SQL querying and database manipulation project

Project Description

The project encompasses a variety of functionalities and includes SQL filter commands, demonstrating proficiency in SQL querying and database manipulation. Key features of the project include:

1. **Retrieving Customer Information:**

2. **Fetching Product Details by Category:** .

3. **Finding Orders Placed Within a Specific Date Range:**

4. **Updating Employee Information:**

5. **Inserting New Products or Customers Into the Database:**

6. **Deleting Outdated Records:**

Joining Multiple Tables to Generate Reports:

This project showcases advanced SQL skills, including the use of filter commands to retrieve, manipulate, and analyze data within a relational database environment.

Tables(There are 8 tables)

>Customers:This table contains information about customers, such as their name, contact details,and addresses.

Number of Records: 91

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany

>Categories:Stores the categories of products available in the database. Each product belongs to a specific category.

Number of Records: 8

CategoryID	CategoryName	Description
1	Beverages	Soft drinks, coffees, teas, beers, and ales
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
3	Confections	Desserts, candies, and sweet breads
4	Dairy Products	Cheeses
5	Grains/Cereals	Breads, crackers, pasta, and cereal
6	Meat/Poultry	Prepared meats
7	Produce	Dried fruit and bean curd
8	Seafood	Seaweed and fish

>Employees:Records data about employees who work for the company. This can include their name, birthdate , and other relevant information.

Number of Records: 10

EmployeeID	LastName	FirstName	BirthDate	Photo	Notes
1	Davolio	Nancy	12/8/1968	EmpID1.pic	Education includes a BA in psychology from Colorado State University. She also completed (The Art of the Cold Call). Nancy is a member of 'Toastmasters International'.
2	Fuller	Andrew	2/19/1952	EmpID2.pic	Andrew received his BTS commercial and a Ph.D. in international marketing from the University of Dallas. He is fluent in French and Italian and reads German. He joined the company as a sales representative, was promoted to sales manager and was then named vice president of sales. Andrew is a member of the Sales Management Roundtable, the Seattle Chamber of Commerce, and the Pacific Rim Importers Association.
3	Leverling	Janet	8/30/1963	EmpID3.pic	Janet has a BS degree in chemistry from Boston College).

>OrderDetails:Stores the details of each product included in an order, including the quantity and unit price.

Number of Records: 518

OrderDetailID	OrderID	ProductID	Quantity
1	10248	11	12
2	10248	42	10
3	10248	72	5
4	10249	14	9
5	10249	51	40
6	10250	41	10
7	10250	51	35
8	10250	65	15

>Orders:Contains information about individual orders placed by customers, including the customer who placed the order, the employee who handled it, and the order date.

Number of Records: 196

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10248	90	5	7/4/1996	3
10249	81	6	7/5/1996	1
10250	34	4	7/8/1996	2
10251	84	3	7/8/1996	1
10252	76	4	7/9/1996	2
10253	34	3	7/10/1996	2
10254	14	5	7/11/1996	2
10255	68	9	7/12/1996	3

>Products:Holds details about each product available for purchase, including its name, price, supplier, and category.

Number of Records: 77

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35
6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25
7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30
8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40

>Shippers:Stores information about companies responsible for shipping orders to customers.

Number of Records: 3

ShipperID	ShipperName	Phone
1	Speedy Express	(503) 555-9831
2	United Package	(503) 555-3199
3	Federal Shipping	(503) 555-9931

>Suppliers:Contains data about the suppliers who provide products to the company, including their contact information and address.

Number of Records: 29

SupplierID	SupplierName	ContactName	Address	City	PostalCode	Country	Phone
1	Exotic Liquid	Charlotte Cooper	49 Gilbert St.	Londona	EC1 4SD	UK	(171) 555-2222
2	New Orleans Cajun Delights	Shelley Burke	P.O. Box 78934	New Orleans	70117	USA	(100) 555-4822
3	Grandma Kelly's Homestead	Regina Murphy	707 Oxford Rd.	Ann Arbor	48104	USA	(313) 555-5735
4	Tokyo Traders	Yoshi Nagase	9-8 Sekimai Musashino-shi	Tokyo	100	Japan	(03) 3555-5011
5	Cooperativa de Quesos 'Las Cabras'	Antonio del Valle Saavedra	Calle del Rosal 4	Oviedo	33007	Spain	(98) 598 76 54

Data Queries and Manipulation

>Fetching product details by category

SELECT * FROM Products WHERE CategoryID = 1;

SQL Statement:

Get your own SQL server

```
SELECT * FROM Products WHERE CategoryID = 1;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 12

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
24	Guaraná Fantástica	10	1	12 - 355 ml cans	4.5
34	Sasquatch Ale	16	1	24 - 12 oz bottles	14
35	Steeleye Stout	16	1	24 - 12 oz bottles	18
38	Côte de Blaye	18	1	12 - 75 cl bottles	263.5
39	Chartreuse verte	18	1	750 cc per bottle	18
43	Ipoh Coffee	20	1	16 - 500 g tins	46

>>Explanation: The first part of the query retrieves data from the "Products" table. The asterisk (*) indicates that all columns should be included in the output. The FROM clause specifies the source table, which is "Products".

>>Output Explanation: This query returns all products that belong to a specific category, where the CategoryID is equal to 1. It filters the products based on their category. For example, if CategoryID 1 represents the category "Beverages", the output would include all products categorized as beverages. The WHERE clause restricts the results to only those rows where the CategoryID column has a value of 1, indicating the specified category.

>Finding orders placed within a specific date range:

SELECT * FROM Orders WHERE OrderDate BETWEEN #1996-01-01# AND #1996-07-12#;

SQL Statement:

Get your own SQL server

```
SELECT * FROM Orders WHERE OrderDate BETWEEN #1996-01-01# AND #1996-07-12#;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 8

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10248	90	5	7/4/1996	3
10249	81	6	7/5/1996	1
10250	34	4	7/8/1996	2
10251	84	3	7/8/1996	1
10252	76	4	7/9/1996	2
10253	34	3	7/10/1996	2
10254	14	5	7/11/1996	2
10255	68	9	7/12/1996	3

>>**Explanation:** The first part of the query retrieves data from the "Orders" table. The asterisk (*) indicates that all columns should be included in the output. The FROM clause specifies the source table, which is "Orders". The WHERE clause filters the results based on a condition related to the "OrderDate" column.
>>**Output Explanation:** This query returns all orders that were placed within a specific date range. The BETWEEN keyword is used to specify a range, and the pound sign (#) is often used to denote date literals in some SQL databases. In this case, the query filters orders where the "OrderDate" falls between January 1, 1996, and July 12, 1996. The output would include all orders placed within this date range.

>WHERE Clause

SELECT * FROM Products WHERE Price > 50;

SQL Statement:

Get your own SQL server

```
SELECT * FROM Products WHERE Price > 50;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 7

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97
18	Carnarvon Tigers	7	8	16 kg pkg.	62.5
20	Sir Rodney's Marmalade	8	3	30 gift boxes	81
29	Thüringer Rostbratwurst	12	6	50 bags x 30 sausgs.	123.79
38	Côte de Blaye	18	1	12 - 75 cl bottles	263.5
51	Manjimup Dried Apples	24	7	50 - 300 g pkgs.	53
59	Raclette Courdavault	28	4	5 kg pkg.	55

>>**Explanation:** The first part of the query retrieves data from the "Products" table. The asterisk (*) indicates that all columns should be included in the output. The FROM clause specifies the source table, which is "Products". The WHERE clause filters the results based on a condition related to the "Price" column.
>>**Output Explanation:** This query returns all products from the "Products" table where the price is greater than 50. It filters the products based on their price, specifically selecting those with a price higher than 50. The output would include details of products that meet this condition, such as ProductID, ProductName, SupplierID, CategoryID, Unit, and Price.

>OR Operator:

SELECT * FROM Customers WHERE Country = 'USA' OR Country = 'Canada';

SQL Statement:

Get your own SQL server

```
SELECT * FROM Customers WHERE Country = 'USA' OR Country = 'Canada';
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 16

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada
32	Great Lakes Food Market	Howard Snyder	2732 Baker Blvd.	Eugene	97403	USA
36	Hungry Coyote Import Store	Yoshi Latimer	City Center Plaza 516 Main St.	Elgin	97827	USA
42	Laughing Bacchus Wine Cellars	Yoshi Tannamuri	1900 Oak St.	Vancouver	V3F 2K1	Canada
43	Lazy K Kountry Store	John Steel	12 Orchestra Terrace	Walla Walla	99362	USA
45	Let's Stop N Shop	Jaime Yorres	87 Polk St. Suite 5	San Francisco	94117	USA
48	Lonesome Pine Restaurant	Fran Wilson	89 Chiaroscuro Rd.	Portland	97219	USA
51	Mère Paillarde	Jean Fresnière	43 rue St. Laurent	Montréal	H1J 1C3	Canada

>>**Explanation:** This SQL query retrieves data from the "Customers" table. The asterisk (*) indicates that all columns from the table should be included in the output. The FROM clause specifies the source table, which is "Customers". The WHERE clause filters the results based on two conditions related to the "Country" column.
>>**Output Explanation:** This query returns all customers from the "Customers" table who are from either the USA or Canada. The WHERE clause uses the OR operator to combine two conditions: Country = 'USA' and Country = 'Canada'. This means that the output will include

customers whose country is either 'USA' or 'Canada'. The output would include details of customers such as CustomerID, CustomerName, ContactName, Address, City, PostalCode, Country, and Phone for those meeting the specified conditions.

>LIKE Operator

SELECT * FROM Customers WHERE ContactName LIKE 'A%';

SQL Statement:

Get your own SQL server

```
SELECT * FROM Customers WHERE ContactName LIKE 'A%';
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 10

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
19	Eastern Connection	Ann Devon	35 King George	London	WX3 6FW	UK
21	Familia Arquibaldo	Aria Cruz	Rua Orós, 92	São Paulo	05442-030	Brazil
31	Gourmet Lanchonetes	André Fonseca	Av. Brasil, 442	Campinas	04876-786	Brazil
41	La maison d'Asie	Annette Roulet	1 rue Alsace-Lorraine	Toulouse	31000	France
52	Morgenstern Gesundkost	Alexander Feuer	Heerstr. 22	Leipzig	04179	Germany
69	Romero y tomillo	Alejandra Camino	Gran Vía, 1	Madrid	28001	Spain

>>**Explanation:** This SQL query retrieves data from the "Customers" table. The asterisk (*) indicates that all columns from the table should be included in the output. The FROM clause specifies the source table, which is "Customers". The WHERE clause filters the results based on a condition using the LIKE operator with a pattern match.

>>**Output Explanation:** This query returns all customers from the "Customers" table whose contact names start with the letter 'A'. The LIKE operator is used with the pattern 'A%', where '%' is a wildcard character that matches any sequence of characters, and 'A' specifies that the contact name must begin with 'A'. Therefore, the output would include details of customers whose contact names start with 'A', such as CustomerID, CustomerName, ContactName, Address, City, PostalCode, Country, and Phone.

>GROUP BY Clause

SELECT CategoryID, COUNT(*) AS ProductCount FROM Products GROUP BY CategoryID;

SQL Statement:

Get your own SQL server

```
SELECT CategoryID, COUNT(*) AS ProductCount FROM Products GROUP BY CategoryID;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 8

CategoryID	ProductCount
1	12
2	12
3	13
4	10
5	7
6	6
7	5
8	12

>>**Explanation:** This SQL query retrieves data from the "Products" table and performs a count of products for each unique CategoryID. The SELECT statement specifies the columns to be included in the output: CategoryID and the count of products for each category, aliased as "ProductCount". The COUNT(*) function is used to count the number of rows for each group defined by the GROUP BY clause.

>>**Output Explanation:** This query generates a summary of product counts for each category in the database. It groups the products by their CategoryID and calculates the number of products within each category. The output would consist of two columns: CategoryID and ProductCount, where each row represents a unique category along with the count of products belonging to that category.

>HAVING Clause

SELECT CategoryID, COUNT(*) AS ProductCount FROM Products GROUP BY CategoryID HAVING COUNT(*) > 5;

SQL Statement:

```
SELECT CategoryID, COUNT(*) AS ProductCount FROM Products GROUP BY CategoryID HAVING COUNT(*) > 5;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 7

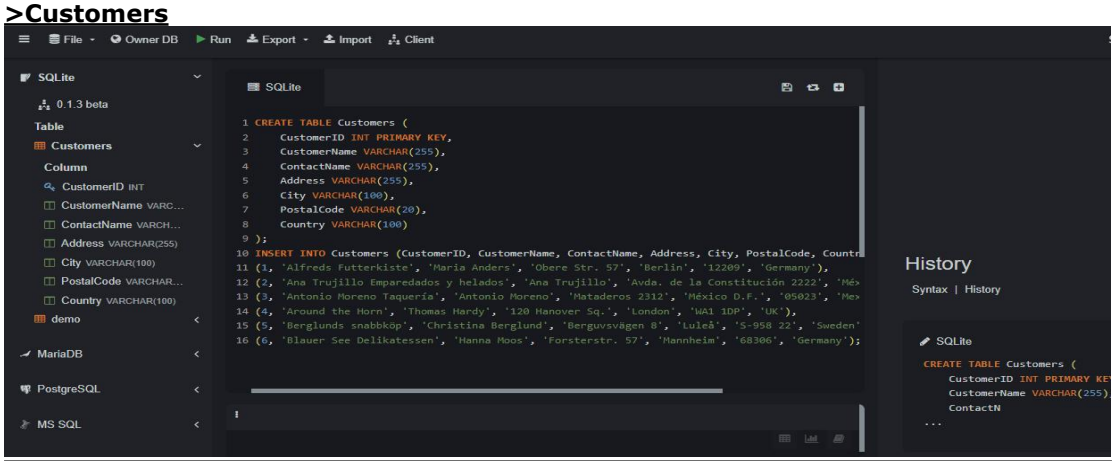
CategoryID	ProductCount
1	12
2	12
3	13
4	10
5	7
6	6
8	12

>>Explanation: This SQL query retrieves data from the "Products" table and performs a count of products for each unique CategoryID. The SELECT statement specifies the columns to be included in the output: CategoryID and the count of products for each category, aliased as "ProductCount". The COUNT(*) function is used to count the number of rows for each group defined by the GROUP BY clause. Additionally, the HAVING clause is used to filter the groups based on the aggregated result of the COUNT(*) function.

>>Output Explanation: This query generates a summary of product counts for each category in the database, similar to the previous query. However, the HAVING clause filters the results to include only those categories where the count of products is greater than 5. Therefore, the output would consist of CategoryID and ProductCount columns for categories with more than 5 products. This allows for the retrieval of categories with a significant number of associated products.

Commands for creating tables

>Customers



This code will create the Customers table with the appropriate columns and data types, and then insert the provided data into the table.

>Categories

```sql
CREATE TABLE Categories (
    CategoryID INT PRIMARY KEY,
    CategoryName VARCHAR(255),
    Description TEXT
);
INSERT INTO Categories (CategoryID, CategoryName, Description) VALUES
(1, 'Beverages', 'Soft drinks, coffees, teas, beers, and ales'),
(2, 'Condiments', 'Sweet and savory sauces, relishes, spreads, and seasonings'),
(3, 'Confections', 'Desserts, candies, and sweet breads'),
(4, 'Dairy Products', 'Cheeses'),
(5, 'Grains/Cereals', 'Breads, crackers, pasta, and cereal'),
(6, 'Meat/Poultry', 'Prepared meats'),
(7, 'Produce', 'Dried fruit and bean curd'),
(8, 'Seafood', 'Seaweed and fish');
```

This code will create the Categories table with the appropriate columns and data types, and then insert the provided data into the table.

>Employees

```sql
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    LastName VARCHAR(255),
    FirstName VARCHAR(255),
    BirthDate DATE,
    Photo VARCHAR(255),
    Notes TEXT
);
INSERT INTO Employees (EmployeeID, LastName, FirstName, BirthDate, Photo, Notes) VALUES
(1, 'Davolio', 'Nancy', '1968-12-08', 'EmpID1.pic', 'Education includes a BA in psychology from
(2, 'Fuller', 'Andrew', '1952-02-19', 'EmpID2.pic', 'Andrew received his BTS commercial and a PH
(3, 'Leverling', 'Janet', '1963-08-30', 'EmpID3.pic', 'Janet has a BS degree in chemistry from B
(4, 'Peacock', 'Margaret', '1958-09-19', 'EmpID4.pic', 'Margaret holds a BA in English literatur
(5, 'Buchanan', 'Steven', '1955-03-04', 'EmpID5.pic', 'Steven Buchanan graduated from St. Andrew
(6, 'Suyama', 'Michael', '1963-07-02', 'EmpID6.pic', 'Michael is a graduate of Sussex Universit
```

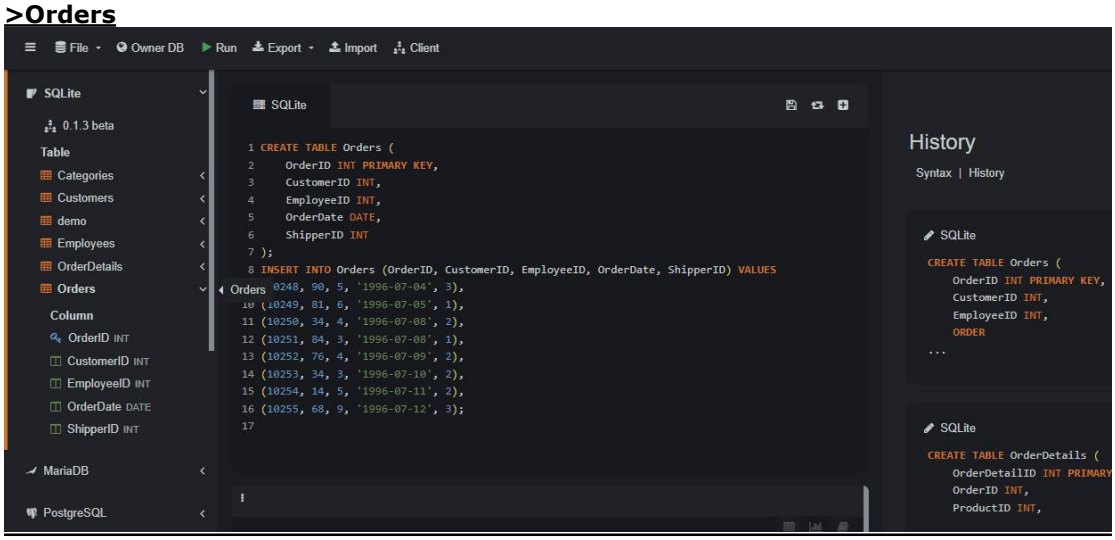
This code will create the Employees table with the appropriate columns and data types, and then insert the provided data into the table.

>OrderDetails

```sql
    OrderDetailID INT PRIMARY KEY,
    OrderID INT,
    ProductID INT,
    Quantity INT
);
INSERT INTO OrderDetails (OrderDetailID, OrderID, ProductID, Quantity) VALUES
(1, 10248, 11, 12),
(2, 10248, 42, 10),
(3, 10248, 72, 5),
(4, 10249, 14, 9),
(5, 10249, 51, 40),
(6, 10250, 41, 10),
(7, 10250, 51, 35),
(8, 10250, 65, 15);
```

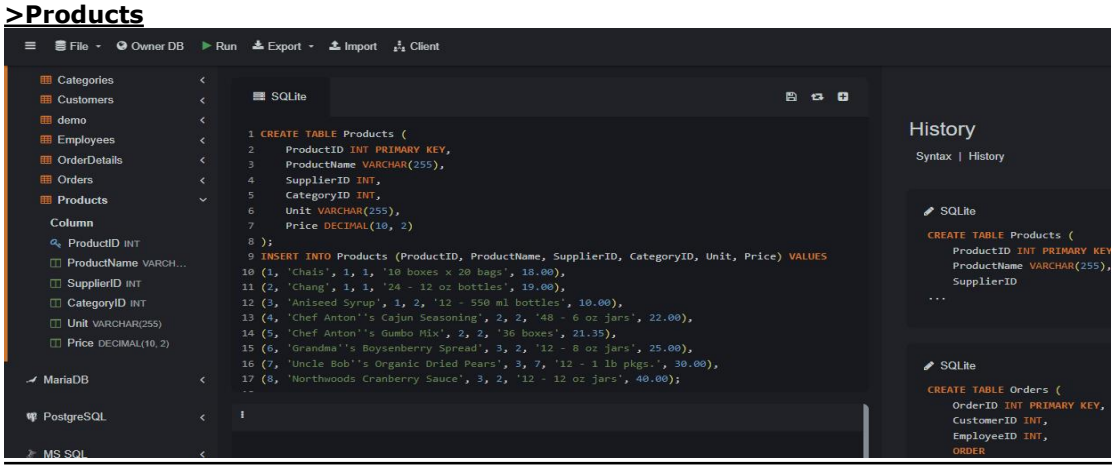
This code will create the OrderDetails table with the appropriate columns and data types, and then insert the provided data into the table.

>Orders



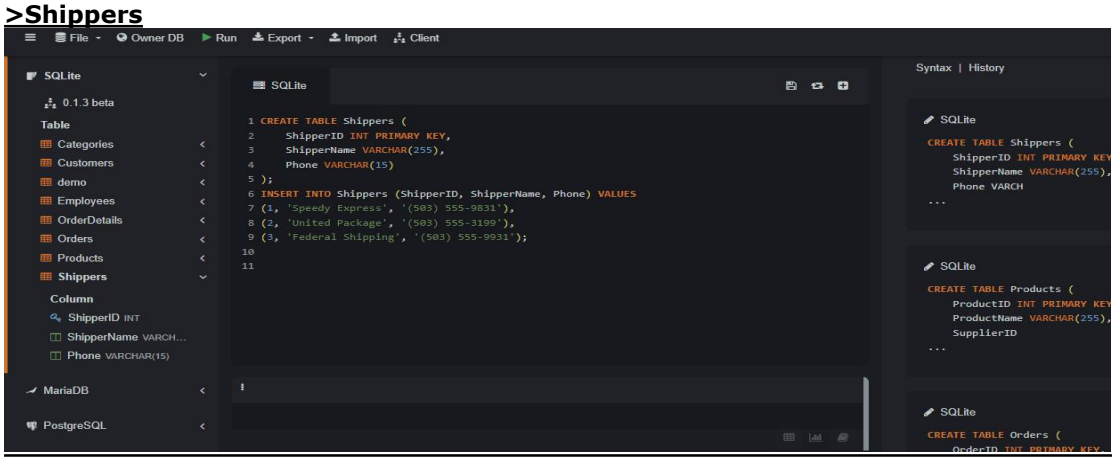
This code will create the Orders table with the appropriate columns and data types, and then insert the provided data into the table.

>Products



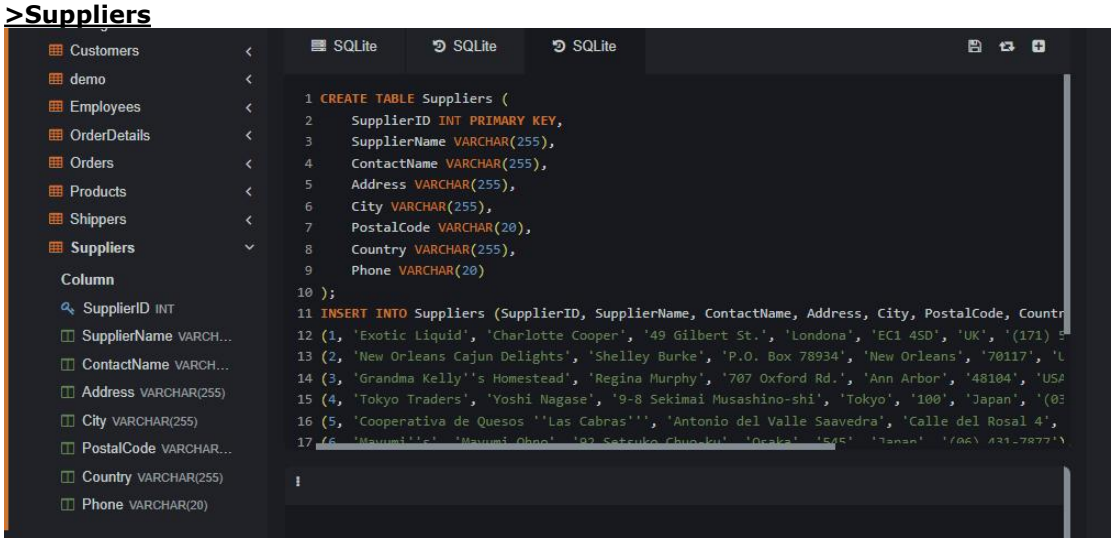
This code will create the Products table with the appropriate columns and data types, and then insert the provided data into the table.

>Shippers



This code will create the Products table with the appropriate columns and data types, and then insert the provided data into the table.

>Suppliers



This code will create the Suppliers table with the appropriate columns and data types, and then insert the provided data into the table.

Conclusion

So, here's what I've done: I've created a database layout for managing inventory. There are tables for products, categories, suppliers, customers, employees, orders, order details, and shippers. Each table holds specific details crucial for keeping track of inventory, like product names, supplier information, customer details, and more.

To make it more real, I've added some example data to these tables. This data includes product names, supplier contacts, customer addresses, and other relevant information. It's like stocking up a store with items before opening.

With this setup, I've essentially built the foundation for an inventory management system. It's equipped to handle tasks such as monitoring stock levels, processing orders, and generating reports to analyze sales performance.

In essence, this project represents a hands-on approach to creating an inventory management system from scratch. It showcases how SQL commands can be used to structure a database and manage its contents, all within a single-person project framework.