Scenario: Let's consider a retail business that wants to analyze sales data across different dimensions like time, product, and location.

1. Star Schema Example

The **Star Schema** is a simple database design in a data warehouse, where a central fact table is directly connected to several dimension tables.

- Fact Table: Sales
  - Columns: `Sales_ID`, `Product_ID`, `Date_ID`, `Store_ID`, `Quantity_Sold`, `Total_Sales_Amount`
- Dimension Tables:
  - Product Dimension:
    - `Product_ID`, `Product_Name`, `Category`, `Brand`
  - Date Dimension:
    - `Date_ID`, `Date`, `Month`, `Quarter`, `Year`
  - Store Dimension:
    - `Store_ID`, `Store_Name`, `City`, `State`, `Country`

Sample Data:

- Fact Table (Sales):

Sales_ID	Product_ID	Date_ID	Store_ID	Quantity_Sold	Total_Sales_Amount
1	101	20230101	1	5	$500
2	102	20230102	2	3	$300

- Product Dimension:

Product_ID	Product_Name	Category	Brand
101	Laptop	Electronics	Dell
102	Smartphone	Electronics	Apple

- Date Dimension:

Date_ID	Date	Month	Quarter	Year
20230101	01-Jan-2023	Jan	Q1	2023
20230102	02-Jan-2023	Jan	Q1	2023

- Store Dimension:

Store_ID	Store_Name	City	State	Country
1	Store A	New York	NY	USA
2	Store B	San Francisco	CA	USA

2. Snowflake Schema Example

The **Snowflake Schema** is a more normalized version of the star schema, where dimension tables are further broken down into related tables.

- **Fact Table: Sales**
  - Same as in the Star Schema:
  - **Columns**: `` `Sales_ID` ``, `` `Product_ID` ``, `` `Date_ID` ``, `` `Store_ID` ``, `` `Quantity_Sold` ``, `` `Total_Sales_Amount` ``
- **Dimension Tables**:
  - **Product Dimension**:
    - `` `Product_ID` ``, `` `Product_Name` ``, `` `Category_ID` ``, `` `Brand_ID` ``
  - **Category Dimension** (Normalized from Product Dimension):
    - `` `Category_ID` ``, `` `Category_Name` ``
  - **Brand Dimension** (Normalized from Product Dimension):
    - `` `Brand_ID` ``, `` `Brand_Name` ``
  - **Date Dimension**:
    - `` `Date_ID` ``, `` `Date` ``, `` `Month_ID` ``, `` `Quarter_ID` ``, `` `Year` ``
  - **Month Dimension** (Normalized from Date Dimension):
    - `` `Month_ID` ``, `` `Month_Name` ``
  - **Quarter Dimension** (Normalized from Date Dimension):
    - `` `Quarter_ID` ``, `` `Quarter_Name` ``
  - **Store Dimension**:
    - `` `Store_ID` ``, `` `Store_Name` ``, `` `City_ID` ``
  - **City Dimension** (Normalized from Store Dimension):
    - `` `City_ID` ``, `` `City_Name` ``, `` `State_ID` ``
  - **State Dimension** (Normalized from City Dimension):
    - `` `State_ID` ``, `` `State_Name` ``, `` `Country_ID` ``
  - **Country Dimension** (Normalized from State Dimension):
    - `` `Country_ID` ``, `` `Country_Name` ``

Sample Data:

- Fact Table (Sales):

Sales_ID	Product_ID	Date_ID	Store_ID	Quantity_Sold	Total_Sales_Amount
1	101	20230101	1	5	$500
2	102	20230102	2	3	$300

- Product Dimension:

Product_ID	Product_Name	Category_ID	Brand_ID
101	Laptop	10	20
102	Smartphone	10	21

- Category Dimension:

Category_ID	Category_Name
10	Electronics

- Brand Dimension:

Brand_ID	Brand_Name
20	Dell
21	Apple

- Date Dimension:

Date_ID	Date	Month_ID	Quarter_ID	Year
20230101	01-Jan-2023	1	1	2023
20230102	02-Jan-2023	1	1	2023

- Month Dimension:

Month_ID	Month_Name
1	January

- Quarter Dimension:

Quarter_ID	Quarter_Name
1	Q1

- **Store Dimension**:

Store_ID	Store_Name	City_ID
1	Store A	100
2	Store B	101

- **City Dimension**:

City_ID	City_Name	State_ID
100	New York	200
101	San Francisco	201

- **State Dimension**:

State_ID	State_Name	Country_ID
200	NY	300
201	CA	300

- **Country Dimension**:

Country_ID	Country_Name
300	USA

Comparison

- **Star Schema**:
  - Easier to understand and navigate.
  - Denormalized with fewer joins required in queries.
  - Faster query performance but potentially more data redundancy.
- **Snowflake Schema**:
  - More normalized, which reduces redundancy and saves storage space.
  - More complex queries with multiple joins.
  - Typically used in scenarios where data integrity and storage efficiency are critical.

These schemas are foundational structures in a data warehouse, designed to efficiently organize and query large volumes of data.