What is Data Modeling?

Data modeling helps us organize data in a way that makes it easy to use, understand, and analyze.

In simple terms:

- **Data** is like raw information (e.g., names, addresses, sales numbers).
- **Modeling** is the process of structuring this data so it's useful.

Why Do We Need Data Modeling?

Without data modeling:

- Data can be messy and hard to work with.
- It becomes difficult to find the information you need.
- Reports and analyses might not make sense.

With data modeling:

- Data is organized logically.
- It's easier to build systems like databases, reports, or dashboards.
- Everyone using the data understands how it's structured.

How Does Data Modeling Work?

Think of data modeling as arranging things into tables (like Excel sheets) and connecting them properly. Here's a simple breakdown:

1. **Identify the Data**

- Figure out what kind of information you're dealing with.
- Example:
  - *Customer data*: Name, age, address.
  - *Sales data*: Product name, price, quantity sold.

2. **Organize the Data**
   - Group related data together.
   - Example:
     - A "*Customer*" table with columns like Name, Age, Address.
     - A "*Sales*" table with columns like Product, Price, Quantity.
3. **Connect the Data**
   - Link related tables.
   - Example:
     - Each customer can have multiple sales records, so you connect the "*Customer*" table to the "*Sales*" table using a unique ID (like Customer ID).
4. **Define Rules**
   - Set rules for how the data should behave.
   - Example:
     - Every customer must have a unique ID.
     - Prices in the "*Sales*" table cannot be negative.

Types of Data Models

There are three main types of data models, depending on how detailed they are:

1. **Conceptual Model**
   - The big picture.
   - What data do we need? Who will use it?
   - Example: "*We need customer and sales data.*"

2. **Logical Model**
   - More detailed.
   - Defines the structure of the data (tables, columns, relationships).
   - Example: "*The 'Customer' table has Name, Age, Address. The 'Sales' table has Product, Price, Quantity.*"
3. **Physical Model**
   - The actual implementation.
   - How the data is stored in a database.
   - Example: *Writing SQL queries to create tables and relationships.*

Tools Used for Data Modeling

There are tools that help create data models visually. Some popular ones are:

- **Microsoft Excel**: For simple data organization.
- **Power BI/Tableau**: For creating visual models and dashboards.
- **SQL**: For creating and managing databases.
- **ER Diagram Tools**: For drawing diagrams of how tables are connected.

Snowflake and Star Schema

Snowflake and Star Schema are two popular approaches to organizing data in a database, especially for analytics and reporting purposes. Both are types of **dimensional modeling**, which is a method used to structure data so it's easy to query and analyze.

What is Dimensional Modeling?

Dimensional modeling is a way to organize data into two main types of tables:

- **Fact Tables**: These store the "measures" or "metrics" (numbers you want to analyze, like sales amount, quantity sold, etc.).
- **Dimension Tables**: These store descriptive information about the data (like customer name, product details, date, etc.).

The goal is to make it easy to answer business questions like:

- "*How many products were sold last month?*"
- "*What was the total revenue from a specific region?*"

What is Star Schema?

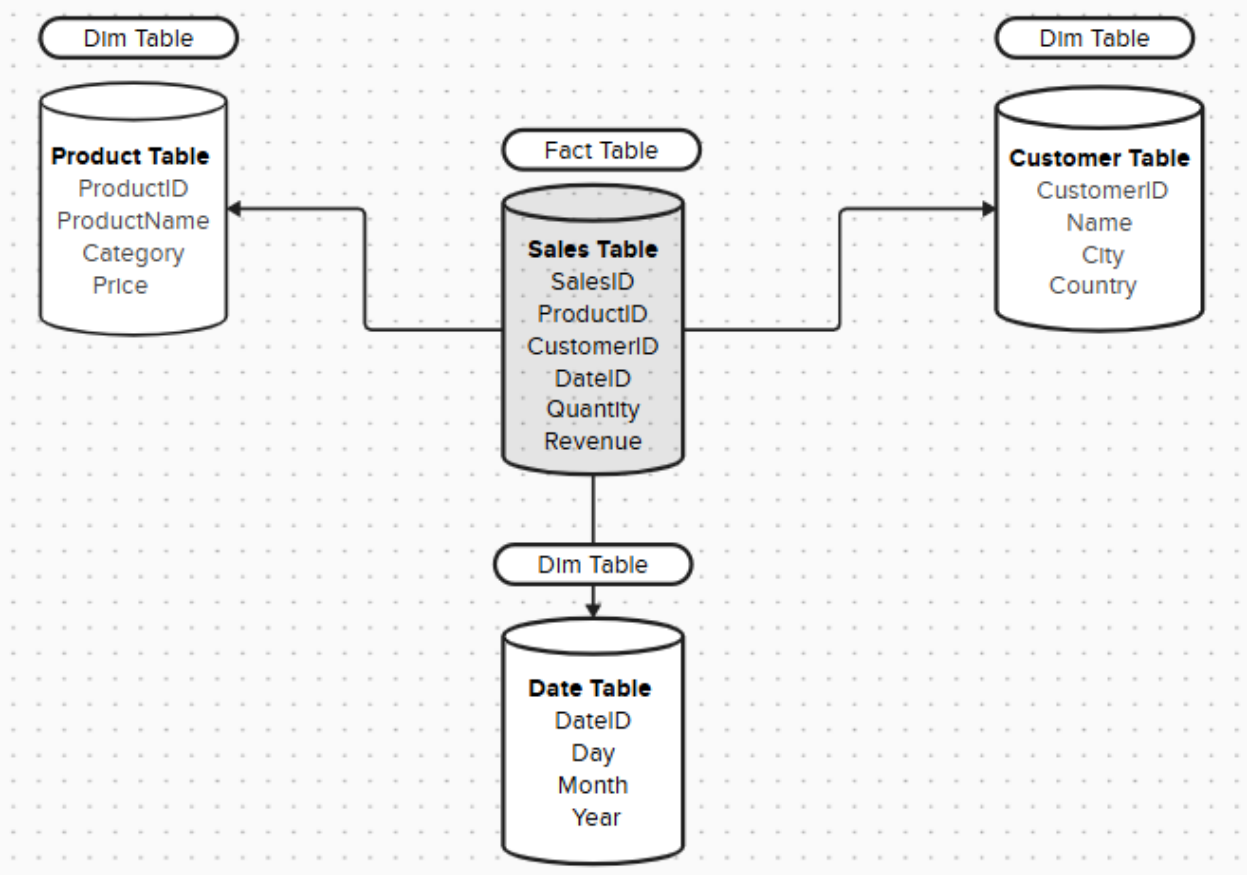
Structure:

- One central **Fact Table** surrounded by multiple **Dimension Tables**.
- The relationships between the fact table and dimension tables are direct (no intermediate tables).

Example:
Imagine a retail company that wants to track sales data.

- **Fact Table**: *Sales*
  - Columns: SalesID, ProductID, CustomerID, DateID, Quantity, Revenue.
- **Dimension Tables**:
  - *Product*: Product details like ProductID, ProductName, Category, Price.
  - *Customer*: Customer details like CustomerID, Name, City, Country.
  - *Date*: Date details like DateID, Day, Month, Year.

Diagram:



Key Features:

- Simple and easy to understand.
- Fast for querying because there are fewer joins.
- Best suited for small to medium-sized datasets.

What is Snowflake Schema?

Structure:

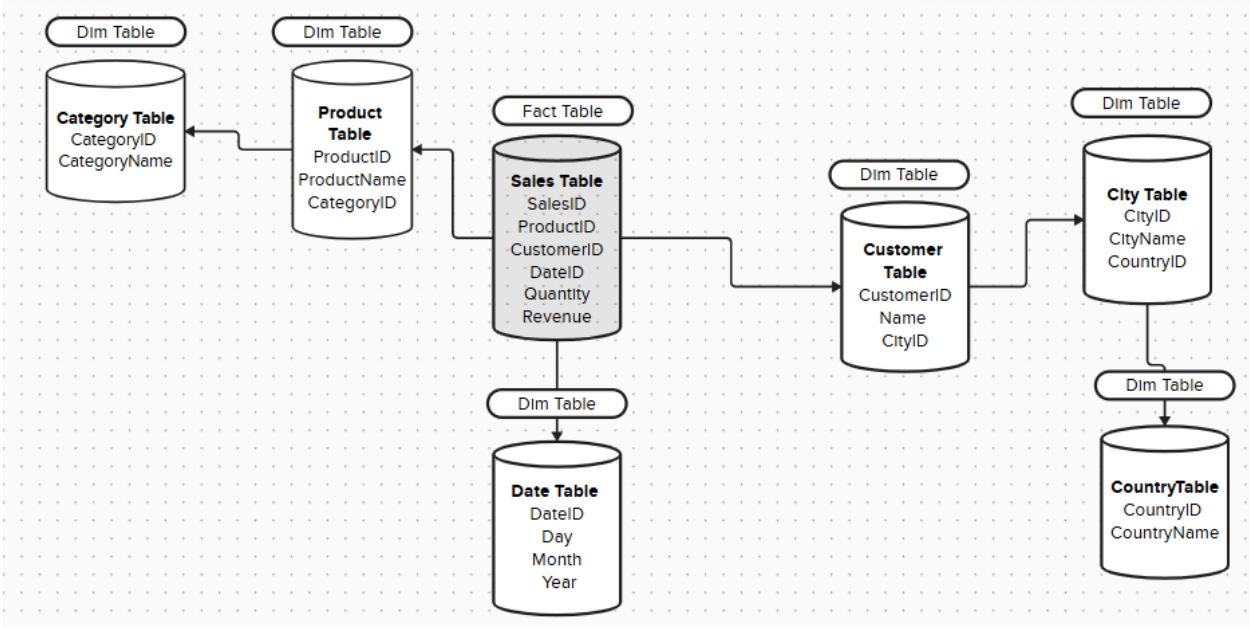
- The dimension tables are **normalized** (broken down further into sub-dimensions).
- Some dimension tables are connected to other dimension tables.

Example:

Using the same retail company example, but now with normalized dimensions:

- **Fact Table**: *Sales*
  - Columns: SalesID, ProductID, CustomerID, DateID, Quantity, Revenue.
- **Dimension Tables**:
  - *Product* → Connected to a *Category* table.
    - Product: ProductID, ProductName, CategoryID.
    - Category: CategoryID, CategoryName.
  - *Customer* → Connected to a *City* table and *Country* table.
    - Customer: CustomerID, Name, CityID.
    - City: CityID, CityName, CountryID.
    - Country: CountryID, CountryName.
  - *Date*: DateID, Day, Month, Year.

Diagram:



Key Features:

- More complex than Star Schema due to additional normalization.
- Reduces data redundancy (e.g., instead of storing full city and country names repeatedly, they're stored in separate tables).
- Slower for querying because there are more joins.
- Best suited for large datasets where storage efficiency is important.

Comparison: Star Schema vs. Snowflake Schema

FEATURE	STAR SCHEMA	SNOWFLAKE SCHEMA
Complexity	Simpler and easier to understand.	More complex due to normalization.
Query Performance	Faster queries (fewer joins).	Slower queries (more joins).
Data Redundancy	Higher redundancy (data is repeated).	Lower redundancy (data is normalized).
Storage Efficiency	Less efficient (repeated data).	More efficient (normalized data).
Use Case	Small to medium-sized datasets.	Large datasets with high redundancy.

When to Use Each Schema?

- **Star Schema**:
  - When simplicity and query performance are priorities.
  - For smaller datasets where storage efficiency isn't a concern.
  - Ideal for business intelligence tools like Power BI, Tableau, etc.
- **Snowflake Schema**:
  - When dealing with very large datasets where storage efficiency matters.
  - When you need to reduce data redundancy and maintain strict normalization.
  - Commonly used in data warehouses for enterprise-level systems.

Summary

- **Star Schema**: Simple, fast, and great for small to medium datasets. Fact table is directly connected to dimension tables.
- **Snowflake Schema**: Normalized, efficient for large datasets, but slower for queries due to more joins.
- Both are widely used in data warehousing and analytics. The choice depends on your dataset size, performance needs, and storage constraints.

PROJECT ON DATA MODELING USING SQL

We'll follow the process of **Conceptual → Logical → Physical modeling** and then implement both **Star Schema** and **Snowflake Schema** using SQL.

Step 1: Conceptual Model

The conceptual model is the high-level understanding of the data and its relationships. It defines:

- What **entities (tables)** are involved?
- What **attributes (columns)** describe each entity?
- How the entities are **related** to each other?

Example Scenario:

Let's use a **Retail Sales System** as our example. The system tracks sales data for products sold to customers.

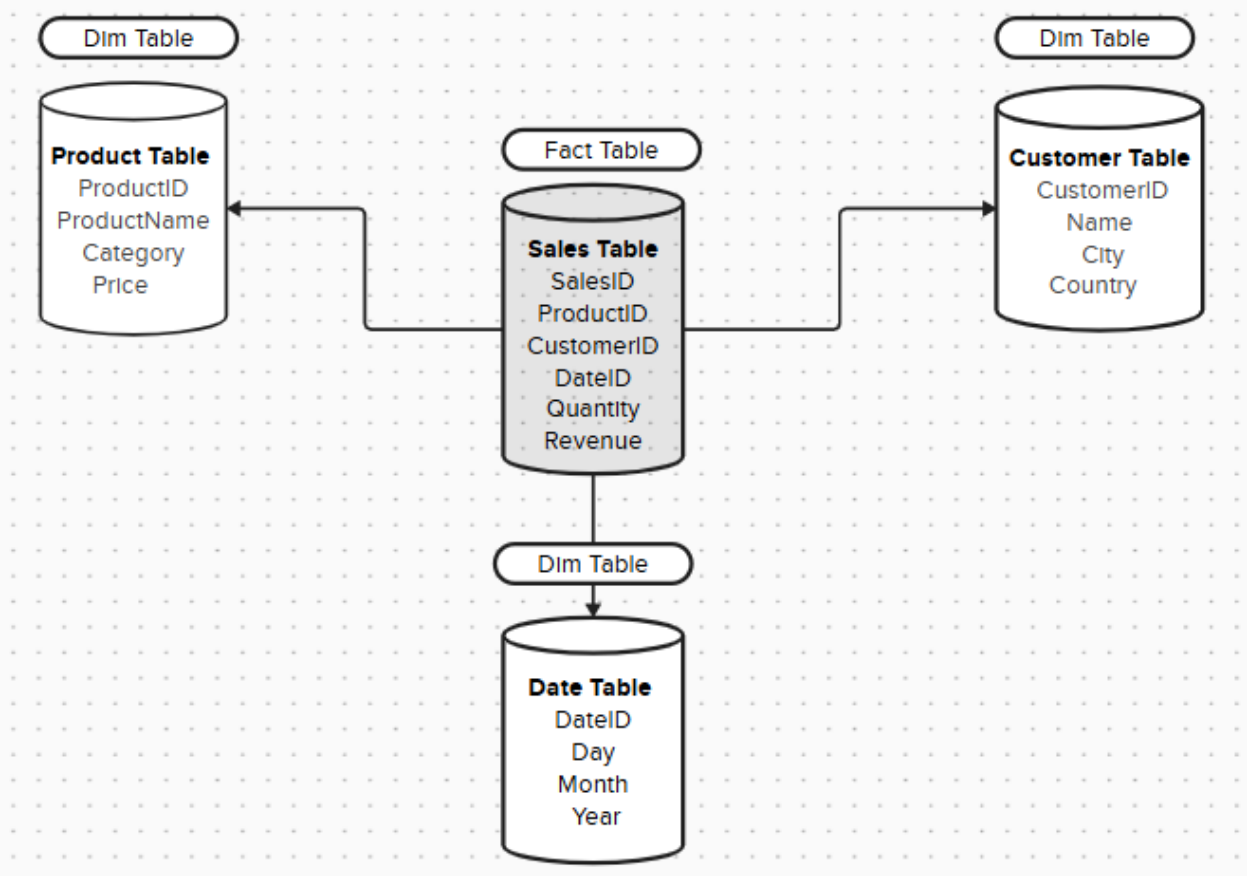
Entities (Tables):

- **Sales**: Tracks sales transactions.
- **Product**: Stores product details.
- **Customer**: Stores customer details.
- **Date**: Stores date-related information.

Relationships:

- A sale involves **one Product**, **one Customer**, and **one Date**.
- Each product belongs to a **category**.
- Each customer lives in a **city**, which belongs to a **country**.

Diagram:

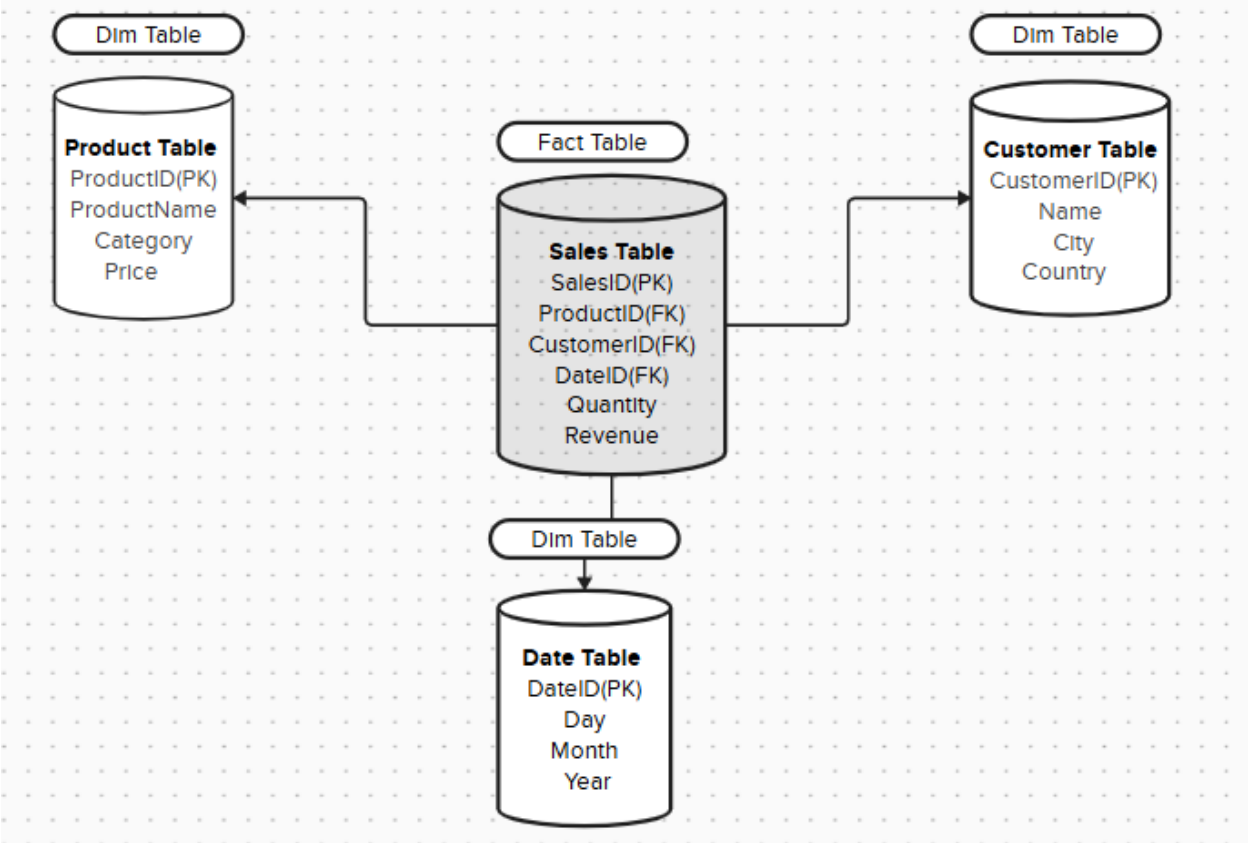


Step 2: Logical Model

The logical model adds more detail to the conceptual model. It specifies:

- The **structure of tables** (columns and data types).
- **Primary keys** (unique identifiers for each table).
- **Foreign keys** (relationships between tables).

Diagram:



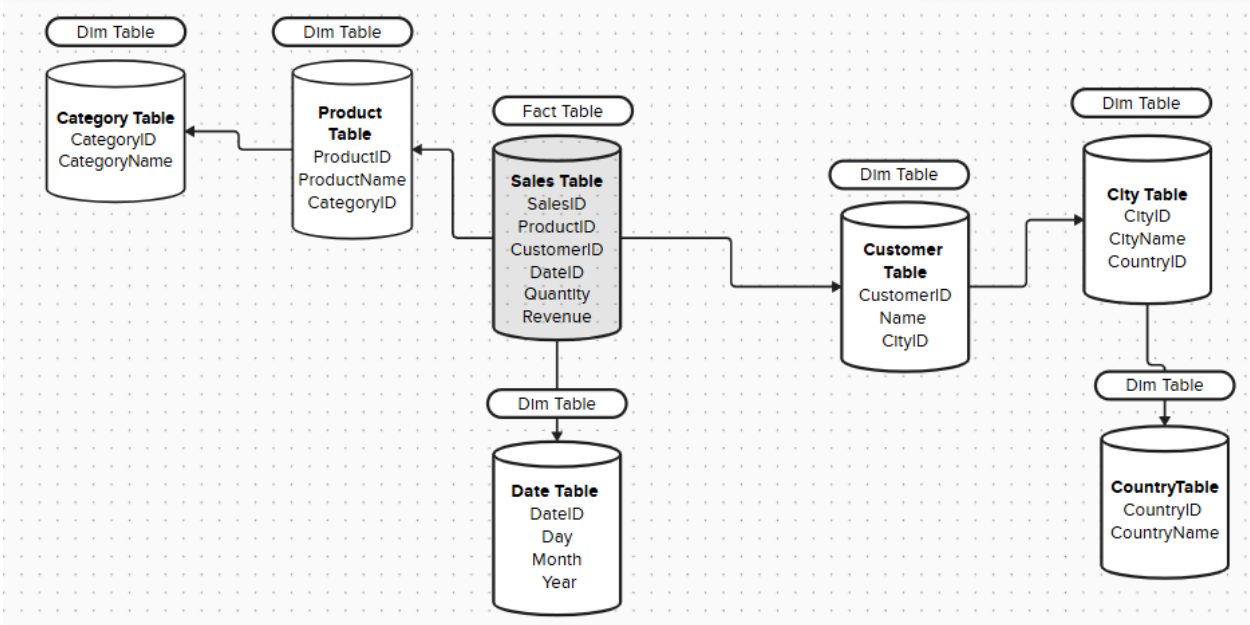
Star Schema Logical Model:

- **Fact Table**: **Sales** (contains measures like quantity and revenue).
- **Dimension Tables**: **Product**, **Customer**, **Date**.

Snowflake Schema Logical Model:

- Same as the Star Schema, but dimension tables are **normalized**:
  - **Product** is linked to **Category**.
  - **Customer** is linked to **City** and **Country**.

Diagram:



Step 3: Physical Model

The physical model is the actual implementation in SQL. It includes:

- Creating tables with **proper data types**.
- Defining **primary and foreign keys**.
- Inserting **sample records** for testing.

Step 3: Physical Model

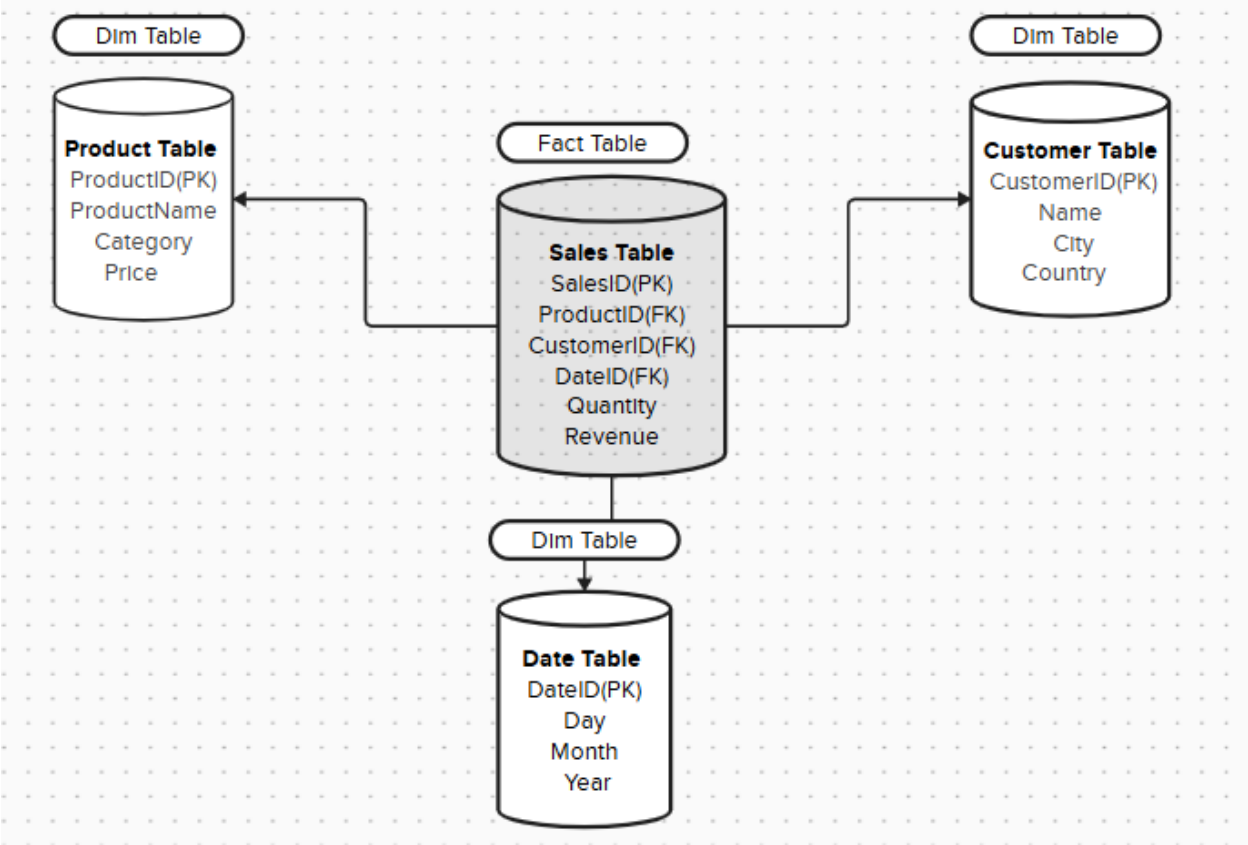
The physical model is the actual implementation in SQL. It includes:

- Creating tables with **proper data types**.

- Defining **primary and foreign keys**.
- Inserting **sample records** for testing.

SQL Implementation

Star Schema



```sql
-- Create Dimension Tables for Star Schema
CREATE TABLE Product (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100),
    Category VARCHAR(50),
    Price DECIMAL(10, 2)
);

CREATE TABLE Customer (
    CustomerID INT PRIMARY KEY,
    Name VARCHAR(100),
    City VARCHAR(50),
    Country VARCHAR(50)
);

CREATE TABLE Date (
    DateID INT PRIMARY KEY,
    Day INT,
    Month INT,
    Year INT
);
```

```sql
-- Create Fact Table for Star schema
CREATE TABLE Sales (
    SalesID INT PRIMARY KEY,
    ProductID INT,
    CustomerID INT,
    DateID INT,
    Quantity INT,
    Revenue DECIMAL(10, 2),
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID),
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
    FOREIGN KEY (DateID) REFERENCES Date(DateID)
);
```

```sql
-- Insert Sample Data into Dimension Tables for star schema
INSERT INTO Product (ProductID, ProductName, Category, Price)
VALUES (1, 'Laptop', 'Electronics', 1000.00),
       (2, 'Shirt', 'Apparel', 50.00);

INSERT INTO Customer (CustomerID, Name, City, Country)
VALUES (1, 'John Doe', 'Mumbai', 'India'),
       (2, 'Jane Smith', 'New York', 'USA');

INSERT INTO Date (DateID, Day, Month, Year)
VALUES (1, 1, 1, 2023),
       (2, 15, 1, 2023);

-- Insert Sample Data into Fact Table
INSERT INTO Sales (SalesID, ProductID, CustomerID, DateID, Quantity, Revenue)
VALUES (1, 1, 1, 1, 1, 1000.00),
       (2, 2, 2, 2, 2, 100.00);
```

Product Table (Dim)

	ProductID	ProductName	Category	Price
1	1	Laptop	Electronics	1000.00
2	2	Shirt	Apparel	50.00

Customer Table (Dim)

	CustomerID	Name	City	Country
1	1	John Doe	Mumbai	India
2	2	Jane Smith	New York	USA

Date Table (Dim)

	DateID	Day	Month	Year
1	1	1	1	2023
2	2	15	1	2023

Sales Table (Fact)

	SalesID	ProductID	CustomerID	DateID	Quantity	Revenue
1	1	1	1	1	1	1000.00
2	2	2	2	2	2	100.00

Testing the Data Models

To test the models, you can run queries to analyze the data. For example:

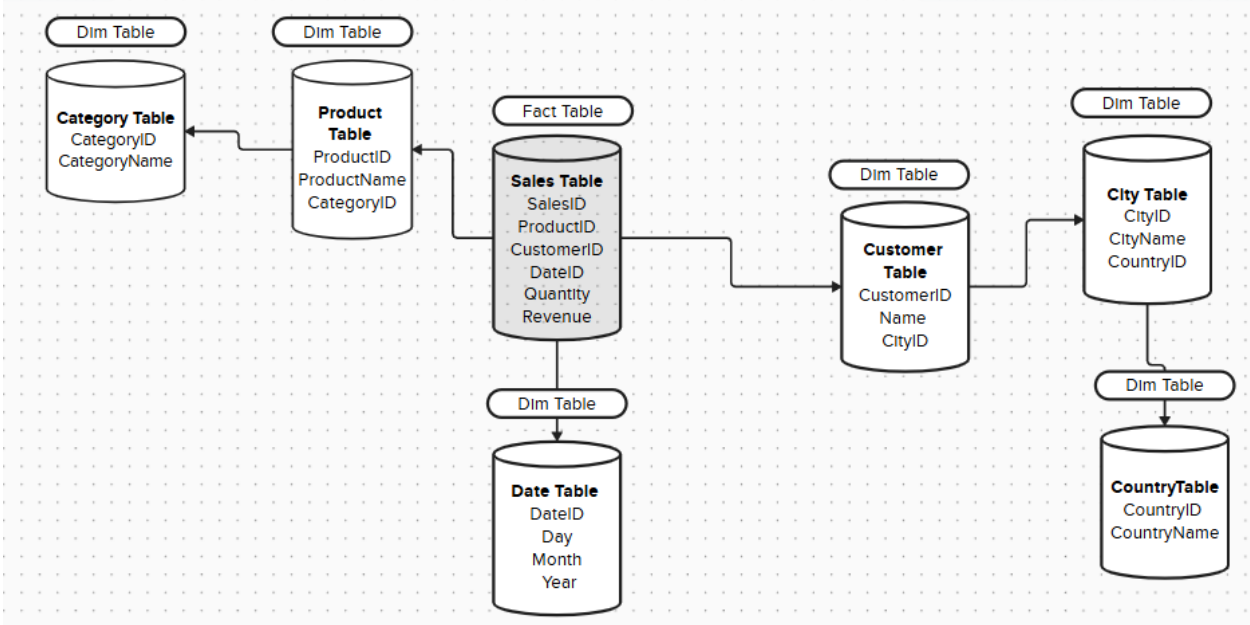
Query in Star Schema:

```
--Query in Star schema
SELECT
    s.SalesID,
    p.ProductName,
    c.Name AS CustomerName,
    d.Day,
    d.Month,
    d.Year,
    s.Quantity,
    s.Revenue
FROM
    Sales s
JOIN Product p ON s.ProductID = p.ProductID
JOIN Customer c ON s.CustomerID = c.CustomerID
JOIN Date d ON s.DateID = d.DateID;
```

Results Messages

	SalesID	ProductName	CustomerName	Day	Month	Year	Quantity	Revenue
	1	Laptop	John Doe	1	1	2023	1	1000.00
2	2	Shirt	Jane Smith	15	1	2023	2	100.00

Snowflake Schema




```sql
-- Create Normalized Dimension Tables in Snowflake schema
CREATE TABLE Category (
    CategoryID INT PRIMARY KEY,
    CategoryName VARCHAR(50)
);

CREATE TABLE Product (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100),
    CategoryID INT,
    Price DECIMAL(10, 2),
    FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID)
);

CREATE TABLE Country (
    CountryID INT PRIMARY KEY,
    CountryName VARCHAR(50)
);

CREATE TABLE City (
    CityID INT PRIMARY KEY,
    CityName VARCHAR(50),
    CountryID INT,
    FOREIGN KEY (CountryID) REFERENCES Country(CountryID)
);
```

```sql
CREATE TABLE Customer (
    CustomerID INT PRIMARY KEY,
    Name VARCHAR(100),
    CityID INT,
    FOREIGN KEY (CityID) REFERENCES City(CityID)
);

CREATE TABLE Date (
    DateID INT PRIMARY KEY,
    Day INT,
    Month INT,
    Year INT
);
```

```sql
-- Create Fact Table in Snowflake Schema
CREATE TABLE Sales (
    SalesID INT PRIMARY KEY,
    ProductID INT,
    CustomerID INT,
    DateID INT,
    Quantity INT,
    Revenue DECIMAL(10, 2),
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID),
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
    FOREIGN KEY (DateID) REFERENCES Date(DateID)
);
```

Category Table (Dim)

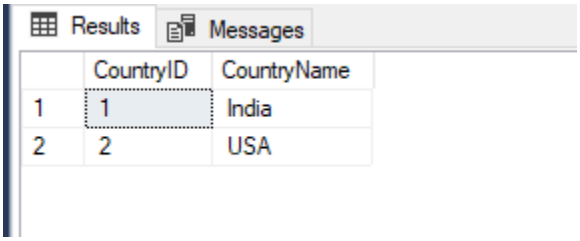
Results | Messages

	CategoryID	CategoryName
1	1	Electronics
2	2	Apparel

Product Table (Dim)

	ProductID	ProductName	CategoryID	Price
1	1	Laptop	1	1000.00
2	2	Shirt	2	50.00

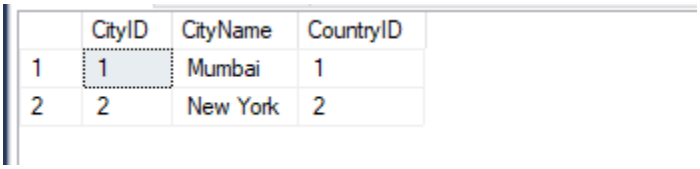
Country Table (Dim)



Results | Messages

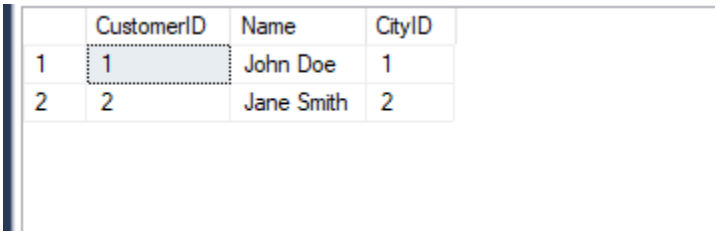
	CountryID	CountryName
1	1	India
2	2	USA

City Table (Dim)



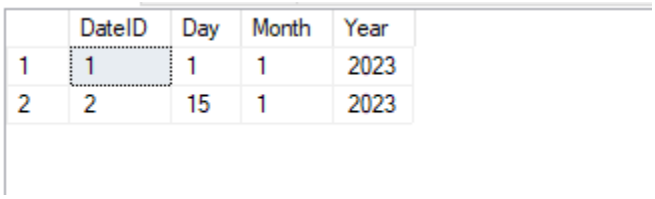
	CityID	CityName	CountryID
1	1	Mumbai	1
2	2	New York	2

Customer Table (Dim)



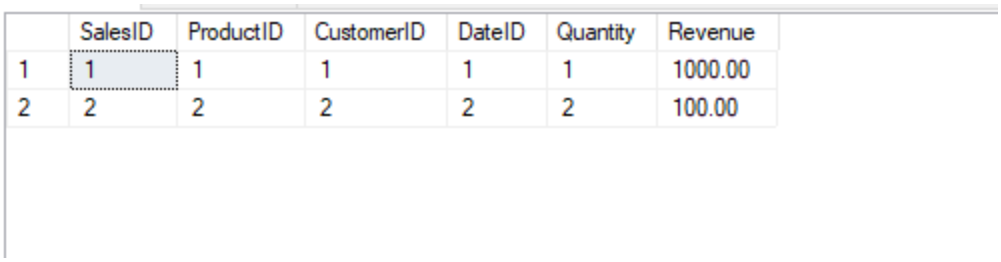
	CustomerID	Name	CityID
1	1	John Doe	1
2	2	Jane Smith	2

Date Table (Dim)



	DateID	Day	Month	Year
1	1	1	1	2023
2	2	15	1	2023

Sales Table (Fact)



	SalesID	ProductID	CustomerID	DateID	Quantity	Revenue
1	1	1	1	1	1	1000.00
2	2	2	2	2	2	100.00

Testing the Data Models

To test the models, you can run queries to analyze the data. For example:

Query in Snowflake Schema:

```sql
SELECT
    s.SalesID,
    p.ProductName,
    cat.CategoryName,
    c.Name AS CustomerName,
    ci.CityName,
    co.CountryName,
    d.Day,
    d.Month,
    d.Year,
    s.Quantity,
    s.Revenue
FROM
    Sales s
JOIN Product p ON s.ProductID = p.ProductID
JOIN Category cat ON p.CategoryID = cat.CategoryID
JOIN Customer c ON s.CustomerID = c.CustomerID
JOIN City ci ON c.CityID = ci.CityID
JOIN Country co ON ci.CountryID = co.CountryID
JOIN Date d ON s.DateID = d.DateID;
```

00 %

Results | Messages

	SalesID	ProductName	CategoryName	CustomerName	CityName	CountryName	Day	Month	Year	Quantity	Revenue
1	1	Laptop	Electronics	John Doe	Mumbai	India	1	1	2023	1	1000.00
2	2	Shirt	Apparel	Jane Smith	New York	USA	15	1	2023	2	100.00

Key Takeaways

- Star Schema is simpler and faster for querying because it has fewer joins.
- Snowflake Schema reduces redundancy and is more storage-efficient but requires more joins.
- Both schemas are useful depending on the size of your dataset and performance requirements.