| **Name:** | **James Lewis** |
| --- | --- |
| **Roll No:** | **32** |
| **Class/Sem:** | TE/V |
| **Experiment No.:** | 1 |
| **Title:** | Data Warehouse Construction – Star schema and Snowflake schema |
| **Date of Performance:** |  |
| **Date of Submission:** |  |
| **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:

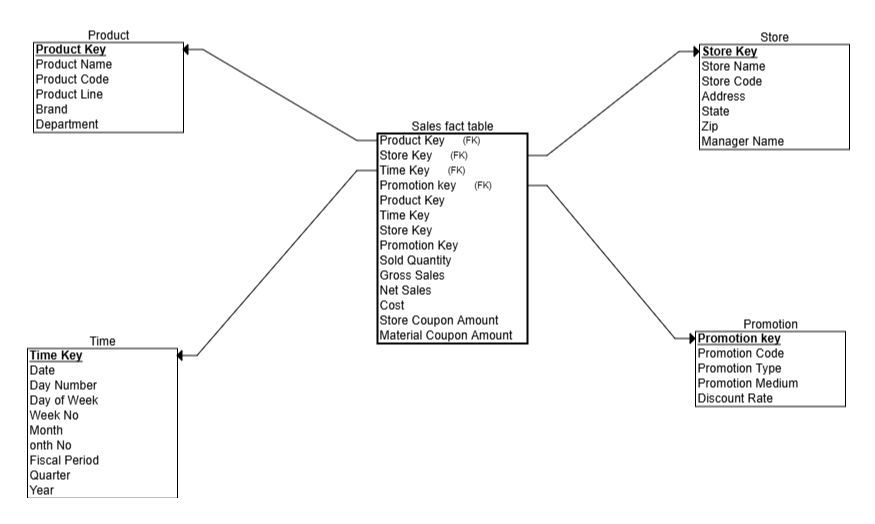
* Step 1: Start with the fact table as in the Star Schema.
* Step 2: Identify the dimension tables and further normalize them by breaking them into multiple related tables (e.g., split "Location" into "Country" and "City").
* Step 3: Establish relationships between these normalized dimension tables and 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.

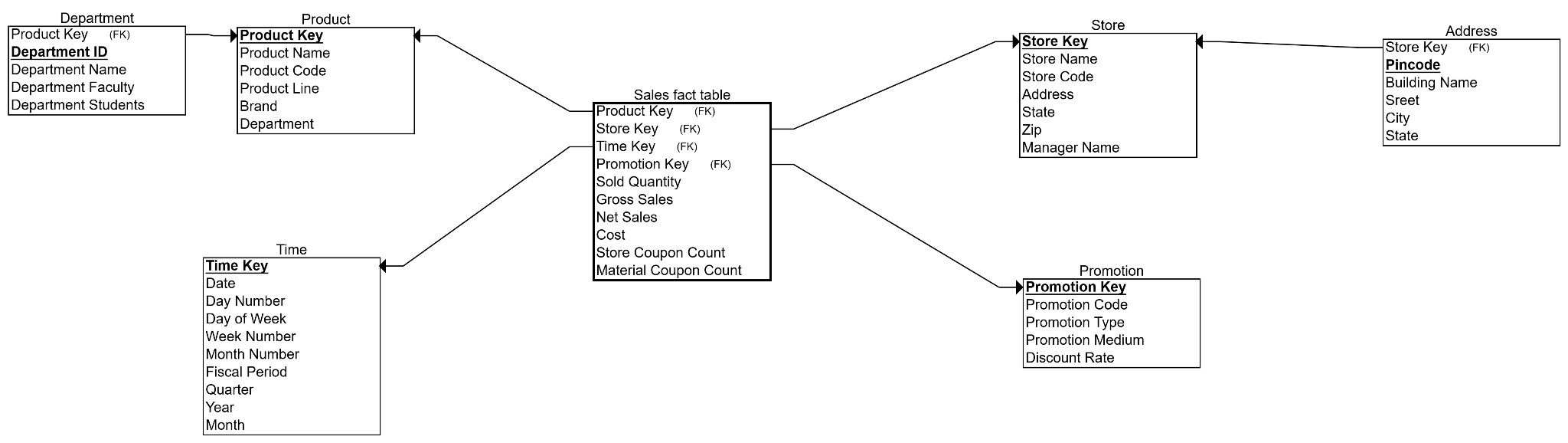
3. Fact Constellation Schema:

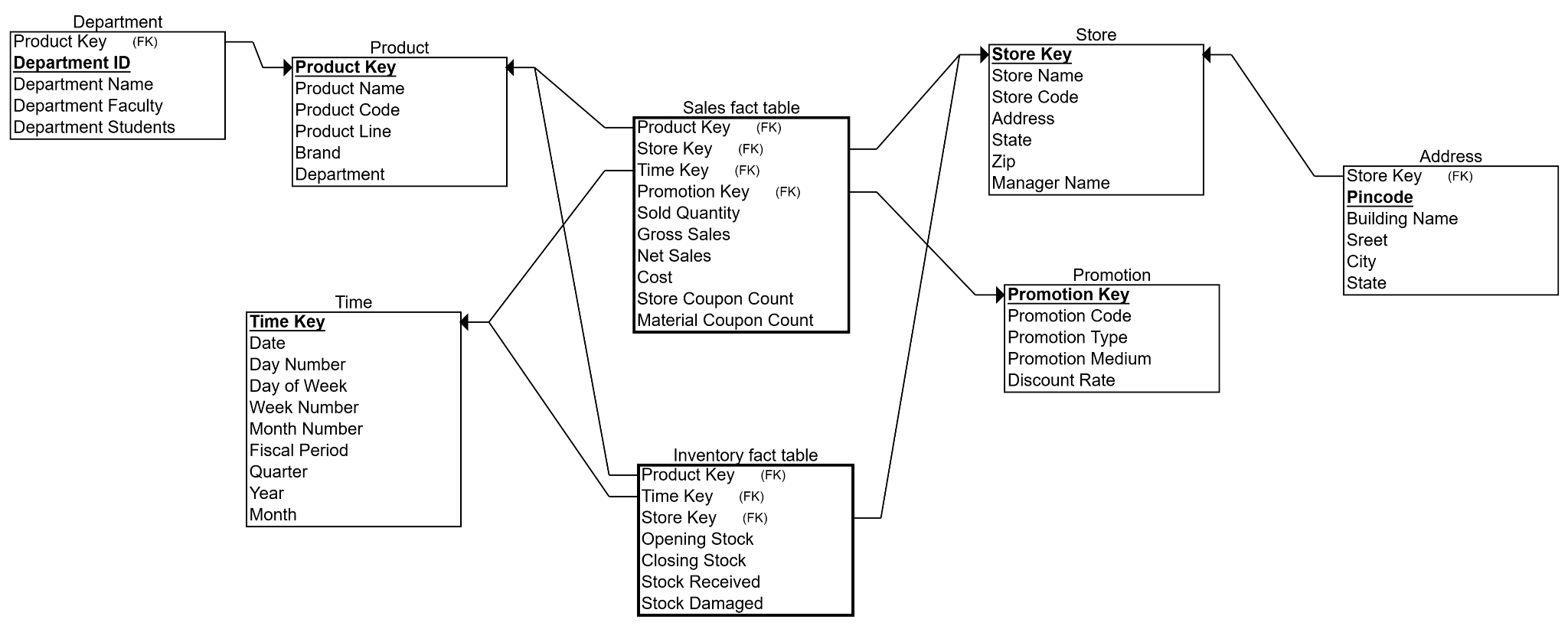
* Step 1: Identify multiple fact tables representing different processes or subjects (e.g., sales and inventory).
* Step 2: Identify the shared dimension tables that will connect to these fact tables.
* Step 3: Define relationships between each fact table and the shared dimension tables.
* Step 4: Draw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

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

1. **Star Schema**

****

1. **Snowflake schema  
   **
2. **Constellation Schema**

****

**Conclusion:**

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

**Star Schema:**

* 1. Easier to maintain due to its simple, denormalized structure.
  2. Faster query performance because of fewer table joins.
  3. Uses more storage space because of data redundancy.
  4. Scales well for read-heavy analytical workloads.

**Snowflake Schema:**

* 1. More complex and harder to maintain due to normalized tables.
  2. Potentially slower queries because of increased joins.
  3. More storage-efficient by minimizing data redundancy.
  4. Scalability depends on database optimization; may require more tuning for query performance.

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

Managing ETL complexity in a Fact Constellation Schema (also called a galaxy schema) involves addressing the challenges arising from multiple fact tables sharing dimension tables. Here’s how you can effectively manage that complexity:

### Managing ETL Complexity in Fact Constellation Schema

1. Modular ETL Design:  
   * Break down ETL processes into smaller, reusable modules.
   * Handle each fact table’s loading separately but coordinate shared dimension loads.
   * Use parameterized workflows to accommodate differences across fact tables.
2. Dimension Table Consistency:  
   * Load and maintain shared dimension tables carefully to ensure consistency.
   * Use a single source of truth for dimensions to avoid discrepancies.
   * Implement slowly changing dimension (SCD) techniques consistently across facts.
3. Incremental and Parallel Loads:  
   * Use incremental data loads where possible to reduce processing time.
   * Parallelize ETL jobs for different fact tables to improve efficiency.
4. Clear Dependency Management:  
   * Maintain a clear dependency order: dimensions must be loaded and validated before fact tables.
   * Use orchestration tools (like Apache Airflow, Azure Data Factory) to manage task dependencies.
5. Metadata Management and Documentation:  
   * Document schema relationships and ETL workflows to ease troubleshooting.
   * Use metadata-driven ETL tools to automate mappings and transformations.
6. Error Handling and Logging:  
   * Implement robust error detection and logging mechanisms.
   * Handle failures gracefully and enable easy reruns of failed jobs.
7. Data Quality Checks:  
   * Validate data at each ETL stage, especially in shared dimensions, to prevent propagation of errors.