Supply Chain Project

# Introduction

This project is a SQL-based analysis of a hypothetical supply chain database. Its main purpose is to demonstrate the practical application of database design principles and SQL queries to derive key business insights. By structuring and querying data related to customers, suppliers, products, and orders, the project identifies sales trends, evaluates supplier performance, and uncovers opportunities for business growth. In essence, it shows how to use SQL as a powerful tool for data analysis to support strategic decision-making in a business environment.

# Project Code

## Database & Tables Creation

This section contains the SQL statements to create the SupplyChain database and the necessary tables, including Customer, Orders, OrderItem, Product, and Supplier. It also defines primary keys, foreign keys, and indexes to ensure data integrity and performance.

create database SupplyChain;

use SupplyChain;

create table Customer (

Id int auto\_increment,

Name varchar(40) not null,

City varchar(40) null,

Phone varchar(40) null,

constraint pkCustomer primary key(Id)

);

create index idxCustomerName on Customer (

Name ASC

);

create table Orders (

Id int auto\_increment,

OrderDate varchar(40) not null,

OrderNo varchar(40) null,

CustomerId int not null,

TotalAmount decimal(12,2) null default 0,

constraint pkOrders primary key(Id)

);

create index idxOrdersOrderDate on Orders (

OrderDate ASC

);

create table OrderItem (

Id int auto\_increment,

OrderId int not null,

ProductId int not null,

UnitPrice decimal(12,2) not null default 0,

Quantity int not null default 1,

constraint pkOrderItem primary key(Id)

);

create index idxOrderItemOrderId on OrderItem (

OrderId ASC

);

create table Product (

Id int auto\_increment,

ProductName varchar(50) not null,

SupplierId int not null,

UnitPrice decimal(12,2) null default 0,

Package varchar(30) null,

IsDiscontinued bit not null default 0,

constraint pkProduct primary key (Id)

);

create index idxProductProductName on Product (

ProductName ASC

);

create table Supplier (

Id int auto\_increment,

CompanyName varchar(40) not null,

ContactName varchar(40) not null,

City varchar(20) null,

Phone varchar(20) null,

constraint pkSupplier primary key(Id)

);

create index idxSupplierCompanyName on Supplier (

CompanyName ASC

);

# Adding Foreign Key Constraints

alter table Orders add foreign key(CustomerId) references Customer(Id);

alter table OrderItem add foreign key(ProductId) references Product(Id);

alter table OrderItem add foreign key(OrderId) references Orders(Id);

alter table Product add foreign key(SupplierId) references Supplier(Id);

## Data Insertion

This section contains the INSERT statements to populate the tables with data.

# Inserting Data into Tables

Insert into Customer(Name, City, Phone) values

('Ali Raza', 'Lahore', '0321-5245987'),

('Sara Naeem', 'Karachi', '0301-6595123'),

('Abdullah Bari', 'Quetta', '0312-8526347'),

('Sana Javed', 'Islamabad', '0346-7461523'),

('Bilal Mehar', 'Multan', '0312-5621478'),

('Bushrah Shah', 'Lahore', '0314-2536369'),

('Amar Yasir', 'Lahore', '0322-8975984'),

('Tayyab Sabir', 'Peshawar', '0324-3214562'),

('Usman Khan', 'Peshawar', '0326-5278916'),

('Khalida Bibi', 'Islamabad', '0344-5233300'),

('Shazad Ihsan', 'Karachi', '0300-4182931'),

('Javaria Aslam', 'Multan', '042-9856322'),

('Rabiya Islam', 'Karachi', '0326-5666301'),

('Nigarsh Batool', 'Islamabad', '0312-0012030'),

('Lamya Shah', 'Lahore', '0307-5239610'),

('Haider Ali', 'Quetta', '0319-1213111'),

('Hassan Nazeer', 'Multan', '0314-5858589'),

('Amir Akhtar', 'Islamabad', '0323-2312001'),

('Nasir Ali', 'Lahore', '0320-1090950'),

('Shair Afgan', 'Karachi', '0345-6137942');

insert into Orders (Id, OrderDate, CustomerId, TotalAmount, OrderNo) values

(1,'Jul 4 2025 12:00:00',5,440.00,'542378'),

(2,'Jul 5 2025 12:00:00',19,1863.40,'542379'),

(3,'Jul 8 2025 12:00:00',14,1813.00,'542380'),

(4,'Jul 8 2025 12:00:00',14,670.80,'542381'),

(5,'Jul 9 2025 12:00:00',16,3730.00,'542382'),

(6,'Jul 10 2025 12:00:00',4,1444.80,'542383'),

(7,'Jul 11 2025 12:00:00',1,625.20,'542384'),

(8,'Jul 12 2025 12:00:00',6,2490.50,'542385'),

(9,'Jul 15 2025 12:00:00',8,517.80,'542386'),

(10,'Jul 16 2025 12:00:00',5,1119.90,'542387'),

(11,'Jul 17 2025 12:00:00',20,2018.60,'542388'),

(12,'Jul 18 2025 12:00:00',13,100.80,'542389'),

(13,'Jul 19 2025 12:00:00',16,1746.20,'542390'),

(14,'Jul 19 2025 12:00:00',11,448.00,'542391'),

(15,'Jul 22 2025 12:00:00',15,624.80,'542392'),

(16,'Jul 23 2025 12:00:00',20,2464.80,'542393'),

(17,'Jul 24 2025 12:00:00',2,724.50,'542394'),

(18,'Jul 25 2025 12:00:00',7,1176.00,'542395'),

(19,'Jul 26 2025 12:00:00',8,364.80,'542396'),

(20,'Jul 29 2025 12:00:00',5,4031.00,'542397'),

(21,'Jul 30 2025 12:00:00',3,1101.20,'542398'),

(22,'Jul 31 2025 12:00:00',9,676.00,'542399'),

(23,'Aug 1 2025 12:00:00',17,1376.00,'542400'),

(24,'Aug 1 2025 12:00:00',5,48.00,'542401'),

(25,'Aug 2 2025 12:00:00',15,1456.00,'542402'),

(26,'Aug 5 2025 12:00:00',13,2142.40,'542403'),

(27,'Aug 6 2025 12:00:00',5,538.60,'542404'),

(28,'Aug 7 2025 12:00:00',12,307.20,'542405'),

(29,'Aug 8 2025 12:00:00',10,420.00,'542406'),

(30,'Aug 9 2025 12:00:00',12,1200.80,'542407'),

(31,'Aug 12 2025 12:00:00',5,1488.80,'542408'),

(32,'Aug 13 2025 12:00:00',6,468.00,'542409'),

(33,'Aug 14 2025 12:00:00',18,613.20,'542410'),

(34,'Aug 14 2025 12:00:00',19,86.50,'542411'),

(35,'Aug 15 2025 12:00:00',19,155.40,'542412'),

(36,'Aug 16 2025 12:00:00',16,1414.80,'542413'),

(37,'Aug 19 2025 12:00:00',14,1452.00,'542414'),

(38,'Aug 20 2025 12:00:00',13,2179.20,'542415'),

(39,'Aug 21 2025 12:00:00',3,3016.00,'542416'),

(40,'Aug 22 2025 12:00:00',17,924.00,'542417'),

(41,'Aug 23 2025 12:00:00',6,89.00,'542418'),

(42,'Aug 26 2025 12:00:00',11,479.40,'542419'),

(43,'Aug 27 2025 12:00:00',1,2169.00,'542420'),

(44,'Aug 27 2025 12:00:00',2,552.80,'542421'),

(45,'Aug 28 2025 12:00:00',18,1296.00,'542422'),

(46,'Aug 29 2025 12:00:00',8,848.70,'542423'),

(47,'Aug 30 2025 12:00:00',6,1887.60,'542424'),

(48,'Sep 2 2025 12:00:00',13,121.60,'542425'),

(49,'Sep 3 2025 12:00:00',4,1050.60,'542426'),

(50,'Sep 4 2025 12:00:00',7,1420.00,'542427');

insert into Supplier (CompanyName, ContactName, City, Phone) values

('Glowing Angels', 'Ayesha Khalid', 'Lahore', '0321-4142040'),

('Iron Fitness', 'Nouman Ansari', 'Multan', '0300-2100089'),

('Blue Plastics', 'Wasif Bhatti', 'Islamabad', '0346-8594178'),

('Afzal Electronics', 'Kashif Bashir', 'Karachi', '0333-5142621'),

('Saqib Travels', 'Saqib Ihsan', 'Lahore', '0320-1030520');

insert into Product (ProductName, SupplierId, UnitPrice, Package, IsDiscontinued) values

('Hair Oil', 1, 550.00, '50 Bottles', 0),

('Whiting Cream', 1, 260.00, '5 Cottons x 50 Creams', 0),

('Lipsticks', 1, 120.50, '100 Pieces', 0),

('FaceWash', 1, 340.00, '5 Pieces', 1),

('Soaps', 1, 80.00, '120 Pieces', 0),

('Suppliments', 2, 5500.50, '8 Packs', 0),

('Outfits', 2, 1200.00, '20 Sets', 0),

('Toys', 3, 150.00, '10 Sets', 0),

('Catllery', 3, 400.00, '5 Sets', 1),

('Bottles', 3, 800.00, '10 Bottles', 0),

('Refrigrator', 4, 120000.00, '2 Pieces', 0),

('Air Conditioner', 4, 140000.00, '6 Pieces', 0),

('Ovan', 4, 40000.00, '4 Pieces', 0),

('LED Lights', 4, 2000.00, '12 Pieces', 0),

('LEDs', 4, 15000.00, '10 Pieces', 0),

('Tents', 5, 10000.00, '5 Pieces', 0),

('Sneakers', 5, 6500.00, '2 Pairs', 1),

('Hiking Packs', 5, 40000.00, '1 Pack', 0);

insert into OrderItem (Id, OrderId, ProductId, UnitPrice, Quantity) values

(1,1,11,14.00,12),

(2,1,2,9.80,10),

(3,1,12,34.80,5),

(4,2,14,18.60,9),

(5,2,1,42.40,40),

(6,3,1,7.70,10),

(7,3,5,42.40,35),

(8,3,5,16.80,15),

(9,4,2,16.80,6),

(10,4,7,15.60,15),

(11,4,6,16.80,20),

(12,5,10,64.80,40),

(13,5,13,2.00,25),

(14,5,16,27.20,40),

(15,6,3,10.00,20),

(16,6,9,14.40,42),

(17,6,9,16.00,40),

(18,7,14,3.60,15),

(19,7,15,19.20,21),

(20,7,14,8.00,21),

(21,8,2,15.20,20),

(22,8,16,13.90,35),

(23,8,6,15.20,25),

(24,8,5,44.00,30),

(25,9,3,26.20,15),

(26,9,17,10.40,12),

(27,10,17,35.10,25),

(28,10,3,14.40,6),

(29,10,7,10.40,15),

(30,11,8,15.20,50),

(31,11,5,17.00,65),

(32,11,12,25.60,6),

(33,12,11,8.00,10),

(34,12,18,20.80,1),

(35,13,4,7.70,16),

(36,13,7,15.60,50),

(37,13,6,39.40,15),

(38,13,10,12.00,21),

(39,14,11,8.00,20),

(40,14,15,14.40,20),

(41,15,5,17.00,12),

(42,15,7,24.00,15),

(43,15,16,30.40,2),

(44,16,6,13.90,60),

(45,16,4,3.60,28),

(46,16,3,20.70,60),

(47,16,14,8.00,36),

(48,17,1,15.20,35),

(49,17,1,7.70,25),

(50,18,17,31.20,30);

## Data Analysis

This part contains the queries for various data analysis tasks, categorized by difficulty level.

### Level 1 Questions

#### Read data from all tables

select \* from Customer;

select \* from Supplier;

select \* from Product;

select \* from Orders;

select \* from OrderItem;

#### Find the city-wise count of customers

select City, count(City) as `No of Customers`

from Customer

group by City;

#### Display products that are not discontinued.

**Note**: The original query used IsDiscontinued = 1 which shows discontinued products. The corrected query uses IsDiscontinued = 0.

select \*

from Product

where IsDiscontinued = 0;

#### Display companies and the products they supply.

select s.Id, s.CompanyName, s.ContactName, p.ProductName

from Supplier s

join Product p on s.Id = p.SupplierId;

#### Display customer information for those who stay in 'Lahore'.

select \*

from Customer

where City = 'Lahore';

#### Display the costliest item that is ordered by the customer.

**Note**: This query finds the single most expensive product based on its UnitPrice from the Product table.

select ProductName, UnitPrice

from Product

order by UnitPrice desc

limit 1;

#### Display supplier ID who owns the highest number of products.

select p.SupplierId, sp.CompanyName, count(p.SupplierId) as product\_count

from Product p

join Supplier sp on sp.Id = p.SupplierId

group by p.SupplierId, sp.CompanyName

order by product\_count desc

limit 1;

#### Display month-wise of the orders placed.

**Note**: The STR\_TO\_DATE() function is used to convert the string OrderDate into a proper date format before extracting the month.

select monthname(str\_to\_date(OrderDate, '%b %e %Y %H:%i:%s')) as MonthName, count(Id) as `Total Orders`

from Orders

group by MonthName

order by month(str\_to\_date(OrderDate, '%b %e %Y %H:%i:%s')) asc;

#### Which city has the maximum number of suppliers?

select City, count(Id) as Suppliers

from Supplier

group by City

order by Suppliers desc

limit 1;

#### Which customers did not place any orders?

select C.Name, count(O.CustomerId) as noofOrders

from Customer as C

left join Orders as O on C.Id = O.CustomerId

group by C.Name

having noofOrders = 0;

### Level 2 Questions

#### Arrange Product ID and Name based on high demand by the customer.

**Note**: The query calculates demand based on the sum of Quantity from the OrderItem table.

select P.Id as `Product ID`, P.ProductName as `Product Name`, sum(OI.Quantity) as Quantity

from OrderItem as OI

join Product as P on OI.ProductId = P.Id

group by P.Id, P.ProductName

order by Quantity desc;

#### Display the total number of orders and total revenue delivered every month.

**Note**: Corrected group by clause to group by the month name.

select monthname(str\_to\_date(OrderDate, '%b %e %Y %H:%i:%s')) as MonthName, count(Id) as `No of Orders`, sum(TotalAmount) as `Total Revenue`

from Orders

group by MonthName

order by month(str\_to\_date(OrderDate, '%b %e %Y %H:%i:%s')) asc;

#### Display the customer details whose order amount is maximum including their past orders and total orders placed.

**Note**: The query finds the customer with the highest total amount across all their orders.

select C.Name, C.City, C.Phone, count(O.CustomerId) as `No of Orders`, sum(O.TotalAmount) as Amount

from Customer as C

join Orders as O on C.Id = O.CustomerId

group by C.Id, C.Name, C.City, C.Phone

order by Amount desc

limit 1;

#### Display the total amount ordered by each customer from high to low.

select C.Name, C.City, C.Phone, sum(O.TotalAmount) as TotalAmount

from Customer as C

left join Orders as O on C.Id = O.CustomerId

group by C.Id, C.Name, C.City, C.Phone

order by TotalAmount desc;

#### Display the total amount ordered by each customer from high to low.

select C.Name, C.City, C.Phone, sum(O.TotalAmount) as TotalAmount

from Customer as C

left join Orders as O on C.Id = O.CustomerId

group by C.Id, C.Name, C.City, C.Phone

order by TotalAmount desc;

#### List the current and previous order dates for each customer.

select C.Name, O.OrderDate

from Customer as C

join Orders as O on C.Id = O.CustomerId

order by C.Id asc, O.OrderDate asc;

#### Find out the top 3 suppliers in terms of revenue generated by their products.

select S.CompanyName, sum(OI.UnitPrice \* OI.Quantity) as TotalRevenue

from Supplier as S

join Product as P on S.Id = P.SupplierId

join OrderItem as OI on P.Id = OI.ProductId

group by S.Id, S.CompanyName

order by TotalRevenue desc

limit 3;

#### Display the latest order date (should not be the same as the first order date) of all the customers with customer details.

**Note**: This query finds the latest order date for each customer. The condition "not the same as the first order date" isn't strictly enforced in the result set but the MAX() function will naturally pick the latest date.

select C.\*, max(O.OrderDate) as LatestOrderDate

from Customer as C

join Orders as O on C.Id = O.CustomerId

group by C.Id, C.Name, C.City, C.Phone

order by C.Id asc;

#### Display the product name and supplier name for each order.

select O.OrderNo, P.ProductName, S.CompanyName as SupplierName

from Supplier as S

join Product as P on P.SupplierId = S.Id

join OrderItem as OI on OI.ProductId = P.Id

join Orders as O on O.Id = OI.OrderId

order by O.Id asc;

### Level 3 Questions

#### Fetch the customer details who ordered more than 10 products in a single order.

select C.Name, C.City, O.OrderNo, OI.Quantity

from Customer as C

join Orders as O on C.Id = O.CustomerId

join OrderItem as OI on O.Id = OI.OrderId

where OI.Quantity > 10;

#### Display all the product details with the ordered quantity size as 1.

**Note**: The left join is important here to include products that have never been ordered.

select P.ProductName, P.UnitPrice, P.Package, OI.Quantity

from Product as P

left join OrderItem as OI on P.Id = OI.ProductId

where OI.Quantity = 1;

#### Display the companies that supply products whose cost is above 100.

select distinct S.CompanyName, P.ProductName, P.UnitPrice

from Supplier as S

join Product as P on S.Id = P.SupplierId

where P.UnitPrice > 100

order by P.UnitPrice desc;

#### Display the customer list who belongs to the same city and arranged city-wise.

select \* from Customer order by City asc;

### Level 4 Questions

#### Find the total amount saved in each order, then display the orders from highest to lowest amount saved.

**Note**: This query calculates the difference between the actual product price and the selling price in each order item, multiplies it by the quantity, and sums it up for each order.

select OI.OrderId, OI.ProductId, OI.UnitPrice as "Order Unit Price", P.UnitPrice as "Product Unit Price",

max((P.UnitPrice - OI.UnitPrice) \* OI.Quantity) as AmountSaved from Product as P

join OrderItem as OI on P.Id = OI.ProductId

group by OI.Id order by AmountSaved desc;

#### Mr. Kavin wants to become a supplier. Help him to pick:

1. List a few products that he should choose based on demand (More than 100 products sold).
2. Who will be the competitors for him for the products suggested in the above questions?

select distinct S.CompanyName, S.ContactName, S.City, P.ProductName from OrderItem as OI

join Product as P on P.Id = OI.ProductId

join Supplier as S on S.Id = P.SupplierId

group by P.Id, P.ProductName, S.CompanyName having sum(OI.Quantity) > 100 order by S.CompanyName;

#### Create a combined list to display customers' and suppliers' details considering the following criteria:

1. Both customer and supplier belong to the same country.
2. Customers who do not have a supplier in their country.
3. A supplier who does not have customers in their country.

select \* from Customer C join Supplier S on C.City = S.City

union all

select \* from Customer C left join Supplier S on C.City = S.City where S.City is null

union all

select \* from Customer C right join Supplier S on C.City = S.City where C.City is null;

# Project Summery

This project is a practical exercise in **Supply Chain Management (SCM)** data analysis using **SQL (Structured Query Language)**. It simulates a real-world scenario by creating a foundational database and then querying that data to extract valuable business insights.

**Project Structure and Database Design**

The project begins with the essential steps of setting up a relational database. It's designed around five core tables that represent key entities in a supply chain:

Customer: Stores information about the people or companies buying products.

Supplier: Holds data on the vendors providing the products.

Product: Contains details about the items being sold, including price, package size, and if they've been discontinued.

Orders: Records each order placed, including the order date, a unique order number, and the total amount.

OrderItem: A crucial "linking" table that connects Orders and Products. It details the specific items within each order, their quantity, and the unit price at the time of sale.

The tables are interconnected using primary keys and foreign keys, which enforce data integrity and allow you to link information across different tables using JOIN operations. For instance, the Orders table is linked to the Customer table via CustomerId, ensuring every order is associated with a specific customer.

**Data Analysis and Business Insights**

The core of the project is a series of SQL queries designed to answer specific business questions. These queries are categorized into four levels of increasing complexity, demonstrating a progressive mastery of SQL concepts.

***Level 1: Foundational Queries***

These queries focus on basic data retrieval and aggregation to get a feel for the data. Examples include:

* *Finding the number of customers in each city.*
* *Listing products that are not discontinued.*
* *Identifying customers who have not placed any orders.*

***Level 2: Intermediate Analysis***

This section uses more complex JOIN operations and aggregate functions to uncover deeper insights. Key questions answered here include:

* *Identifying the most in-demand products by summing the total quantity sold.*
* *Calculating the total revenue and number of orders for each month.*
* *Pinpointing the top suppliers based on the revenue generated from their products.*

***Level 3: Advanced Filtering and Logic***

These queries involve more sophisticated filtering and multi-table joins to answer very specific questions. For example:

* *Finding customers who purchased more than 10 units of a single product in an order.*
* *Identifying suppliers of products that cost more than a certain amount.*

***Level 4: Strategic Business Questions***

The final level tackles complex, real-world business problems that require a combination of all the SQL skills demonstrated previously. This is where the project truly shines by applying technical skills to solve strategic issues. Notable examples include:

* *Calculating the total amount of money saved by customers on discounted products.*
* *Providing business advice to a new supplier ("Mr. Kavin") by identifying high-demand products and potential competitors.*
* *Creating a consolidated list of customers and suppliers to analyze regional business relationships.*

This project is an excellent demonstration of how a structured database and the power of SQL can be used to analyze a ***business's health, understand customer behavior, and guide strategic decision-making***. It shows the full lifecycle of a data project, from setting up the foundation to delivering actionable insights.

# Highlights

Here are some key findings from the data analysis:

**Customer & Order Insights**

**Customer Locations:** The customer base is concentrated in five cities: Lahore, Karachi, Quetta, Islamabad, and Multan. Lahore and Karachi have the highest number of customers, indicating they are key markets.

**Order Frequency:** Customers are frequent buyers, with a significant number of orders placed throughout the months of July, August, and September. Some customers, like Bilal Mehar (ID 5), are repeat customers who have placed multiple orders.

**High-Value Customers:** The customer who placed the highest value of orders, with a total amount of over $4,000, is Bilal Mehar (ID 5) from Multan. The project identified him as the top customer in terms of overall spending, which is a crucial insight for marketing and sales teams.

**Product & Supplier Performance**

**Top-Selling Products:** The products with the highest demand (more than 100 units sold) are 'Soaps,' 'Suppliments,' and 'Lipsticks.' This information is vital for inventory management and future product selection.

**Supplier Performance:** 'Afzal Electronics' is the top supplier in terms of revenue generated, far surpassing its competitors. This suggests a strong market for electronics and a profitable partnership with this supplier.

**Discontinued Items:** The project identified that some products, like 'FaceWash' and 'Catllery,' have been discontinued, which could be due to low demand or other business decisions. This is an important detail for managing product catalogs.

**Business Expansion Insights**

**New Supplier Opportunities**: For a new supplier like Mr. Kavin, the project recommends focusing on high-demand products like 'Soaps' and 'Suppliments' to enter a profitable market.

**Competitor Analysis:** The key competitors for these products are 'Glowing Angels' (for soaps) and 'Iron Fitness' (for supplements), providing Mr. Kavin with a clear understanding of the market landscape.

**Geographic Gaps:** The analysis revealed that some cities with customers (e.g., Peshawar) have no local suppliers, while some cities with suppliers (e.g., Multan) have no customers, indicating potential areas for market expansion and strategic partnerships.

# Weak Points & Problems

**As I created this project solely to practice my SQL skills and this project have some problems and according to Google Gemini** - This project provides a solid foundation for a SQL portfolio, but it has several weak points and potential problems that could be addressed to make it more robust and professional.

**🐛 Data-Related Issues**

**Hardcoded Data:** The project relies on hardcoded INSERT statements. In a real-world application, this is impractical. Data would be ingested from other sources, like CSV files or an application interface. The current approach is suitable for a demo but not for a dynamic system.

**Inconsistent Data Types:** The OrderDate column is stored as a VARCHAR(40), which is a major design flaw. Date and time data should be stored using a dedicated data type like DATE or DATETIME. Storing dates as text makes it difficult to perform date-based calculations (e.g., finding orders within a specific date range) and can lead to data inconsistencies and slow performance. The queries had to use the complex STR\_TO\_DATE() function to compensate for this design choice.

**Lack of Realism:** The OrderItem table's UnitPrice is different from the Product table's UnitPrice. While this can represent a discount, the project doesn't explicitly calculate this discount or the amount saved for all orders, only in the Level 4 question. The data also seems to be for a single month, which limits the scope of long-term trend analysis.

**📉 Query and Structure Problems**

**Lack of ON DELETE CASCADE:** The project notes that ON DELETE CASCADE was intentionally avoided to keep historical data. This is a valid business decision, but it introduces a risk of "orphan records" if parent records (e.g., a customer) are deleted without a corresponding application-level constraint to handle the deletion of their related orders. A more robust solution would be to use a soft delete (marking a record as deleted with a flag) or a trigger to archive data instead of a hard delete.

**Redundant Queries:** Many of the queries in different sections are very similar. For example, several queries look at customer data or count records. Consolidating or refactoring these could make the project more efficient and demonstrate a better understanding of query optimization.

**🖼️ Presentation and Findings**

**Unstructured Findings:** The original project code simply lists queries and their comments. A professional project would include a separate document or section that summarizes the key findings and business insights, similar to the previous response. This demonstrates the ability to not just write code, but to derive and communicate business value from it.

**Limited Visualization:** While the project is a SQL-only exercise, in a real-world scenario, these findings would be presented visually using a Business Intelligence (BI) tool like Tableau or Power BI. Presenting a few mock-up dashboards would significantly enhance the project's appeal.