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 contains **unique, non-null values**.
- In Power BI, while there's no formal *primary key* constraint like in SQL, it's used conceptually for relationships.

---

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

- **One-to-many (1:*):** The most common, e.g., one customer → many sales transactions.
- **Many-to-many (*:*):** Used when both tables have non-unique keys.

---

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

- Go to **Model view**.
- Drag a field from one table (key) to the corresponding field in another table.
- Set **Cardinality** (1:* or *:*) and **Cross filter direction** (single or both).
- Click **OK** to confirm.

---

4. What is a "star schema"?
A **star schema** is a data model design where:

- **Fact table** (contains numeric measures like sales, quantities) is in the center.
- **Dimension tables** (customers, products, dates) surround it, connected by keys.
  It looks like a star in diagram view.

---

5. Which table is typically the fact table in a sales dataset?
Sales — because it contains transactional data like Quantity, Amount, Date, ProductID, CustomerID.

---

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

- Ensure CustomerID in Customers is **unique** → "1" side.
- CustomerID in Sales is **repeated** → "*" side.
- In **Model view**, drag CustomerID from Customers → CustomerID in Sales, set cardinality **1:* (single direction)**.

---

7. Why is ProductID in Sales.csv a foreign key?
Because it **refers** to a ProductID in the **Products** table to get product details (name, price). It's not unique in Sales.

---

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

- In **Power Query**, select ProductID in both tables.
- Transform → **Data Type** → choose the same type (e.g., Whole Number).
- Apply changes and re-create the relationship.

---

9. Explain why a star schema improves performance.

- Reduces **data duplication** and storage size.
- Simplifies relationships → avoids complex joins.
- Improves **DAX performance** because each dimension is filtered directly into the fact table without chaining.

---

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

DAX

CopyEdit

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

---

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

- Remove one of the conflicting relationships.
- Use **inactive relationships** with USERELATIONSHIP() in DAX when needed.
- Break into a **star schema** to avoid loops.

---

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

- Duplicate the **Date** table twice (OrderDateTable, ShipDateTable).
- Link **OrderDateTable[Date]** → Sales[OrderDate] and **ShipDateTable[Date]** → Sales[ShipDate].
- This allows filtering by each date independently.

---

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

- Create a **bridge table** with CustomerID + ProductID (all combinations that exist).
- Link Customers → Bridge (1:*), Products → Bridge (1:*).
- The bridge resolves *:*.

---

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

- When you need **both tables to filter each other**, e.g., for certain calculations in many-to-many scenarios.
- Avoid using it in large models—it can cause slow performance and incorrect results.

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

DAX

CopyEdit

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

This counts only sales with matching Customers, ignoring “orphan” rows.