Name:	Pranita Kumbhar
Roll No:	70
Class/Sem:	TE/V
Experiment No.:	1
Title:	Data Warehouse Construction – Star schema and Snowflake schema
Date of Performance:	17/07/25
Date of Submission:	24/07/25
Marks:	
Sign of Faculty:	

Aim: To Build a Data Warehouse – Star Schema, Snowflake Schema and Fact Constellation Schema

Objective: A data warehouse is a large store of data collected from multiple sources within a business. The objective of a data warehouse system is to provide consolidated, flexible, meaningful data storage to the end user for reporting and analysis.

Theory:

In general, the warehouse design process consists of the following steps:

1. Choose a business process to model (e.g., orders, invoices, shipments, inventory, account administration, sales, or the general ledger). If the business process is organizational and involves multiple complex object collections, a data warehouse model should be followed. However, if the process is departmental and focuses on the analysis of one kind of business process, a data mart model should be chosen.
2. Choose the business process grain, which is the fundamental, atomic level of data to be represented in the fact table for this process (e.g., individual transactions, individual daily snapshots, and so on).
3. Choose the dimensions that will apply to each fact table record. Typical dimensions are time, item, customer, supplier, warehouse, transaction type, and status.
4. Choose the measures that will populate each fact table record. Typical measures are numeric additive quantities like dollars sold and units sold.

Steps to Draw Star, Snowflake, and Fact Constellation Schemas

1. Star Schema:

- Step 1: Identify the central fact table, which contains quantitative data (e.g., sales, revenue).
- Step 2: Determine the dimension tables related to the fact table, such as time, product, customer, etc.
- Step 3: Define the relationships between the fact table and each dimension table, usually a one-to-many relationship.
- Step 4: Draw the fact table at the center and connect it to each dimension table using lines, creating a star-like structure.

2. Snowflake Schema:

- Step 1: Start with the fact table as in the Star Schema.
- Step 2: Identify the dimension tables and further normalize them by breaking them into multiple related tables (e.g., split "Location" into "Country" and "City").
- Step 3: Establish relationships between these normalized dimension tables and the fact table.
- Step 4: Draw the fact table at the center, then connect it to the dimension tables, which in turn connect to their sub-tables, forming a snowflake-like structure.

3. Fact Constellation Schema:

- Step 1: Identify multiple fact tables representing different processes or subjects (e.g., sales and inventory).
- Step 2: Identify the shared dimension tables that will connect to these fact tables.
- Step 3: Define relationships between each fact table and the shared dimension tables.
- Step 4: Draw all fact tables and connect them to the shared dimension tables, creating a constellation of facts and dimensions.

Problem Statement:

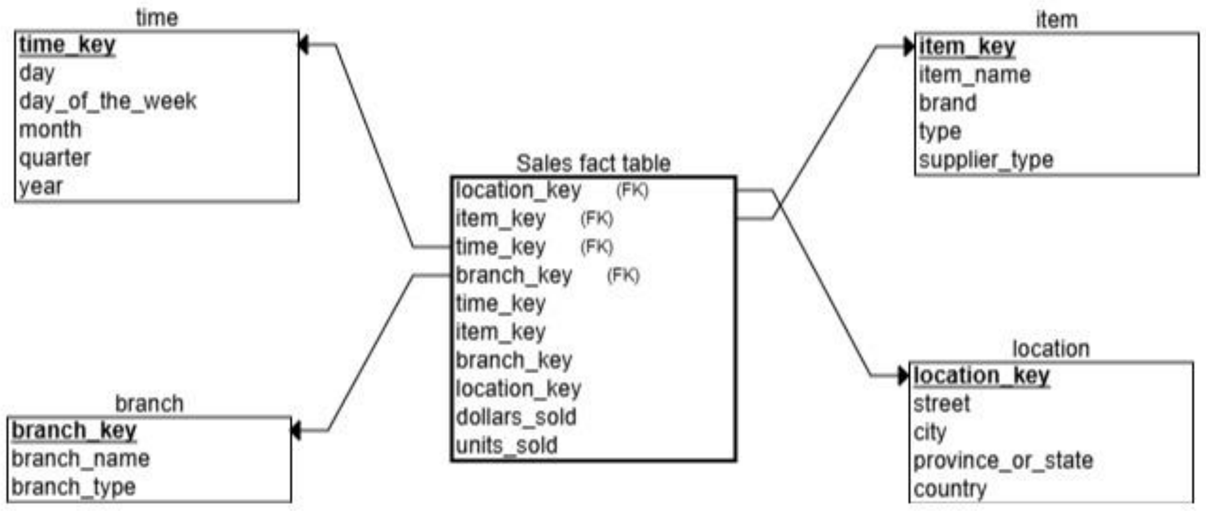
1. Sales Data Warehouse Schema (Star/Snowflake Schema)
Design a data warehouse to analyze sales performance across multiple branches, locations, and time periods. The schema should capture sales facts (dollars sold, units sold) and link them to dimensions such as time, item, branch, and location to enable aggregation, trend analysis, and reporting.

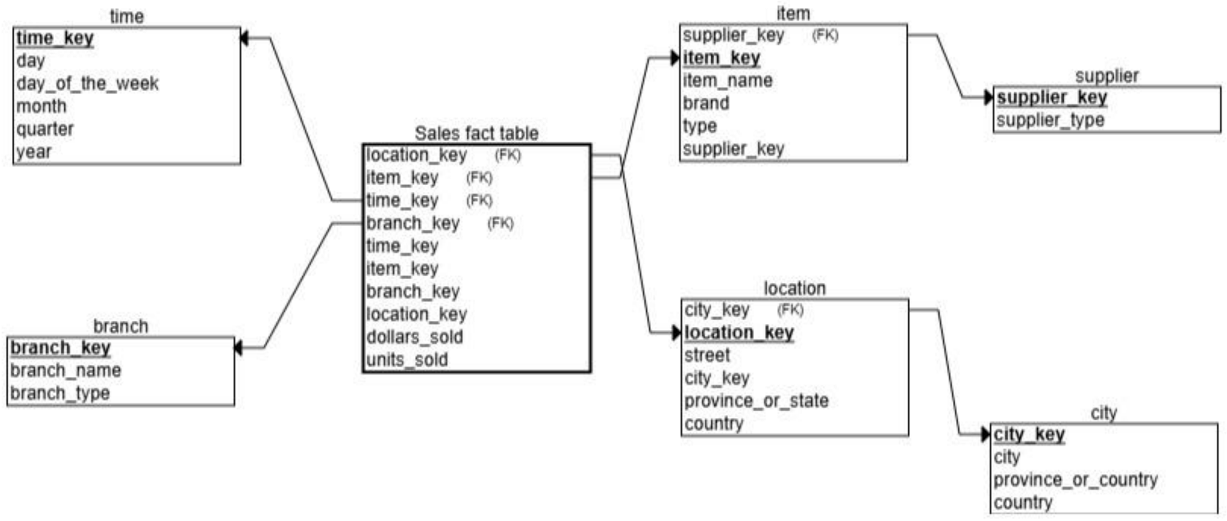
2. Enhanced Sales Schema with Normalized Dimensions (Snowflake Schema)
Develop a normalized sales data warehouse that allows analysis of sales at detailed location and supplier levels. The schema should facilitate multi-level aggregation (e.g., city → location) and provide insights into item performance, supplier contributions, and regional sales trends.

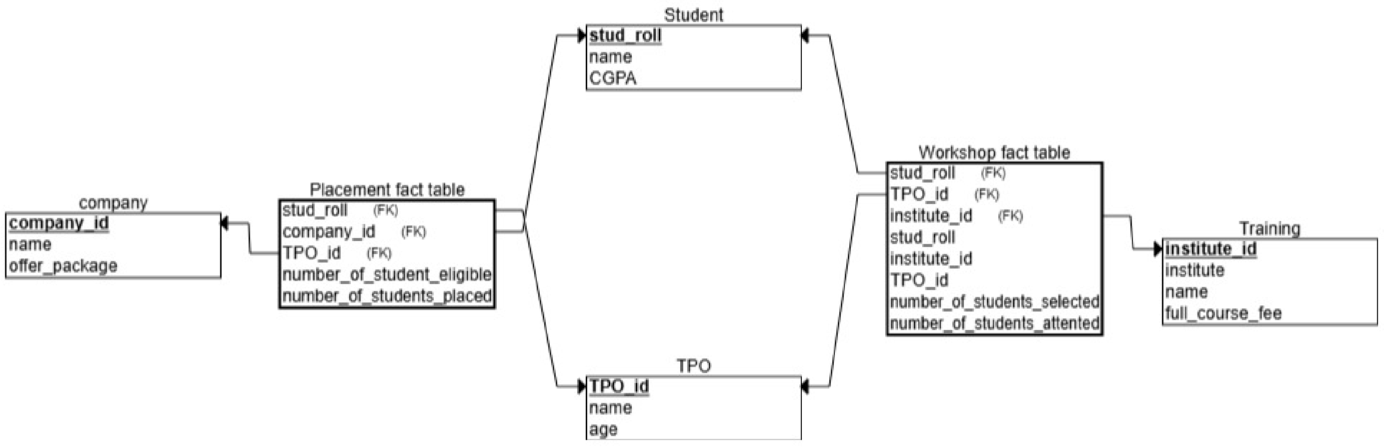
3. Student Placement and Workshop Schema

Create a data warehouse to track student placements and workshop participation. The schema should store placement details, student information, TPO coordination, and training/workshop data to analyze student eligibility, placement outcomes, and training effectiveness.

Construction of Star schema, Snowflake schema and Fact Constellation Schema:







Conclusion:

1. How does the Snowflake Schema compare to the Star Schema in terms of ease of maintenance and scalability?

Star Schema

- **Structure**: Central fact table + denormalized dimension tables (all attributes in one table per dimension).
- **Ease of Maintenance**:
    - **Easier to design and query** → only one join per dimension.
    - **Harder to maintain if attributes change** → since dimension tables are wide and redundant (e.g., city and country info repeated for every row).
- **Scalability**:
    - Works well for **smaller to medium datasets** where query speed is more important than storage efficiency.
    - As data grows large, storage redundancy and update anomalies can become issues.

Snowflake Schema

- **Structure**: Central fact table + normalized dimension tables (split into multiple related tables).
- **Ease of Maintenance**:
    - **More maintainable** because attributes are stored only once (e.g., separate City, Country, Continent tables).
    - Updates and changes are easier and cleaner (less redundancy).

- **Scalability**:
    - Scales **better for very large datasets** because it reduces storage and avoids data duplication.
    - But queries become **more complex and slower** (need multiple joins).

2. How do you manage the complexity of ETL (Extract, Transform, Load) processes in a Fact Constellation Schema?

1. Modularize the ETL Process

- **Separate ETL workflows** for **dimensions** and **facts**:
    - First, extract and load **shared dimension tables** (e.g., Date, Product, Customer) only **once**.
    - Then, process each **fact table** (e.g., Sales Fact, Inventory Fact) in separate, modular ETL jobs.
- This prevents duplication and allows you to reuse the same dimension data across multiple facts.

Benefit: Easier debugging, maintenance, and parallel execution.

2.Implement a Data Staging Area

- Before loading data into the warehouse:
    - Extract raw data from different sources into a **staging database**.
    - Apply **transformation and validation rules** here (e.g., data type conversions, cleansing).
- This ensures that by the time data reaches the fact constellation schema, it's clean and standardized.

Benefit: Isolates complex transformations and avoids corrupting the warehouse if something goes wrong.

3. Use Surrogate Keys for Shared Dimensions

- Since multiple fact tables reference common dimensions, using **surrogate keys** (system-generated IDs) ensures:
    - Consistent references across all fact tables.
    - Simplified joins and referential integrity.

For example:
Customer_Dim (Customer_Key, Name, Region) → referenced by
both Sales_Fact and Returns_Fact.
Benefit: Keeps relationships stable even if source system keys change.

4. **Apply Incremental & Parallel Loading**

- Instead of reloading entire tables each time:
    - Use **incremental ETL** (only new or changed records).
    - Load **multiple fact tables in parallel** where possible, since they share dimensions.

Benefit: Improves performance and scalability, especially in large constellation schemas.

5. **Use ETL Workflow Orchestration Tools**

- Use tools like:
    - **Apache Airflow**, **Talend**, **Informatica**, or **Azure Data Factory**
- These tools allow:
    - **Dependency management** between dimension and fact loads
    - **Scheduling**, error handling, and retries
    - **Visual DAGs** (Directed Acyclic Graphs) to manage complex ETL flows

Benefit: Makes complex ETL pipelines manageable and auditable.

6. **Maintain Metadata & Data Lineage**

- Keep a **metadata repository** to track:
    - Source-to-target mappings
    - Transformation rules
    - Table dependencies
- Track **data lineage** so you know where each data point originates.

Benefit: Simplifies debugging and future schema evolution.