1. What is a primary key in a table?

- A **primary key** is a column (or set of columns) that **uniquely identifies each row** in a table.
- Example: In `Customers.csv`, `CustomerID` is the primary key (no duplicates, no nulls).

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

1. **One-to-Many (1:*):** Most common. Example: One Customer can have many Sales.
2. **Many-to-Many (*:*):** Less common, used when both tables can have multiple matches.

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

- Go to **Model View** → drag `CustomerID` from **Sales** to `CustomerID` in **Customers**.
- Choose relationship type (`1:*` or `*:*`), and cross-filter direction (single or both).

◈ 4. What is a "star schema"?

- A **star schema** is a model design with:
  - **Fact table** in the center (transactions: Sales).
  - **Dimension tables** around it (lookup info: Customers, Products, Dates).
- Looks like a shape → improves performance and readability.

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

- **Sales.csv** → because it stores **transactions** (Quantity, ProductID, CustomerID, Date).

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

- In **Sales**, `CustomerID` repeats (many sales per customer).
- In **Customers**, `CustomerID` is unique.
  Relationship: `Customers[CustomerID] (1)` → `Sales[CustomerID] (*)`.

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

- Because it **references** the primary key `ProductID` in `Products.csv`.
- Sales table does not own the Product info; it just points to it.

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

- Example: `Products[ProductID]` is **text**, but `Sales[ProductID]` is **number**.
- Solution: In Power Query, change data types so both match (`Whole Number` or `Text`).

◈ 9. Explain why a star schema improves performance.

- Star schema avoids **many-to-many joins**.
- Queries become simpler (one fact table, multiple dimensions).
- Power BI can use **columnar storage + compression** efficiently.

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

In **DAX**:

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

`RELATED()` pulls in the Product Price from the dimension table.

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

- Circular relationships = loops between tables → **bad** for performance.
  Solutions:
- Remove one relationship and use **DAX LOOKUP** instead.
- Use **bridge tables** (intermediate tables).
- Keep schema in **star shape** (no loops).

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

- Duplicate the `Date` table twice: `OrderDate`, `ShipDate`.
- Link:
  - `Sales[OrderDate] → OrderDate[Date]`
  - `Sales[ShipDate] → ShipDate[Date]`
- Both use the same underlying calendar, but serve different roles.

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

- Example: Customers buy many Products, Products are bought by many Customers.
  Solution:

- Introduce a **bridge table** (Sales) with `CustomerID` and `ProductID`.
- Then relationships are *1: from Customers → Sales** and *1: from Products → Sales**.

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

- Bidirectional filtering (both directions) should be used only when:
  - Reports need **filter propagation both ways** (e.g., filtering Customers also filters Products through Sales).
- Use carefully → can cause ambiguity and performance slowdown.

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

If a Customer is missing in Customers but exists in Sales:

```
ValidCustomerSales =
CALCULATE(
    SUM(Sales[Quantity]),
    FILTER(
        Sales,
        NOT ISBLANK(RELATED(Customers[CustomerID]))
    )
)
```