Lesson 5

Topic: Data Modeling Basics
Prerequisites: Download customer.csv, product.csv, sales.csv

1. **What is a primary key in a table?**

   A primary key is a column (or a set of columns) in a table that uniquely identifies each row in that table.
   It must have unique values and cannot be NULL.
   It helps ensure data integrity and is used to create relationships between tables.

2. **Name the two types of table relationships in Power BI.**

   Two Types of Table Relationships in Power BI

   In Power BI, tables can be connected through relationships that help model your data correctly. The two main types of relationships are:

---

1. One-to-Many (1:*) Relationship

- Most common type of relationship.
- One row in Table A relates to many rows in Table B.

  Example:

- One customer (CustomerID) in the Customer table → can have many orders in the Sales table.

Customer (1) -> Sales (*)

---

2. Many-to-Many (:) Relationship

- Used when both tables can have multiple matching rows on both sides.
- Usually requires a bridge table or composite model to manage properly.

Example:

If a product can belong to multiple categories and categories can have multiple products, you'd have a many-to-many relationship.

3. **How do you create a relationship between two tables in Power BI?**

    1. Load Your Tables
       First, import your data files (customer.csv, product.csv, sales.csv) into Power BI Desktop using the Get Data option.
    2. Go to the Model View

- On the left sidebar, click the Model icon (it looks like three connected tables).
- This view shows all tables and their existing relationships visually.

    3. Create a Relationship

- Find the common column (key) between the two tables you want to relate. For example, customer.csv and sales.csv might share CustomerID, or product.csv and sales.csv share ProductID.
- Click and drag the key column from one table and drop it onto the matching column in the other table.

    4. Set Relationship Properties
       After you drag-and-drop:

- Power BI will open a Create Relationship dialog box.
- Confirm the columns selected are the ones you want to relate.
- Choose the Cardinality (usually Many to One (*:1) when connecting fact tables like sales to dimension tables like customers).
- Choose the Cross filter direction (usually Single or Both, depending on your analysis needs).
- Click OK to create the relationship.

    5. Verify the Relationship

- You should now see a line connecting the two tables in Model view, representing the relationship.
- This relationship enables Power BI to aggregate and filter data correctly across these tables.

4. **What is a "star schema"?**

   Structure of a Star Schema:

   - Central Fact Table:
     This is the main table containing measurable, quantitative data (facts), such as sales transactions, order quantities, or revenue. It usually contains foreign keys referencing dimension tables.
   - Surrounding Dimension Tables:
     These tables describe the context or attributes related to the facts, such as customers, products, time, or regions. They provide descriptive details and are connected directly to the fact table.

   Why use a Star Schema?

   - Simplifies Queries:
     Because everything connects directly to the fact table, it’s easy to write queries and understand relationships.
   - Optimized for Reporting:
     Star schema is designed to improve query performance in BI tools like Power BI, making aggregations and filtering faster.
   - Easy to Understand:
     The clear separation of facts and dimensions makes the data model intuitive.

   Example using your data:

   - Fact Table:
     Sales — with sales transactions (OrderID, CustomerID, ProductID, Quantity, OrderDate)
   - Dimension Tables:
     - Customer (CustomerID, Name, Region)
     - Products (ProductID, ProductName, Category, Price)

     In this setup, the fact table Sales sits at the center, and the dimension tables Customer and Products are linked to it, forming a star-like shape.

5. **Which table is typically the fact table in a sales dataset?**

   In a sales dataset, the table that is typically the fact table is the Sales table (or sometimes called the Transactions or Orders table).

   Why?

   - The fact table contains the measurable, quantitative data about business processes, such as sales amounts, quantities, or revenue.
   - In the sales dataset, the Sales table records individual transactions with details like OrderID, CustomerID, ProductID, Quantity, and OrderDate.
   - It usually has foreign keys linking to dimension tables (e.g., Customers, Products) and contains the numeric facts used for analysis.

6. **Link Sales.csv to Customers.csv using CustomerID (one-to-many).**

   1. Load both tables (Sales.csv and Customers.csv) into Power BI via Get Data > CSV.
   2. Go to Model view by clicking the Model icon on the left sidebar.
   3. Create the relationship:

   - Find the CustomerID column in the Sales table.
   - Drag the CustomerID field from Sales and drop it onto the CustomerID column in the Customers table.

   4. Set the relationship details:

   - The Cardinality should be set to Many to One (*:1) — because many sales records relate to one customer.
   - The Cross filter direction usually stays Single (from Customers to Sales) unless you have a specific reason to set it to Both.

   5. Click OK to create the relationship.
   6. Verify that a line appears between Sales and Customers tables in Model view, showing the one-to-many connection.

7. **Why is ProductID in Sales.csv a foreign key?**

   Because it references the ProductID in the Products table to identify which product was sold.

   Explanation:

- A foreign key is a column in one table that points to a primary key in another table.
- In this case:
  - In the Products table, ProductID is the primary key — each product has a unique ID.
  - In the Sales table, ProductID appears multiple times — it refers to the product being sold in each transaction.

8. **Fix a relationship error where ProductID has mismatched data types.**

   Step 1: Go to Power Query Editor

1. In Power BI Desktop, click Transform Data to open Power Query Editor.

   Step 2: Check Data Types

1. In the Sales table, find the ProductID column.
2. Look at the icon next to the column name:
   - ABC = text
   - 123 = number
3. Do the same for the ProductID column in the Products table.

   Step 3: Change Data Types (if needed)

1. If one of the columns is not a number, click the data type icon next to the column name.
2. Select Whole Number (or Text in both if needed, but usually numbers are better for IDs).
3. Click Close & Apply (top-left) to save changes and return to Power BI.

   Step 4: Recreate the Relationship

Now go to Model view, delete the old relationship (if it's still broken), and:

- Drag ProductID from Sales
- Drop onto ProductID in Products
- Confirm Many-to-One relationship

9. **Explain why a star schema improves performance.**

1. Simpler Joins

- In a star schema, all dimension tables connect directly to the fact table.
- Power BI performs fewer joins to retrieve data, which makes queries faster and more efficient.

---

2. Optimized for Columnar Storage

- Power BI uses the VertiPaq engine, which compresses columns very well.
- In a star schema, dimension tables are small and descriptive, and fact tables are large and numeric, which is ideal for compression and fast retrieval.

---

3. Faster Aggregations

- The fact table contains pre-cleaned, numeric data (e.g., Quantity, Price).
- Aggregations like SUM, COUNT, or AVERAGE are performed efficiently without complex joins or filters.

---

4. Improved Filter & Slicer Performance

- Slicers and filters apply to dimension tables, which are smaller and more optimized.

- These filters quickly flow into the fact table for slicing data, speeding up dashboard interactions.

---

5. Better Query Plan & DAX Optimization

- DAX and Power BI's query engine are designed to work best with star schema.
- Query folding and indexing are more effective in this clean structure.

10. Add a new column TotalSales in Sales (Quantity * Price from Products).

1. In Power BI Desktop, go to the **Data view**.
2. Select the **Sales** table in the Fields pane.
3. Click **New column** (on the toolbar).
4. Enter this DAX formula:

DAX
TotalSales = Sales[Quantity] * RELATED(Products[Price])

11. Optimize a model with circular relationships—how would you resolve it?

1. Remove One of the Relationships

- Identify the relationship causing the loop.
- Delete the least important or redundant relationship from Model view.

2. Use DAX Instead of a Physical Relationship

- Instead of creating a direct relationship, use DAX functions like:
  - LOOKUPVALUE()
  - RELATED()
  - CALCULATE() with FILTER()

Example:

DAX

CustomerRegion = LOOKUPVALUE(Customers[Region], Customers[CustomerID], Sales[CustomerID])

This pulls the region info from Customers without needing a direct relationship.

3. Use Bridge Tables (Role-playing Dimensions)

- If two tables need to link to the same table in different ways, create a bridge table.

Example:

- Dates table used for OrderDate and ShipDate — duplicate the Dates table to use both.

4. Use One-to-Many Relationships Only Where Logical

- Ensure that your relationships follow a star schema — facts in the center, dimensions surrounding.

12.Create a role-playing dimension for OrderDate and ShipDate.

Step 1: Create a Date Table

If you don't already have one, create a Date table:

1. Go to Modeling > New table and enter this DAX:

   DAX

   Date = CALENDAR(DATE(2022, 1, 1), DATE(2024, 12, 31))

2. Add additional columns (optional):

   DAX

   Year = YEAR(Date[Date])

   Month = FORMAT(Date[Date], "MMMM")

   MonthNumber = MONTH(Date[Date])

Step 2: Create Two Copies for Role-Playing

Power BI doesn't allow two active relationships from the same table to one table. So:

1. Duplicate the Date table:

- In Power Query: right-click the Date query > Duplicate > rename it as ShipDateTable.

2. Now you have:
   - Date for OrderDate
   - ShipDateTable for ShipDate

Step 3: Create Relationships

- Go to Model view.
- Link Sales[OrderDate] → Date[Date] (active)
- Link Sales[ShipDate] → ShipDateTable[Date] (active)

13. Handle a many-to-many relationship between Customers and Products.

1. Go to Model View

Create the following relationships:

- Sales[CustomerID] → Customers[CustomerID] (Many-to-One)
- Sales[ProductID] → Products[ProductID] (Many-to-One)

This creates an indirect many-to-many connection between Customers and Products through Sales.

14. Use bidirectional filtering sparingly—when is it appropriate?

In Power BI, relationships between tables usually filter in one direction (e.g., from a dimension table like Customers to a fact table like Sales).

Bidirectional filtering allows filters to flow in both directions between tables.

Why Use It Sparingly?

- It increases complexity in your data model
- It may create ambiguity or circular relationships

- It can slow down performance significantly

When Is It Appropriate?

Use bidirectional filtering only when it’s necessary to make specific reports or visuals work correctly.

Appropriate Scenarios:

1. Many-to-Many Relationships
    - When you're using a bridge table that connects two tables with no clear direction
    - Example: Salesperson ↔ Customer ↔ Sales
2. Complex Filtering Requirements
    - When you need filters from a fact table to flow up to a dimension table (rare but possible)
    - Example: A report where filtering by product category should also limit the customer list based on sales.
3. Role-Based Security
    - When defining row-level security, sometimes bi-directional filtering is required to ensure correct user-based filters flow.

15. Write DAX to enforce referential integrity if a CustomerID is deleted.

DAX Solution: Use RELATED() with IF(ISBLANK())

You can create a calculated column in the Sales table to detect if a CustomerID no longer exists in the Customers table:

DAX

```
CustomerExists =
IF(
    ISBLANK(RELATED(Customers[Name])),
    "Missing Customer",
    "Valid Customer"
```

)

Explanation:

- RELATED(Customers[Name]): Tries to bring in the customer's name from the Customers table.
- ISBLANK(...): Checks if that customer exists.
- The formula returns:
  - "Valid Customer" if the ID exists in the Customers table
  - "Missing Customer" if not — i.e., referential integrity is broken

---

🛠️ Use Case:

You can use this column in:

- Reports to highlight missing customers
- Filters to exclude broken records
- Data validation steps before sharing dashboards

🚀 Optional: Filter Out Invalid Data Automatically

You could also create a measure to count only valid sales:

DAX

```
ValidSales =
CALCULATE(
    SUM(Sales[Quantity]),
    NOT ISBLANK(RELATED(Customers[Name]))
)
```