 'P008', 4),
('OI008', 'O009', 'P009', 2),
('OI009', 'O010', 'P010', 4),
('OI010', 'O011', 'P011', 2),
('OI011', 'O012', 'P012', 3),
```

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

OrderItemID	OrderID	ProductID	Quantity
OI001	O002	P002	4
OI002	O003	P003	2
OI003	O004	P004	3
OI004	O005	P005	2
OI005	O006	P006	4
OI006	O007	P007	2

```sql
INSERT INTO Inventory (InventoryID, ProductID, WarehouseID, Quantity) VALUES
('INV001', 'P001', 'W002', 108),
('INV002', 'P002', 'W003', 145),
('INV003', 'P003', 'W004', 162),
('INV004', 'P004', 'W005', 55),
('INV005', 'P005', 'W001', 121),
('INV006', 'P006', 'W002', 152),
('INV007', 'P007', 'W003', 71),
('INV008', 'P008', 'W004', 183),
('INV009', 'P009', 'W005', 132),
('INV010', 'P010', 'W001', 85),
('INV011', 'P011', 'W002', 197),
```

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

InventoryID	ProductID	WarehouseID	Quantity
INV001	P001	W002	108
INV002	P002	W003	145
INV003	P003	W004	162
INV004	P004	W005	55
INV005	P005	W001	121
INV006	P006	W002	152

```sql
INSERT INTO Payment (PaymentID, OrderID, PaymentDate, PaymentMethod, Amount) VALUES
('PAY001', 'O001', '2025-01-02', 'Credit Card', 2471.64),
('PAY002', 'O002', '2025-01-03', 'Cash on Delivery', 4669.59),
('PAY003', 'O003', '2025-01-04', 'Credit Card', 3548.83),
('PAY004', 'O004', '2025-01-05', 'Cash on Delivery', 2206.28),
('PAY005', 'O005', '2025-01-06', 'Net Banking', 1814.61),
('PAY006', 'O006', '2025-01-07', 'Credit Card', 3721.0),
('PAY007', 'O007', '2025-01-08', 'Debit Card', 3192.54),
('PAY008', 'O008', '2025-01-09', 'Credit Card', 2939.9),
('PAY009', 'O009', '2025-01-10', 'Debit Card', 1435.36),
('PAY010', 'O010', '2025-01-11', 'UPI', 4884.18),
('PAY011', 'O011', '2025-01-12', 'Cash on Delivery', 2750.44),
```

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

PaymentID	OrderID	PaymentDate	PaymentMethod	Amount
PAY001	O001	2025-01-02	Credit Card	2471.64
PAY002	O002	2025-01-03	Cash on Delivery	4669.59
PAY003	O003	2025-01-04	Credit Card	3548.83
PAY004	O004	2025-01-05	Cash on Delivery	2206.28
PAY005	O005	2025-01-06	Net Banking	1814.61
PAY006	O006	2025-01-07	Credit Card	3721

```sql

INSERT INTO Shipping (ShippingID, OrderID, ShipDate, Carrier, TrackingNumber) VALUES
('SHP001', 'O001', '2025-01-03', 'Blue Dart', 'TN0000001'),
('SHP002', 'O002', '2025-01-04', 'Blue Dart', 'TN0000002'),
('SHP003', 'O003', '2025-01-05', 'DHL', 'TN0000003'),
('SHP004', 'O004', '2025-01-06', 'DHL', 'TN0000004'),
('SHP005', 'O005', '2025-01-07', 'Blue Dart', 'TN0000005'),
('SHP006', 'O006', '2025-01-08', 'FedEx', 'TN0000006'),
('SHP007', 'O007', '2025-01-09', 'FedEx', 'TN0000007'),
('SHP008', 'O008', '2025-01-10', 'Blue Dart', 'TN0000008'),
('SHP009', 'O009', '2025-01-11', 'FedEx', 'TN0000009'),
('SHP010', 'O010', '2025-01-12', 'DHL', 'TN0000010'),
```

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

ShippingID	OrderID	ShipDate	Carrier	TrackingNumber
SHP001	O001	2025-01-03	Blue Dart	TN0000001
SHP002	O002	2025-01-04	Blue Dart	TN0000002
SHP003	O003	2025-01-05	DHL	TN0000003
SHP004	O004	2025-01-06	DHL	TN0000004
SHP005	O005	2025-01-07	Blue Dart	TN0000005
SHP006	O006	2025-01-08	FedEx	TN0000006

```sql
INSERT INTO Review (ReviewID, CustomerID, ProductID, Rating, Comment) VALUES
('R001', 'CUST001', 'P001', 5, 'Excellent product! Highly recommend.'),
('R002', 'CUST002', 'P002', 2, 'Not satisfied, expected better quality.'),
('R003', 'CUST003', 'P003', 3, 'Decent product, but could be better.'),
('R004', 'CUST004', 'P004', 5, 'Excellent product! Highly recommend.'),
('R005', 'CUST005', 'P005', 3, 'Decent product, but could be better.'),
('R006', 'CUST006', 'P006', 4, 'Great quality, worth the price.'),
('R007', 'CUST007', 'P007', 4, 'Great quality, worth the price.'),
('R008', 'CUST008', 'P008', 1, 'Poor product, would not recommend.'),
('R009', 'CUST009', 'P009', 2, 'Not satisfied, expected better quality.'),
('R010', 'CUST010', 'P010', 4, 'Great quality, worth the price.');
SELECT * FROM REVIEW;
```

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

ReviewID	CustomerID	ProductID	Rating	Comment
R001	CUST001	P001	5	Excellent product! Highly recommend.
R002	CUST002	P002	2	Not satisfied, expected better quality.
R003	CUST003	P003	3	Decent product, but could be better.
R004	CUST004	P004	5	Excellent product! Highly recommend.
R005	CUST005	P005	3	Decent product, but could be better.
R006	CUST006	P006	4	Great quality, worth the price.

Retrieve

A. Retrieve all Products

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

```sql

-- Review Queries
-- Retrieve a list of all products along with their category names.
SELECT p.ProductID, p.Name AS ProductName, c.Name AS CategoryName
FROM Product p
JOIN Category c ON p.CategoryID = c.CategoryID;

-- List all orders where a coupon was applied, along with the coupon code and discount value.
SELECT o.OrderID, c.Code, c.DiscountValue
```

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

ProductID	ProductName	CategoryName
P005	Laptop	Electronics
P010	Headphones	Electronics
P015	Smartphone	Electronics
P020	Smartwatch	Electronics
P001	T-Shirt	Clothing
P006	Dress	Clothing

B. Orders with Applied Coupons

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

with the coupon code and discount value.

```
-- List all orders where a coupon was applied, along with the coupon code and discount value.
SELECT o.OrderID, c.Code, c.DiscountValue
FROM `Order` o
JOIN Coupon c ON o.CouponID = c.CouponID
WHERE o.CouponID IS NOT NULL;

-- Get the most recent 10 customer reviews with product names and ratings.
SELECT p.Name AS ProductName, r.Rating, r.Comment
```

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

OrderID	Code	DiscountValue
O001	SAVE10	0.1
O006	SAVE10	0.1
O011	SAVE10	0.1
O016	SAVE10	0.1
O002	SAVE20	0.2
O007	SAVE20	0.2

C. Recent Customer Reviews

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

```
-- Get the most recent 10 customer reviews with product names and ratings.
SELECT p.Name AS ProductName, r.Rating, r.Comment
FROM Review r
JOIN Product p ON r.ProductID = p.ProductID
ORDER BY r.ReviewID DESC
LIMIT 10;

-- Count the number of orders from each city.
```

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

OrderID	Code	DiscountValue
O001	SAVE10	0.1
O006	SAVE10	0.1
O011	SAVE10	0.1
O016	SAVE10	0.1
O002	SAVE20	0.2
O007	SAVE20	0.2

Aggregation

A. Top 5 Best-Selling Products

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

sold.

```
-- Find the top 5 products based on the total quantity sold.
SELECT p.Name, SUM(oi.Quantity) AS TotalSold
FROM OrderItem oi
JOIN Product p ON oi.ProductID = p.ProductID
GROUP BY p.Name
ORDER BY TotalSold DESC
LIMIT 5;

-- Calculate the total spending of each customer.
SELECT c.FirstName, c.LastName, SUM(p.Amount) AS TotalSpent
```

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

Name	TotalSold
T-Shirt	11
Vase	11
Textbook	9
Moisturizer	8
Dress	7

B. Customer Spending

Problem Statement: Calculate the total spending of each customer.

```
-- Calculate the total spending of each customer.
SELECT c.FirstName, c.LastName, ROUND(SUM(p.Amount),2) AS TotalSpent
FROM Customer c
JOIN `Order` o ON c.CustomerID = o.CustomerID
JOIN Payment p ON o.OrderID = p.OrderID
GROUP BY c.CustomerID;

-- Get the total stock of each product in each warehouse.
```

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

FirstName	LastName	TotalSpent
Aarav	Sharma	5222.08
Vihaan	Gupta	5956.14
Aadhya	Kapoor	8030.09
Ishaan	Verma	2889.3
Riya	Mehta	3006.51
Anaya	Singh	7856.23

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
FROM Product p
JOIN Review r ON p.ProductID = r.ProductID
GROUP BY p.Name;
```

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

Name	AverageRating
T-Shirt	5.0000
Vase	3.0000
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.

```
-- Calculate the total revenue generated through each payment method.
SELECT PaymentMethod, ROUND(SUM(Amount),2) AS TotalRevenue
FROM Payment
GROUP BY PaymentMethod;
```

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

PaymentMethod	TotalRevenue
Credit Card	14616.91
Cash on Delivery	9626.31
Net Banking	11029.92
Debit Card	11030.47
UPI	6076.08

E. Orders Shipped by Each Carrier

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

```sql
-- Count the number of orders shipped by each carrier.
SELECT Carrier, COUNT(*) AS TotalOrders
FROM Shipping
GROUP BY Carrier;
```

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

Carrier	TotalOrders
Blue Dart	7
DHL	6
FedEx	3
India Post	3
DTDC	1

F. Products Below Reorder Level

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

```sql
-- Find products where the total stock across warehouses is less than 100.
SELECT p.Name, SUM(i.Quantity) AS TotalStock
FROM Product p
JOIN Inventory i ON p.ProductID = i.ProductID
GROUP BY p.Name
HAVING TotalStock < 100;
```

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

Name	TotalStock
Headphones	85
Rug	92
Lipstick	89

G. Customer with Maximum Orders

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

```
-- Find the customer who placed the maximum number of orders.
SELECT c.FirstName, c.LastName, COUNT(o.OrderID) AS TotalOrders
FROM Customer c
JOIN `Order` o ON c.CustomerID = o.CustomerID
GROUP BY c.CustomerID
ORDER BY TotalOrders DESC
LIMIT 1;
```

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

FirstName	LastName	TotalOrders
Aarav	Sharma	2

H. Average Order Value

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

```
-- Calculate the average payment amount for all completed orders.
SELECT ROUND(AVG(p.Amount),2) AS AverageOrderValue
FROM Payment p
JOIN `Order` o ON p.OrderID = o.OrderID
WHERE o.Status = 'Delivered';
```

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

AverageOrderValue
2424.84

I. Average Delivery Time by each Carrier

Problem Statement: Calculate the average number of days taken to

deliver by each carrier.

```
-- Calculate the average number of days taken to deliver by each carrier.
SELECT Carrier, AVG(DATEDIFF(ShipDate, OrderDate)) AS AvgDeliveryTime
FROM Shipping s
JOIN `Order` o ON s.OrderID = o.OrderID
GROUP BY Carrier;
```

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

Carrier	AvgDeliveryTime
Blue Dart	2.0000
DHL	2.0000
FedEx	2.0000
India Post	2.0000
DTDC	2.0000

Update

A. Increase Price

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

```
-- Increase the price of all products in the "Electronics" category by 10%.
UPDATE Product p
JOIN Category c ON p.CategoryID = c.CategoryID
SET p.Price = p.Price * 1.10
WHERE c.Name = 'Electronics';
SELECT * FROM PRODUCT;
```

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

ProductID	Name	Description	Price	Stock	CategoryID
P001	T-Shirt	Comfortable cotton t-shirt	898.06	95	C002
P002	Vase	Ceramic vase with a minimalist touch	893.35	85	C003
P003	Cookbook	Delicious recipes from around the world	406.6	68	C004
P004	Moisturizer	Hydrating cream for soft skin	248.63	82	C005
P005	Laptop	High-performance laptop for professionals	145.321	30	C001
P006	Dress	Elegant evening dress	309.94	74	C002

B. Correct Phone Number

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

is "vihaan@mail.com".

```
-- Correct the phone number of a customer whose email is "vihaan@mail.com".
UPDATE Customer
SET Phone = '9876543210'
WHERE Email = 'vihaan@mail.com';
SELECT * FROM customer;
```

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

CustomerID	FirstName	LastName	Email	Phone
CUST001	Aarav	Sharma	aarav@mail.com	9896320196
CUST002	Vihaan	Gupta	vihaan@mail.com	9876543210
CUST003	Aadhya	Kapoor	aadhya@mail.com	9843525953
CUST004	Ishaan	Verma	ishaan@mail.com	9835543154
CUST005	Riya	Mehta	riya@mail.com	9872090049
CUST006	Anaya	Singh	anaya@mail.com	9840206667

C. Extend Discount Expiry Date

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

```
-- Extend the expiry date of a discount named "Summer Sale" by 30 days.
UPDATE Discount
SET ExpiryDate = ExpiryDate + INTERVAL 30 DAY
WHERE Name = 'Summer Sale';
SELECT * FROM DISCOUNT;
```

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

DiscountID	Name	DiscountPercentage	ExpiryDate
DISC001	Summer Sale	5	2025-07-30
DISC002	Festive Bonanza	10	2025-07-01
DISC003	Clearance Offer	15	2025-07-02
DISC004	New Year Special	20	2025-07-03
NULL	NULL	NULL	NULL

Delete

A. Delete Poor Reviews

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

```
-- Delete all reviews with a rating of 1.
DELETE FROM Review
WHERE Rating = 1;
SELECT * FROM REVIEW;
```

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

ReviewID	CustomerID	ProductID	Rating	Comment
R004	CUST004	P004	5	Excellent product! Highly recommend.
R005	CUST005	P005	3	Decent product, but could be better.
R006	CUST006	P006	4	Great quality, worth the price.
R007	CUST007	P007	4	Great quality, worth the price.
R009	CUST009	P009	2	Not satisfied, expected better quality.
R010	CUST010	P010	4	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.
-- Delete from the child table (OrderItem) first
DELETE oi
FROM OrderItem oi
JOIN `Order` o ON oi.OrderID = o.OrderID
WHERE o.Status = 'Delivered'
  AND o.OrderDate < CURDATE() - INTERVAL 70 DAY;

-- Delete from the Payment table
DELETE p
FROM Payment p
JOIN `Order` o ON p.OrderID = o.OrderID
WHERE o.Status = 'Delivered'
  AND o.OrderDate < CURDATE() - INTERVAL 70 DAY;
```

```sql
-- Delete from the Shipping table
DELETE s
FROM Shipping s
JOIN `Order` o ON s.OrderID = o.OrderID
WHERE o.Status = 'Delivered'
  AND o.OrderDate < CURDATE() - INTERVAL 70 DAY;

-- Finally, delete from the Order table
DELETE FROM `Order`
WHERE Status = 'Delivered'
  AND OrderDate < CURDATE() - INTERVAL 70 DAY;
SELECT * FROM `ORDER`;
```

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

OrderID	OrderDate	Status	CustomerID	CouponID	DiscountID
O002	2025-01-02	Shipped	CUST002	COUP002	DISC002
O003	2025-01-03	Pending	CUST003	COUP003	DISC003
O004	2025-01-04	Cancelled	CUST004	COUP004	DISC004
O005	2025-01-05	Delivered	CUST005	COUP005	DISC001
O006	2025-01-06	Shipped	CUST006	COUP001	DISC002

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

```sql
-- Normal Testing
-- Check for non-atomic data (columns that may contain multiple values)
SELECT TABLE_NAME, COLUMN_NAME, COLUMN_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_SCHEMA = 'ecommerce'
  AND (COLUMN_TYPE LIKE 'SET%' OR COLUMN_TYPE LIKE 'ENUM%');
```

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

TABLE_NAME	COLUMN_NAME	COLUMN_TYPE

B. Check for Duplicate Values

```sql
-- Check for duplicate rows in each table

SELECT 'Warehouse' AS TableName, COUNT(*) AS Total_Rows, COUNT(DISTINCT CONCAT_WS(',', WarehouseID, Location)) AS Unique_Rows
FROM Warehouse
UNION ALL
SELECT 'Category', COUNT(*), COUNT(DISTINCT CONCAT_WS(',', CategoryID, Name))
FROM Category
UNION ALL
SELECT 'Product', COUNT(*), COUNT(DISTINCT CONCAT_WS(',', ProductID, Name, Description, Price, Stock, CategoryID))
FROM Product
UNION ALL
```

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

TableName	Total_Rows	Unique_Rows
Warehouse	5	5
Category	5	5
Product	20	20
Inventory	20	20
Customer	10	10
Address	10	10
Coupon	5	5
Discount	4	4
Order	19	19
OrderItem	29	29
Payment	19	19
Shipping	19	19
Review	9	9

C. Check for Inconsistent Data Types

```sql
-- Check for inconsistent data types in each table
SELECT TABLE_NAME, COLUMN_NAME, COLUMN_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_SCHEMA = 'ecommerce'
  AND DATA_TYPE NOT IN ('varchar', 'int', 'float', 'text', 'date');
```

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

TABLE_NAME	COLUMN_NAME	COLUMN_TYPE

---

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.