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. It must contain **unique** and **non-null** values.

---

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

- **One-to-many (1:*)**
- **Many-to-many (*:*)**

---

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

- Go to **Model view**
- Drag the **key column** from one table to the matching column in another
- Or use **Manage Relationships → New**

---

4. What is a "star schema"?

A **star schema** has:

- A **central fact table** (e.g., Sales)
- Surrounded by **dimension tables** (e.g., Products, Customers, Date)
  It's simple, fast, and ideal for analytics.

---

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

Usually, the **Sales** table is the fact table—it records transactional data.

---

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

- CustomerID in `Customers` is **primary key**
- CustomerID in `Sales` is **foreign key**
- Relationship: **Customers (1) → Sales (*)**

---

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

Because it links each sales record to a **specific product** in the Products table—it refers to a key from another table.

---

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

- Go to **Power Query** → ensure both `ProductID` columns are the **same data type** (e.g., Text or Whole Number)
- Then recreate the relationship

---

9. Why does a star schema improve performance?

- Simplifies joins
- Reduces redundant data
- Supports **efficient columnar storage**
- Works well with **VertiPaq** compression engine

---

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

Use **DAX**:

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

---

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

- **Avoid bidirectional filters** unless necessary
- Use **bridge tables** or **DAX** instead of creating direct loops
- Split roles (e.g., one Date table per role)

---

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

- Duplicate the Date table → rename as `OrderDateTable`, `ShipDateTable`
- Create separate relationships:
  - Sales[OrderDate] → OrderDateTable[Date]
  - Sales[ShipDate] → ShipDateTable[Date]
- Use DAX to control which one is active

---

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

- Create a **bridge table** (e.g., Purchases or CustomerProduct) with both IDs
- Connect **Customers → Bridge → Products** using one-to-many links

---

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

- When using **many-to-many** relationships
- When you need filters to **flow in both directions** for specific calculations (e.g., slicer impacts both tables)

⚠️ Can impact performance or cause ambiguity—use cautiously.

---

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

```
ValidSales =
CALCULATE (
    COUNTROWS(Sales),
    NOT(ISBLANK(RELATED(Customers[CustomerID])))
)
```

Or, in a calculated column:

```
IsValidCustomer = NOT(ISBLANK(RELATED(Customers[CustomerID])))
```

Use it to **flag or filter out orphaned rows**.