DATA MODELING PROJECT

ONLINE STORE DATABASE SYSTEM

BY GROUP STYX

Group Members:

1. **Srinivas Pallapu**
2. **Shreya Padala**
3. **Sai Chaitanya**

PROJECT DESCRIPTION:

Online mart is a shopping platform that provides customers with the convenience of purchasing products online. The goal of the system is to improve shopping experience by offering a customized shopping service that involves a dedicated personal shopper who will take care of the shopping on behalf of the customer.

The Database will maintain customer data, including personal information, buying history, and payment details. This database will enable the system to provide personalized suggestions for upcoming purchases based on the customer's buying history and preferences.

One of the key objectives of the database is to ensure safe deliveries by maintaining accurate records of delivery addresses and tracking the progress of deliveries. The system will also maintain a record of payments, ensuring that all payments are processed accurately and securely.

The Online Mart management system aims to increase customer productivity by minimizing the time and effort required to complete a shopping trip. By providing a personal shopper who handles the actual shopping, customers can focus on other tasks, such as work or family commitments.

The scope of the database:

- The database of the online mart will store customer information such as name, address, email, phone number, and other relevant details. This data will be utilized to enhance the shopping experience for customers and keep track of their interactions with the online store.
- The database system will maintain a record of the customer’s buying history, including past purchases, returns, and exchanges. This data will be used to provide personalized recommendations for upcoming purchases.
- The database system will maintain a record of payment details, including credit card information, payment history, and payment processing. This data will be used to ensure accurate payment processing and secure transactions.
- The database system will maintain a record of product data, including product descriptions, images, prices, and availability. This data will be used to provide customers with accurate and up-to-date information about the products available for purchase.
- The database system will maintain a record of personal shopper data, including availability, location, and shopping history. This data will be used to match customers with personal shoppers and to track the progress of shopping trips.

The objectives of the database:

- Personalized Shopping Experience: The database system will use customer data, buying history, and product data to provide personalized recommendations and enhance the shopping experience.
- Efficient Payment Processing: The database system will ensure accurate payment processing and secure transactions.
- Safe Deliveries: The database system will maintain accurate records of delivery addresses and track the progress of deliveries to ensure safe and timely deliveries.
- Increased Productivity: The database system will minimize the time and effort required to complete a shopping trip by providing a personal shopper who handles the actual shopping.

User requirements:

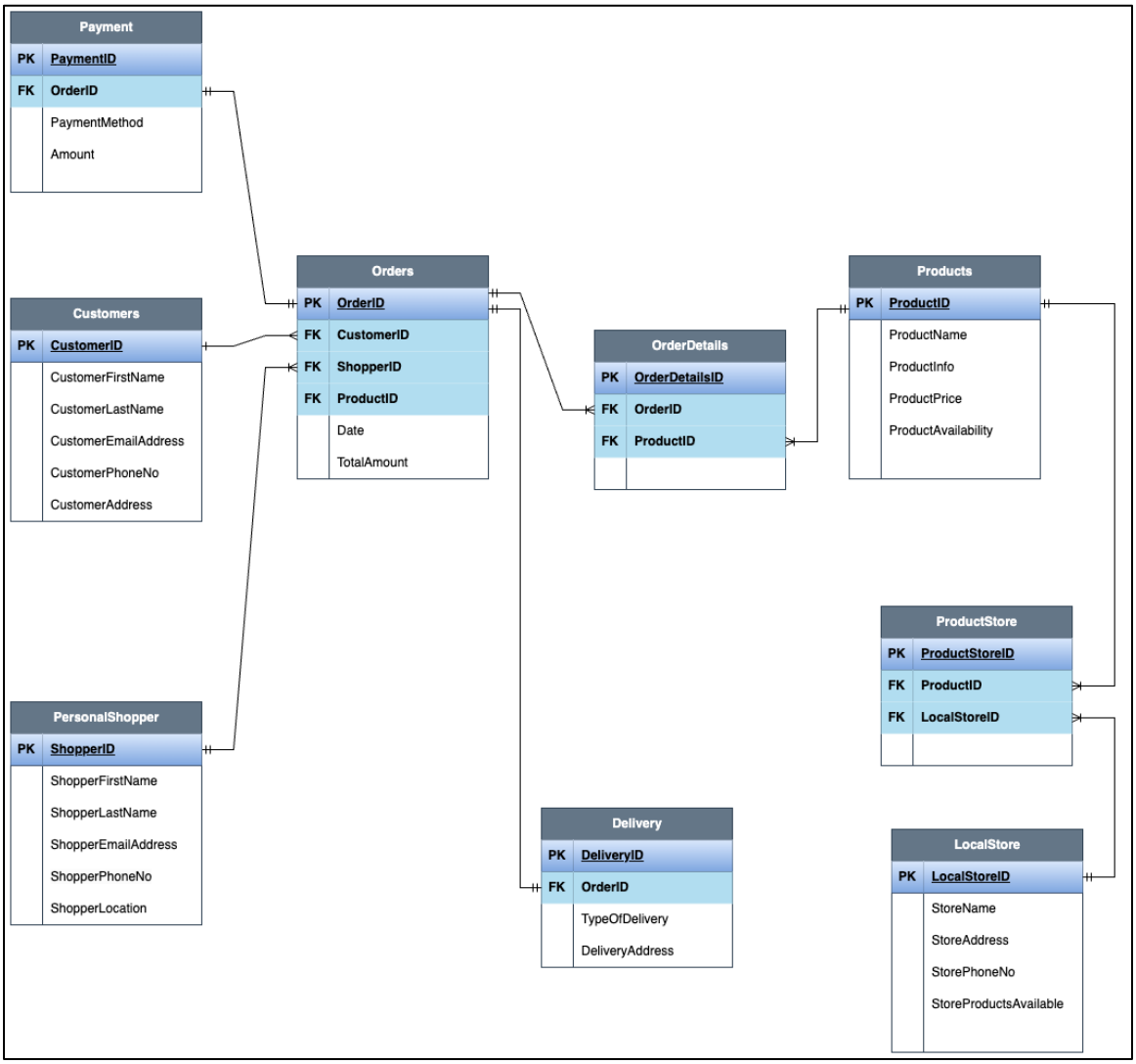
1. Customers can create an account and view their order history.
2. Customers can search for products by name or availability status.
3. Customers can add products to their cart and check out with their preferred payment method.
4. Personal shoppers can view their assigned customers and their current orders.
5. Personal shoppers can communicate with their customers through email or phone.
6. Each product can be available in multiple local stores.
7. Customers can select a local store to pick up their order.
8. Customers can track the status of their delivery or pickup order.
9. The system can automatically calculate the total amount of each order, including tax and shipping fees.
10. Customers can leave feedback or ratings on products or personal shoppers.

Business rules:

1. Customers can place orders and have a personal shopper assigned to them.
2. Each customer can have only one personal shopper assigned to them at a time.
3. Each order must have a customer, a personal shopper, and at least one product.
4. Each product can be available in multiple local stores.
5. Each order can have multiple products.
6. Each product can appear in multiple orders.
7. Each order can have one payment and one delivery.
8. Each payment must be for one order and one customer.
9. Each delivery must be for one order.
10. Every customer must possess a distinct customer ID along with their first and last names, phone number, and address.

11. Orders of every customer is maintained, and it consists of customer id, shopper id, product id, date of purchase and total amount.
12. Every personal shopper must contain shopperid, firstname, lastname, phone number and location.
13. Local store contains store name, address, phone number, products availability and their status.
14. Every order detail has order id and product id included in it.

Entity Relationship Diagram:



Data Dictionary:

S No.	Table Name	Attribute Name	Description	Format	Range	Type	Key	FK referenced table
1	Customers	CustomerID	Unique number given to customer	99999	10000-99999	INT	PK	
		CustomerFirstName	customer's First Name	xxxxxxx		VARCHAR		
		CustomerLastName	customer's last Name	xxxxx		VARCHAR		
		CustomerEmailAddress	customer's email address	xxx@gmail.com		VARCHAR		
		CustomerPhoneNo	customer's phone number	9999999999		VARCHAR		
		CustomerAddress	customer's Address	xxxxxxx		VARCHAR		
2	PersonalShopper	ShopperID	Unique number given to personal shopper	999999	100000-999999	INT	PK	
		ShopperFirstName	shopper's first name	xxxxxxx		VARCHAR		
		ShopperLastName	shopper's last name	xxxxxxx		VARCHAR		
		ShopperEmailAddress	shopper's email address	xxx@gmail.com		VARCHAR		
		ShopperPhoneNumber	shopper's phone number	9999999999		INT		
		ShopperLocation	shopper's location	xxxxxxx		VARCHAR		

3	Orders	OrderID	Unique number given to order	9999999	1000000-9999999	INT	PK	
		CustomerID	Unique number given to customer	99999	10000-99999	INT	FK	Customers
		ShopperID	Unique number given to personal shopper	999999	100000-999999	INT	FK	PersonalShopper
		ProductID	unique number given to the product	99999999	10000000-99999999	INT	FK	Products
		Date	date of order	mm/dd/yy		DATE		
		TotalAmount	total amount for order	9,999,999		FLOAT		
4	OrderDetails	OrderDetailsID	Unique number given to order details	999999999	100000000-999999999	INT	PK	
		OrderID	Unique number given to order	9999999	1000000-9999999	INT	FK	Orders
		ProductID	unique number given to the product	99999999	10000000-99999999	INT	FK	Products

5	Products	ProductID	unique number given to the product	99999999	10000000-99999999	INT	PK	
		ProductName	product's name	xxxxxxx		VARCHAR		
		ProductInfo	product's info	xxxxxxx		VARCHAR		
		ProductPrice	product price	99.8		FLOAT		
		ProductAvailability	product availability	available/not available		VARCHAR		
6	ProductStore	ProductStoreID	Unique number given to product store	9999999999	1000000000-9999999999	BIGINT	PK	
		ProductID	unique number given to the product	99999999	10000000-99999999	INT	FK	Products
		LocalStoreID	unique number given to the local store	99999999999	1000000000-9999999999	BIGINT	FK	LocalStore
7	LocalStore	LocalStoreID	unique number given to the local store	99999999999	1000000000-9999999999	BIGINT	PK	
		StoreName	Store name	xxxxxxx		VARCHAR		
		StoreAddress	store address	xxxx		VARCHAR		
		StorePhoneNo	store phone number	9999999999		VARCHAR		
		ProductsAvailablility	product availability	available/not available		VARCHAR		

8	Payment	PaymentID	Unique number given to payment	1	1-999999999999	BIGINT	PK	
		OrderID	Unique number given to order	9999999	1000000-99999999	INT	FK	Orders
		PaymentMethod	method of payment(credit card/debit card/cash on delivery)	credit card/debit card/cash on delivery		VARCHAR		
		Amount	amount paid	9,999,999		FLOAT		
9	Delivery	DeliveryID	Unique number given to delivery	1	1-999999999999	BIGINT	PK	
		OrderID	Unique number given to order	9999999	1000000-99999999	INT	FK	Orders
		TypeOfDelivery	type delivery pickup or door delivery	Pick up / door		VARCHAR		
		DeliveryAddress	Delivery Address	xxxxxxx		VARCHAR		

Entity Generation and Data Entry:

Customers Table

1. **Create table statement**

```
CREATE TABLE `Customers` (
  `CustomerID` int NOT NULL,
  `CustomerFirstName` varchar(50)  NOT NULL,
  `CustomerLastName` varchar(50)  NOT NULL,
  `CustomerEmailAddress` varchar(100)  NOT NULL,
  `CustomerPhoneNo` varchar(20)  NOT NULL,
  `CustomerAddress` varchar(200)  NOT NULL
) ;
```

```
CREATE TABLE Customers (
  CustomerID INT NOT NULL AUTO_INCREMENT,
  CustomerFirstName VARCHAR(50) NOT NULL,
  CustomerLastName VARCHAR(50) NOT NULL,
  CustomerEmailAddress VARCHAR(100) NOT NULL,
  CustomerPhoneNo VARCHAR(20) NOT NULL,
  CustomerAddress VARCHAR(200) NOT NULL,
  PRIMARY KEY (CustomerID)
);
```

2. **Insert Statement**

```
INSERT INTO `Customers` (`CustomerID`, `CustomerFirstName`, `CustomerLastName`,
`CustomerEmailAddress`, `CustomerPhoneNo`, `CustomerAddress`) VALUES
(10000, 'John', 'Doe', 'johndoe@example.com', '1234567890', '123 Main St'),
(10001, 'Jane', 'Doe', 'janedoe@example.com', '0987654321', '456 Oak Ave'),
(10002, 'Bob', 'Smith', 'bobsmith@example.com', '5555555555', '789 Maple Rd'),
```

(10003, 'David', 'Lee', 'david.lee@hotmail.com', '3456789012', '789 Maple Rd'),

(10004, 'Sarah', 'Brown', 'sarah.brown@gmail.com', '4567890123', '345 Cedar St'),

(10005, 'Michael', 'Johnson', 'michaelj@gmail.com', '5678901234', '678 Elm St'),

(10006, 'Rachel', 'Davis', 'rachel.davis@yahoo.com', '6789012345', '901 Pine Ave'),

(10007, 'Andrew', 'Wilson', 'andrew.wilson@gmail.com', '7890123456', '234 Oak St'),

(10008, 'Elizabeth', 'Taylor', 'elizabeth.taylor@hotmail.com', '8901234567', '567 Maple Ave'),

(10009, 'William', 'Jones', 'william.jones@yahoo.com', '9012345678', '890 Cedar Rd'),

(10010, 'Emily', 'Lee', 'emily.lee@gmail.com', '1234567891', '123 Pine St'),

(10011, 'Ethan', 'Brown', 'ethan.brown@yahoo.com', '2345678902', '456 Maple Ave'),

(10012, 'Olivia', 'Johnson', 'olivia.johnson@hotmail.com', '3456789013', '789 Cedar St'),

(10013, 'Daniel', 'Davis', 'daniel.davis@gmail.com', '4567890124', '345 Oak Ave'),

(10014, 'Sophia', 'Wilson', 'sophia.wilson@yahoo.com', '5678901235', '678 Pine Rd'),

(10015, 'Matthew', 'Taylor', 'matthew.taylor@gmail.com', '6789012346', '901 Maple Ave'),

(10016, 'Isabella', 'Jones', 'isabella.jones@yahoo.com', '7890123457', '234 Cedar St'),

(10017, 'Benjamin', 'Lee', 'benjamin.lee@hotmail.com', '8901234568', '567 Pine Ave'),

(10018, 'Amelia', 'Brown', 'amelia.brown@gmail.com', '9012345679', '890 Oak St'),

(10019, 'Mia', 'Johnson', 'mia.johnson@yahoo.com', '1234567892', '123 Maple Rd'),

(10020, 'Alexander', 'Davis', 'alexander.davis@hotmail.com', '2345678903', '456 Cedar St'),

(10021, 'Abigail', 'Wilson', 'abigail.wilson@gmail.com', '3456789014', '789 Pine Ave'),

(10022, 'Ryan', 'Taylor', 'ryan.taylor@yahoo.com', '4567890125', '345 bryan st'),

(10023, 'Alice', 'Anderson', 'aliceanderson@example.com', '1112223333', '789 Elm St'),

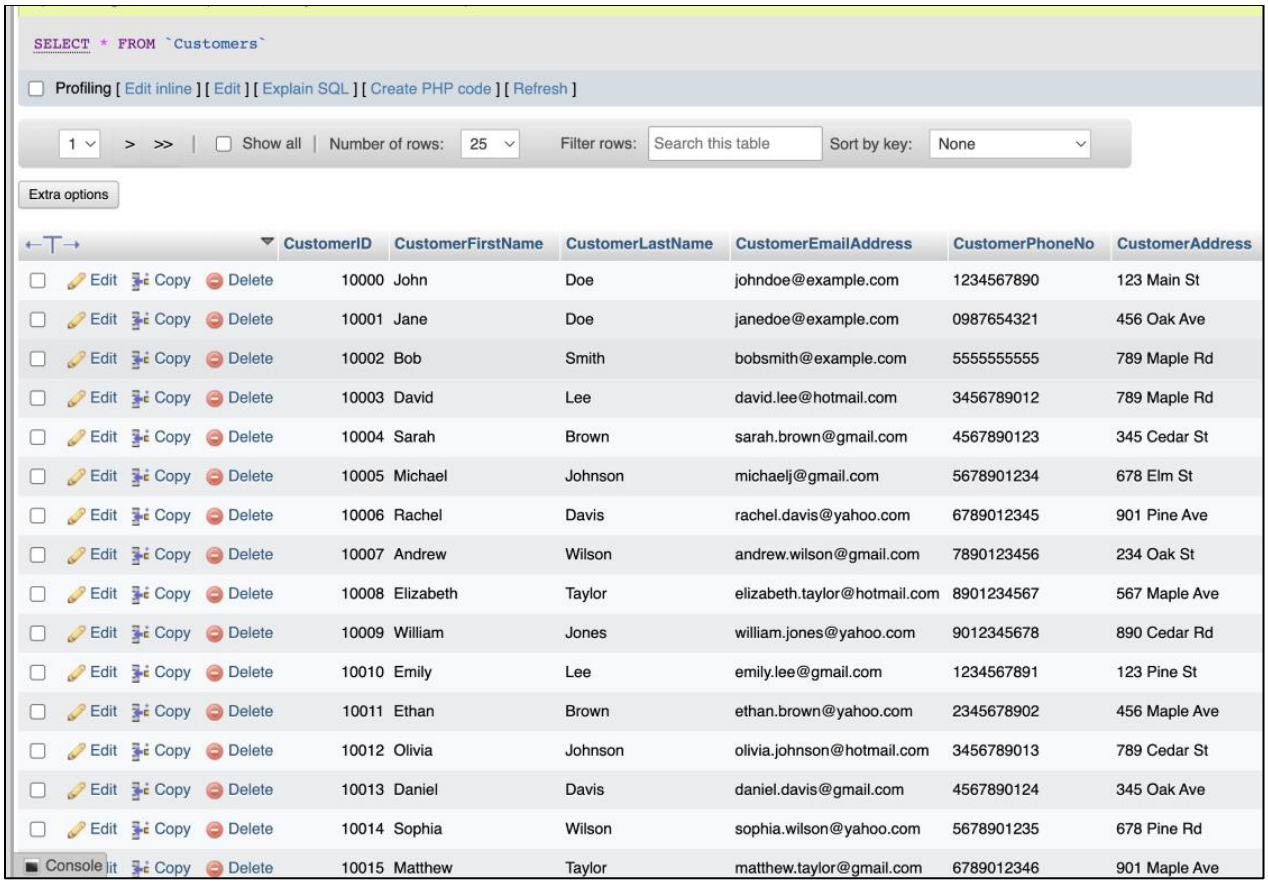
(10024, 'Mike', 'Miller', 'mikemiller@example.com', '4445556666', '456 Maple Rd'),

(10025, 'Mike', 'tyson', 'miketyson@example.com', '4436355353', '537 fry Rd');

```
INSERT INTO Customers (CustomerID, CustomerFirstName, CustomerLastName, CustomerEmailAddress, CustomerPhoneNo, CustomerAddress) VALUES
(10000, 'John', 'Doe', 'johndoe@example.com', '1234567890', '123 Main St'),
(10001, 'Jane', 'Doe', 'janedoe@example.com', '0987654321', '456 Oak Ave'),
(10002, 'Bob', 'Smith', 'bobsmith@example.com', '5555555555', '789 Maple Rd'),
(10003, 'David', 'Lee', 'david.lee@hotmail.com', '3456789012', '789 Maple Rd'),
(10004, 'Sarah', 'Brown', 'sarah.brown@gmail.com', '4567890123', '345 Cedar St'),
(10005, 'Michael', 'Johnson', 'michaelj@gmail.com', '5678901234', '678 Elm St'),
(10006, 'Rachel', 'Davis', 'rachel.davis@yahoo.com', '6789012345', '901 Pine Ave'),
(10007, 'Andrew', 'Wilson', 'andrew.wilson@gmail.com', '7890123456', '234 Oak St'),
(10008, 'Elizabeth', 'Taylor', 'elizabeth.taylor@hotmail.com', '8901234567', '567 Maple Ave'),
(10009, 'William', 'Jones', 'william.jones@yahoo.com', '9012345678', '890 Cedar Rd'),
(10010, 'Emily', 'Lee', 'emily.lee@gmail.com', '1234567891', '123 Pine St'),
(10011, 'Ethan', 'Brown', 'ethan.brown@yahoo.com', '2345678902', '456 Maple Ave'),
(10012, 'Olivia', 'Johnson', 'olivia.johnson@hotmail.com', '3456789013', '789 Cedar St'),
(10013, 'Daniel', 'Davis', 'daniel.davis@gmail.com', '4567890124', '345 Oak Ave'),
(10014, 'Sophia', 'Wilson', 'sophia.wilson@yahoo.com', '5678901235', '678 Pine Rd'),
(10015, 'Matthew', 'Taylor', 'matthew.taylor@gmail.com', '6789012346', '901 Maple Ave'),
(10016, 'Isabella', 'Jones', 'isabella.jones@yahoo.com', '7890123457', '234 Cedar St'),
(10017, 'Benjamin', 'Lee', 'benjamin.lee@hotmail.com', '8901234568', '567 Pine Ave'),
(10018, 'Amelia', 'Brown', 'amelia.brown@gmail.com', '9012345679', '890 Oak St'),
(10019, 'Mia', 'Johnson', 'mia.johnson@yahoo.com', '1234567892', '123 Maple Rd'),
(10020, 'Alexander', 'Davis', 'alexander.davis@hotmail.com', '2345678903', '456 Cedar St'),
(10021, 'Abigail', 'Wilson', 'abigail.wilson@gmail.com', '3456789014', '789 Pine Ave'),
(10022, 'Ryan', 'Taylor', 'ryan.taylor@yahoo.com', '4567890125', '345 bryan st'),
(10023, 'Alice', 'Anderson', 'aliceanderson@example.com', '1112223333', '789 Elm St'),
(10024, 'Mike', 'Miller', 'mikemiller@example.com', '4445556666', '456 Maple Rd'),
(10025, 'Mike', 'tyson', 'miketyson@example.com', '4436355353', '537 fry Rd');
```

3. Results:

SELECT * FROM Customers;



```
SELECT * FROM `Customers`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1 > >> | Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	CustomerID	CustomerFirstName	CustomerLastName	CustomerEmailAddress	CustomerPhoneNo	CustomerAddress
Edit Copy Delete	10000	John	Doe	johndoe@example.com	1234567890	123 Main St
Edit Copy Delete	10001	Jane	Doe	janedoe@example.com	0987654321	456 Oak Ave
Edit Copy Delete	10002	Bob	Smith	bobsmith@example.com	5555555555	789 Maple Rd
Edit Copy Delete	10003	David	Lee	david.lee@hotmail.com	3456789012	789 Maple Rd
Edit Copy Delete	10004	Sarah	Brown	sarah.brown@gmail.com	4567890123	345 Cedar St
Edit Copy Delete	10005	Michael	Johnson	michaelj@gmail.com	5678901234	678 Elm St
Edit Copy Delete	10006	Rachel	Davis	rachel.davis@yahoo.com	6789012345	901 Pine Ave
Edit Copy Delete	10007	Andrew	Wilson	andrew.wilson@gmail.com	7890123456	234 Oak St
Edit Copy Delete	10008	Elizabeth	Taylor	elizabeth.taylor@hotmail.com	8901234567	567 Maple Ave
Edit Copy Delete	10009	William	Jones	william.jones@yahoo.com	9012345678	890 Cedar Rd
Edit Copy Delete	10010	Emily	Lee	emily.lee@gmail.com	1234567891	123 Pine St
Edit Copy Delete	10011	Ethan	Brown	ethan.brown@yahoo.com	2345678902	456 Maple Ave
Edit Copy Delete	10012	Olivia	Johnson	olivia.johnson@hotmail.com	3456789013	789 Cedar St
Edit Copy Delete	10013	Daniel	Davis	daniel.davis@gmail.com	4567890124	345 Oak Ave
Edit Copy Delete	10014	Sophia	Wilson	sophia.wilson@yahoo.com	5678901235	678 Pine Rd
Edit Copy Delete	10015	Matthew	Taylor	matthew.taylor@gmail.com	6789012346	901 Maple Ave

Console

4. **Description of the table Customers:**

   The **Customers** table that stores information about customers. Each row in the table represents a single customer and includes columns for their ID, first name, last name, email address, phone number, and physical address.

PersonalShopper Table

1. **Create table statement**

CREATE TABLE `PersonalShopper` (

`ShopperID` int NOT NULL,

`ShopperFirstName` varchar(50) NOT NULL,

`ShopperLastName` varchar(50) NOT NULL,

`ShopperEmailAddress` varchar(100) NOT NULL,

`ShopperPhoneNumber` varchar(20) NOT NULL,

`ShopperLocation` varchar(200) NOT NULL

);

```
CREATE TABLE PersonalShopper (
  ShopperID INT NOT NULL AUTO_INCREMENT,
  ShopperFirstName VARCHAR(50) NOT NULL,
  ShopperLastName VARCHAR(50) NOT NULL,
  ShopperEmailAddress VARCHAR(100) NOT NULL,
  ShopperPhoneNumber VARCHAR(20) NOT NULL,
  ShopperLocation VARCHAR(200) NOT NULL,
  PRIMARY KEY (ShopperID)
);
```

2. **Insert table statement**

INSERT INTO `PersonalShopper` (`ShopperID`, `ShopperFirstName`, `ShopperLastName`, `ShopperEmailAddress`, `ShopperPhoneNumber`, `ShopperLocation`) VALUES

(100001, 'John', 'Doe', 'john.doe@gmail.com', '9876543210', 'New York'),

(100002, 'Alice', 'Smith', 'alice.smith@gmail.com', '1234567890', 'Los Angeles'),

(100003, 'Robert', 'Johnson', 'robert.johnson@gmail.com', '7865432190', 'Chicago'),

(100004, 'Emily', 'Davis', 'emily.davis@gmail.com', '7890123456', 'San Francisco'),

(100005, 'Michael', 'Wilson', 'michael.wilson@gmail.com', '2345678901', 'Houston'),

(100006, 'Jessica', 'Brown', 'jessica.brown@gmail.com', '9012345678', 'Seattle'),

(100007, 'David', 'Garcia', 'david.garcia@gmail.com', '3456789012', 'Miami'),

(100008, 'Olivia', 'Martinez', 'olivia.martinez@gmail.com', '6789012345', 'Dallas'),

(100009, 'Daniel', 'Anderson', 'daniel.anderson@gmail.com', '9012345678', 'Phoenix'),

(100010, 'Samantha', 'Taylor', 'samantha.taylor@gmail.com', '2345678901', 'Philadelphia'),

(100011, 'Joseph', 'Hernandez', 'joseph.hernandez@gmail.com', '7890123456', 'Boston'),

(100012, 'Ava', 'Moore', 'ava.moore@gmail.com', '9012345678', 'Washington DC'),

(100013, 'Christopher', 'Martin', 'christopher.martin@gmail.com', '3456789012', 'Atlanta'),

(100014, 'Mia', 'Jackson', 'mia.jackson@gmail.com', '6789012345', 'Denver'),

(100015, 'Andrew', 'Lee', 'andrew.lee@gmail.com', '9012345678', 'San Diego'),

(100016, 'Sophia', 'Gonzalez', 'sophia.gonzalez@gmail.com', '2345678901', 'Austin'),

(100017, 'William', 'Perez', 'william.perez@gmail.com', '7890123456', 'San Antonio'),

(100018, 'Isabella', 'Turner', 'isabella.turner@gmail.com', '9012345678', 'Las Vegas'),

(100019, 'Ethan', 'Phillips', 'ethan.phillips@gmail.com', '3456789012', 'Kansas City'),

(100020, 'Charlotte', 'Campbell', 'charlotte.campbell@gmail.com', '6789012345', 'New Orleans'),

(100021, 'Alexander', 'Parker', 'alexander.parker@gmail.com', '9012345678', 'Nashville'),

(100022, 'Amelia', 'Evans', 'amelia.evans@gmail.com', '2345678901', 'Seattle'),

(100023, 'Ryan', 'Edwards', 'ryan.edwards@gmail.com', '7890123456', 'Orlando'),

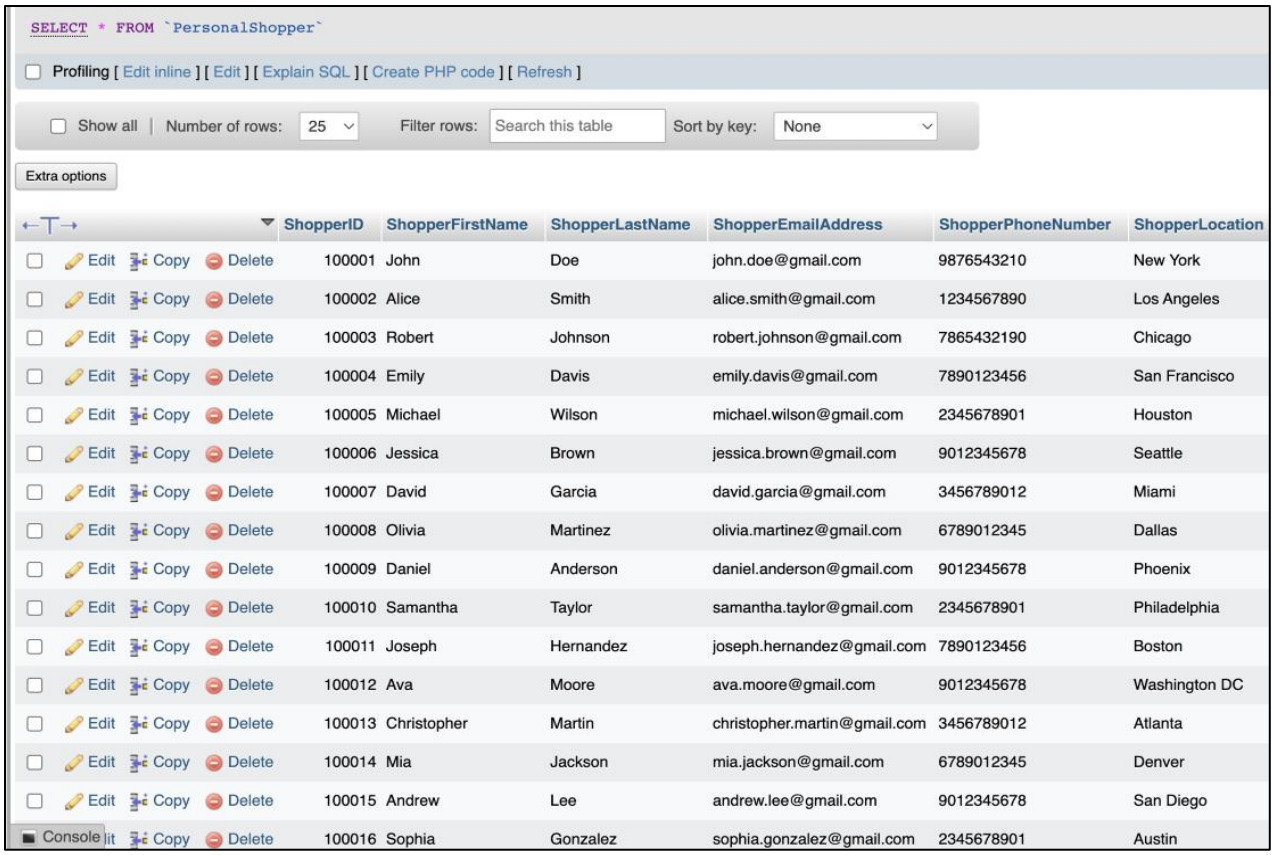
(100024, 'Madison', 'Collins', 'madison.collins@gmail.com', '9012345678', 'Salt'),

(100025, 'Madry', 'Coloni', 'Madry.Coloni@gmail.com', '901234324', 'denton');

```sql
INSERT INTO PersonalShopper (ShopperID, ShopperFirstName, ShopperLastName, ShopperEmailAddress, ShopperPhoneNumber, ShopperLocation) VALUES
(100001, 'John', 'Doe', 'john.doe@gmail.com', '9876543210', 'New York'),
(100002, 'Alice', 'Smith', 'alice.smith@gmail.com', '1234567890', 'Los Angeles'),
(100003, 'Robert', 'Johnson', 'robert.johnson@gmail.com', '7865432190', 'Chicago'),
(100004, 'Emily', 'Davis', 'emily.davis@gmail.com', '7890123456', 'San Francisco'),
(100005, 'Michael', 'Wilson', 'michael.wilson@gmail.com', '2345678901', 'Houston'),
(100006, 'Jessica', 'Brown', 'jessica.brown@gmail.com', '9012345678', 'Seattle'),
(100007, 'David', 'Garcia', 'david.garcia@gmail.com', '3456789012', 'Miami'),
(100008, 'Olivia', 'Martinez', 'olivia.martinez@gmail.com', '6789012345', 'Dallas'),
(100009, 'Daniel', 'Anderson', 'daniel.anderson@gmail.com', '9012345678', 'Phoenix'),
(100010, 'Samantha', 'Taylor', 'samantha.taylor@gmail.com', '2345678901', 'Philadelphia'),
(100011, 'Joseph', 'Hernandez', 'joseph.hernandez@gmail.com', '7890123456', 'Boston'),
(100012, 'Ava', 'Moore', 'ava.moore@gmail.com', '9012345678', 'Washington DC'),
(100013, 'Christopher', 'Martin', 'christopher.martin@gmail.com', '3456789012', 'Atlanta'),
(100014, 'Mia', 'Jackson', 'mia.jackson@gmail.com', '6789012345', 'Denver'),
(100015, 'Andrew', 'Lee', 'andrew.lee@gmail.com', '9012345678', 'San Diego'),
(100016, 'Sophia', 'Gonzalez', 'sophia.gonzalez@gmail.com', '2345678901', 'Austin'),
(100017, 'William', 'Perez', 'william.perez@gmail.com', '7890123456', 'San Antonio'),
(100018, 'Isabella', 'Turner', 'isabella.turner@gmail.com', '9012345678', 'Las Vegas'),
(100019, 'Ethan', 'Phillips', 'ethan.phillips@gmail.com', '3456789012', 'Kansas City'),
(100020, 'Charlotte', 'Campbell', 'charlotte.campbell@gmail.com', '6789012345', 'New Orleans'),
(100021, 'Alexander', 'Parker', 'alexander.parker@gmail.com', '9012345678', 'Nashville'),
(100022, 'Amelia', 'Evans', 'amelia.evans@gmail.com', '2345678901', 'Seattle'),
(100023, 'Ryan', 'Edwards', 'ryan.edwards@gmail.com', '7890123456', 'Orlando'),
(100024, 'Madison', 'Collins', 'madison.collins@gmail.com', '9012345678', 'Salt'),
(100025, 'Madry', 'Coloni', 'Madry.Coloni@gmail.com', '901234324', 'denton');
```

3. Results

SELECT * FROM PersonalShopper;



```
SELECT * FROM `PersonalShopper`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	ShopperID	ShopperFirstName	ShopperLastName	ShopperEmailAddress	ShopperPhoneNumber	ShopperLocation
Edit Copy Delete	100001	John	Doe	john.doe@gmail.com	9876543210	New York
Edit Copy Delete	100002	Alice	Smith	alice.smith@gmail.com	1234567890	Los Angeles
Edit Copy Delete	100003	Robert	Johnson	robert.johnson@gmail.com	7865432190	Chicago
Edit Copy Delete	100004	Emily	Davis	emily.davis@gmail.com	7890123456	San Francisco
Edit Copy Delete	100005	Michael	Wilson	michael.wilson@gmail.com	2345678901	Houston
Edit Copy Delete	100006	Jessica	Brown	jessica.brown@gmail.com	9012345678	Seattle
Edit Copy Delete	100007	David	Garcia	david.garcia@gmail.com	3456789012	Miami
Edit Copy Delete	100008	Olivia	Martinez	olivia.martinez@gmail.com	6789012345	Dallas
Edit Copy Delete	100009	Daniel	Anderson	daniel.anderson@gmail.com	9012345678	Phoenix
Edit Copy Delete	100010	Samantha	Taylor	samantha.taylor@gmail.com	2345678901	Philadelphia
Edit Copy Delete	100011	Joseph	Hernandez	joseph.hernandez@gmail.com	7890123456	Boston
Edit Copy Delete	100012	Ava	Moore	ava.moore@gmail.com	9012345678	Washington DC
Edit Copy Delete	100013	Christopher	Martin	christopher.martin@gmail.com	3456789012	Atlanta
Edit Copy Delete	100014	Mia	Jackson	mia.jackson@gmail.com	6789012345	Denver
Edit Copy Delete	100015	Andrew	Lee	andrew.lee@gmail.com	9012345678	San Diego
Edit Copy Delete	100016	Sophia	Gonzalez	sophia.gonzalez@gmail.com	2345678901	Austin

Console

4. **Description of the** PersonalShopper **table:**

   The PersonalShopper table contains information about personal shoppers, including their unique ShopperID, first name, last name, email address, phone number, and location. The table is used to store data about individuals who offer personal shopping services to customers.

Products Table

1. **Create table statement**

```
CREATE TABLE `Products` (
  `ProductID` int NOT NULL,
  `ProductName` varchar(100)  NOT NULL,
  `ProductInfo` varchar(200)  NOT NULL,
  `ProductPrice` float NOT NULL,
  `ProductAvailability` varchar(20)  NOT NULL
```

);

```
CREATE TABLE Products (
  ProductID INT NOT NULL AUTO_INCREMENT,
  ProductName VARCHAR(100) NOT NULL,
  ProductInfo VARCHAR(200) NOT NULL,
  ProductPrice FLOAT NOT NULL,
  ProductAvailability VARCHAR(20) NOT NULL,
  PRIMARY KEY (ProductID)
);
```

2. Insert statement

INSERT INTO `Products` (`ProductID`, `ProductName`, `ProductInfo`, `ProductPrice`, `ProductAvailability`) VALUES

(10000001, 'Bananas', 'Fresh bananas from South America', 0.99, 'Available'),

(10000002, 'Apples', 'Crisp red apples from Washington', 1.49, 'Available'),

(10000003, 'Oranges', 'Juicy oranges from Florida', 1.19, 'Available'),

(10000004, 'Carrots', 'Organic carrots from California', 1.29, 'Available'),

(10000005, 'Broccoli', 'Fresh broccoli from local farm', 1.99, 'Not Available'),

(10000006, 'Tomatoes', 'Vine-ripened tomatoes from local farm', 2.49, 'Available'),

(10000007, 'Lettuce', 'Crisp lettuce from local farm', 1.99, 'Available'),

(10000008, 'Cucumbers', 'Fresh cucumbers from local farm', 1.29, 'Available'),

(10000009, 'Potatoes', 'Organic potatoes from Idaho', 1.49, 'Available'),

(10000010, 'Onions', 'Sweet onions from local farm', 0.99, 'Available'),

(10000011, 'Chicken Breast', 'Boneless, skinless chicken breast', 3.99, 'Not Available'),

(10000012, 'Ground Beef', 'Fresh ground beef, 90% lean', 5.99, 'Available'),

(10000013, 'Salmon Fillet', 'Fresh Atlantic salmon fillet', 10.99, 'Available'),

(10000014, 'Tilapia Fillet', 'Fresh tilapia fillet', 7.99, 'Available'),

(10000015, 'Pork Chops', 'Bone-in pork chops', 4.99, 'Available'),

(10000016, 'Bacon', 'Hickory-smoked bacon', 3.99, 'Not Available'),

(10000017, 'Eggs', 'Farm-fresh large eggs', 1.99, 'Available'),

(10000018, 'Milk', 'Whole milk, 1 gallon', 2.49, 'Available'),

(10000019, 'Bread', 'Freshly baked whole wheat bread', 1.99, 'Available'),

(10000020, 'Pasta', 'Organic spaghetti pasta', 2.99, 'Available'),

(10000021, 'Rice', 'Long grain white rice, 2 lbs', 1.99, 'Not Available'),

(10000022, 'Cereal', 'Honey Nut Oat cereal, 16 oz', 3.49, 'Available'),

(10000023, 'Peanut Butter', 'Creamy peanut butter, 16 oz', 2.99, 'Available'),

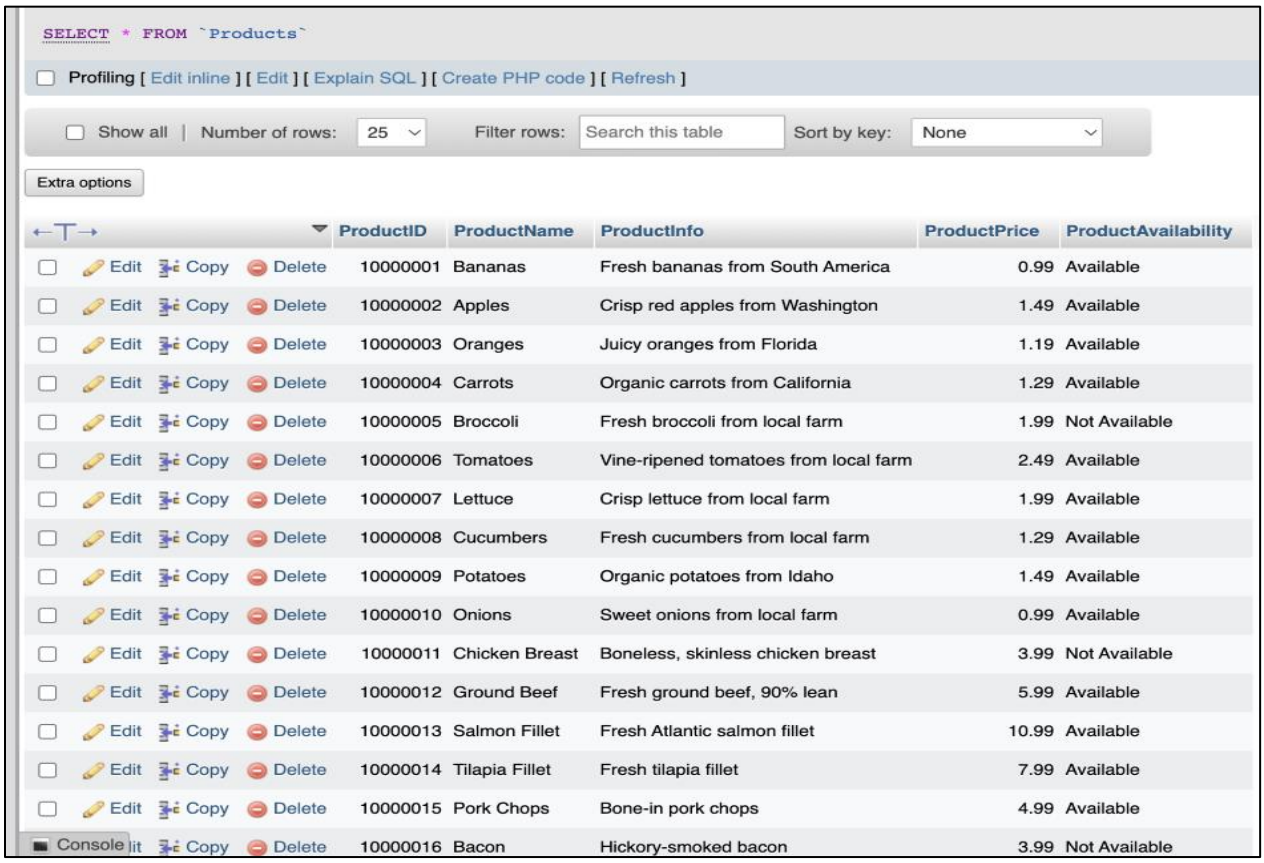
(10000024, 'Jelly', 'Grape jelly, 18 oz', 1.99, 'Available'),

(10000025, 'Cheese', 'Sharp cheddar cheese, 8 oz', 3.49, 'Available');

```sql
INSERT INTO Products (ProductID, ProductName, ProductInfo, ProductPrice, ProductAvailability)
VALUES (10000001, 'Bananas', 'Fresh bananas from South America', 0.99, 'Available'),
       (10000002, 'Apples', 'Crisp red apples from Washington', 1.49, 'Available'),
       (10000003, 'Oranges', 'Juicy oranges from Florida', 1.19, 'Available'),
       (10000004, 'Carrots', 'Organic carrots from California', 1.29, 'Available'),
       (10000005, 'Broccoli', 'Fresh broccoli from local farm', 1.99, 'Not Available'),
       (10000006, 'Tomatoes', 'Vine-ripened tomatoes from local farm', 2.49, 'Available'),
       (10000007, 'Lettuce', 'Crisp lettuce from local farm', 1.99, 'Available'),
       (10000008, 'Cucumbers', 'Fresh cucumbers from local farm', 1.29, 'Available'),
       (10000009, 'Potatoes', 'Organic potatoes from Idaho', 1.49, 'Available'),
       (10000010, 'Onions', 'Sweet onions from local farm', 0.99, 'Available'),
       (10000011, 'Chicken Breast', 'Boneless, skinless chicken breast', 3.99, 'Not Available'),
       (10000012, 'Ground Beef', 'Fresh ground beef, 90% lean', 5.99, 'Available'),
       (10000013, 'Salmon Fillet', 'Fresh Atlantic salmon fillet', 10.99, 'Available'),
       (10000014, 'Tilapia Fillet', 'Fresh tilapia fillet', 7.99, 'Available'),
       (10000015, 'Pork Chops', 'Bone-in pork chops', 4.99, 'Available'),
       (10000016, 'Bacon', 'Hickory-smoked bacon', 3.99, 'Not Available'),
       (10000017, 'Eggs', 'Farm-fresh large eggs', 1.99, 'Available'),
       (10000018, 'Milk', 'Whole milk, 1 gallon', 2.49, 'Available'),
       (10000019, 'Bread', 'Freshly baked whole wheat bread', 1.99, 'Available'),
       (10000020, 'Pasta', 'Organic spaghetti pasta', 2.99, 'Available'),
       (10000021, 'Rice', 'Long grain white rice, 2 lbs', 1.99, 'Not Available'),
       (10000022, 'Cereal', 'Honey Nut Oat cereal, 16 oz', 3.49, 'Available'),
       (10000023, 'Peanut Butter', 'Creamy peanut butter, 16 oz', 2.99, 'Available'),
       (10000024, 'Jelly', 'Grape jelly, 18 oz', 1.99, 'Available'),
       (10000025, 'Cheese', 'Sharp cheddar cheese, 8 oz', 3.49, 'Available');
```

3. Results

SELECT * FROM Products;



```
SELECT * FROM `Products`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	ProductID	ProductName	ProductInfo	ProductPrice	ProductAvailability
Edit Copy Delete	10000001	Bananas	Fresh bananas from South America	0.99	Available
Edit Copy Delete	10000002	Apples	Crisp red apples from Washington	1.49	Available
Edit Copy Delete	10000003	Oranges	Juicy oranges from Florida	1.19	Available
Edit Copy Delete	10000004	Carrots	Organic carrots from California	1.29	Available
Edit Copy Delete	10000005	Broccoli	Fresh broccoli from local farm	1.99	Not Available
Edit Copy Delete	10000006	Tomatoes	Vine-ripened tomatoes from local farm	2.49	Available
Edit Copy Delete	10000007	Lettuce	Crisp lettuce from local farm	1.99	Available
Edit Copy Delete	10000008	Cucumbers	Fresh cucumbers from local farm	1.29	Available
Edit Copy Delete	10000009	Potatoes	Organic potatoes from Idaho	1.49	Available
Edit Copy Delete	10000010	Onions	Sweet onions from local farm	0.99	Available
Edit Copy Delete	10000011	Chicken Breast	Boneless, skinless chicken breast	3.99	Not Available
Edit Copy Delete	10000012	Ground Beef	Fresh ground beef, 90% lean	5.99	Available
Edit Copy Delete	10000013	Salmon Fillet	Fresh Atlantic salmon fillet	10.99	Available
Edit Copy Delete	10000014	Tilapia Fillet	Fresh tilapia fillet	7.99	Available
Edit Copy Delete	10000015	Pork Chops	Bone-in pork chops	4.99	Available
Console Edit Copy Delete	10000016	Bacon	Hickory-smoked bacon	3.99	Not Available

4. **Description of the Products table:**

   The "Products" table stores information about various products, including their unique product ID, name, information, price, and availability. This table can be used by an e-commerce website, grocery store, or any other business that deals with the sale and management of various products. The table includes columns such as "ProductID", "ProductName", "ProductInfo", "ProductPrice", and "ProductAvailability" which contain the specific details of each product.

LocalStore Table

1. **Create table statement**

   CREATE TABLE `LocalStore` (

   `LocalStoreID` bigint NOT NULL,

   `StoreName` varchar(100) NOT NULL,

   `StoreAddress` varchar(200) NOT NULL,

   `StorePhoneNo` varchar(20) NOT NULL,

`ProductsAvailability` varchar(20) NOT NULL

);

```
CREATE TABLE LocalStore (
  LocalStoreID INT NOT NULL AUTO_INCREMENT,
  StoreName VARCHAR(100) NOT NULL,
  StoreAddress VARCHAR(200) NOT NULL,
  StorePhoneNo VARCHAR(20) NOT NULL,
  ProductsAvailability VARCHAR(20) NOT NULL,
  PRIMARY KEY (LocalStoreID)
);
```

2. Insert statement

INSERT INTO `LocalStore` (`LocalStoreID`, `StoreName`, `StoreAddress`, `StorePhoneNo`, `ProductsAvailability`) VALUES

(10000000001, 'Sunrise Groceries', '123 Main St, Anytown, USA', '1234567890', 'available'),

(10000000002, 'Green Garden Market', '456 Elm St, Anytown, USA', '2345678901', 'available'),

(10000000003, 'Farmers Market', '789 Oak St, Anytown, USA', '3456789012', 'available'),

(10000000004, 'Food Co-op', '111 Maple Ave, Anytown, USA', '4567890123', 'available'),

(10000000005, 'Organic Harvest', '222 Oak St, Anytown, USA', '5678901234', 'available'),

(10000000006, 'Supermart', '333 Pine St, Anytown, USA', '6789012345', 'available'),

(10000000007, 'Quality Foods', '444 Maple Ave, Anytown, USA', '7890123456', 'available'),

(10000000008, 'Greenway Market', '555 Elm St, Anytown, USA', '8901234567', 'available'),

(10000000009, 'Main Street Grocery', '666 Main St, Anytown, USA', '9012345678', 'available'),

(10000000010, 'Healthy Habits', '777 Oak St, Anytown, USA', '2109876543', 'available'),

(10000000011, 'Friendly Market', '888 Pine St, Anytown, USA', '3210987654', 'available'),

(10000000012, 'Natural Foods', '999 Elm St, Anytown, USA', '4321098765', 'available'),

(10000000013, 'Green Acres Market', '101 Main St, Anytown, USA', '5432109876', 'available'),

(10000000014, 'Good Earth Foods', '202 Oak St, Anytown, USA', '6543210987', 'available'),

(10000000015, 'Local Harvest', '303 Pine St, Anytown, USA', '7654321098', 'available'),

(10000000016, 'Sunflower Market', '404 Maple Ave, Anytown, USA', '8765432109', 'available'),

(10000000017, 'Whole Foods', '505 Elm St, Anytown, USA', '9876543210', 'available'),

(10000000018, 'Fresh Choice', '606 Oak St, Anytown, USA', '1098765432', 'available'),

(10000000019, 'Natural Grocers', '707 Pine St, Anytown, USA', '2198765431', 'available'),

(10000000020, 'Farmer\'s Market', '808 Maple Ave, Anytown, USA', '3298765430', 'available'),

(10000000021, 'Produce Junction', '909 Elm St, Anytown, USA', '4398765429', 'available'),

(10000000022, 'Healthy Habits Market', '1010 Oak St, Anytown, USA', '5498765428', 'available'),

(10000000023, 'Green Fresh Market', '1111 Pine St, Anytown, USA', '6598765427', 'available'),

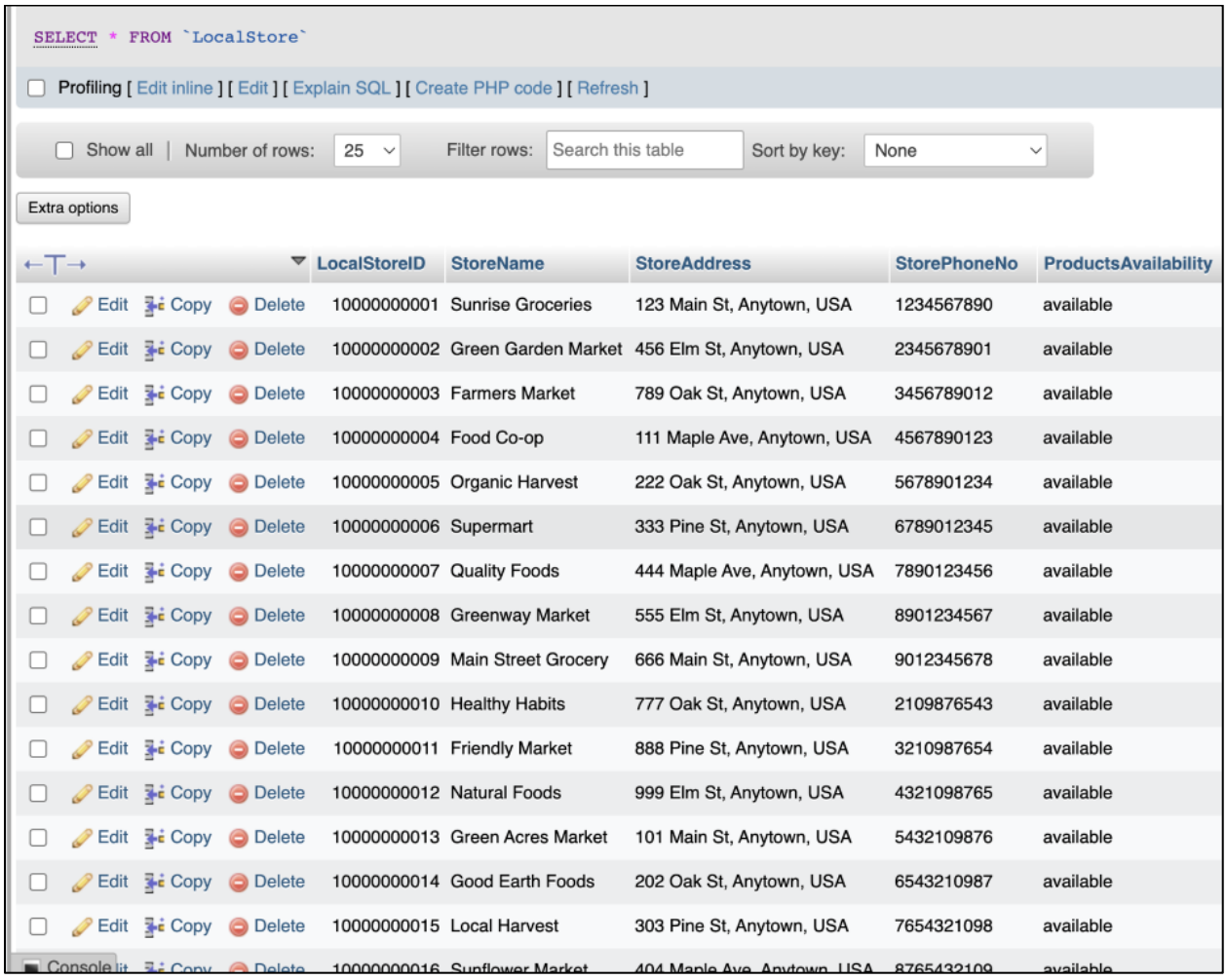
(10000000024, 'Better Health Market', '1212 Maple Ave, Anytown, USA', '7698765426', 'available'),

(10000000025, 'Manohar store', '3925 N Elm st', '8798765425', 'not available');

```
INSERT INTO LocalStore (LocalStoreID, StoreName, StoreAddress, StorePhoneNo, ProductsAvailablility) VALUES
(10000000001, 'Sunrise Groceries', '123 Main St, Anytown, USA', '1234567890', 'available'),
(10000000002, 'Green Garden Market', '456 Elm St, Anytown, USA', '2345678901', 'available'),
(10000000003, 'Farmers Market', '789 Oak St, Anytown, USA', '3456789012', 'available'),
(10000000004, 'Food Co-op', '111 Maple Ave, Anytown, USA', '4567890123', 'available'),
(10000000005, 'Organic Harvest', '222 Oak St, Anytown, USA', '5678901234', 'available'),
(10000000006, 'Supermart', '333 Pine St, Anytown, USA', '6789012345', 'available'),
(10000000007, 'Quality Foods', '444 Maple Ave, Anytown, USA', '7890123456', 'available'),
(10000000008, 'Greenway Market', '555 Elm St, Anytown, USA', '8901234567', 'available'),
(10000000009, 'Main Street Grocery', '666 Main St, Anytown, USA', '9012345678', 'available'),
(10000000010, 'Healthy Habits', '777 Oak St, Anytown, USA', '2109876543', 'available'),
(10000000011, 'Friendly Market', '888 Pine St, Anytown, USA', '3210987654', 'available'),
(10000000012, 'Natural Foods', '999 Elm St, Anytown, USA', '4321098765', 'available'),
(10000000013, 'Green Acres Market', '101 Main St, Anytown, USA', '5432109876', 'available'),
(10000000014, 'Good Earth Foods', '202 Oak St, Anytown, USA', '6543210987', 'available'),
(10000000015, 'Local Harvest', '303 Pine St, Anytown, USA', '7654321098', 'available'),
(10000000016, 'Sunflower Market', '404 Maple Ave, Anytown, USA', '8765432109', 'available'),
(10000000017, 'Whole Foods', '505 Elm St, Anytown, USA', '9876543210', 'available'),
(10000000018, 'Fresh Choice', '606 Oak St, Anytown, USA', '1098765432', 'available'),
(10000000019, 'Natural Grocers', '707 Pine St, Anytown, USA', '2198765431', 'available'),
(10000000020, 'Farmer\'s Market', '808 Maple Ave, Anytown, USA', '3298765430', 'available'),
(10000000021, 'Produce Junction', '909 Elm St, Anytown, USA', '4398765429', 'available'),
(10000000022, 'Healthy Habits Market', '1010 Oak St, Anytown, USA', '5498765428', 'available'),
(10000000023, 'Green Fresh Market', '1111 Pine St, Anytown, USA', '6598765427', 'available'),
(10000000024, 'Better Health Market', '1212 Maple Ave, Anytown, USA', '7698765426', 'available'),
(10000000025, 'Manohar store','3925 N Elm st','8798765425','not available');
```

3. Results

SELECT * FROM LocalStore;



	LocalStoreID	StoreName	StoreAddress	StorePhoneNo	ProductsAvailability
Edit Copy Delete	10000000001	Sunrise Groceries	123 Main St, Anytown, USA	1234567890	available
Edit Copy Delete	10000000002	Green Garden Market	456 Elm St, Anytown, USA	2345678901	available
Edit Copy Delete	10000000003	Farmers Market	789 Oak St, Anytown, USA	3456789012	available
Edit Copy Delete	10000000004	Food Co-op	111 Maple Ave, Anytown, USA	4567890123	available
Edit Copy Delete	10000000005	Organic Harvest	222 Oak St, Anytown, USA	5678901234	available
Edit Copy Delete	10000000006	Supermart	333 Pine St, Anytown, USA	6789012345	available
Edit Copy Delete	10000000007	Quality Foods	444 Maple Ave, Anytown, USA	7890123456	available
Edit Copy Delete	10000000008	Greenway Market	555 Elm St, Anytown, USA	8901234567	available
Edit Copy Delete	10000000009	Main Street Grocery	666 Main St, Anytown, USA	9012345678	available
Edit Copy Delete	10000000010	Healthy Habits	777 Oak St, Anytown, USA	2109876543	available
Edit Copy Delete	10000000011	Friendly Market	888 Pine St, Anytown, USA	3210987654	available
Edit Copy Delete	10000000012	Natural Foods	999 Elm St, Anytown, USA	4321098765	available
Edit Copy Delete	10000000013	Green Acres Market	101 Main St, Anytown, USA	5432109876	available
Edit Copy Delete	10000000014	Good Earth Foods	202 Oak St, Anytown, USA	6543210987	available
Edit Copy Delete	10000000015	Local Harvest	303 Pine St, Anytown, USA	7654321098	available
Edit Copy Delete	10000000016	Sunflower Market	404 Maple Ave, Anytown, USA	8765432109	available

4. **Description of the LocalStore table:**

   The LocalStore table stores information about local stores that sell products. It contains columns for the store's ID, name, address, phone number, and the availability of products. The table can be used to keep track of the different stores that carry certain products and their respective locations.

Orders Table

1. **Create table statement**

   CREATE TABLE `Orders` (

   `OrderID` int NOT NULL,

   `CustomerID` int NOT NULL,

`ShopperID` int NOT NULL,

`ProductID` int NOT NULL,

`Date` date NOT NULL,

`TotalAmount` float NOT NULL

) ;

```
CREATE TABLE Orders (
  OrderID INT NOT NULL AUTO_INCREMENT,
  CustomerID INT NOT NULL,
  ShopperID INT NOT NULL,
  ProductID INT NOT NULL,
  Date DATE NOT NULL,
  TotalAmount FLOAT NOT NULL,
  PRIMARY KEY (OrderID),
  FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
  FOREIGN KEY (ShopperID) REFERENCES PersonalShopper(ShopperID),
  FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
```

2. Insert Statement

INSERT INTO `Orders` (`OrderID`, `CustomerID`, `ShopperID`, `ProductID`, `Date`, `TotalAmount`) VALUES

(1000001, 10001, 100001, 10000001, '2022-01-02', 200.75),

(1000002, 10002, 100002, 10000002, '2022-01-03', 50.25),

(1000003, 10003, 100003, 10000003, '2022-01-04', 75),

(1000004, 10004, 100004, 10000004, '2022-01-05', 150),

(1000005, 10005, 100005, 10000005, '2022-01-06', 25.5),

(1000006, 10006, 100006, 10000006, '2022-01-07', 75.25),

(1000007, 10007, 100007, 10000007, '2022-01-08', 100),

(1000008, 10008, 100008, 10000008, '2022-01-09', 300.75),

(1000009, 10009, 100009, 10000009, '2022-01-10', 500),

(1000010, 10010, 100010, 10000010, '2022-01-11', 120.25),

(1000011, 10011, 100011, 10000011, '2022-01-12', 80.5),

(1000012, 10012, 100012, 10000012, '2022-01-13', 150),

(1000013, 10013, 100013, 10000013, '2022-01-14', 90.25),

(1000014, 10014, 100014, 10000014, '2022-01-15', 200.5),

(1000015, 10015, 100015, 10000015, '2022-01-16', 75),

(1000016, 10016, 100016, 10000016, '2022-01-17', 50.25),

(1000017, 10017, 100017, 10000017, '2022-01-18', 100.75),

(1000018, 10018, 100018, 10000018, '2022-01-19', 150),

(1000019, 10019, 100019, 10000019, '2022-01-20', 25.5),

(1000020, 10020, 100020, 10000020, '2022-01-21', 75.25),

(1000021, 10021, 100021, 10000021, '2022-01-22', 100),

(1000022, 10022, 100022, 10000022, '2022-01-23', 300.75),

(1000023, 10023, 100023, 10000023, '2022-01-24', 40.75),

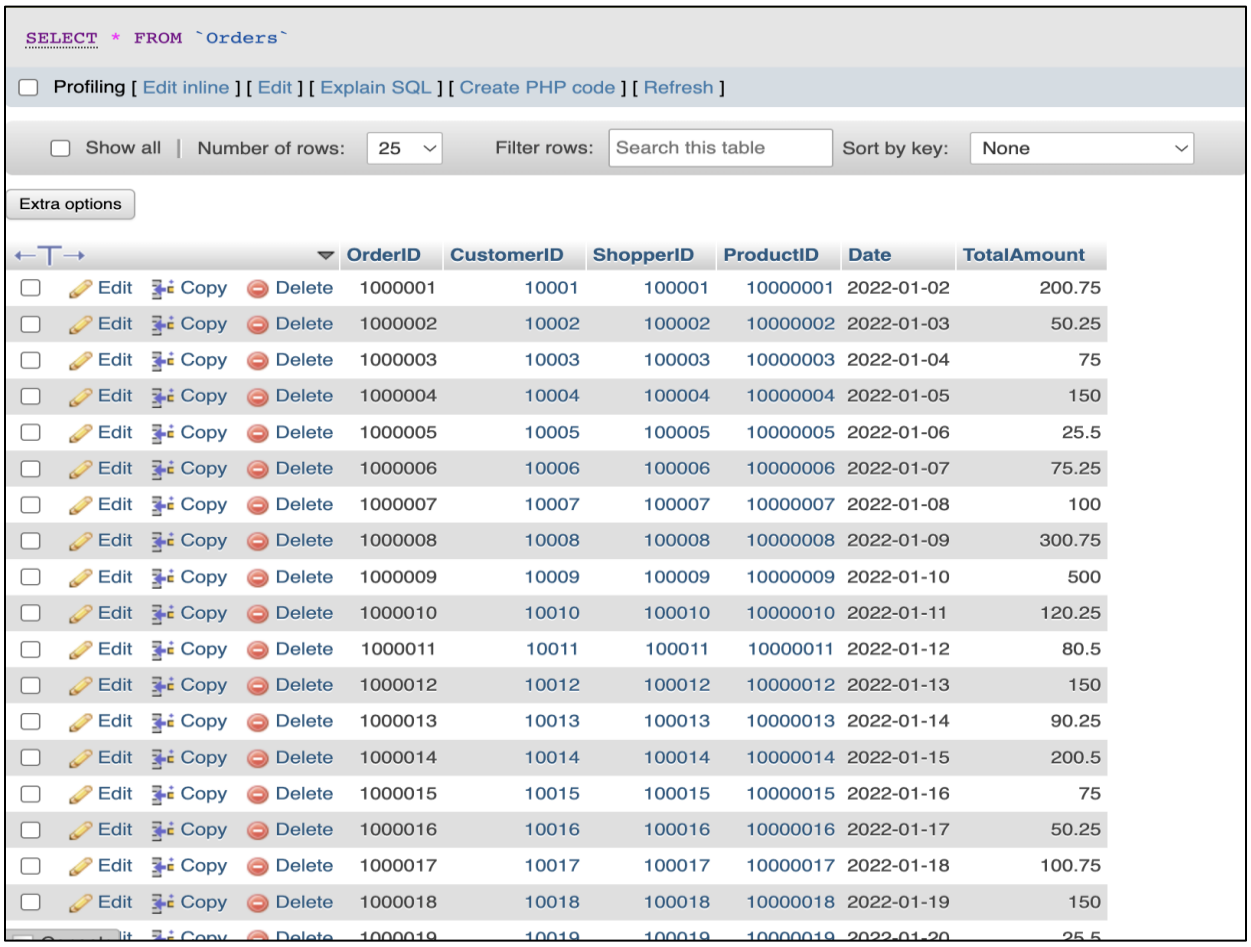
(1000024, 10024, 100024, 10000024, '2022-01-25', 100.5),

(1000025, 10025, 100025, 10000025, '2022-01-25', 30.5);

```sql
INSERT INTO Orders (OrderID, CustomerID, ShopperID, ProductID, Date, TotalAmount) VALUES
(1000001, 10001, 100001, 10000001, '2022-01-02', 200.75),
(1000002, 10002, 100002, 10000002, '2022-01-03', 50.25),
(1000003, 10003, 100003, 10000003, '2022-01-04', 75),
(1000004, 10004, 100004, 10000004, '2022-01-05', 150),
(1000005, 10005, 100005, 10000005, '2022-01-06', 25.5),
(1000006, 10006, 100006, 10000006, '2022-01-07', 75.25),
(1000007, 10007, 100007, 10000007, '2022-01-08', 100),
(1000008, 10008, 100008, 10000008, '2022-01-09', 300.75),
(1000009, 10009, 100009, 10000009, '2022-01-10', 500),
(1000010, 10010, 100010, 10000010, '2022-01-11', 120.25),
(1000011, 10011, 100011, 10000011, '2022-01-12', 80.5),
(1000012, 10012, 100012, 10000012, '2022-01-13', 150),
(1000013, 10013, 100013, 10000013, '2022-01-14', 90.25),
(1000014, 10014, 100014, 10000014, '2022-01-15', 200.5),
(1000015, 10015, 100015, 10000015, '2022-01-16', 75),
(1000016, 10016, 100016, 10000016, '2022-01-17', 50.25),
(1000017, 10017, 100017, 10000017, '2022-01-18', 100.75),
(1000018, 10018, 100018, 10000018, '2022-01-19', 150),
(1000019, 10019, 100019, 10000019, '2022-01-20', 25.5),
(1000020, 10020, 100020, 10000020, '2022-01-21', 75.25),
(1000021, 10021, 100021, 10000021, '2022-01-22', 100),
(1000022, 10022, 100022, 10000022, '2022-01-23', 300.75),
(1000023, 10023, 100023, 10000023, '2022-01-24', 40.75),
(1000024, 10024, 100024, 10000024, '2022-01-25', 100.5),
(1000025, 10025, 100025, 10000025, '2022-01-25', 30.5);
```

3. **Results**

SELECT * FROM Orders;



```
SELECT * FROM `Orders`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	OrderID	CustomerID	ShopperID	ProductID	Date	TotalAmount
Edit Copy Delete	1000001	10001	100001	10000001	2022-01-02	200.75
Edit Copy Delete	1000002	10002	100002	10000002	2022-01-03	50.25
Edit Copy Delete	1000003	10003	100003	10000003	2022-01-04	75
Edit Copy Delete	1000004	10004	100004	10000004	2022-01-05	150
Edit Copy Delete	1000005	10005	100005	10000005	2022-01-06	25.5
Edit Copy Delete	1000006	10006	100006	10000006	2022-01-07	75.25
Edit Copy Delete	1000007	10007	100007	10000007	2022-01-08	100
Edit Copy Delete	1000008	10008	100008	10000008	2022-01-09	300.75
Edit Copy Delete	1000009	10009	100009	10000009	2022-01-10	500
Edit Copy Delete	1000010	10010	100010	10000010	2022-01-11	120.25
Edit Copy Delete	1000011	10011	100011	10000011	2022-01-12	80.5
Edit Copy Delete	1000012	10012	100012	10000012	2022-01-13	150
Edit Copy Delete	1000013	10013	100013	10000013	2022-01-14	90.25
Edit Copy Delete	1000014	10014	100014	10000014	2022-01-15	200.5
Edit Copy Delete	1000015	10015	100015	10000015	2022-01-16	75
Edit Copy Delete	1000016	10016	100016	10000016	2022-01-17	50.25
Edit Copy Delete	1000017	10017	100017	10000017	2022-01-18	100.75
Edit Copy Delete	1000018	10018	100018	10000018	2022-01-19	150
Edit Copy Delete	1000019	10019	100019	10000019	2022-01-20	25.5

4. **Description of the table:**

The "Orders" table stores information about orders placed by customers, including details such as the order ID, customer ID, shopper ID, product ID, date of the order, and total amount. The table is likely used by an e-commerce or retail business to keep track of sales and customer purchases.

OrderDetails

1. **Create table statement**

CREATE TABLE `OrderDetails` (

`OrderDetailsID` bigint NOT NULL,

`OrderID` int NOT NULL,

`ProductID` int NOT NULL

) ;

```
CREATE TABLE OrderDetails (
  OrderDetailsID BIGINT NOT NULL AUTO_INCREMENT,
  OrderID INT NOT NULL,
  ProductID INT NOT NULL,
  PRIMARY KEY (OrderDetailsID),
  FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
  FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
```

2. **Insert statement**

INSERT INTO `OrderDetails` (`OrderDetailsID`, `OrderID`, `ProductID`) VALUES

(100000001, 1000001, 10000001),

(100000002, 1000001, 10000002),

(100000003, 1000002, 10000003),

(100000004, 1000002, 10000004),

(100000005, 1000003, 10000005),

(100000006, 1000004, 10000006),

(100000007, 1000005, 10000007),

(100000008, 1000005, 10000008),

(100000009, 1000006, 10000009),

(100000010, 1000007, 10000010),

(100000011, 1000007, 10000011),

(100000012, 1000007, 10000012),

(100000013, 1000008, 10000013),

(100000014, 1000009, 10000014),

(100000015, 1000009, 10000015),

(100000016, 1000010, 10000016),

(100000018, 1000011, 10000018),

(100000019, 1000012, 10000019),

(100000020, 1000013, 10000020),

(100000021, 1000014, 10000021),

(100000022, 1000015, 10000022),

(100000023, 1000016, 10000023),

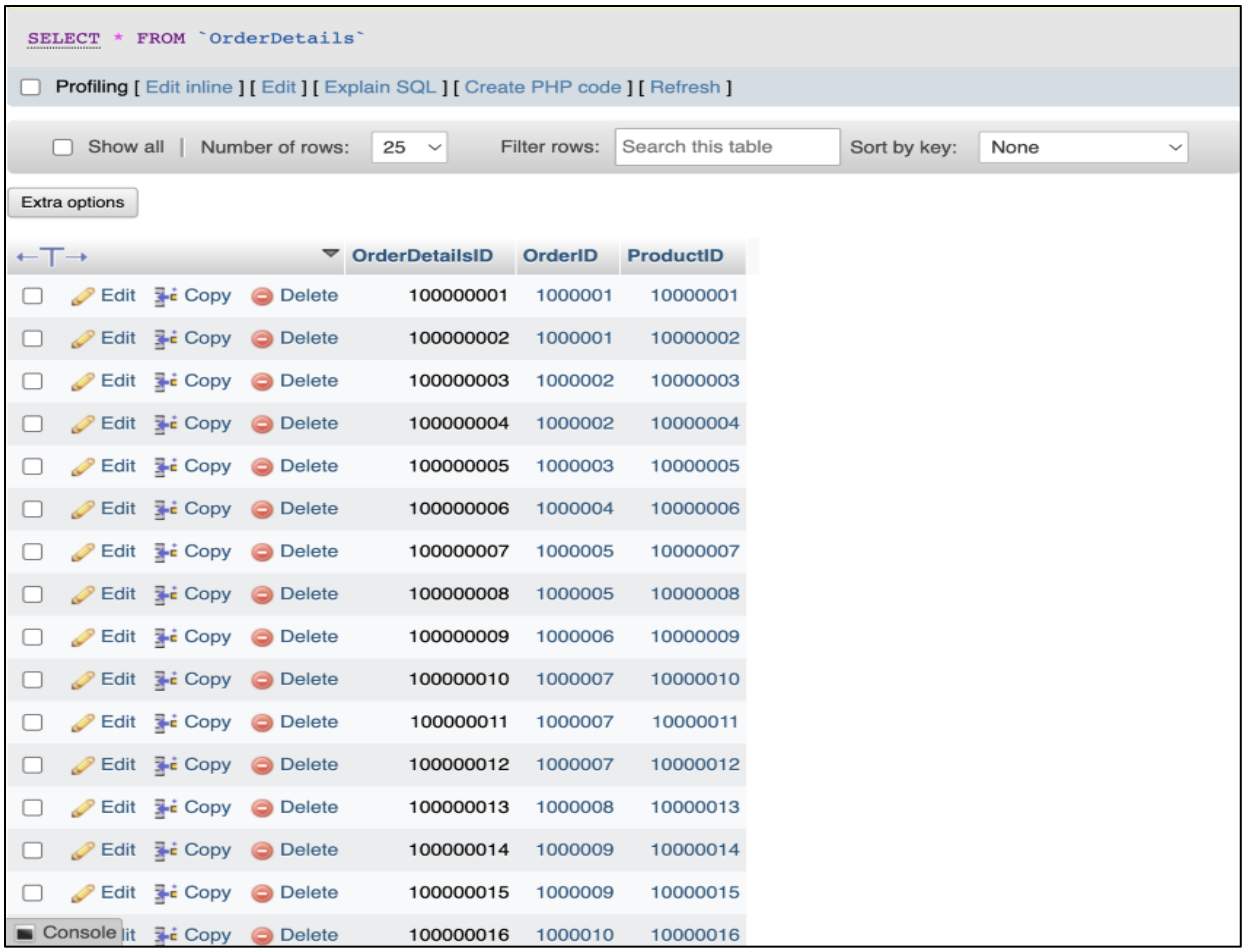
(100000024, 1000017, 10000022),

(100000025, 1000018, 10000023);

```
INSERT INTO OrderDetails (OrderDetailsID, OrderID, ProductID)
VALUES  (100000001, 1000001, 10000001),
        (100000002, 1000001, 10000002),
        (100000003, 1000002, 10000003),
        (100000004, 1000002, 10000004),
        (100000005, 1000003, 10000005),
        (100000006, 1000004, 10000006),
        (100000007, 1000005, 10000007),
        (100000008, 1000005, 10000008),
        (100000009, 1000006, 10000009),
        (100000010, 1000007, 10000010),
        (100000011, 1000007, 10000011),
        (100000012, 1000007, 10000012),
        (100000013, 1000008, 10000013),
        (100000014, 1000009, 10000014),
        (100000015, 1000009, 10000015),
        (100000016, 1000010, 10000016),
        (100000018, 1000011, 10000018),
        (100000019, 1000012, 10000019),
        (100000020, 1000013, 10000020),
        (100000021, 1000014, 10000021),
        (100000022, 1000015, 10000022),
        (100000023, 1000016, 10000023),
        (100000024, 1000017, 10000022),
        (100000025, 1000018, 10000023);
```

3. **Results**

SELECT * FROM OrderDetails;



4. **Description of the table:**

   The **OrderDetails** table stores details about the products that were ordered in each order. Each row in the table represents a specific order and the product(s) that were included in that order.

ProductStore

1. **Create table statement**

```
CREATE TABLE `ProductStore` (
  `ProductStoreID` bigint NOT NULL,
  `ProductID` int NOT NULL,
  `LocalStoreID` bigint NOT NULL
) ;
```

```
CREATE TABLE ProductStore (
  ProductStoreID BIGINT NOT NULL AUTO_INCREMENT,
  ProductID INT NOT NULL,
  LocalStoreID INT NOT NULL,
  PRIMARY KEY (ProductStoreID),
  FOREIGN KEY (ProductID) REFERENCES Products(ProductID),
  FOREIGN KEY (LocalStoreID) REFERENCES LocalStore(LocalStoreID)
);
```

2. **Insert statement**

INSERT INTO `ProductStore` (`ProductStoreID`, `ProductID`, `LocalStoreID`) VALUES

(1000000001, 10000002, 10000000001),

(1000000002, 10000006, 10000000002),

(1000000003, 10000003, 10000000001),

(1000000004, 10000008, 10000000005),

(1000000005, 10000004, 10000000004),

(1000000006, 10000001, 10000000003),

(1000000007, 10000007, 10000000002),

(1000000008, 10000005, 10000000005),

(1000000009, 10000002, 10000000004),

(1000000010, 10000003, 10000000003),

(1000000011, 10000008, 10000000001),

(1000000012, 10000006, 10000000004),

(1000000013, 10000005, 10000000002),

(1000000014, 10000001, 10000000005),

(1000000015, 10000004, 10000000003),

(1000000016, 10000007, 10000000001),

(1000000017, 10000008, 10000000003),

(1000000018, 10000002, 10000000002),

(1000000019, 10000005, 10000000005),

(1000000020, 10000003, 10000000004),

(1000000021, 10000001, 10000000003),

(1000000022, 10000004, 10000000001),

(1000000023, 10000006, 10000000005),

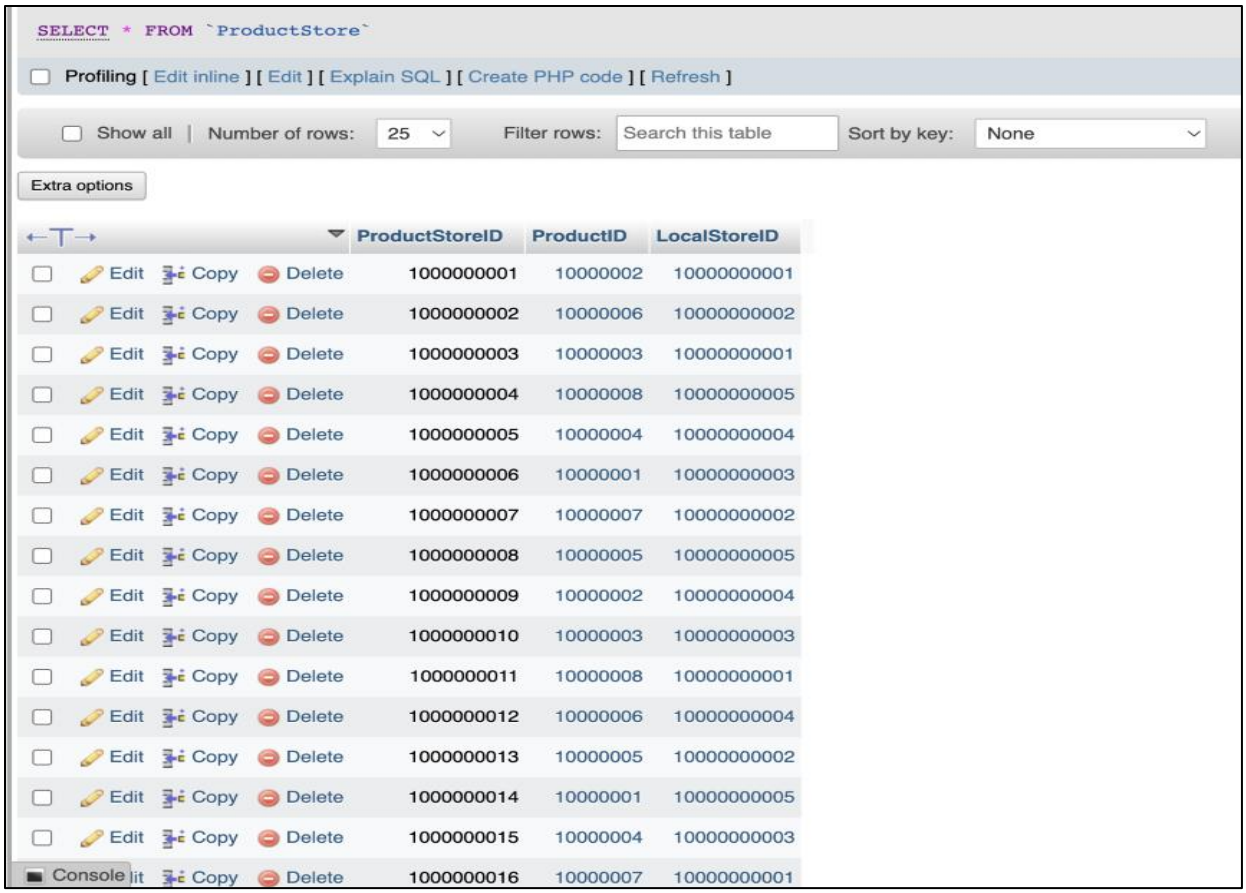
(1000000024, 10000007, 10000000004),

(1000000025, 10000008, 10000000002);

```
INSERT INTO ProductStore (ProductStoreID, ProductID, LocalStoreID)
VALUES
(1000000001, 10000002, 10000000001),
(1000000002, 10000006, 10000000002),
(1000000003, 10000003, 10000000001),
(1000000004, 10000008, 10000000005),
(1000000005, 10000004, 10000000004),
(1000000006, 10000001, 10000000003),
(1000000007, 10000007, 10000000002),
(1000000008, 10000005, 10000000005),
(1000000009, 10000002, 10000000004),
(1000000010, 10000003, 10000000003),
(1000000011, 10000008, 10000000001),
(1000000012, 10000006, 10000000004),
(1000000013, 10000005, 10000000002),
(1000000014, 10000001, 10000000005),
(1000000015, 10000004, 10000000003),
(1000000016, 10000007, 10000000001),
(1000000017, 10000008, 10000000003),
(1000000018, 10000002, 10000000002),
(1000000019, 10000005, 10000000005),
(1000000020, 10000003, 10000000004),
(1000000021, 10000001, 10000000003),
(1000000022, 10000004, 10000000001),
(1000000023, 10000006, 10000000005),
(1000000024, 10000007, 10000000004),
(1000000025, 10000008, 10000000002);
```

3. **Results**

SELECT * FROM ProductStore;



4. **Description of the table:**

   The **ProductStore** table is used to track the relationship between products and stores. Each record in this table represents the availability of a product in a specific store. The **ProductStoreID** column is a unique identifier for each record, the **ProductID** column refers to the unique identifier of the product that is available in the store, and the **LocalStoreID** column refers to the unique identifier of the store where the product is available.

Payment

1. **Create table statement**

   CREATE TABLE `Payment` (

   `PaymentID` bigint NOT NULL,

   `OrderID` int NOT NULL,

`PaymentMethod` varchar(50) NOT NULL,

`Amount` float NOT NULL

) ;

```
CREATE TABLE Payment (
  PaymentID BIGINT NOT NULL AUTO_INCREMENT,
  OrderID INT NOT NULL,
  PaymentMethod VARCHAR(50) NOT NULL,
  Amount FLOAT NOT NULL,
  PRIMARY KEY (PaymentID),
  FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
);
```

2. **Insert statement**

INSERT INTO `Payment` (`PaymentID`, `OrderID`, `PaymentMethod`, `Amount`) VALUES

(1, 1000001, 'credit card', 50),

(2, 1000002, 'cash', 25),

(3, 1000003, 'debit card', 100),

(4, 1000004, 'credit card', 75),

(5, 1000005, 'cash', 30),

(6, 1000006, 'debit card', 80),

(7, 1000007, 'credit card', 40),

(8, 1000008, 'cash', 20),

(9, 1000009, 'debit card', 60),

(10, 1000010, 'credit card', 90),

(11, 1000011, 'cash', 35),

(12, 1000012, 'debit card', 70),

(13, 1000013, 'credit card', 45),

(14, 1000014, 'cash', 55),

(15, 1000015, 'debit card', 65),

(16, 1000016, 'credit card', 80),

(17, 1000017, 'cash', 75),

(18, 1000018, 'debit card', 50),

(19, 1000019, 'credit card', 30),

(20, 1000020, 'cash', 25),

(21, 1000021, 'debit card', 100),

(22, 1000022, 'credit card', 70),

(23, 1000023, 'cash', 45),

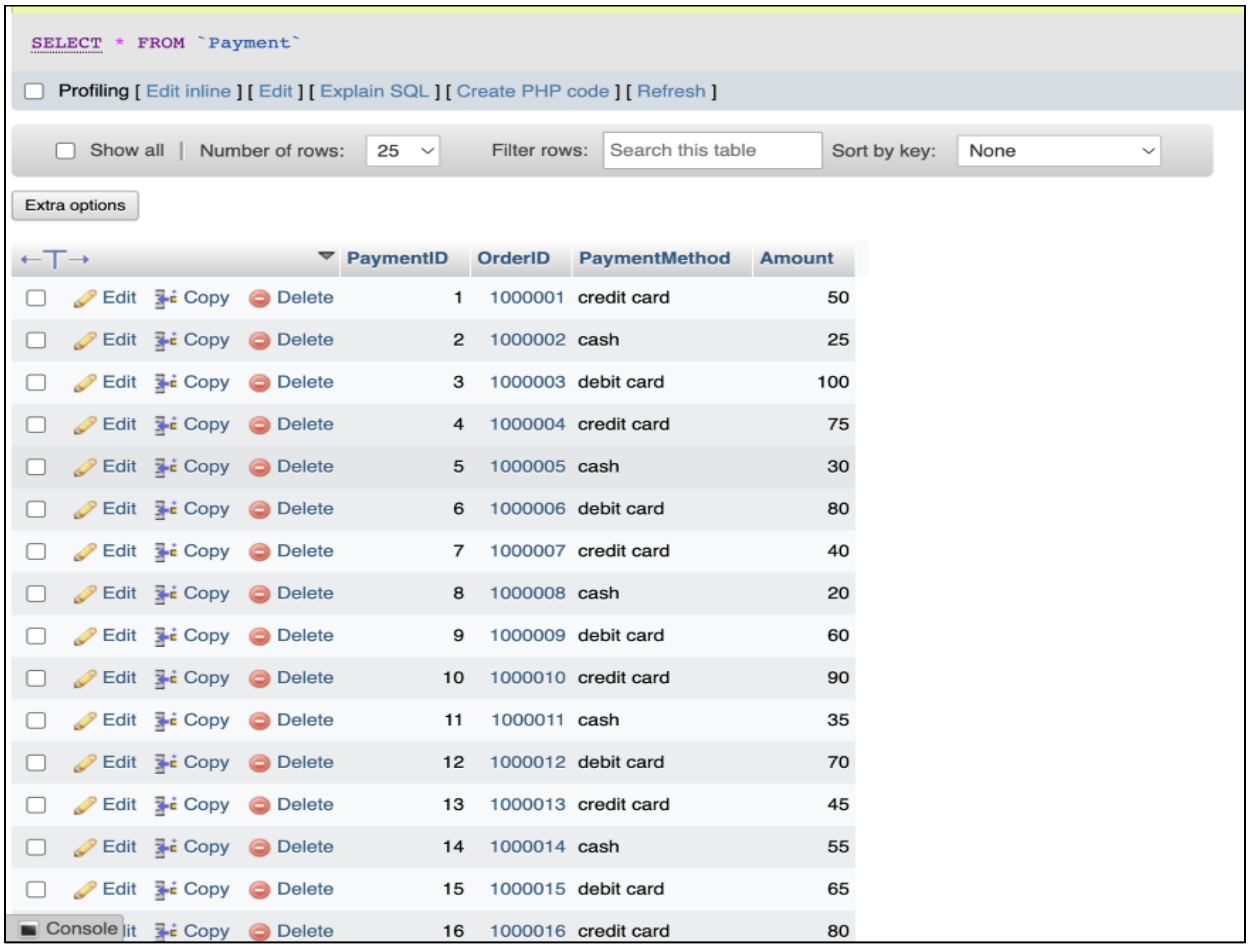
(24, 1000024, 'debit card', 60),

(25, 1000025, 'credit card', 85);

```sql
INSERT INTO Payment (PaymentID, OrderID, PaymentMethod, Amount)
VALUES
(1, 1000001, 'credit card', 50.00),
(2, 1000002, 'cash', 25.00),
(3, 1000003, 'debit card', 100.00),
(4, 1000004, 'credit card', 75.00),
(5, 1000005, 'cash', 30.00),
(6, 1000006, 'debit card', 80.00),
(7, 1000007, 'credit card', 40.00),
(8, 1000008, 'cash', 20.00),
(9, 1000009, 'debit card', 60.00),
(10, 1000010, 'credit card', 90.00),
(11, 1000011, 'cash', 35.00),
(12, 1000012, 'debit card', 70.00),
(13, 1000013, 'credit card', 45.00),
(14, 1000014, 'cash', 55.00),
(15, 1000015, 'debit card', 65.00),
(16, 1000016, 'credit card', 80.00),
(17, 1000017, 'cash', 75.00),
(18, 1000018, 'debit card', 50.00),
(19, 1000019, 'credit card', 30.00),
(20, 1000020, 'cash', 25.00),
(21, 1000021, 'debit card', 100.00),
(22, 1000022, 'credit card', 70.00),
(23, 1000023, 'cash', 45.00),
(24, 1000024, 'debit card', 60.00),
(25, 1000025, 'credit card', 85.00);
```

3. Results

SELECT * FROM Payment;



```
SELECT * FROM `Payment`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	PaymentID	OrderID	PaymentMethod	Amount
Edit Copy Delete	1	1000001	credit card	50
Edit Copy Delete	2	1000002	cash	25
Edit Copy Delete	3	1000003	debit card	100
Edit Copy Delete	4	1000004	credit card	75
Edit Copy Delete	5	1000005	cash	30
Edit Copy Delete	6	1000006	debit card	80
Edit Copy Delete	7	1000007	credit card	40
Edit Copy Delete	8	1000008	cash	20
Edit Copy Delete	9	1000009	debit card	60
Edit Copy Delete	10	1000010	credit card	90
Edit Copy Delete	11	1000011	cash	35
Edit Copy Delete	12	1000012	debit card	70
Edit Copy Delete	13	1000013	credit card	45
Edit Copy Delete	14	1000014	cash	55
Edit Copy Delete	15	1000015	debit card	65
Console Edit Copy Delete	16	1000016	credit card	80

4. **Description of the table:**

   The **Payment** table represents the payments made by customers for their orders. Each row in the table represents a single payment made by a customer for a specific order.

Delivery

1. Create table statement

   CREATE TABLE ‘Delivery’ (

   ‘DeliveryID’ BIGINT PRIMARY KEY,

   ‘OrderID’ INT,

   ‘ TypeOfDelivery’ VARCHAR(50),

   ‘DeliveryAddress’ VARCHAR(100),

FOREIGN KEY (‘OrderID’) REFERENCES ‘Orders’(‘OrderID’)

);

```
CREATE TABLE Delivery (
    DeliveryID BIGINT NOT NULL AUTO_INCREMENT,
    OrderID INT,
    TypeOfDelivery VARCHAR(50),
    DeliveryAddress VARCHAR(100),
    PRIMARY KEY(DeliveryID),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
);
```

2. Insert statement

INSERT INTO `Delivery` (`DeliveryID`, `OrderID`, `TypeOfDelivery`, `DeliveryAddress`) VALUES

(1, 1000001, 'pickup', '123 Main St.'),

(2, 1000002, 'door delivery', '456 Oak Ave.'),

(3, 1000003, 'pickup', '789 Pine Rd.'),

(4, 1000004, 'door delivery', '1011 Elm St.'),

(5, 1000005, 'pickup', '1213 Maple Ave.'),

(6, 1000006, 'door delivery', '1415 Walnut St.'),

(7, 1000007, 'pickup', '1617 Chestnut St.'),

(8, 1000008, 'door delivery', '1819 Spruce St.'),

(9, 1000009, 'pickup', '2021 Birch Rd.'),

(10, 1000010, 'door delivery', '2223 Cedar Ave.'),

(11, 1000011, 'pickup', '2425 Elmwood Dr.'),

(12, 1000012, 'door delivery', '2627 Walnut St.'),

(13, 1000013, 'pickup', '2829 Pine Rd.'),

(14, 1000014, 'door delivery', '3031 Oak Ave.'),

(15, 1000015, 'pickup', '3233 Main St.'),

(16, 1000016, 'door delivery', '3435 Maple Ave.'),

(17, 1000017, 'pickup', '3637 Chestnut St.'),

(18, 1000018, 'door delivery', '3839 Spruce St.'),

(19, 1000019, 'pickup', '4041 Birch Rd.'),

(20, 1000020, 'door delivery', '4243 Cedar Ave.'),

(21, 1000021, 'pickup', '4445 Elmwood Dr.'),

(22, 1000022, 'door delivery', '4647 Walnut St.'),

(23, 1000023, 'pickup', '4849 Pine Rd.'),

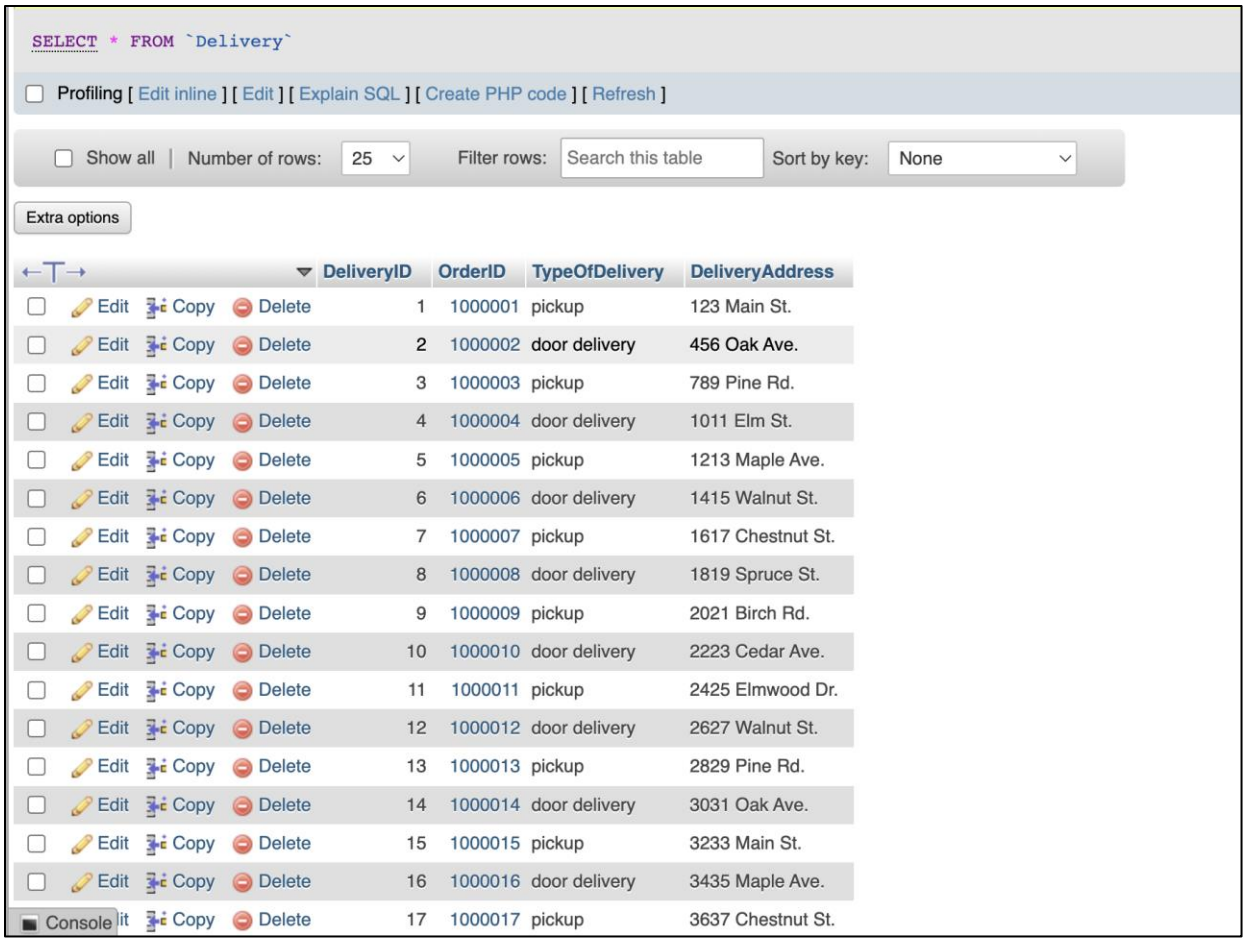
(24, 1000024, 'door delivery', '5051 Oak Ave.'),

(25, 1000025, 'pickup', '5253 Main St.');

```
INSERT INTO Delivery (DeliveryID, OrderID, TypeOfDelivery, DeliveryAddress)
VALUES (1, 1000001, 'pickup', '123 Main St.'),
       (2, 1000002, 'door delivery', '456 Oak Ave.'),
       (3, 1000003, 'pickup', '789 Pine Rd.'),
       (4, 1000004, 'door delivery', '1011 Elm St.'),
       (5, 1000005, 'pickup', '1213 Maple Ave.'),
       (6, 1000006, 'door delivery', '1415 Walnut St.'),
       (7, 1000007, 'pickup', '1617 Chestnut St.'),
       (8, 1000008, 'door delivery', '1819 Spruce St.'),
       (9, 1000009, 'pickup', '2021 Birch Rd.'),
       (10, 1000010, 'door delivery', '2223 Cedar Ave.'),
       (11, 1000011, 'pickup', '2425 Elmwood Dr.'),
       (12, 1000012, 'door delivery', '2627 Walnut St.'),
       (13, 1000013, 'pickup', '2829 Pine Rd.'),
       (14, 1000014, 'door delivery', '3031 Oak Ave.'),
       (15, 1000015, 'pickup', '3233 Main St.'),
       (16, 1000016, 'door delivery', '3435 Maple Ave.'),
       (17, 1000017, 'pickup', '3637 Chestnut St.'),
       (18, 1000018, 'door delivery', '3839 Spruce St.'),
       (19, 1000019, 'pickup', '4041 Birch Rd.'),
       (20, 1000020, 'door delivery', '4243 Cedar Ave.'),
       (21, 1000021, 'pickup', '4445 Elmwood Dr.'),
       (22, 1000022, 'door delivery', '4647 Walnut St.'),
       (23, 1000023, 'pickup', '4849 Pine Rd.'),
       (24, 1000024, 'door delivery', '5051 Oak Ave.'),
       (25, 1000025, 'pickup', '5253 Main St.');
```

3. Results

SELECT * FROM Delivery;



```
SELECT * FROM `Delivery`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	DeliveryID	OrderID	TypeOfDelivery	DeliveryAddress
Edit Copy Delete	1	1000001	pickup	123 Main St.
Edit Copy Delete	2	1000002	door delivery	456 Oak Ave.
Edit Copy Delete	3	1000003	pickup	789 Pine Rd.
Edit Copy Delete	4	1000004	door delivery	1011 Elm St.
Edit Copy Delete	5	1000005	pickup	1213 Maple Ave.
Edit Copy Delete	6	1000006	door delivery	1415 Walnut St.
Edit Copy Delete	7	1000007	pickup	1617 Chestnut St.
Edit Copy Delete	8	1000008	door delivery	1819 Spruce St.
Edit Copy Delete	9	1000009	pickup	2021 Birch Rd.
Edit Copy Delete	10	1000010	door delivery	2223 Cedar Ave.
Edit Copy Delete	11	1000011	pickup	2425 Elmwood Dr.
Edit Copy Delete	12	1000012	door delivery	2627 Walnut St.
Edit Copy Delete	13	1000013	pickup	2829 Pine Rd.
Edit Copy Delete	14	1000014	door delivery	3031 Oak Ave.
Edit Copy Delete	15	1000015	pickup	3233 Main St.
Edit Copy Delete	16	1000016	door delivery	3435 Maple Ave.
Edit Copy Delete	17	1000017	pickup	3637 Chestnut St.

Console

4. **Description of the table:**

   The Delivery table stores information about the delivery of orders, including the delivery ID, order ID, type of delivery (pickup or door delivery), and delivery address. The table has a foreign key that references the OrderID column in the Orders table to link orders with their corresponding deliveries.

Data Retrieval and Reports:

1. List all products and their availability in each local store.

   Query:

   SELECT ls.StoreName, p.ProductName, ls.ProductsAvailability

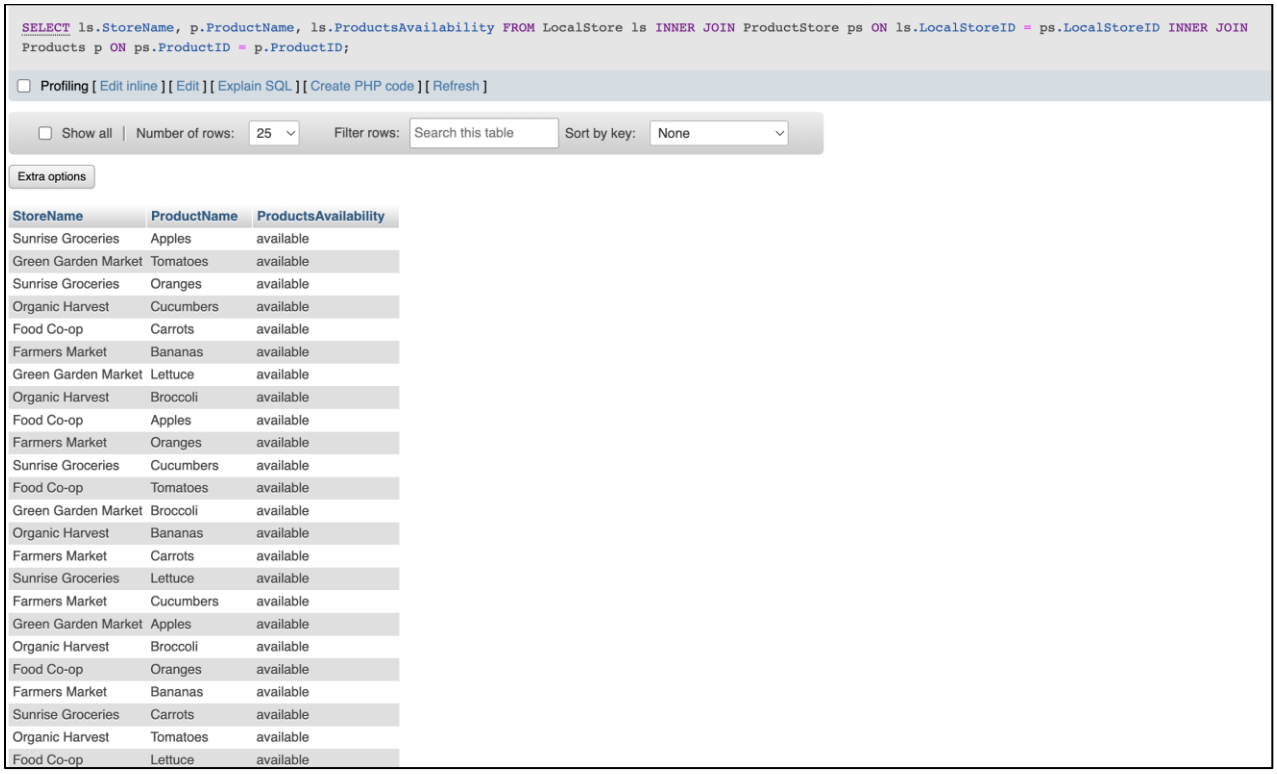
   FROM LocalStore ls

   INNER JOIN ProductStore ps ON ls.LocalStoreID = ps.LocalStoreID

   INNER JOIN Products p ON ps.ProductID = p.ProductID;

```sql
SELECT ls.StoreName, p.ProductName, ls.ProductsAvailability
FROM LocalStore ls
INNER JOIN ProductStore ps ON ls.LocalStoreID = ps.LocalStoreID
INNER JOIN Products p ON ps.ProductID = p.ProductID;
```

Result:



SELECT ls.StoreName, p.ProductName, ls.ProductsAvailability FROM LocalStore ls INNER JOIN ProductStore ps ON ls.LocalStoreID = ps.LocalStoreID INNER JOIN Products p ON ps.ProductID = p.ProductID;

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

StoreName	ProductName	ProductsAvailability
Sunrise Groceries	Apples	available
Green Garden Market	Tomatoes	available
Sunrise Groceries	Oranges	available
Organic Harvest	Cucumbers	available
Food Co-op	Carrots	available
Farmers Market	Bananas	available
Green Garden Market	Lettuce	available
Organic Harvest	Broccoli	available
Food Co-op	Apples	available
Farmers Market	Oranges	available
Sunrise Groceries	Cucumbers	available
Food Co-op	Tomatoes	available
Green Garden Market	Broccoli	available
Organic Harvest	Bananas	available
Farmers Market	Carrots	available
Sunrise Groceries	Lettuce	available
Farmers Market	Cucumbers	available
Green Garden Market	Apples	available
Organic Harvest	Broccoli	available
Food Co-op	Oranges	available
Farmers Market	Bananas	available
Sunrise Groceries	Carrots	available
Organic Harvest	Tomatoes	available
Food Co-op	Lettuce	available

Explanation:

This SQL SELECT statement retrieves the names of all local stores, along with the names of the products they carry and the availability status of each product. The statement uses inner joins to link the LocalStore, ProductStore, and Products tables.

2. List the FirstName, LastName and Order ID Of the Customers Who chose Door Delivery as the type of delivery?

Query:

SELECT c.CustomerFirstName, c.CustomerLastName, o.OrderID

FROM Customers c

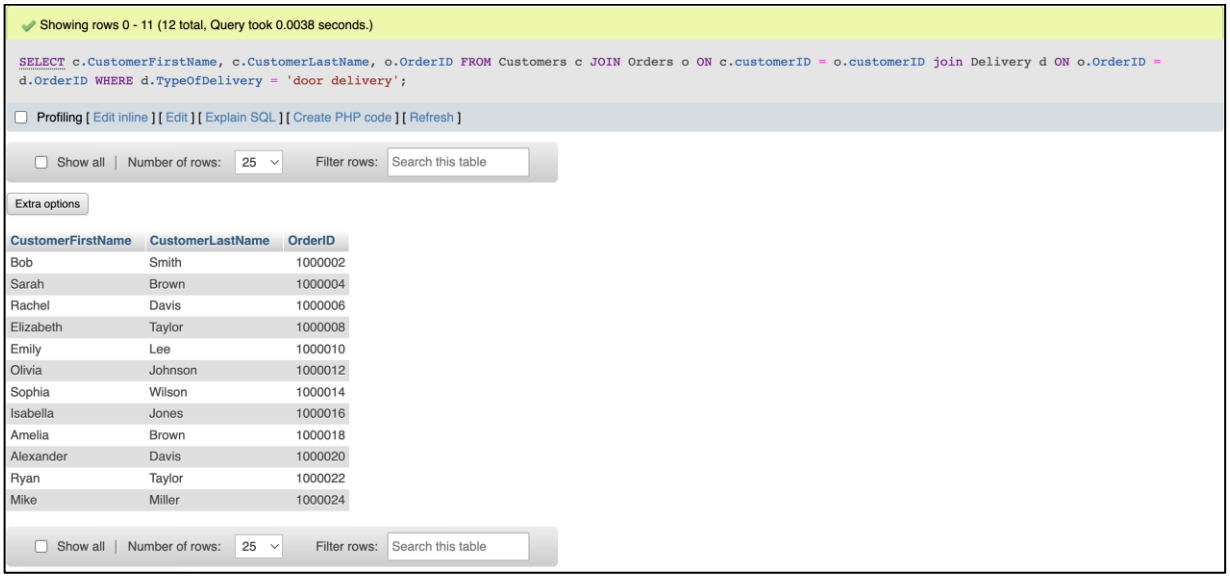
JOIN Orders o ON c.customerID = o.customerID

join Delivery d ON o.OrderID = d.OrderID

WHERE d.TypeOfDelivery = 'door delivery';

```
USE groupstyx;
SELECT c.CustomerFirstName, c.CustomerLastName, o.OrderID
FROM Customers c
JOIN Orders o ON c.customerID = o.customerID
join Delivery d ON o.OrderID = d.OrderID
WHERE d.TypeOfDelivery = 'door delivery';
```

Result:


Showing rows 0 - 11 (12 total, Query took 0.0038 seconds.)

SELECT c.CustomerFirstName, c.CustomerLastName, o.OrderID FROM Customers c JOIN Orders o ON c.customerID = o.customerID join Delivery d ON o.OrderID = d.OrderID WHERE d.TypeOfDelivery = 'door delivery';

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

CustomerFirstName	CustomerLastName	OrderID
Bob	Smith	1000002
Sarah	Brown	1000004
Rachel	Davis	1000006
Elizabeth	Taylor	1000008
Emily	Lee	1000010
Olivia	Johnson	1000012
Sophia	Wilson	1000014
Isabella	Jones	1000016
Amelia	Brown	1000018
Alexander	Davis	1000020
Ryan	Taylor	1000022
Mike	Miller	1000024

Show all | Number of rows: 25 Filter rows: Search this table

Explanation: CustomerFirstName and CustomerLastName from the Customers table as well as the OrderID from the Orders table are the columns that the SELECT statement specifies to retrieve. Customers and Orders are the two tables that are specified in the FROM statement. The customerID column in the Customers table is connected to the equivalent column in the Orders table by the JOIN condition. The second JOIN statement uses the OrderID column to connect the Orders table with the Delivery table.

3. List the Details of Customers who ordered onions?

Query:

SELECT c.* FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

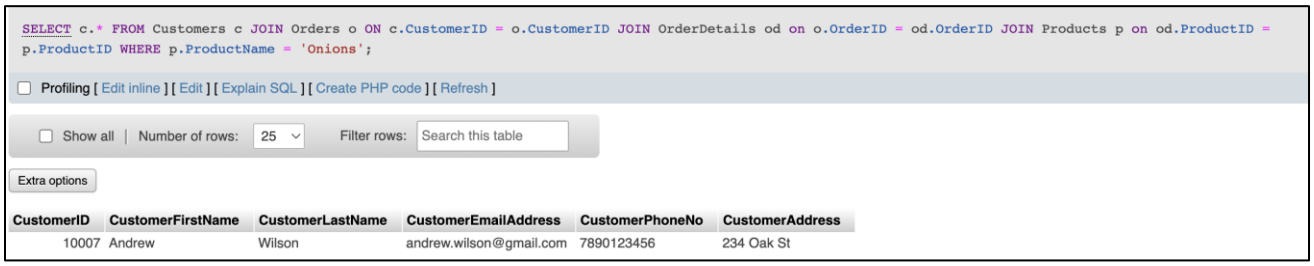
JOIN OrderDetails od on o.OrderID = od.OrderID

JOIN Products p on od.ProductID = p.ProductID

WHERE p.ProductName = 'Onions';

```
USE groupstyx;
SELECT c.* FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
JOIN OrderDetails od on o.OrderID = od.OrderID
JOIN Products p on od.ProductID = p.ProductID
WHERE p.ProductName = 'Onions';
```

Result:



Explanation: This query brings data on clients who have ordered onions from a database. Using the CustomerID as the common identifier, it first links the Customers table with the Orders table. The OrderID is then used as the common identifier to join the OrderDetails table to the Orders table. Finally, it uses the ProductID as the common identifier to combine the Products table to the OrderDetails table. The WHERE clause of the query then has a restriction that limits the results to just orders with the product name "Onions." All columns from the Customers table that satisfy this requirement are chosen by the SELECT statement at the start of the query.

4. List the Details of Customers whose order Amount is above 150?

Query:

Select c.* from Customers c

join Orders o on c.CustomerID = o.CustomerID

where TotalAmount > 150;

```sql
USE groupstyx;
Select c.* from Customers c
join Orders o on c.CustomerID = o.CustomerID
where TotalAmount > 150;
```

Result:

```
Select c.* from Customers c join Orders o on c.CustomerID = o.CustomerID where TotalAmount > 150;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

CustomerID	CustomerFirstName	CustomerLastName	CustomerEmailAddress	CustomerPhoneNo	CustomerAddress
10001	Jane	Doe	janedoe@example.com	0987654321	456 Oak Ave
10008	Elizabeth	Taylor	elizabeth.taylor@hotmail.com	8901234567	567 Maple Ave
10009	William	Jones	william.jones@yahoo.com	9012345678	890 Cedar Rd
10014	Sophia	Wilson	sophia.wilson@yahoo.com	5678901235	678 Pine Rd
10022	Ryan	Taylor	ryan.taylor@yahoo.com	4567890125	345 bryan st

Explanation: The database is being prompted by the query to retrieve data from the "Customers" and "Orders" tables. In order for the generated data to show the customers who have placed orders, it wants to combine these tables based on the "CustomerID" field. Then, any customers whose "TotalAmount" (presumably the total amount they spent on orders) is less than or equal to 150 are excluded from the query. The query then chooses any columns (denoted by "c.*") from the "Customers" table that satisfy these requirements. The query is requesting a list of clients who have placed orders and whose cumulative order total exceeds 150.

5. List the Details of the PesonalShopper who stay in Denton or Dallas?

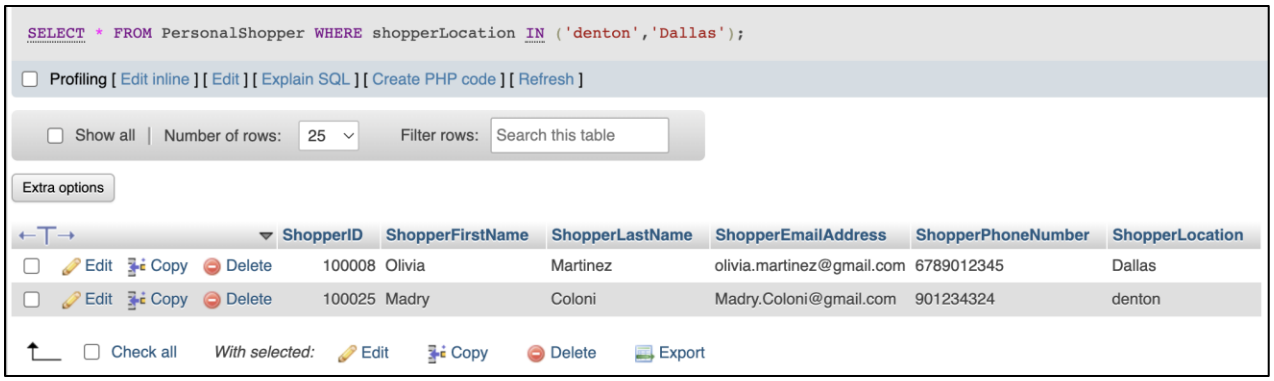
   Query:

   SELECT * FROM PersonalShopper

   WHERE shopperLocation IN ('denton','Dallas');

```sql
USE groupstyx;
SELECT * FROM PersonalShopper
WHERE shopperLocation IN ('denton','Dallas');
```

Result:



Explanation: This query is requesting to retrieve all the information stored in a table named "PersonalShopper" but only for the rows where the location of the shopper is either "Denton" or "Dallas". The "*" character denotes that all columns in the table will be included in the query result.

6. Count the total number of orders made by each customer.

Query:

SELECT c.CustomerFirstName, c.CustomerLastName, COUNT(o.OrderID) AS TotalOrders

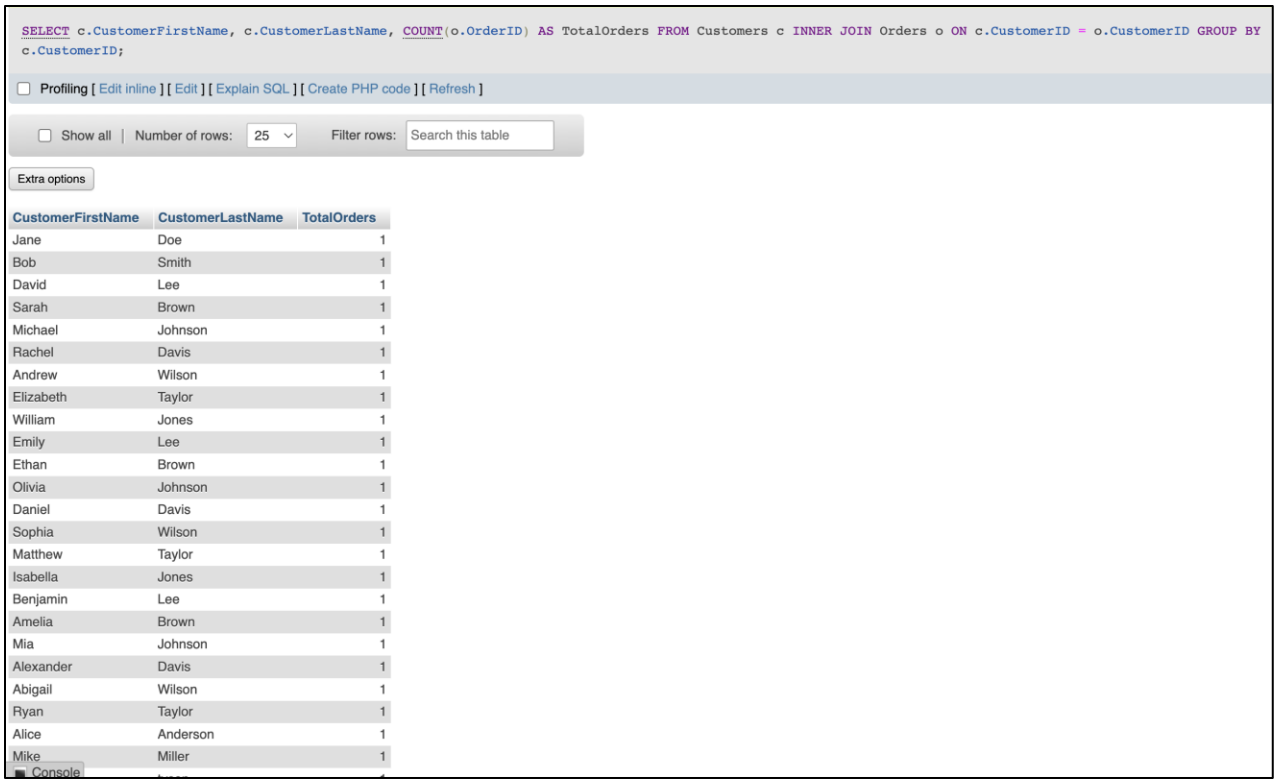
FROM Customers c

INNER JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID;

```
USE groupstyx;
SELECT c.CustomerFirstName, c.CustomerLastName, COUNT(o.OrderID) AS TotalOrders
FROM Customers c
INNER JOIN Orders o ON c.CustomerID = o.CustomerID
GROUP BY c.CustomerID;
```

Result:



```
SELECT c.CustomerFirstName, c.CustomerLastName, COUNT(o.OrderID) AS TotalOrders FROM Customers c INNER JOIN Orders o ON c.CustomerID = o.CustomerID GROUP BY c.CustomerID;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

CustomerFirstName	CustomerLastName	TotalOrders
Jane	Doe	1
Bob	Smith	1
David	Lee	1
Sarah	Brown	1
Michael	Johnson	1
Rachel	Davis	1
Andrew	Wilson	1
Elizabeth	Taylor	1
William	Jones	1
Emily	Lee	1
Ethan	Brown	1
Olivia	Johnson	1
Daniel	Davis	1
Sophia	Wilson	1
Matthew	Taylor	1
Isabella	Jones	1
Benjamin	Lee	1
Amelia	Brown	1
Mia	Johnson	1
Alexander	Davis	1
Abigail	Wilson	1
Ryan	Taylor	1
Alice	Anderson	1
Mike	Miller	1

Console

Explanation:

This SQL SELECT statement retrieves the first and last names of all customers, along with the total number of orders each customer has made. The statement uses an inner join to link the Customers and Orders tables, and the GROUP BY clause to group the results by the CustomerID field.

Conclusion: The Online Mart project offers a complete solution for an online retail platform that works to enhance the effectiveness and convenience of the client buying experience. By offering a personalized service that saves clients time and effort, Safe delivery, effective payment processing, and individualized recommendations are all made possible by the database system.