SOL CAFÉ MANAGEMENT SYSTEM

Brewing Success, One Data-Driven Decision at a Time

BY RAHUL PATEL

GitHub
LinkedIn

Table of Contents

Section Title	Page Number
1. Introduction	2
2. Business Challenges	2
3. Project Objectives	3
4. Entity Relationship Diagram (ERD)	4
5. System Overview and Final Field List	5
6. Relationships Between Tables	7
7. SQL Views	8
8. Sample Queries	12
9. Future Enhancements	14
10. Conclusion	14

1. Introduction

Sol Café is a growing coffee business that operates both online and in-store. As the business expanded, it became harder to manage information like sales, customers, employees, products, and shipments. To solve this, we designed a relational database that keeps all this data organized and connected.

Our goal was to build a system that's easy to use, reliable, and helps the business make better decisions. We used SQL to create views that show useful insights, like total revenue, best-selling products, and loyal customers. This project helped Sol Café streamline its operations and prepare for future growth.

2. Business Challenges

Sol Café faced several challenges typical of a growing business:

- Scattered Information - Sales, customer, and inventory data were kept in different spreadsheets and paper files. This caused confusion, data errors, and made it hard to find what was needed quickly.
- No Clear View of Customer or Sales - There was no single place to track customer habits or see which products were selling well. Without this insight, marketing and sales strategies were based on guesswork.
- Trouble Tracking Employees and Shipments - It was hard to see who was doing what among staff. Deliveries from suppliers were also difficult to track, leading to delays or inventory issues.
- Online vs In Store Sales not Connected - Online and in-store sales were recorded separately, so comparing performance across channels was time-consuming and often inaccurate.

A unified database would solve these issues and enable better decision-making.

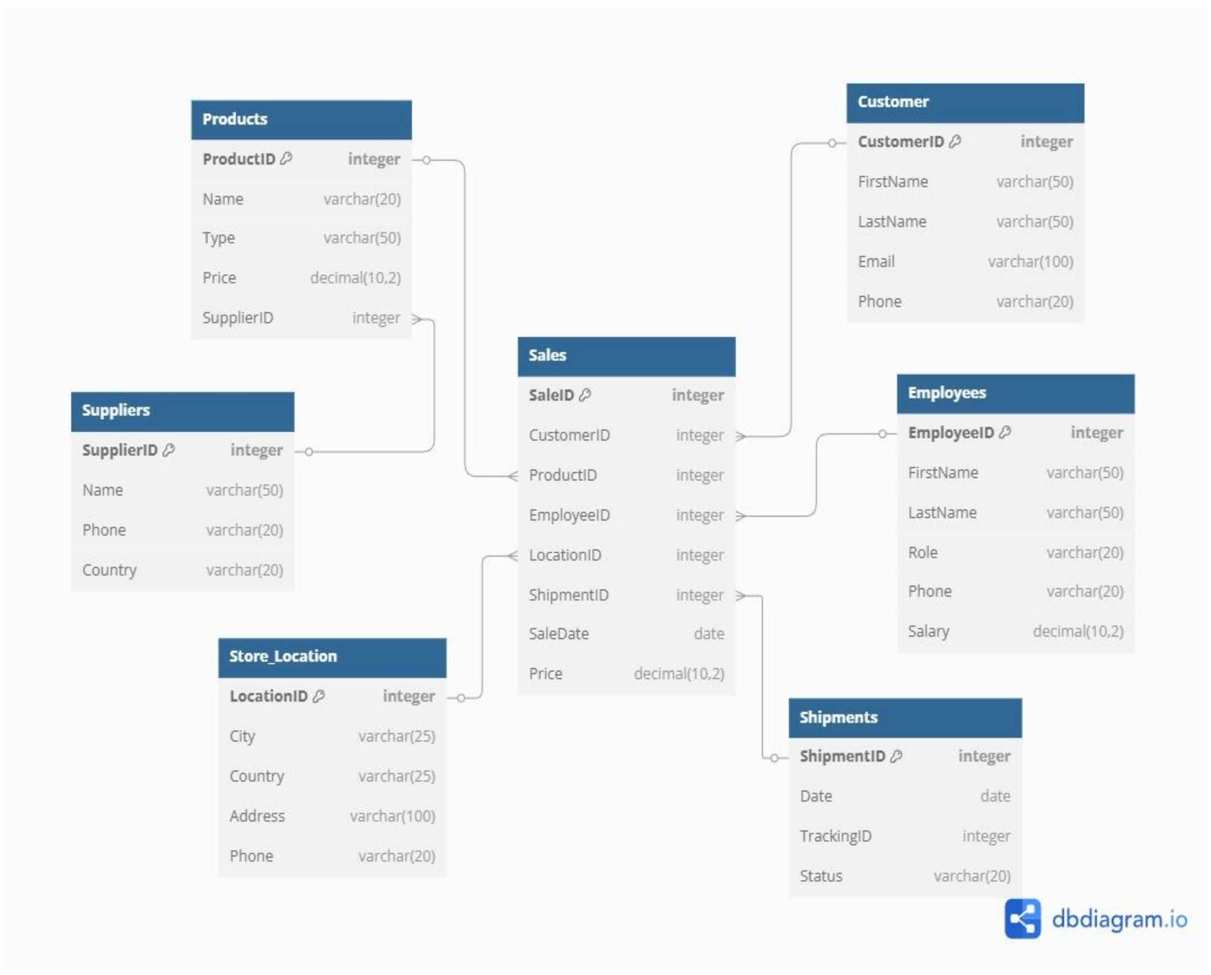
3. Project Objectives

The main objectives for this database project were:

- Organize all business data in one system - Consolidate information that was previously scattered across spreadsheets, files, and disconnected systems into a single, unified database. This ensures better accessibility, consistency, and control over critical business data.
- Establish clear relationships between entities - Define how key parts of the business, such as customers, sales, employees, products, suppliers, and shipments interact with each other. This relational structure eliminates data duplication and allows for accurate linking of records across tables.
- Enable real-time analysis of sales, products, employees, and customers - Support day-to-day decision-making by enabling fast, accurate queries about sales performance, customer activity, product trends, and employee contributions. This allows managers to respond to business changes promptly.
- Support both in-store and online transactions - Design the system to handle orders from both physical store locations and the online store. This ensures Sol Café can monitor and compare performance across all channels in one place.
- Provide views for quick insights and reporting - Create SQL views that summarize key metrics, such as total revenue, best-selling products, employee sales totals, shipment statuses, and repeat customers so managers can access actionable insights without digging through raw data.

4. Entity Relationship Diagram

The Entity Relationship Diagram (ERD) illustrates how the core components of Sol Café's operations such as customers, sales, products, shipments, etc. are connected. It visually represents the relationships between tables and ensures a clear, organized data structure for the database.



5. Key Tables & Their Roles

A) Customers Table

Field Name	Data Type	Description
CustomerID	Int (PK)	Unique ID assigned to each customer
FirstName	Varchar (50)	Customers First Name
LastName	Varchar (50)	Customers Last Name
Email	Varchar (100)	Customers Email
Phone	Varchar (20)	Customer Phone Number

B) Suppliers Table

Field Name	Data Type	Description
SupplierID	Int (PK)	Unique ID assigned to each supplier
Name	Varchar (50)	Suppliers Name
Phone	Varchar (20)	Suppliers Phone Number
Country	Varchar (20)	Suppliers' country

C) Products Table

Field Name	Data Type	Description
ProductID	Int (PK)	Unique ID assigned to each product
Name	Varchar (20)	Product Name
Type	Varchar (50)	Type of bean or drink
SupplierID	Int (FK)	References the supplier providing the product
Price	Decimal (10, 2)	Price per unit

D) Stores Location Table

Field Name	Data Type	Description
StoreLocationID	Int (PK)	Unique ID assigned to each store location
City	Varchar (25)	City where store is located
Country	Varchar (25)	Country where the store is located
Address	Varchar (100)	Physical address
Phone	Varchar (20)	Stores contact number

E) Employees Table

Field Name	Data Type	Description
EmployeeID	Int (PK)	Unique ID assigned to each employee
FirstName	Varchar (50)	Employees first name
LastName	Varchar (50)	Employees last name
Role	Varchar (20)	Job Title
Phone	Varchar (20)	Contact Number
Salary	Decimal (10, 2)	Base Salary per month

F) Shipments Table

Field Name	Data Type	Description
ShipmentID	Int (PK)	Unique ID assigned for each shipment
SaleID	Int (FK)	References the sale the shipment is processing
Date	Date	Date of dispatch
Status	Varchar (20)	Status of the shipment

G) Sales Table

Field Name	Data Type	Description
SaleID	Int (PK)	Unique ID assigned for each sale
CustomerID	Int (FK)	References the customer who made the purchase
ProductID	Int (FK)	References the product being sold
EmployeeID	Int (FK)	References the employee who processes the sale
LocationID	Int (FK)	References the store where the sale was made
SaleDate	Date	Date the sale was made
Price	Decimal (10, 2)	The price of the products sold

6. Relationship Between Tables

Customer → Sales

- One customer can be linked to many sales.

Product → Sales

- Each product can appear in multiple sales.

Employee → Sales

- One employee can process many sales.

Store_Location → Sales

- Each store location can have multiple sales.

Sales → Shipments

- Each sale is associated with one shipment.

Supplier → Products

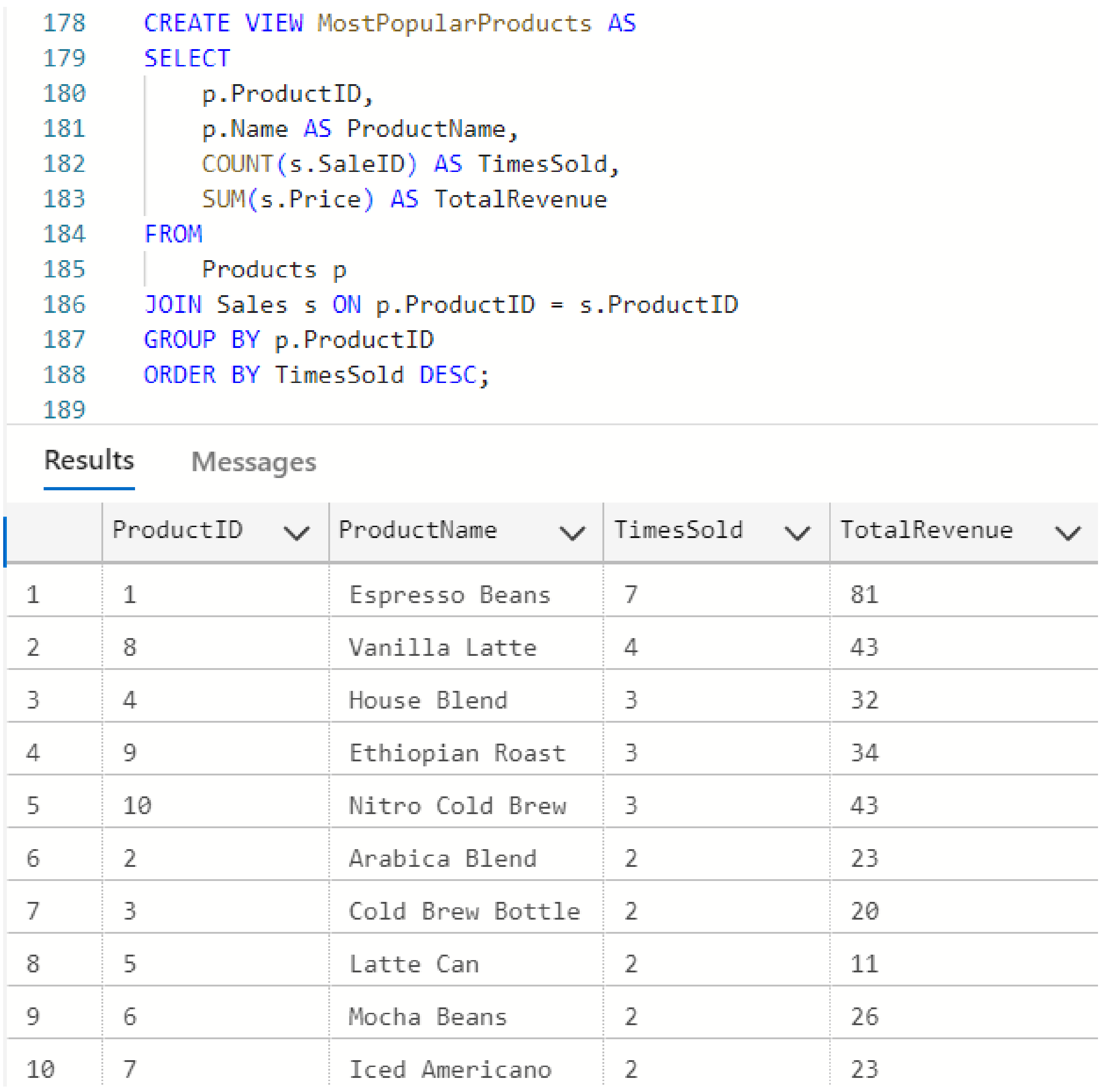
- One supplier can provide many products.

GitHub
LinkedIn

7. SQL Views

View 1 – Most Popular Products

This view helps us identify the most popular products of Sol Café by showing us how many times a product has been sold and the revenue generated from it.



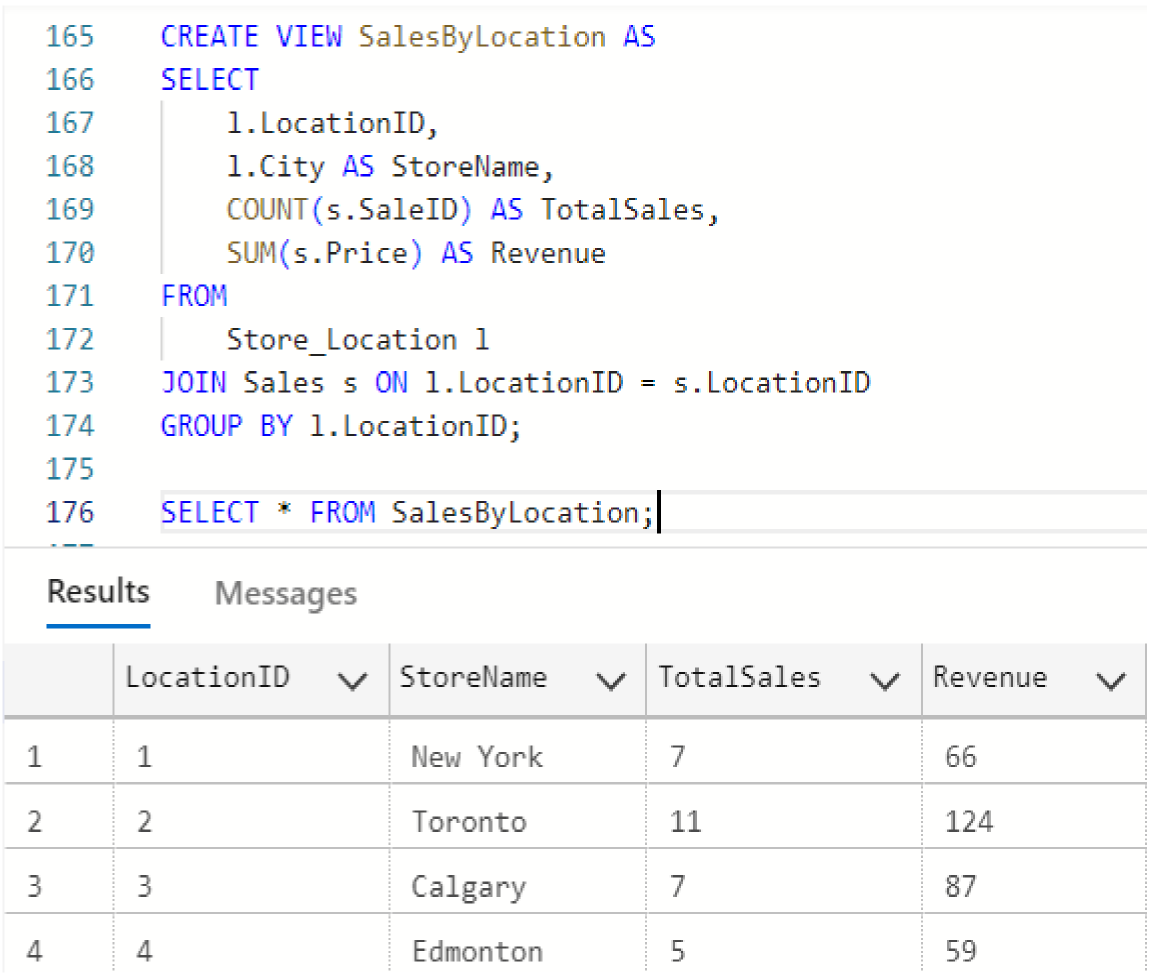
```sql
CREATE VIEW MostPopularProducts AS
SELECT
    p.ProductID,
    p.Name AS ProductName,
    COUNT(s.SaleID) AS TimesSold,
    SUM(s.Price) AS TotalRevenue
FROM
    Products p
JOIN Sales s ON p.ProductID = s.ProductID
GROUP BY p.ProductID
ORDER BY TimesSold DESC;
```

Results   Messages

	ProductID	ProductName	TimesSold	TotalRevenue
1	1	Espresso Beans	7	81
2	8	Vanilla Latte	4	43
3	4	House Blend	3	32
4	9	Ethiopian Roast	3	34
5	10	Nitro Cold Brew	3	43
6	2	Arabica Blend	2	23
7	3	Cold Brew Bottle	2	20
8	5	Latte Can	2	11
9	6	Mocha Beans	2	26
10	7	Iced Americano	2	23

View 2 – Sales By Location

This view helps us check which location is the most profitable and is generating the most revenue. It also helps us identify the weaker locations and promote them in a different way.



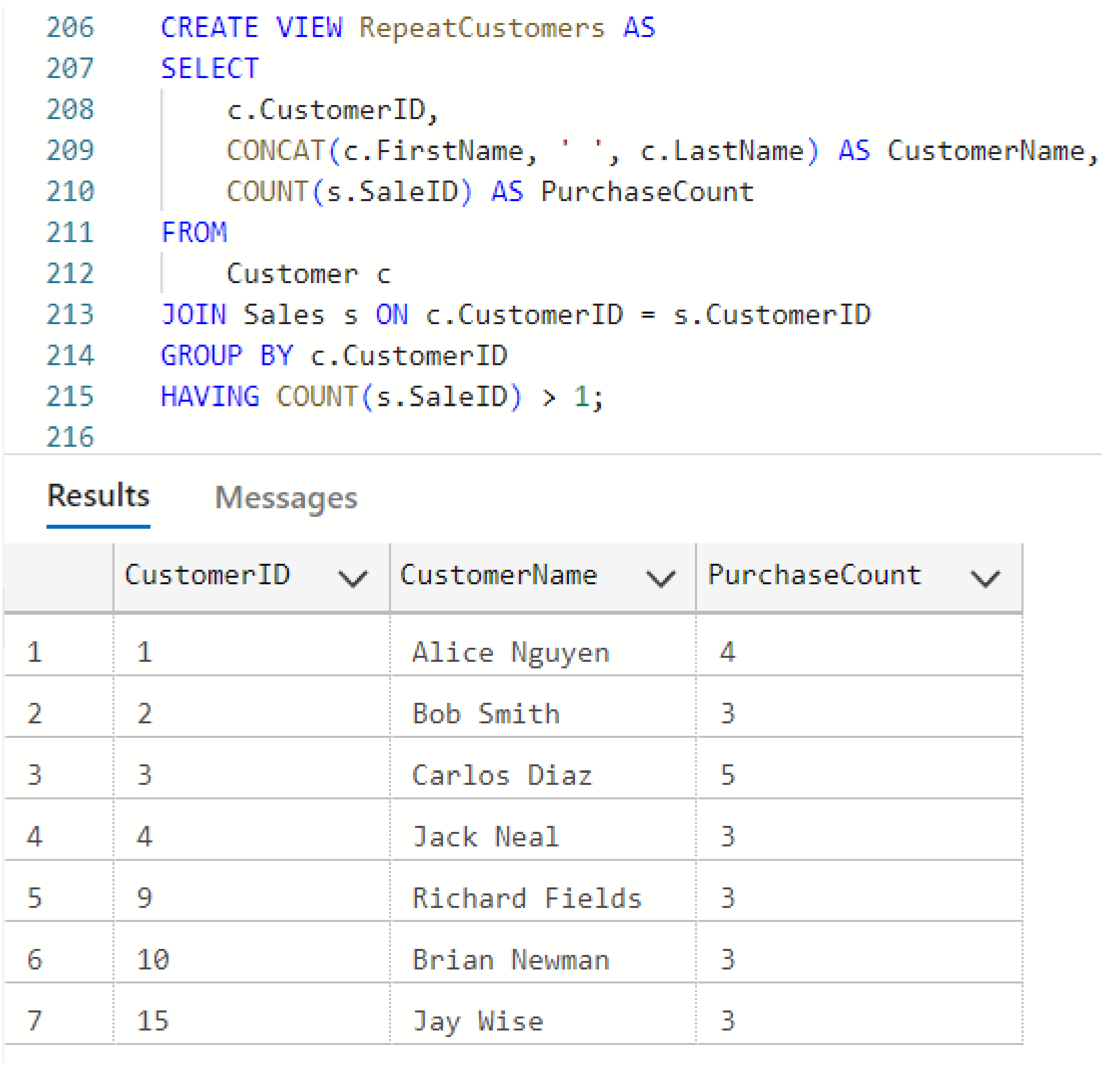
```sql
CREATE VIEW SalesByLocation AS
SELECT
    l.LocationID,
    l.City AS StoreName,
    COUNT(s.SaleID) AS TotalSales,
    SUM(s.Price) AS Revenue
FROM
    Store_Location l
JOIN Sales s ON l.LocationID = s.LocationID
GROUP BY l.LocationID;

SELECT * FROM SalesByLocation;
```

	LocationID	StoreName	TotalSales	Revenue
1	1	New York	7	66
2	2	Toronto	11	124
3	3	Calgary	7	87
4	4	Edmonton	5	59

View 3 – Repeat Customers

This view helps us see who our loyal customers are so we can offer them loyalty programs. It also helps us identify the customers we are not retaining so we can push targeted marketing campaigns.



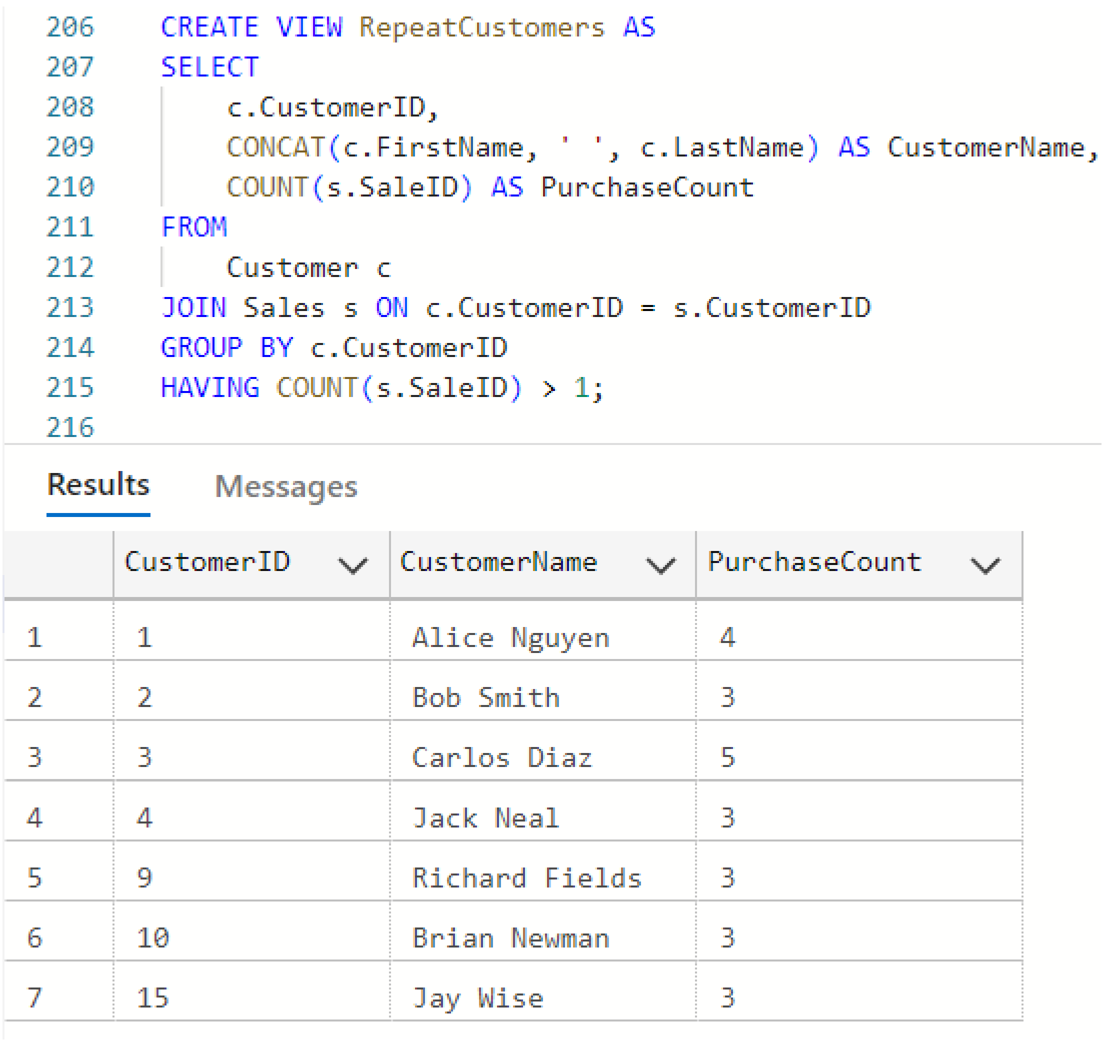
```sql
CREATE VIEW RepeatCustomers AS
SELECT
    c.CustomerID,
    CONCAT(c.FirstName, ' ', c.LastName) AS CustomerName,
    COUNT(s.SaleID) AS PurchaseCount
FROM
    Customer c
JOIN Sales s ON c.CustomerID = s.CustomerID
GROUP BY c.CustomerID
HAVING COUNT(s.SaleID) > 1;
```

Results Messages

	CustomerID	CustomerName	PurchaseCount
1	1	Alice Nguyen	4
2	2	Bob Smith	3
3	3	Carlos Diaz	5
4	4	Jack Neal	3
5	9	Richard Fields	3
6	10	Brian Newman	3
7	15	Jay Wise	3

View 4 – Customer Purchase History

This view helps us identify customer purchase history so we can see the customers preferences and launch the future products according to the customer preferences. This will also help us increase customer retention.



```sql
CREATE VIEW RepeatCustomers AS
SELECT
    c.CustomerID,
    CONCAT(c.FirstName, ' ', c.LastName) AS CustomerName,
    COUNT(s.SaleID) AS PurchaseCount
FROM
    Customer c
JOIN Sales s ON c.CustomerID = s.CustomerID
GROUP BY c.CustomerID
HAVING COUNT(s.SaleID) > 1;
```

Results   Messages

	CustomerID	CustomerName	PurchaseCount
1	1	Alice Nguyen	4
2	2	Bob Smith	3
3	3	Carlos Diaz	5
4	4	Jack Neal	3
5	9	Richard Fields	3
6	10	Brian Newman	3
7	15	Jay Wise	3

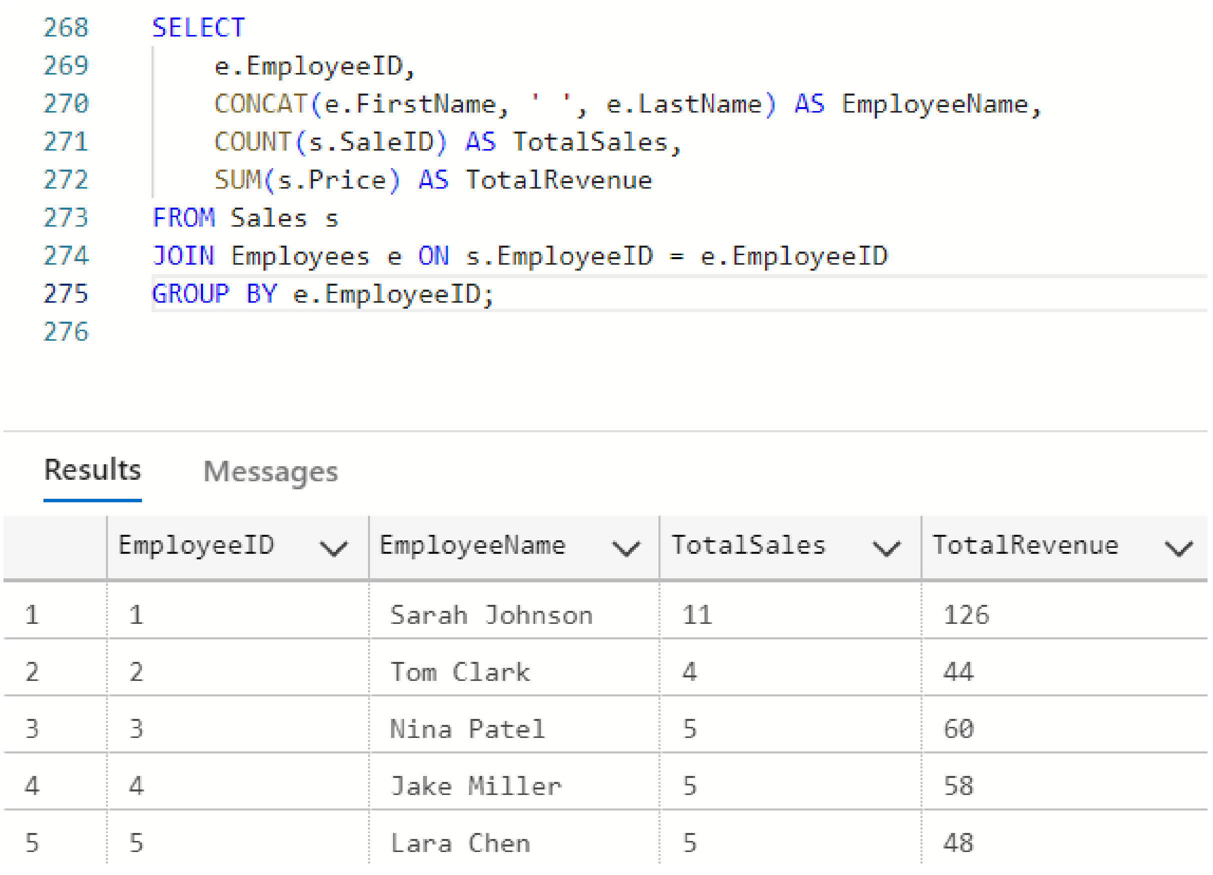
GitHub
LinkedIn

8. Sample Queries

There are also some sample queries that we can run with this sample database that we have created and explore the system:

1. Total Sales by Each Employee

```
SELECT
    e.EmployeeID,
    CONCAT(e.FirstName, ' ', e.LastName) AS EmployeeName,
    COUNT(s.SaleID) AS TotalSales,
    SUM(s.Price) AS TotalRevenue
FROM Sales s
JOIN Employees e ON s.EmployeeID = e.EmployeeID
GROUP BY e.EmployeeID;
```

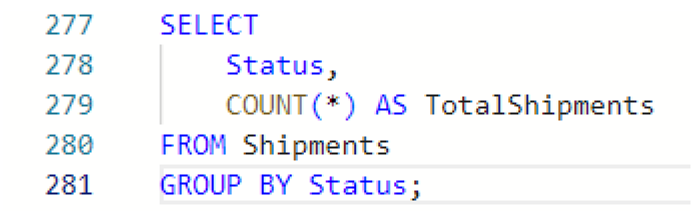


```
SELECT
    e.EmployeeID,
    CONCAT(e.FirstName, ' ', e.LastName) AS EmployeeName,
    COUNT(s.SaleID) AS TotalSales,
    SUM(s.Price) AS TotalRevenue
FROM Sales s
JOIN Employees e ON s.EmployeeID = e.EmployeeID
GROUP BY e.EmployeeID;
```

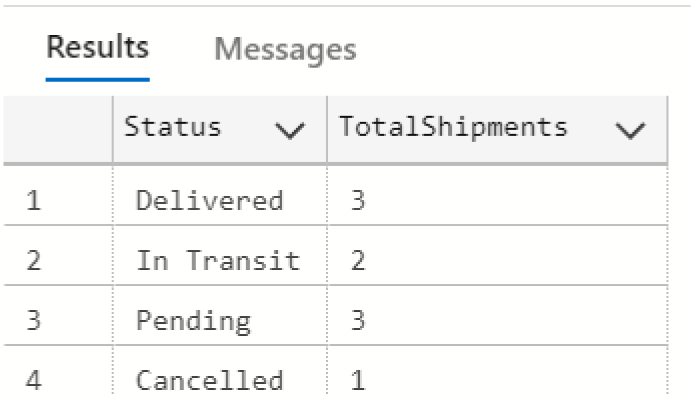
	EmployeeID	EmployeeName	TotalSales	TotalRevenue
1	1	Sarah Johnson	11	126
2	2	Tom Clark	4	44
3	3	Nina Patel	5	60
4	4	Jake Miller	5	58
5	5	Lara Chen	5	48

2. Shipment Status

```
SELECT
    Status,
    COUNT(*) AS TotalShipments
FROM Shipments
GROUP BY Status;
```



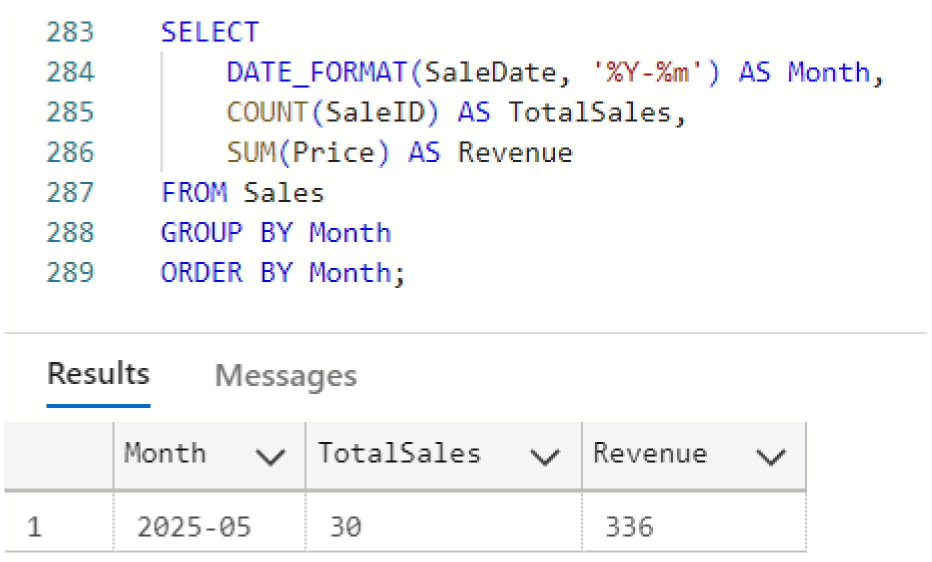
```
SELECT
    Status,
    COUNT(*) AS TotalShipments
FROM Shipments
GROUP BY Status;
```



	Status	TotalShipments
1	Delivered	3
2	In Transit	2
3	Pending	3
4	Cancelled	1

3. Sales per Month

```
SELECT
    DATE_FORMAT(SaleDate, '%Y-%m') AS Month,
    COUNT(SaleID) AS TotalSales,
    SUM(Price) AS Revenue
FROM Sales
GROUP BY Month
ORDER BY Month;
```



```sql
SELECT
    DATE_FORMAT(SaleDate, '%Y-%m') AS Month,
    COUNT(SaleID) AS TotalSales,
    SUM(Price) AS Revenue
FROM Sales
GROUP BY Month
ORDER BY Month;
```

	Month	TotalSales	Revenue
1	2025-05	30	336

9. Future Improvements

As Sol Café continues to grow, the database can be improved with new features to support more advanced needs:

- **Inventory Tracking** – To monitor stock levels and restock automatically.
- **Loyalty Program** – To reward repeat customers with points or offers.
- **Customer Reviews** – To collect feedback on products and service.
- **Returns Handling** – To track returned items and refund status.
- **Discounts and Promotions** – To manage special offers and seasonal pricing.
- **Payment Details** – To track payment methods and statuses.
- **User Roles and Permissions** – To control who can access or edit data.

These updates would make the system more powerful and useful for everyday operations.

10. Conclusion

The Sol Café database project helped turn scattered business data into a clear, organized system. It makes it easy to track sales, customers, employees, and shipments all in one place. With helpful SQL views, the café can now get quick insights and make better decisions.

Overall, this database sets a strong foundation for Sol Café to run smoothly and grow in the future. It also makes daily tasks faster and reduces errors by keeping everything connected. As the business expands, the database can easily be updated to support new features and needs.