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

A **primary key** is a column (or combination of columns) that uniquely identifies each row in a table.
✔ **Example:** CustomerID in Customers table.

✅ 2. Name two types of table relationships in Power BI.

1. **One-to-Many (1:*)** – e.g. each customer has many sales.
2. **Many-to-Many (*:*)** – e.g. customers and products if each can link to multiple entries in both tables (rare; requires special handling).

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

Steps:

1. Go to **Model View**.
2. Drag the key column from one table to the matching column in the other table.
3. Ensure **data types match** and *cardinality (1: or :)** is correct.

✅ 4. What is a "star schema"?

A **star schema** is a data model with:

- A central **fact table** (e.g. Sales) containing measurable data.
- Connected **dimension tables** (e.g. Products, Customers, Dates) providing descriptive attributes.

✔ Structure: Fact table at the center with dimension tables radiating like a star.

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

✔ **Fact Table: Sales** – it stores transactions (quantities, amounts, dates) linked to dimensions like Products and Customers.

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

Steps:

1. Import both tables.

2. Ensure **CustomerID data type** matches in both.
3. Go to **Model View**, drag **CustomerID** from Customers to Sales.
4. Confirm **Cardinality: One-to-Many (1:*).**

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

Because **ProductID** in Sales references the **Products table**, defining which product was sold.
✔ It **connects** Sales data to its corresponding product details.

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

Steps:

1. Go to **Data View**.
2. Check **ProductID column** in both tables.
3. Convert them to the **same data type** (e.g. Text or Whole Number).
4. Recreate the relationship.

✅ 9. Explain why a star schema improves performance.

✔ **Reasons:**

- Simplifies relationships (avoids complex joins).
- Minimizes **filter propagation issues**.
- Enhances **query efficiency** as Power BI's VertiPaq engine optimizes star schemas well.

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

Steps:

1. Create a relationship between **Sales[ProductID]** and **Products[ProductID]**.
2. Create a **calculated column** in Sales:

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

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

✔ **Solutions:**

- Remove or rethink relationships causing loops.
- Create a **bridge table** if needed.
- Avoid bidirectional filtering on multiple paths.

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

Steps:

1. Import or create **Date table**.
2. Create two **copies** of the Date table:
    - **Order Date Dimension**
    - **Ship Date Dimension**
3. Link **Sales[OrderDate]** to Order Date table, and **Sales[ShipDate]** to Ship Date table.

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

✔ **Steps:**

1. Create a **bridge table** listing CustomerID + ProductID combinations.
2. Link Customers → Bridge (CustomerID) and Products → Bridge (ProductID).
3. Use the bridge for analysis of cross-purchases or commonalities.

✅ 14. Use bidirectional filtering sparingly – when is it appropriate?

When you **need filters to flow both ways** to maintain correct context, e.g.:

- Many-to-many relationships.
- Complex calculations requiring dimension tables to filter each other.

✔ **Caution:** Bidirectional filtering can lead to **performance issues and ambiguous relationships** if overused.

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

Example measure to check missing CustomerIDs in Sales:

DAX
КопироватьРедактировать

```
MissingCustomers =
CALCULATE(
    COUNTROWS(Sales),
    ISBLANK(RELATED(Customers[CustomerID]))
)
```