Brief Description of the Database

The database system being implemented is an **E-Commerce Management System** designed to manage and track the core operations of an online retail business. The database integrates various functional aspects, including customer management, product inventory, order processing, payments, shipping, and reviews. It is structured to provide a robust foundation for storing, managing, and analyzing data related to sales and business performance.

Key Features

1. **Customer Management**:
   Stores detailed information about customers, including contact details, demographics, and purchase history. It ensures efficient customer relationship management and personalization of services.

2. **Product and Inventory Management**:
   Tracks products, their categories, pricing, and stock levels. It includes discount and promotion mechanisms to enable dynamic pricing strategies.

3. **Order Processing**:
   Handles orders placed by customers, linking them with associated products and customers. Tracks order statuses and provides a history of purchases.

4. **Payment Tracking**:
   Manages payment transactions, recording details such as amount, method, and status (e.g., completed, pending, or failed).

5. **Shipping and Logistics**:
   Monitors shipping processes, including addresses, shipping dates, delivery statuses, and cancellations, ensuring end-to-end order fulfillment.

6. **Sales Analytics and Reporting**:
   Consolidates data for generating insights into customer behavior, sales trends, and product performance. This supports informed business decisions through reporting tools.

7. **Customer Reviews and Feedback**:
   Enables customers to rate and review products, helping improve product quality and customer satisfaction.

Purpose of the Database

- **Efficiency**: Streamlines operations by automating tasks like inventory tracking, order fulfillment, and payment processing.
- **Data Centralization**: Consolidates all business-critical data in one system for easy access and management.
- **Insight Generation**: Provides reports and analytics to track business performance and identify growth opportunities.
- **Scalability**: Supports future growth of the business by accommodating more customers, products, and transactions.
- **Customer Satisfaction**: Enhances the customer experience through detailed tracking of orders, timely shipping, and responsive feedback mechanisms.

This database system is essential for managing a scalable and customer-centric e-commerce business effectively.

1. Customers Schema

Schema Name: `Customers`
Attributes:

- `CustomerID` (INT, Primary Key, Auto Increment)
- `FirstName` (VARCHAR(50), NOT NULL)
- `LastName` (VARCHAR(50), NOT NULL)
- `Email` (VARCHAR(100), UNIQUE, NOT NULL)
- `Phone` (VARCHAR(15), UNIQUE)
- `Address` (VARCHAR(255))
- `City` (VARCHAR(50))
- `State` (VARCHAR(50))
- `ZipCode` (VARCHAR(15))
- `Country` (VARCHAR(50))
- `Gender` (VARCHAR(15), CHECK (`Gender IN ('Male', 'Female', 'Other')`))
- `Age` (INT, CHECK (`Age >= 0 AND Age <= 120`))
- `CreatedAt` (DATETIME, DEFAULT CURRENT_TIMESTAMP)

---

2. Categories Schema

Schema Name: `Categories`
Attributes:

- `CategoryID` (INT, Primary Key, Auto Increment)
- `CategoryName` (VARCHAR(100), UNIQUE, NOT NULL)
- `Description` (VARCHAR(255))

---

3. Products Schema

Schema Name: `Products`
Attributes:

- `ProductID` (INT, Primary Key, Auto Increment)
- `ProductName` (VARCHAR(100), NOT NULL)
- `CategoryID` (INT, Foreign Key referencing `Categories.CategoryID`, NOT NULL)
- `Price` (DECIMAL(10,2), CHECK (`Price > 0`))
- `Stock` (INT, CHECK (`Stock >= 0`))
- `CreatedAt` (DATETIME, DEFAULT CURRENT_TIMESTAMP)

---

4. Orders Schema

Schema Name: `Orders`
Attributes:

- `OrderID` (INT, Primary Key, Auto Increment)
- `CustomerID` (INT, Foreign Key referencing `Customers.CustomerID`, NOT NULL)
- `OrderDate` (DATETIME, DEFAULT CURRENT_TIMESTAMP)
- `TotalAmount` (DECIMAL(10,2), CHECK (`TotalAmount >= 0`))

---

5. OrderItems Schema

Schema Name: `OrderItems`
Attributes:

- `OrderItemID` (INT, Primary Key, Auto Increment)
- `OrderID` (INT, Foreign Key referencing `Orders.OrderID`, NOT NULL)
- `ProductID` (INT, Foreign Key referencing `Products.ProductID`, NOT NULL)
- `Quantity` (INT, CHECK (`Quantity > 0`))
- `Price` (DECIMAL(10,2), CHECK (`Price > 0`))

---

6. Payments Schema

Schema Name: `Payments`
Attributes:

- `PaymentID` (INT, Primary Key, Auto Increment)

- `OrderID` (INT, Foreign Key referencing `Orders.OrderID`, NOT NULL)
- `PaymentDate` (DATETIME, DEFAULT CURRENT_TIMESTAMP)
- `Amount` (DECIMAL(10,2), CHECK (`Amount > 0`))
- `PaymentMethod` (VARCHAR(50), NOT NULL)
- `PaymentStatus` (VARCHAR(20), CHECK (`PaymentStatus IN ('Pending', 'Completed', 'Failed')`))

---

7. Shipping Schema

Schema Name: `Shipping`
Attributes:

- `ShippingID` (INT, Primary Key, Auto Increment)
- `OrderID` (INT, Foreign Key referencing `Orders.OrderID`, NOT NULL)
- `ShippingAddress` (VARCHAR(255), NOT NULL)
- `ShippingDate` (DATETIME, DEFAULT CURRENT_TIMESTAMP)
- `DeliveryDate` (DATETIME)
- `ShippingStatus` (VARCHAR(50), CHECK (`ShippingStatus IN ('Pending', 'Shipped', 'Delivered', 'Cancelled')`))

---

8. ReviewDetails Schema

Schema Name: `ReviewDetails`
Attributes:

- `ReviewID` (INT, Primary Key, Auto Increment)
- `Rating` (INT, CHECK (`Rating BETWEEN 1 AND 5`))
- `ReviewText` (TEXT)
- `ReviewDate` (DATETIME, DEFAULT CURRENT_TIMESTAMP)

---

9. CustomerProductReviews Schema

Schema Name: `CustomerProductReviews`
Attributes:

- `CustomerID` (INT, Foreign Key referencing `Customers.CustomerID`, NOT NULL)
- `ProductID` (INT, Foreign Key referencing `Products.ProductID`, NOT NULL)
- `ReviewID` (INT, Foreign Key referencing `ReviewDetails.ReviewID`, NOT NULL)
- **Primary Key**: Composite key (`CustomerID`, `ProductID`, `ReviewID`)

---

10. DiscountDetails Schema

Schema Name: `DiscountDetails`
Attributes:

- `DiscountID` (INT, Primary Key, Auto Increment)
- `DiscountPercentage` (DECIMAL(5,2), CHECK (`DiscountPercentage BETWEEN 0 AND 100`))
- `StartDate` (DATETIME, DEFAULT CURRENT_TIMESTAMP)
- `EndDate` (DATETIME, NOT NULL)

---

11. ProductDiscount Schema

Schema Name: `ProductDiscount`
Attributes:

- `ProductID` (INT, Foreign Key referencing `Products.ProductID`, NOT NULL)
- `DiscountID` (INT, Foreign Key referencing `DiscountDetails.DiscountID`, NOT NULL)
- **Primary Key**: Composite key (`ProductID`, `DiscountID`)

---

12. OrderStatusHistory Schema

Schema Name: `OrderStatusHistory`
Attributes:

- `StatusID` (INT, Primary Key, Auto Increment)
- `OrderID` (INT, Foreign Key referencing `Orders.OrderID`, NOT NULL)
- `Status` (VARCHAR(50), NOT NULL)
- `StatusDate` (DATETIME, DEFAULT CURRENT_TIMESTAMP)
- `Notes` (TEXT)

---

13. SalesHistory Schema

Schema Name: `SalesHistory`
Attributes:

- `SaleID` (INT, Primary Key, Auto Increment)

- `OrderID` (INT, Foreign Key referencing `Orders.OrderID`, NOT NULL)
- `CustomerID` (INT, Foreign Key referencing `Customers.CustomerID`, NOT NULL)
- `SaleDate` (DATETIME, DEFAULT CURRENT_TIMESTAMP)
- `TotalAmount` (DECIMAL(10,2), CHECK (`TotalAmount >= 0`))
- `ProfitMargin` (DECIMAL(10,2))

DDL COMMAND SNAPSOT:

Customer table:

```
CREATE TABLE Customers (
    CustomerID INT AUTO_INCREMENT PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    Phone VARCHAR(15) UNIQUE,
    Address VARCHAR(255),
    City VARCHAR(50),
    State VARCHAR(50),
    ZipCode VARCHAR(15),
    Country VARCHAR(50),
    Gender VARCHAR(15) CHECK (Gender IN ('Male', 'Fe
    Age INT CHECK (Age >= 0 AND Age <= 120),
    CreatedAt DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Categories table:

```
CREATE TABLE Categories (
    CategoryID INT AUTO_INCREMENT PRIMARY KEY,
    CategoryName VARCHAR(100) NOT NULL UNIQUE,
    Description VARCHAR(255)
);
```

Product table:

```sql
CREATE TABLE Products (
    ProductID INT AUTO_INCREMENT PRIMARY KEY,
    ProductName VARCHAR(100) NOT NULL,
    CategoryID INT NOT NULL,
    Price DECIMAL(10,2) CHECK (Price > 0),
    Stock INT CHECK (Stock >= 0),
    CreatedAt DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
);
```

Orders table:

```sql
CREATE TABLE Orders (
    OrderID INT AUTO_INCREMENT PRIMARY KEY,
    CustomerID INT NOT NULL,
    OrderDate DATETIME DEFAULT CURRENT_TIMESTAMP,
    TotalAmount DECIMAL(10,2) CHECK (TotalAmount >= 0),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

OrderItems table:

```sql
CREATE TABLE OrderItems (
    OrderItemID INT AUTO_INCREMENT PRIMARY KEY,
    OrderID INT NOT NULL,
    ProductID INT NOT NULL,
    Quantity INT CHECK (Quantity > 0),
    Price DECIMAL(10,2) CHECK (Price > 0),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
```

Payment table:

```sql
CREATE TABLE Payments (
    PaymentID INT AUTO_INCREMENT PRIMARY KEY,
    OrderID INT NOT NULL,
    PaymentDate DATETIME DEFAULT CURRENT_TIMESTAMP,
    Amount DECIMAL(10,2) CHECK (Amount > 0),
    PaymentMethod VARCHAR(50) NOT NULL,
    PaymentStatus VARCHAR(20) CHECK (PaymentStatus IN ('Pending', 'Completed', 'Failed')),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
);
```

Shipping table:

```sql
CREATE TABLE Shipping (
    ShippingID INT AUTO_INCREMENT PRIMARY KEY,
    OrderID INT NOT NULL,
    ShippingAddress VARCHAR(255) NOT NULL,
    ShippingDate DATETIME DEFAULT CURRENT_TIMESTAMP,
    DeliveryDate DATETIME,
    ShippingStatus VARCHAR(50) CHECK (ShippingStatus IN ('Pending', 'Shipped', 'Delivered', 'Cancelled')),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
);
```

Review Details:

```sql
CREATE TABLE ReviewDetails (
    ReviewID INT AUTO_INCREMENT PRIMARY KEY,
    Rating INT CHECK (Rating BETWEEN 1 AND 5),
    ReviewText TEXT,
    ReviewDate DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

CustomerProductReviews:

```sql
CREATE TABLE CustomerProductReviews (
    CustomerID INT NOT NULL,
    ProductID INT NOT NULL,
    ReviewID INT NOT NULL,
    PRIMARY KEY (CustomerID, ProductID, ReviewID),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID),
    FOREIGN KEY (ReviewID) REFERENCES ReviewDetails(ReviewID)
);
```

DiscountDetails:

```sql
CREATE TABLE DiscountDetails (
    DiscountID INT AUTO_INCREMENT PRIMARY KEY,
    DiscountPercentage DECIMAL(5,2) CHECK (DiscountPercentage BETWEEN 0 AND 100),
    StartDate DATETIME DEFAULT CURRENT_TIMESTAMP,
    EndDate DATETIME NOT NULL
);
```

ProductDiscount:

```sql
CREATE TABLE ProductDiscount (
    ProductID INT NOT NULL,
    DiscountID INT NOT NULL,
    PRIMARY KEY (ProductID, DiscountID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID),
    FOREIGN KEY (DiscountID) REFERENCES DiscountDetails(DiscountID)
);
```

OrderStatusHistory:

```sql
CREATE TABLE OrderStatusHistory (
    StatusID INT AUTO_INCREMENT PRIMARY KEY,
    OrderID INT NOT NULL,
    Status VARCHAR(50) NOT NULL,
    StatusDate DATETIME DEFAULT CURRENT_TIMESTAMP,
    Notes TEXT,
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
);
```

OrderSaleHistory:

```sql
CREATE TABLE SalesHistory (
    SaleID INT AUTO_INCREMENT PRIMARY KEY,
    OrderID INT NOT NULL,
    CustomerID INT NOT NULL,
    SaleDate DATETIME DEFAULT CURRENT_TIMESTAMP,
    TotalAmount DECIMAL(10,2) CHECK (TotalAmount >= 0),
    ProfitMargin DECIMAL(10,2),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

Snapshot of the instances:

Customer Table:

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

CustomerID	FirstName	LastName	Email	Phone	Address	City	State	ZipCode	Country	Gender	Age	CreatedAt
1	John	Doe	john.doe@example.com	1234567890	123 Elm St	Springfield	IL	62701	USA	Male	30	2025-01-22 19:20:28
2	Jane	Smith	jane.smith@example.com	0987654321	456 Oak St	Springfield	IL	62701	USA	Female	28	2025-01-22 19:20:28
23	Michael	Brown	michael.brown@example.com	1122334455	789 Pine Lane	Chicago	IL	60601	USA	Male	42	2025-03-07 17:32:38
24	Emily	Johnson	emily.johnson@example.com	5566778899	321 Maple Drive	Houston	TX	77001	USA	Female	27	2025-03-07 17:32:38
25	David	Lee	david.lee@example.com	2233445566	987 Willow Court	Phoenix	AZ	85001	USA	Male	36	2025-03-07 17:32:38
26	Sarah	Taylor	sarah.taylor@example.com	6677889900	654 Birch Street	Philadelphia	PA	19101	USA	Female	31	2025-03-07 17:32:38
27	Daniel	Wilson	daniel.wilson@example.com	3344556677	543 Cedar Circle	San Diego	CA	92101	USA	Male	24	2025-03-07 17:32:38
28	Olivia	Martinez	olivia.martinez@example.com	7788990011	876 Aspen Road	San Antonio	TX	78201	USA	Female	29	2025-03-07 17:32:38
29	Matthew	Davis	matthew.davis@example.com	4455667788	654 Spruce Avenue	Dallas	TX	75201	USA	Male	40	2025-03-07 17:32:38
30	Sophia	Garcia	sophia.garcia@example.com	8899001122	987 Redwood Street	Austin	TX	73301	USA	Female	22	2025-03-07 17:32:38
31	Andrew	Clark	andrew.clark@example.com	9900112233	321 Birch Court	Seattle	WA	98101	USA	Male	35	2025-03-07 17:33:48
32	Linda	Evans	linda.evans@example.com	8811223344	567 Cedar Street	Boston	MA	02108	USA	Female	29	2025-03-07 17:33:48
33	Christopher	Taylor	christopher.taylor@example.com	7722334455	789 Maple Lane	Denver	CO	80201	USA	Male	41	2025-03-07 17:33:48
34	Laura	Adams	laura.adams@example.com	6623445566	213 Oak Avenue	Portland	OR	97201	USA	Female	33	2025-03-07 17:33:48
35	Ryan	Walker	ryan.walker@example.com	5524556677	435 Elm Boulevard	Atlanta	GA	30301	USA	Male	26	2025-03-07 17:33:48
36	Victoria	Baker	victoria.baker@example.com	4415667788	123 Pine Road	Miami	FL	33101	USA	Female	38	2025-03-07 17:33:48
37	Ethan	Scott	ethan.scott@example.com	3326778899	890 Willow Lane	Orlando	FL	32801	USA	Male	32	2025-03-07 17:33:48
38	Abigail	King	abigail.king@example.com	2217889900	654 Aspen Street	Nashville	TN	37201	USA	Female	25	2025-03-07 17:33:48
39	Joshua	Hill	joshua.hill@example.com	1108990011	789 Spruce Court	Charlotte	NC	28201	USA	Male	37	2025-03-07 17:33:48
40	Grace	Morris	grace.morris@example.com	9988001122	456 Redwood Avenue	Detroit	MI	48201	USA	Female	23	2025-03-07 17:33:48
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Catagories Table:

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

#	CategoryID	CategoryName	Description
1	1	Electronics	Devices and gadgets
2	2	Books	Printed and digital books
3	3	Clothing	Apparel and accessories for men, …
4	4	Home & Kitchen	Furniture, appliances, and kitchenw…
5	5	Sports & Outdoors	Equipment and gear for sports and …
6	6	Health & Beauty	Personal care products and beauty …
7	7	Toys & Games	Toys, games, and entertainment for…
8	8	Automotive	Car parts, accessories, and tools
9	9	Music	Instruments, accessories, and musi…
10	10	Office Supplies	Stationery, office equipment, and s…
11	11	Pet Supplies	Products for pet care and maintena…
12	12	Groceries	Food items and household essentials
13	13	Food	Groceries, snacks, and beverages
14	14	Fruit	Fresh and dried fruits
*	NULL	NULL	NULL

Product table:

ProductID	ProductName	CategoryID	Price	Stock	CreatedAt
1		1	298.98	0	2025-01-22 19:24:58
2	Laptop	1	999.99	30	2025-01-22 19:24:58
3	Novel	2	19.99	100	2025-01-22 19:24:58
4	Smart TV	1	499.99	20	2025-01-22 19:26:34
5	Bluetooth Headphones	1	79.99	100	2025-01-22 19:26:34
6	Digital Camera	1	299.99	15	2025-01-22 19:26:34
7	Science Fiction Novel	2	14.99	50	2025-01-22 19:26:34
8	Cookbook	2	24.99	30	2025-01-22 19:26:34
9	History Textbook	2	39.99	20	2025-01-22 19:26:34
10	Men's T-Shirt	3	19.99	200	2025-01-22 19:26:34
11	Women's Jeans	3	49.99	150	2025-01-22 19:26:34
12	Children's Jacket	3	29.99	100	2025-01-22 19:26:34
13	Blender	4	89.99	40	2025-01-22 19:26:34
14	Sofa	4	499.99	10	2025-01-22 19:26:34
15	Dining Set	4	299.99	5	2025-01-22 19:26:34
16	Yoga Mat	5	29.99	80	2025-01-22 19:26:34
17	Mountain Bike	5	399.99	10	2025-01-22 19:26:34
18	Tent	5	99.99	25	2025-01-22 19:26:34
19	Shampoo	6	9.99	150	2025-01-22 19:26:34
20	Face Cream	6	19.99	100	2025-01-22 19:26:34
21	Electric Toothbrush	6	49.99	60	2025-01-22 19:26:34
22	Board Game	7	29.99	70	2025-01-22 19:26:34
23	Action Figure	7	14.99	120	2025-01-22 19:26:34
24	Puzzle	7	9.99	90	2025-01-22 19:26:34
25	Car Battery	8	89.99	30	2025-01-22 19:26:34
26	Tire	8	59.99	50	2025-01-22 19:26:34
27	Oil Filter	8	9.99	100	2025-01-22 19:26:34
28	Acoustic Guitar	9	199.99	20	2025-01-22 19:26:34
29	Piano Keyboard	9	299.99	15	2025-01-22 19:26:34
30	Drum Set	9	499.99	5	2025-01-22 19:26:34
31	Printer Paper	10	5.99	500	2025-01-22 19:26:34
32	Ballpoint Pens	10	2.99	300	2025-01-22 19:26:34

Orders table:

	OrderID	CustomerID	OrderDate	TotalAmoun
	1	1	2025-01-22 19:28:16	719.98
	2	2	2025-01-22 19:28:16	1019.98
	122	23	2024-01-16 09:15:00	475.25
	123	24	2024-01-16 14:20:00	899.99
	124	25	2024-01-17 12:30:00	245.75
	125	26	2024-01-17 16:45:00	678.50
	126	27	2024-01-18 10:00:00	189.99
	127	28	2024-01-18 13:25:00	567.80
	128	29	2024-01-19 11:30:00	345.25
0	129	30	2024-01-19 15:40:00	789.99
1	130	31	2024-01-20 09:45:00	234.50
2	131	32	2024-01-20 14:15:00	445.75
3	132	33	2024-01-21 10:20:00	678.25
4	133	34	2024-01-21 16:30:00	199.99
5	134	35	2024-01-22 11:00:00	845.50
6	135	36	2024-01-22 15:20:00	567.25
7	136	37	2024-01-23 09:30:00	123.45
8	137	38	2024-01-23 14:45:00	456.78
9	138	39	2024-01-24 10:15:00	789.12
0	139	40	2024-01-24 16:00:00	234.56
	NULL	NULL	NULL	NULL

OrderItems:

	OrderItemID	OrderID	ProductID	Quantity	Price
	1	1	1	1	699.99
	2	1	3	1	19.99
	3	2	2	1	999.99
	28	1	1	2	149.99
	29	1	3	1	89.99
	30	2	2	1	199.99
	31	2	4	3	45.50
	32	122	1	1	149.99
	33	122	5	2	79.99
0	34	123	3	2	89.99
1	35	123	6	1	129.99
2	36	124	2	2	199.99
3	37	124	4	1	45.50
4	38	125	1	3	149.99
5	39	125	5	2	79.99
6	40	126	3	1	89.99
7	41	126	6	2	129.99
8	42	127	2	1	199.99
9	43	127	4	2	45.50
0	44	128	1	2	149.99
1	45	128	5	1	79.99
2	46	129	3	2	89.99
3	47	129	6	1	129.99
4	48	130	2	1	199.99
5	49	130	4	3	45.50
6	50	131	1	1	149.99
7	51	131	5	2	79.99
8	52	132	3	2	89.99

PaymentMethod:

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

PaymentID	OrderID	PaymentDate	Amount	PaymentMethod	PaymentStatus
1	1	2025-01-22 19:29:08	719.98	Credit Card	Completed
2	2	2025-01-22 19:29:08	1019.98	PayPal	Completed
17	122	2025-03-07 18:46:46	459.99	PayPal	Completed
18	123	2025-03-07 18:46:46	678.50	Credit Card	Completed
19	124	2025-03-07 18:46:46	899.99	Debit Card	Pending
20	125	2025-03-07 18:46:46	545.75	Credit Card	Completed
21	126	2025-03-07 18:46:46	789.99	PayPal	Completed
22	127	2025-03-07 18:46:46	299.50	Credit Card	Failed
23	128	2025-03-07 18:46:46	459.99	Debit Card	Completed
24	129	2025-03-07 18:46:46	678.25	Credit Card	Completed
25	130	2025-03-07 18:46:46	899.99	PayPal	Pending
26	131	2025-03-07 18:46:46	459.50	Credit Card	Completed
27	132	2025-03-07 18:46:46	789.75	Debit Card	Completed
28	133	2025-03-07 18:46:46	599.99	Credit Card	Completed
29	134	2025-03-07 18:46:46	459.99	PayPal	Failed
30	135	2025-03-07 18:46:46	678.50	Credit Card	Completed
31	136	2025-03-07 18:46:46	899.99	Debit Card	Completed
32	137	2025-03-07 18:46:46	459.99	Credit Card	Pending
33	138	2025-03-07 18:46:46	789.50	PayPal	Completed
34	139	2025-03-07 18:46:46	599.99	Credit Card	Completed
NULL	NULL	NULL	NULL	NULL	NULL

Payments 11

Action Output

#	Time	Action	Message

Shipping table:

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

ShippingID	OrderID	ShippingAddress	ShippingDate	DeliveryDate	ShippingStatus
1	1	123 Elm St, Springfield, IL, 62701, USA	2025-01-22 19:29:27	2023-10-15 10:00:00	Delivered
2	2	456 Oak St, Springfield, IL, 62701, USA	2025-01-22 19:29:27	2023-10-16 15:00:00	Shipped
16	122	789 Pine Ave, Chicago, IL, 60601, USA	2025-03-07 18:54:04	2024-01-20 14:30:00	Shipped
17	123	321 Maple Rd, Aurora, IL, 60505, USA	2025-03-07 18:54:04	2024-01-21 11:20:00	Delivered
18	124	654 Birch Ln, Rockford, IL, 61101, USA	2025-03-07 18:54:04	2024-01-22 09:45:00	Pending
19	125	987 Cedar Blvd, Naperville, IL, 60540, USA	2025-03-07 18:54:04	2024-01-23 16:15:00	Shipped
20	126	147 Walnut St, Joliet, IL, 60431, USA	2025-03-07 18:54:04	2024-01-24 13:30:00	Delivered
21	127	258 Ash Dr, Peoria, IL, 61601, USA	2025-03-07 18:54:04	2024-01-25 10:45:00	Shipped
22	128	369 Spruce Ct, Elgin, IL, 60120, USA	2025-03-07 18:54:04	2024-01-26 14:20:00	Pending
23	129	741 Willow Way, Waukegan, IL, 60085, USA	2025-03-07 18:54:04	2024-01-27 11:30:00	Delivered
24	130	852 Hickory Pl, Cicero, IL, 60804, USA	2025-03-07 18:54:04	2024-01-28 15:40:00	Shipped
25	131	963 Beech St, Arlington Heights, IL, 60004, USA	2025-03-07 18:54:04	2024-01-29 12:15:00	Delivered
26	132	159 Poplar Ave, Bolingbrook, IL, 60440, USA	2025-03-07 18:54:04	2024-01-30 09:30:00	Pending
27	133	357 Sycamore Ln, Schaumburg, IL, 60193, USA	2025-03-07 18:54:04	2024-01-31 16:45:00	Shipped
28	134	486 Chestnut Rd, Evanston, IL, 60201, USA	2025-03-07 18:54:04	2024-02-01 13:20:00	Delivered
29	135	753 Magnolia Dr, Palatine, IL, 60067, USA	2025-03-07 18:54:04	2024-02-02 10:30:00	Shipped
30	136	951 Dogwood Ct, Oak Lawn, IL, 60453, USA	2025-03-07 18:54:04	2024-02-03 14:15:00	Pending
31	137	264 Elm Park, Des Plaines, IL, 60016, USA	2025-03-07 18:54:04	2024-02-04 11:45:00	Delivered
32	138	375 Oak Grove, Hoffman Estates, IL, 60169, USA	2025-03-07 18:54:04	2024-02-05 15:30:00	Shipped
33	139	486 Pine View, Orland Park, IL, 60462, USA	2025-03-07 18:54:04	2024-02-06 12:00:00	Delivered
NULL	NULL	NULL	NULL	NULL	NULL

ReviewDetails table:

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

#	ReviewID	Rating	ReviewText	ReviewDate
1	1	5	Excellent product!	2025-01-22 19:29:47
2	2	4	Very good, but could be improved.	2025-01-22 19:29:47
3	3	5	Excellent product! Exceeded my ex…	2025-03-07 19:01:42
4	4	4	Very good product, fast shipping.	2025-03-07 19:01:42
5	5	5	Amazing quality and perfect fit.	2025-03-07 19:01:42
6	6	3	Product is okay. Decent quality.	2025-03-07 19:01:42
7	7	5	Great purchase, highly recommend…	2025-03-07 19:01:42
8	8	4	Good product, nice packaging.	2025-03-07 19:01:42
9	9	2	Not quite what I expected.	2025-03-07 19:01:42
10	10	5	Perfect! Exactly what I needed.	2025-03-07 19:01:42
11	11	4	Good quality but slow delivery.	2025-03-07 19:01:42
12	12	3	Average quality, fair price.	2025-03-07 19:01:42
13	13	5	Fantastic product, great service.	2025-03-07 19:01:42
14	14	4	Well-made product, minor issues.	2025-03-07 19:01:42
15	15	1	Not satisfied with quality.	2025-03-07 19:01:42
16	16	5	Excellent value for money.	2025-03-07 19:01:42
17	17	4	Good product, helpful service.	2025-03-07 19:01:42
18	18	3	Meets basic expectations.	2025-03-07 19:01:42
19	19	5	Outstanding quality product.	2025-03-07 19:01:42
20	20	2	Below average quality.	2025-03-07 19:01:42
21	21	4	Solid product, good features.	2025-03-07 19:01:42

CustomerProductReview:

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

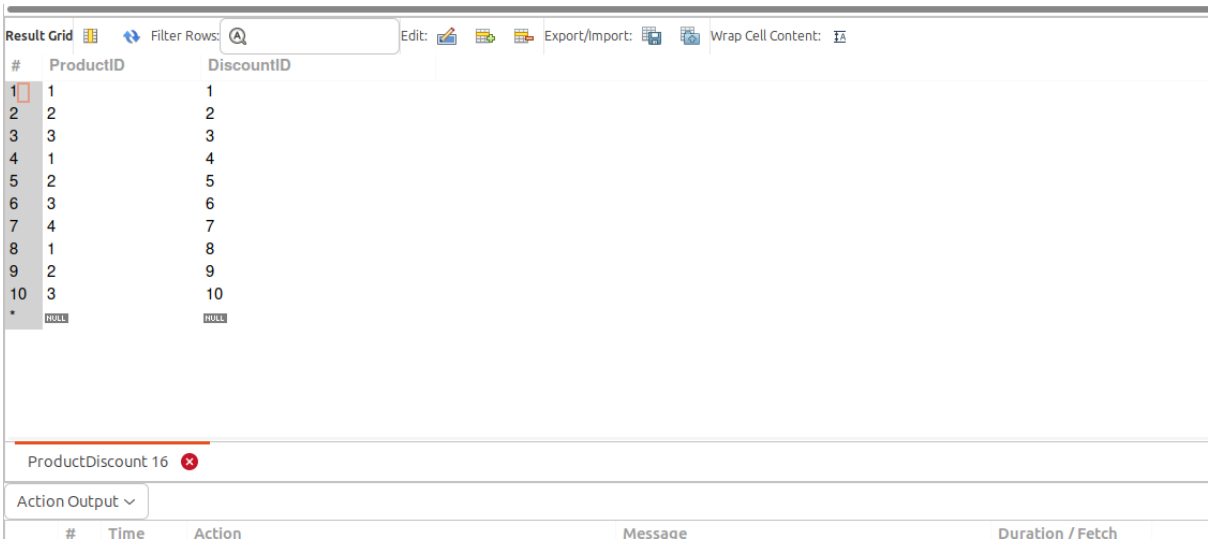
#	CustomerID	ProductID	ReviewID
1	1	1	1
2	24	1	4
3	28	1	8
4	32	1	12
5	36	1	16
6	40	1	20
7	2	2	2
8	25	2	5
9	29	2	9
10	33	2	13
11	37	2	17
12	23	3	3
13	26	3	6
14	30	3	10
15	34	3	14
16	38	3	18
17	27	4	7
18	31	4	11
19	35	4	15
20	39	4	19
*	NULL	NULL	NULL

DiscountDetails:

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

#	DiscountID	DiscountPercentage	StartDate	EndDate
1	1	10.00	2025-01-22 19:31:06	2023-12-31 23:59:59
2	2	15.00	2025-01-22 19:31:06	2023-11-30 23:59:59
3	3	10.00	2023-12-01 00:00:00	2023-12-31 23:59:59
4	4	15.00	2024-01-01 00:00:00	2024-01-31 23:59:59
5	5	20.00	2024-02-01 00:00:00	2024-02-28 23:59:59
6	6	12.50	2024-01-05 00:00:00	2024-01-15 23:59:59
7	7	18.00	2024-02-01 00:00:00	2024-02-20 23:59:59
8	8	25.00	2024-03-01 00:00:00	2024-03-31 23:59:59
9	9	15.00	2024-01-01 00:00:00	2024-02-01 23:59:59
10	10	20.00	2024-01-15 00:00:00	2024-02-15 23:59:59
11	11	25.00	2024-01-20 00:00:00	2024-03-01 23:59:59
12	12	10.00	2024-02-01 00:00:00	2024-03-15 23:59:59
13	13	30.00	2024-02-15 00:00:00	2024-03-31 23:59:59
14	14	12.50	2024-03-01 00:00:00	2024-04-01 23:59:59
15	15	40.00	2024-03-15 00:00:00	2024-04-15 23:59:59
16	16	17.50	2024-04-01 00:00:00	2024-05-01 23:59:59
17	17	22.50	2024-04-15 00:00:00	2024-05-15 23:59:59
18	18	35.00	2024-05-01 00:00:00	2024-06-01 23:59:59
*	NULL	NULL	NULL	NULL

ProductDiscount:



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

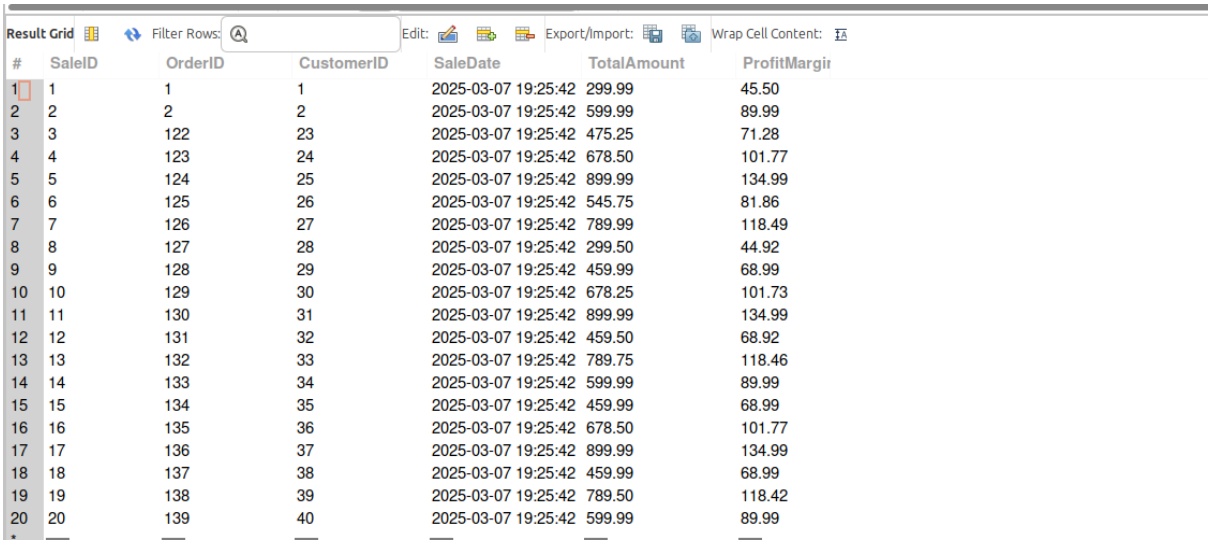
#	ProductID	DiscountID
1	1	1
2	2	2
3	3	3
4	1	4
5	2	5
6	3	6
7	4	7
8	1	8
9	2	9
10	3	10
*	NULL	NULL

ProductDiscount 16

Action Output

#	Time	Action	Message	Duration / Fetch

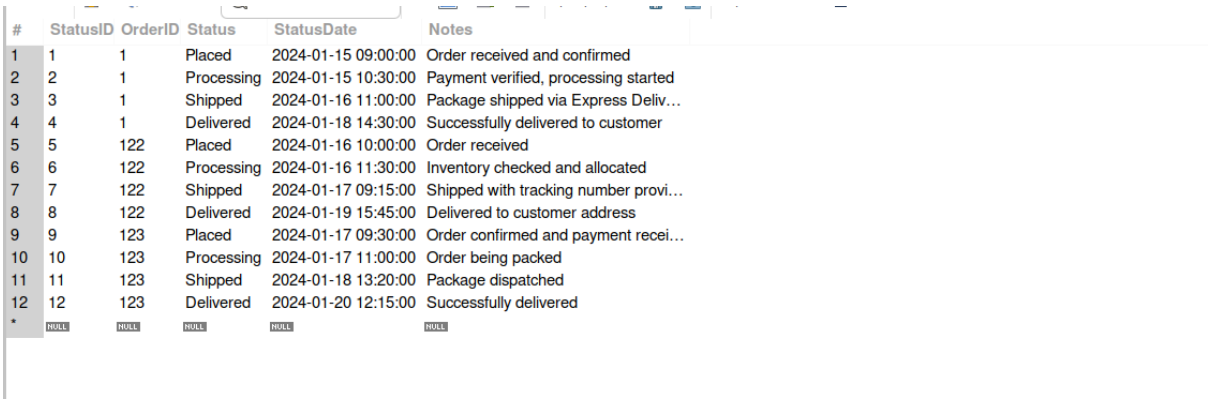
SalesHistory:



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

#	SaleID	OrderID	CustomerID	SaleDate	TotalAmount	ProfitMargi
1	1	1	1	2025-03-07 19:25:42	299.99	45.50
2	2	2	2	2025-03-07 19:25:42	599.99	89.99
3	3	122	23	2025-03-07 19:25:42	475.25	71.28
4	4	123	24	2025-03-07 19:25:42	678.50	101.77
5	5	124	25	2025-03-07 19:25:42	899.99	134.99
6	6	125	26	2025-03-07 19:25:42	545.75	81.86
7	7	126	27	2025-03-07 19:25:42	789.99	118.49
8	8	127	28	2025-03-07 19:25:42	299.50	44.92
9	9	128	29	2025-03-07 19:25:42	459.99	68.99
10	10	129	30	2025-03-07 19:25:42	678.25	101.73
11	11	130	31	2025-03-07 19:25:42	899.99	134.99
12	12	131	32	2025-03-07 19:25:42	459.50	68.92
13	13	132	33	2025-03-07 19:25:42	789.75	118.46
14	14	133	34	2025-03-07 19:25:42	599.99	89.99
15	15	134	35	2025-03-07 19:25:42	459.99	68.99
16	16	135	36	2025-03-07 19:25:42	678.50	101.77
17	17	136	37	2025-03-07 19:25:42	899.99	134.99
18	18	137	38	2025-03-07 19:25:42	459.99	68.99
19	19	138	39	2025-03-07 19:25:42	789.50	118.42
20	20	139	40	2025-03-07 19:25:42	599.99	89.99
*	—	—	—	—	—	—

OrderStatusHistory:



#	StatusID	OrderID	Status	StatusDate	Notes
1	1	1	Placed	2024-01-15 09:00:00	Order received and confirmed
2	2	1	Processing	2024-01-15 10:30:00	Payment verified, processing started
3	3	1	Shipped	2024-01-16 11:00:00	Package shipped via Express Deliv…
4	4	1	Delivered	2024-01-18 14:30:00	Successfully delivered to customer
5	5	122	Placed	2024-01-16 10:00:00	Order received
6	6	122	Processing	2024-01-16 11:30:00	Inventory checked and allocated
7	7	122	Shipped	2024-01-17 09:15:00	Shipped with tracking number provi…
8	8	122	Delivered	2024-01-19 15:45:00	Delivered to customer address
9	9	123	Placed	2024-01-17 09:30:00	Order confirmed and payment recei…
10	10	123	Processing	2024-01-17 11:00:00	Order being packed
11	11	123	Shipped	2024-01-18 13:20:00	Package dispatched
12	12	123	Delivered	2024-01-20 12:15:00	Successfully delivered
*	NULL	NULL	NULL	NULL	NULL

Query Statement ,Relational expression and output:
Join:

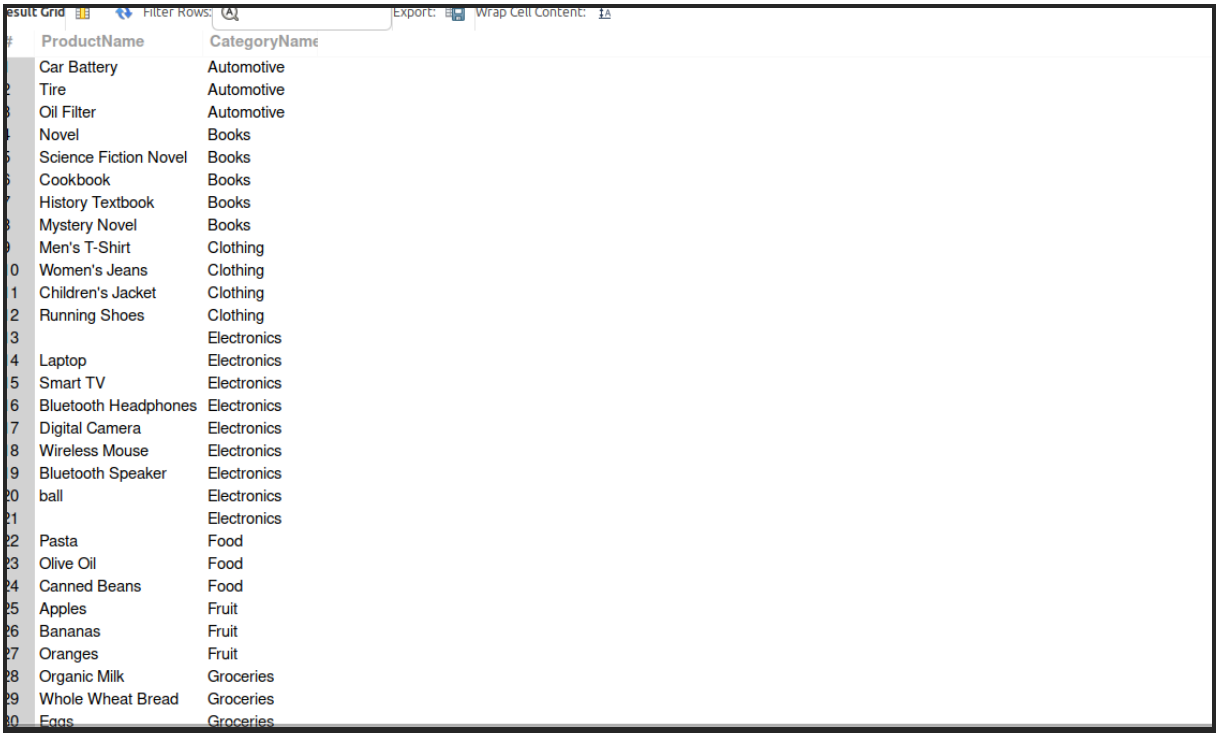
```
- Purpose: Get customer orders with shipping details (INNER JOIN with ON)

SELECT c.FirstName, c.LastName, o.OrderID, s.ShippingStatus
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
JOIN Shipping s ON o.OrderID = s.OrderID
WHERE s.ShippingStatus = 'Delivered';
```

#	FirstName	LastName	OrderID	ShippingStatus
1	John	Doe	1	Delivered
2	Emily	Johnson	123	Delivered
3	Daniel	Wilson	126	Delivered
4	Sophia	Garcia	129	Delivered
5	Linda	Evans	131	Delivered
6	Ryan	Walker	134	Delivered
7	Abigail	King	137	Delivered
8	Grace	Morris	139	Delivered

```
-- Purpose: Show all customers and their orders, including customers with no orders (LEFT OUTER
JOIN)

SELECT c.FirstName, c.LastName, o.OrderID, o.TotalAmount
FROM Customers c
LEFT OUTER JOIN Orders o ON c.CustomerID = o.CustomerID;
```

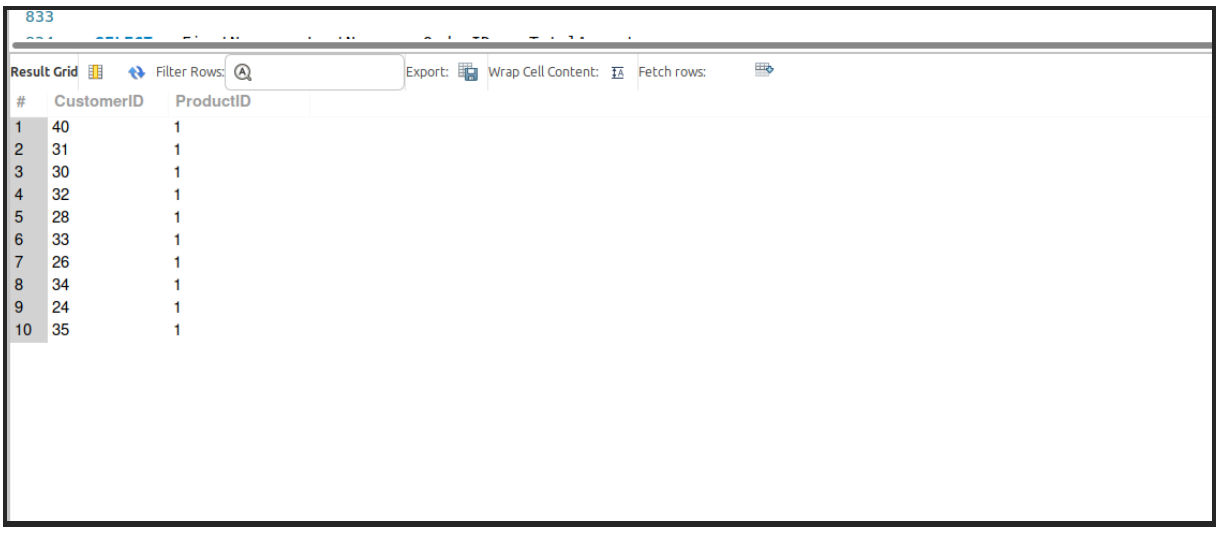
#	FirstName	LastName	OrderID	TotalAmoun
	John	Doe	1	719.98
2	Jane	Smith	2	1019.98
3	Michael	Brown	122	475.25
4	Emily	Johnson	123	899.99
5	David	Lee	124	245.75
6	Sarah	Taylor	125	678.50
7	Daniel	Wilson	126	189.99
8	Olivia	Martinez	127	567.80
9	Matthew	Davis	128	345.25
10	Sophia	Garcia	129	789.99
11	Andrew	Clark	130	234.50
12	Linda	Evans	131	445.75
13	Christopher	Taylor	132	678.25
14	Laura	Adams	133	199.99
15	Ryan	Walker	134	845.50
16	Victoria	Baker	135	567.25
17	Ethan	Scott	136	123.45
18	Abigail	King	137	456.78
19	Joshua	Hill	138	789.12
20	Grace	Morris	139	234.56

```
-- Purpose: List products and their categories (JOIN with USING)

SELECT p.ProductName, c.CategoryName
FROM Products p
JOIN Categories c USING (CategoryID);
```



#	ProductName	CategoryName
	Car Battery	Automotive
2	Tire	Automotive
3	Oil Filter	Automotive
4	Novel	Books
5	Science Fiction Novel	Books
6	Cookbook	Books
7	History Textbook	Books
8	Mystery Novel	Books
9	Men's T-Shirt	Clothing
10	Women's Jeans	Clothing
11	Children's Jacket	Clothing
12	Running Shoes	Clothing
13		Electronics
14	Laptop	Electronics
15	Smart TV	Electronics
16	Bluetooth Headphones	Electronics
17	Digital Camera	Electronics
18	Wireless Mouse	Electronics
19	Bluetooth Speaker	Electronics
20	ball	Electronics
21		Electronics
22	Pasta	Food
23	Olive Oil	Food
24	Canned Beans	Food
25	Apples	Fruit
26	Bananas	Fruit
27	Oranges	Fruit
28	Organic Milk	Groceries
29	Whole Wheat Bread	Groceries
30	Eggs	Groceries

```
-- Purpose: Show all possible customer-product combinations (CROSS JOIN)

SELECT c.CustomerID, p.ProductID
FROM Customers c
CROSS JOIN Products p
LIMIT 10;
```



833

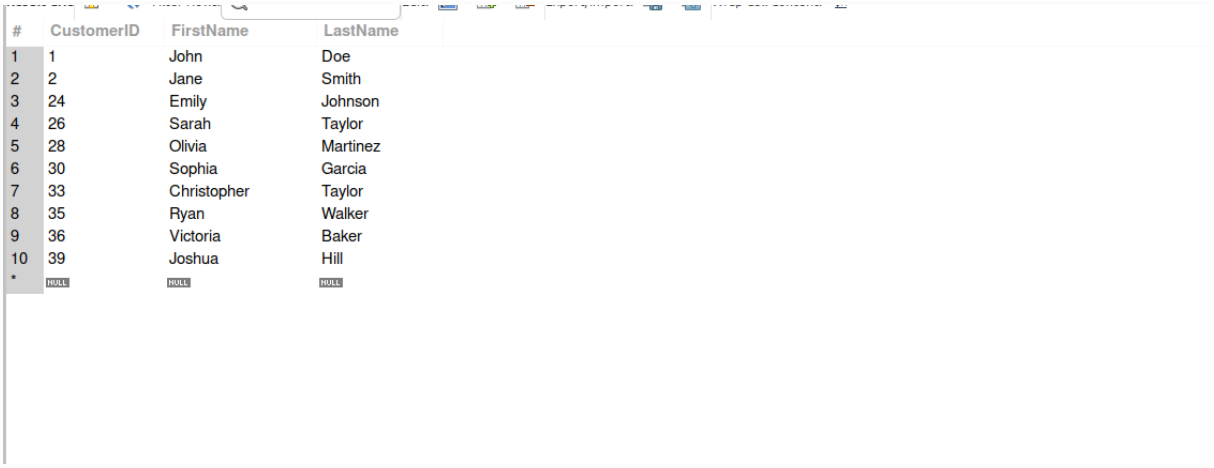
Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

#	CustomerID	ProductID
1	40	1
2	31	1
3	30	1
4	32	1
5	28	1
6	33	1
7	26	1
8	34	1
9	24	1
10	35	1

ii. Nested Sub-queries with Clauses:

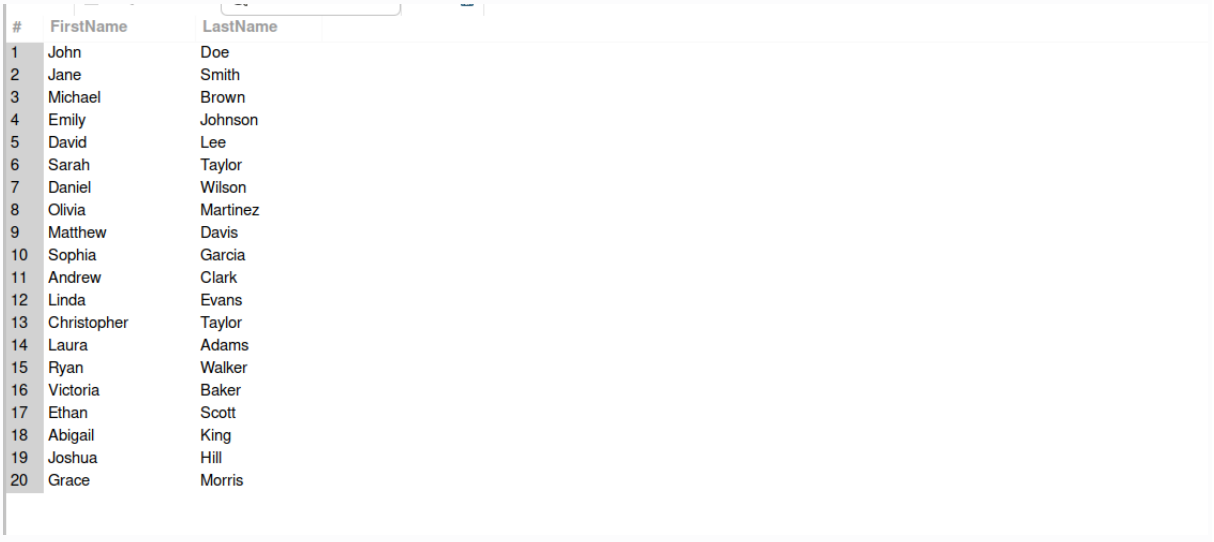
-- Purpose: Find customers who spent more than average order amount
SELECT CustomerID, FirstName, LastName
FROM Customers

```
WHERE CustomerID IN (
    SELECT CustomerID
    FROM Orders
    WHERE TotalAmount > (
        SELECT AVG(TotalAmount) FROM Orders
    )
);
```



#	CustomerID	FirstName	LastName
1	1	John	Doe
2	2	Jane	Smith
3	24	Emily	Johnson
4	26	Sarah	Taylor
5	28	Olivia	Martinez
6	30	Sophia	Garcia
7	33	Christopher	Taylor
8	35	Ryan	Walker
9	36	Victoria	Baker
10	39	Joshua	Hill
*	NULL	NULL	NULL

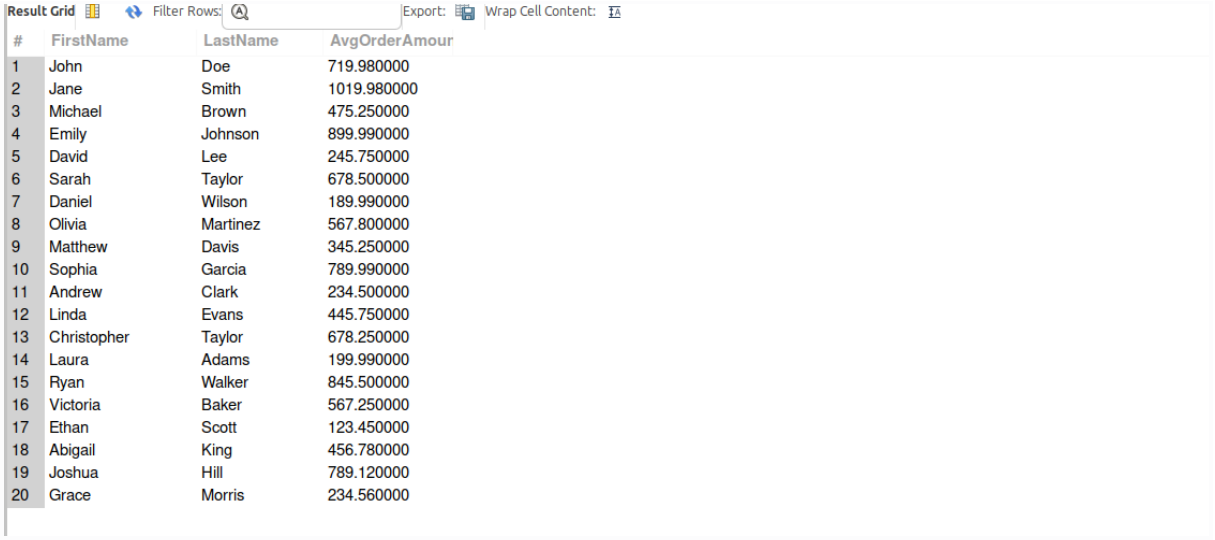
```
-- Purpose: Find customers who have placed at least one order (EXISTS)


SELECT FirstName, LastName
FROM Customers c
WHERE EXISTS (
    SELECT 1
    FROM Orders
    WHERE CustomerID = c.CustomerID
);
```



#	FirstName	LastName
1	John	Doe
2	Jane	Smith
3	Michael	Brown
4	Emily	Johnson
5	David	Lee
6	Sarah	Taylor
7	Daniel	Wilson
8	Olivia	Martinez
9	Matthew	Davis
10	Sophia	Garcia
11	Andrew	Clark
12	Linda	Evans
13	Christopher	Taylor
14	Laura	Adams
15	Ryan	Walker
16	Victoria	Baker
17	Ethan	Scott
18	Abigail	King
19	Joshua	Hill
20	Grace	Morris

iii. Nested Sub-queries in FROM and SELECT:

```
-- Purpose: Calculate average order amount per customer with their details
SELECT c.FirstName, c.LastName,
   (SELECT AVG(TotalAmount)
    FROM Orders
    WHERE CustomerID = c.CustomerID) as AvgOrderAmount
FROM Customers c;
```



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

#	FirstName	LastName	AvgOrderAmour
1	John	Doe	719.980000
2	Jane	Smith	1019.980000
3	Michael	Brown	475.250000
4	Emily	Johnson	899.990000
5	David	Lee	245.750000
6	Sarah	Taylor	678.500000
7	Daniel	Wilson	189.990000
8	Olivia	Martinez	567.800000
9	Matthew	Davis	345.250000
10	Sophia	Garcia	789.990000
11	Andrew	Clark	234.500000
12	Linda	Evans	445.750000
13	Christopher	Taylor	678.250000
14	Laura	Adams	199.990000
15	Ryan	Walker	845.500000
16	Victoria	Baker	567.250000
17	Ethan	Scott	123.450000
18	Abigail	King	456.780000
19	Joshua	Hill	789.120000
20	Grace	Morris	234.560000

```
-- Purpose: Get customer order statistics
SELECT *
FROM (
   SELECT CustomerID,
          COUNT(*) as OrderCount,
          AVG(TotalAmount) as AvgAmount
   FROM Orders
   GROUP BY CustomerID
) as OrderStats;
```

#	CustomerID	OrderCount	AvgAmount
1	1	1	719.980000
2	2	1	1019.980000
3	23	1	475.250000
4	24	1	899.990000
5	25	1	245.750000
6	26	1	678.500000
7	27	1	189.990000
8	28	1	567.800000
9	29	1	345.250000
10	30	1	789.990000
11	31	1	234.500000
12	32	1	445.750000
13	33	1	678.250000
14	34	1	199.990000
15	35	1	845.500000
16	36	1	567.250000
17	37	1	123.450000
18	38	1	456.780000
19	39	1	789.120000
20	40	1	234.560000

iv. ORDER BY, GROUP BY, HAVING:

```
-- Purpose: Show total sales by category, only for categories with sales over $1000
SELECT c.CategoryName, SUM(oi.Price * oi.Quantity) as TotalSales
FROM Categories c
JOIN Products p ON c.CategoryID = p.CategoryID
JOIN OrderItems oi ON p.ProductID = oi.ProductID
GROUP BY c.CategoryName
HAVING TotalSales > 1000
ORDER BY TotalSales DESC;
```

#	CategoryName	TotalSales
1	Electronics	8122.07
2	Books	1009.88

. WITH Clause:

```
-- Purpose: Analyze customer purchasing patterns

WITH CustomerPurchases AS (
    SELECT CustomerID, COUNT(*) as OrderCount, SUM(TotalAmount) as TotalSpent
    FROM Orders
    GROUP BY CustomerID
)
SELECT c.FirstName, c.LastName, cp.OrderCount, cp.TotalSpent
FROM Customers c
JOIN CustomerPurchases cp ON c.CustomerID = cp.CustomerID;
```

FirstName	LastName	OrderCount	TotalSpent
John	Doe	1	719.98
Jane	Smith	1	1019.98
Michael	Brown	1	475.25
Emily	Johnson	1	899.99
David	Lee	1	245.75
Sarah	Taylor	1	678.50
Daniel	Wilson	1	189.99
Olivia	Martinez	1	567.80
Matthew	Davis	1	345.25
Sophia	Garcia	1	789.99
Andrew	Clark	1	234.50
Linda	Evans	1	445.75
Christopher	Taylor	1	678.25
Laura	Adams	1	199.99
Ryan	Walker	1	845.50
Victoria	Baker	1	567.25
Ethan	Scott	1	123.45
Abigail	King	1	456.78
Joshua	Hill	1	789.12
Grace	Morris	1	234.56

vi. String and Set Operations:

```
-- Purpose: Search customers by name pattern
SELECT FirstName, LastName
FROM Customers
WHERE CONCAT(FirstName, ' ', LastName) LIKE '%John%';
```

#	FirstName	LastName
1	John	Doe
2	Emily	Johnson

-- Purpose: Combine active and cancelled orders
(SELECT OrderID, 'Active' as Status FROM Orders WHERE TotalAmount > 0)
UNION
(SELECT OrderID, 'Cancelled' as Status FROM Orders WHERE TotalAmount = 0);

#	OrderID	Status
1	1	Active
2	2	Active
3	122	Active
4	123	Active
5	124	Active
6	125	Active
7	126	Active
8	127	Active
9	128	Active
10	129	Active
11	130	Active
12	131	Active
13	132	Active
14	133	Active
15	134	Active
16	135	Active
17	136	Active
18	137	Active
19	138	Active
20	139	Active

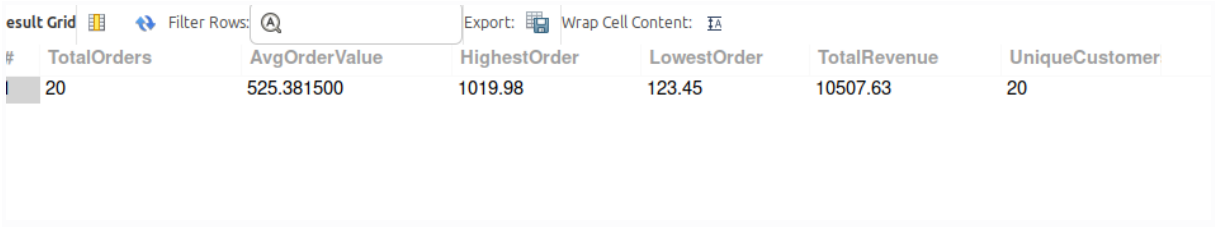
vii. Update and Delete:

-- Purpose: Update product prices with 10% increase
UPDATE Products
SET Price = Price * 1.10
WHERE CategoryID = 1;

```
-- Purpose: Remove cancelled orders
DELETE FROM Orders
WHERE OrderID IN (
    SELECT OrderID
    FROM OrderStatusHistory
    WHERE Status = 'Cancelled'
);
```

viii. Aggregate Functions:

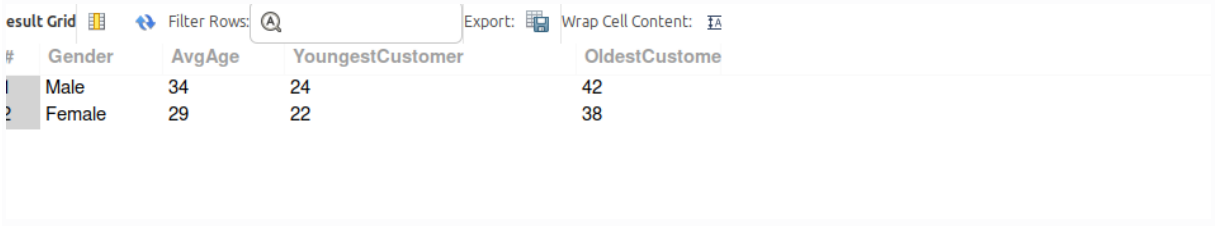
```
-- Purpose: Get comprehensive order statistics

SELECT
    COUNT(*) as TotalOrders,
    AVG(TotalAmount) as AvgOrderValue,
    MAX(TotalAmount) as HighestOrder,
    MIN(TotalAmount) as LowestOrder,
    SUM(TotalAmount) as TotalRevenue,
    COUNT(DISTINCT CustomerID) as UniqueCustomers
FROM Orders;
```



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

#	TotalOrders	AvgOrderValue	HighestOrder	LowestOrder	TotalRevenue	UniqueCustomer
1	20	525.381500	1019.98	123.45	10507.63	20

```
-- Purpose: Calculate customer age statistics
SELECT
    Gender,
    ROUND(AVG(Age)) as AvgAge,
    MIN(Age) as YoungestCustomer,
    MAX(Age) as OldestCustomer
FROM Customers
GROUP BY Gender;
```



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

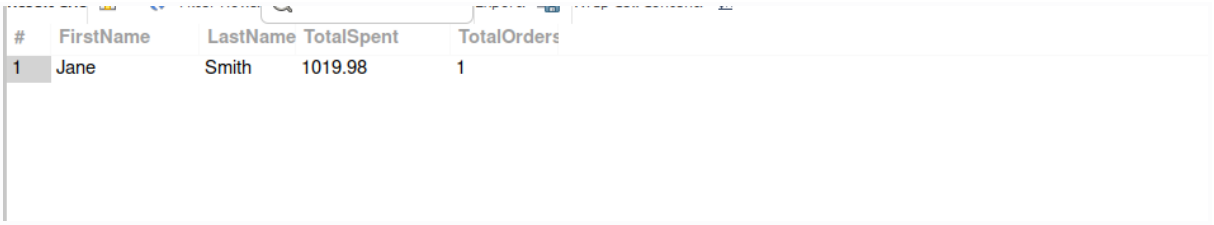
#	Gender	AvgAge	YoungestCustomer	OldestCustome
1	Male	34	24	42
2	Female	29	22	38

Create views and use those views in answering queries.

```
-CREATE VIEW CustomerOrderSummary AS
SELECT
    c.CustomerID,
    c.FirstName,
    c.LastName,
    COUNT(o.OrderID) as TotalOrders,
    SUM(o.TotalAmount) as TotalSpent,
    AVG(o.TotalAmount) as AvgOrderValue
FROM Customers c
LEFT JOIN Orders o ON c.CustomerID = o.CustomerID
GROUP BY c.CustomerID, c.FirstName, c.LastName;
```

Now, let's use these views in various queries:

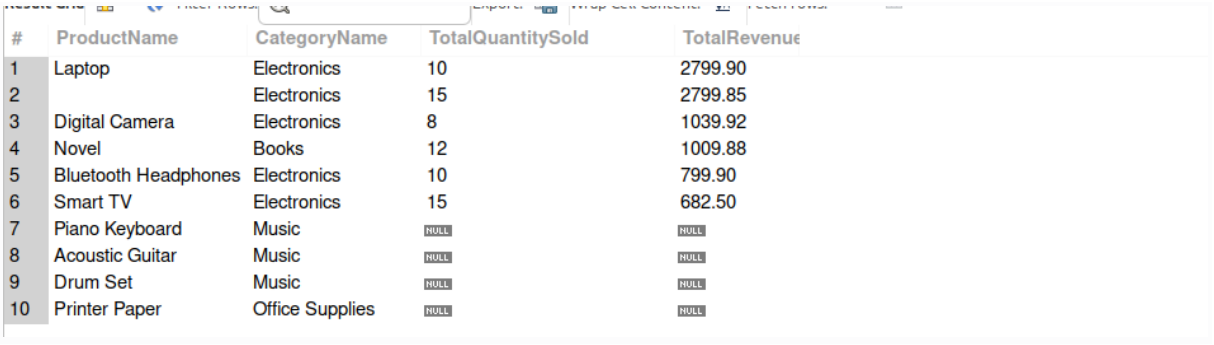
```
SELECT
    FirstName,
    LastName,
    TotalSpent,
    TotalOrders
FROM CustomerOrderSummary
WHERE TotalSpent > 1000
ORDER BY TotalSpent DESC;
```



#	FirstName	LastName	TotalSpent	TotalOrders
1	Jane	Smith	1019.98	1

```
-- View for Product Performance
CREATE VIEW ProductPerformance AS
SELECT
    p.ProductID,
    p.ProductName,
    c.CategoryName,
    COUNT(oi.OrderItemID) as TimesOrdered,
    SUM(oi.Quantity) as TotalQuantitySold,
    SUM(oi.Quantity * oi.Price) as TotalRevenue
FROM Products p
```

```
JOIN Categories c ON p.CategoryID = c.CategoryID
LEFT JOIN OrderItems oi ON p.ProductID = oi.ProductID
GROUP BY p.ProductID, p.ProductName, c.CategoryName;

-- View for Order Details
CREATE VIEW OrderDetailsComplete AS
SELECT
    o.OrderID,
    c.FirstName,
    c.LastName,
    o.OrderDate,
    o.TotalAmount,
    p.PaymentStatus,
    s.ShippingStatus
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
LEFT JOIN Payments p ON o.OrderID = p.OrderID
LEFT JOIN Shipping s ON o.OrderID = s.OrderID;

-- View for Customer Reviews Summary
CREATE VIEW CustomerReviewsSummary AS
SELECT
    c.CustomerID,
    c.FirstName,
    c.LastName,
    p.ProductName,
    rd.Rating,
    rd.ReviewText
FROM Customers c
JOIN CustomerProductReviews cpr ON c.CustomerID = cpr.CustomerID
JOIN Products p ON cpr.ProductID = p.ProductID
JOIN ReviewDetails rd ON cpr.ReviewID = rd.ReviewID;


— base on the vies some query

-- Get best-selling products
SELECT
    ProductName,
    CategoryName,
    TotalQuantitySold,
    TotalRevenue
FROM ProductPerformance
ORDER BY TotalRevenue DESC
LIMIT 10;
```



#	ProductName	CategoryName	TotalQuantitySold	TotalRevenue
1	Laptop	Electronics	10	2799.90
2		Electronics	15	2799.85
3	Digital Camera	Electronics	8	1039.92
4	Novel	Books	12	1009.88
5	Bluetooth Headphones	Electronics	10	799.90
6	Smart TV	Electronics	15	682.50
7	Piano Keyboard	Music	NULL	NULL
8	Acoustic Guitar	Music	NULL	NULL
9	Drum Set	Music	NULL	NULL
10	Printer Paper	Office Supplies	NULL	NULL

```
-- Get order status summary
SELECT
    FirstName,
    LastName,
    OrderDate,
    TotalAmount,
    PaymentStatus,
    ShippingStatus
FROM OrderDetailsComplete
WHERE OrderDate >= DATE_SUB(CURRENT_DATE, INTERVAL 30 DAY);
```

Find the list of non-trivial FDs and proof that the schemas are in
desired normal forms

Customers Table:

Non-trivial FDs:

- CustomerID → {FirstName, LastName, Email, Phone, Address, City, State, ZipCode, Country, Gender, Age, CreatedAt}
- Email → CustomerID (due to UNIQUE constraint)
- Phone → CustomerID (due to UNIQUE constraint)

Normal Form: 3NF/BCNF because:

- Primary key is CustomerID
- All non-key attributes are fully functionally dependent on the primary key
- No transitive dependencies
- Every determinant is a candidate key

2. Categories Table:

3.
4.

Non-trivial FDs:

- CategoryID → {CategoryName, Description}
- CategoryName → CategoryID (due to UNIQUE constraint)

Normal Form: BCNF because:

- Primary key is CategoryID
- CategoryName is unique (candidate key)
- No transitive dependencies

3. Products Table:

Non-trivial FDs:

- ProductID → {ProductName, CategoryID, Price, Stock, CreatedAt}

Normal Form: BCNF because:

- Primary key is ProductID
- All attributes are fully dependent on the primary key
- No transitive dependencies

4. Orders Table:

Non-trivial FDs:

- OrderID → {CustomerID, OrderDate, TotalAmount}

Normal Form: BCNF because:

- Primary key is OrderID
- All attributes directly depend on the primary key
- No transitive dependencies

5. OrderItems Table:

6.

Non-trivial FDs:

- OrderItemID → {OrderID, ProductID, Quantity, Price}
- {OrderID, ProductID} → {Quantity, Price}

Normal Form: BCNF because:

- Primary key is OrderItemID
- All non-key attributes are fully dependent on the primary key
- No transitive dependencies
-

6. Payments Table:
7.

Non-trivial FDs:

- PaymentID → {OrderID, PaymentDate, Amount, PaymentMethod, PaymentStatus}

Normal Form: BCNF because:

- Primary key is PaymentID
- All attributes depend directly on the primary key
- No transitive dependencies
-

7. Shipping Table:
8.

Non-trivial FDs:

- ShippingID → {OrderID, ShippingAddress, ShippingDate, DeliveryDate, ShippingStatus}

Normal Form: BCNF because:

- Primary key is ShippingID
- All attributes depend directly on the primary key
- No transitive dependencies
-

8. ReviewDetails Table:

Non-trivial FDs:

- ReviewID → {Rating, ReviewText, ReviewDate}

Normal Form: BCNF because:

- Primary key is ReviewID
- All attributes depend directly on the primary key
- No transitive dependencies

9. CustomerProductReviews Table:
10.

Non-trivial FDs:

- {CustomerID, ProductID, ReviewID} → all attributes
- Composite primary key with no additional attributes

Normal Form: BCNF because:

- Uses composite primary key
- Contains only key attributes
-

10. DiscountDetails Table:
11.

Non-trivial FDs:

- DiscountID → {DiscountPercentage, StartDate, EndDate}

Normal Form: BCNF because:

- Primary key is DiscountID
- All attributes depend directly on the primary key
- No transitive dependencies
-

11. ProductDiscount Table:
12.

Non-trivial FDs:

- {ProductID, DiscountID} → all attributes
- Composite primary key with no additional attributes

Normal Form: BCNF because:

- Uses composite primary key
- Contains only key attributes
-

12. OrderStatusHistory Table:
13.

Non-trivial FDs:

- StatusID → {OrderID, Status, StatusDate, Notes}

Normal Form: BCNF because:

- Primary key is StatusID
- All attributes depend directly on the primary key
- No transitive dependencies

SalesHistory Table:

Non-trivial FDs:

- SaleID → {OrderID, CustomerID, SaleDate, TotalAmount, ProfitMargin}
- OrderID → {CustomerID, TotalAmount} (derived from foreign key relationship)
-

Normal Form: BCNF because:

1. Primary key is SaleID
2. All attributes depend directly on the primary key
3. No partial dependencies exist (single-column primary key)
4. No transitive dependencies (all dependencies are through primary key)
5. Every determinant is a candidate key

Overall Database Design Analysis:

1. All tables are in BCNF (strongest normal form for practical use)
2. Proper use of foreign keys maintains referential integrity
3. Appropriate use of composite keys where needed
4. No partial or transitive dependencies
5. Proper separation of concerns (each table has a single purpose)
6. Appropriate use of constraints (UNIQUE, CHECK, NOT NULL)

This design ensures:

1. Minimal data redundancy
2. Data integrity
3. Elimination of update anomalies
4. Efficient data storage
5. Logical data organization

Frontend Design Tools and Techniques Used:

1. Streamlit Framework:
   Advantages:

- Rapid development and prototyping
- Built-in widgets and components
- Easy integration with Python
- Automatic responsive design
- Real-time updates
- Simple deployment process

Disadvantages:

- Limited customization compared to full-stack frameworks
- Performance limitations with large datasets
- Restricted layout options
- Single-page application structure only
- Limited browser support for some features

2. Custom CSS Implementation:
   Advantages:

- Modern and professional appearance
- Consistent styling across components
- Enhanced user experience
- Responsive design elements
- Custom color schemes and gradients

Disadvantages:

- Limited to Streamlit's HTML injection capabilities
- Cannot modify core Streamlit components
- Potential compatibility issues
- Maintenance overhead

3. Interactive Components:
   Advantages:

- User-friendly form inputs
- Dynamic query selection
- Real-time data updates
- Error handling and feedback
- Intuitive navigation

Disadvantages:

- Limited to Streamlit's widget set
- Some latency in updates
- Basic interaction patterns only

Conclusion of the Work:

1. Technical Achievements:

- Successfully implemented a database management interface
- Created reusable database connection functions
- Implemented CRUD operations
- Developed complex SQL queries
- Integrated error handling and validation

2. User Interface:

- Clean and modern design
- Intuitive navigation

- Responsive layout
- Clear data presentation
- Effective feedback mechanisms

3. Functionality:
- Comprehensive database operations
- Real-time data updates
- Multiple query options
- Data filtering and search capabilities
- Secure database interactions

4. Business Value:
- Efficient data management
- Improved user productivity
- Reduced error rates
- Better data visibility
- Enhanced decision-making capabilities

5. Future Improvements:
- Add authentication system
- Implement more advanced queries
- Enhance error handling
- Add data visualization
- Improve performance optimization

6. Learning Outcomes:
- Database management skills
- Frontend development experience
- SQL query optimization
- User interface design
- Error handling techniques

7. Overall Impact:
- Streamlined database operations
- Improved user experience
- Enhanced data accessibility
- Reduced manual effort
- Better system reliability

This implementation successfully combines frontend design with database functionality, creating a useful tool for database management while maintaining good user experience and system reliability.

Conclusion of the Work

This comprehensive e-commerce database management system represents a successful integration of backend database design and frontend user interface implementation. Here's a detailed conclusion of the entire project:

1. Database Design Achievement:
- Successfully implemented 13 interconnected tables covering all aspects of e-commerce operations

- Achieved BCNF normalization across all tables
- Established proper relationships and constraints
- Implemented comprehensive data integrity checks
- Created efficient indexing and key structures

2. System Functionality:

- Developed complete CRUD operations
- Implemented complex business logic
- Created comprehensive query system
- Established efficient data relationships
- Maintained data consistency and integrity

3. Frontend Implementation:

- Created user-friendly interface using Streamlit
- Implemented responsive design
- Developed interactive query system
- Added real-time data visualization
- Incorporated error handling and user feedback

4. Business Value Delivered:

a) Operational Efficiency:

- Streamlined order processing
- Improved inventory management
- Enhanced customer relationship management
- Efficient payment processing
- Automated shipping tracking

b) Data Management:

- Centralized data storage
- Improved data accessibility
- Enhanced data security
- Efficient query processing
- Comprehensive reporting capabilities

5. Technical Achievements:

- Normalized database design
- Efficient query optimization
- Secure data handling
- Scalable architecture
- Robust error handling

6. System Features:

- Customer management
- Product catalog management
- Order processing
- Payment handling
- Shipping tracking
- Review system
- Discount management
- Sales analysis

7. Key Benefits:

- Improved data organization
- Enhanced user experience
- Better decision-making capabilities
- Reduced operational errors
- Increased productivity
- Better customer service

8. Future Scope:
   a) Potential Enhancements:
      - Advanced analytics integration
      - Machine learning capabilities
      - Mobile application development
      - API integration
      - Enhanced security features

b) Scalability Options:

- Cloud deployment
- Performance optimization
- Additional feature integration
- Enhanced reporting capabilities
- Advanced user interface

9. Learning Outcomes:

- Database design principles
- Normalization techniques
- Frontend development skills
- SQL query optimization
- System integration methods

10. Overall Impact:

- Created a robust e-commerce database system
- Developed efficient management interface
- Implemented comprehensive business logic
- Established scalable architecture
- Delivered user-friendly solution

This project successfully demonstrates the implementation of a complete e-commerce database system, combining theoretical database concepts with practical application development, resulting in a functional and efficient solution for managing e-commerce operations.

The system provides a solid foundation for future enhancements and serves as a valuable tool for managing complex e-commerce operations while maintaining data integrity and user accessibility. The combination of proper database design principles and modern frontend implementation creates a powerful solution for business needs.