**📑 Methodologies & Implementation Plan (Swiggy Data Pipeline)**

**Week 1 – Planning & System Design**

During the first week the focus is on planning, requirement gathering and high-level architecture. We will perform a comprehensive requirement analysis for all major entities of a Swiggy-like platform—Customers, Customer Address, Restaurants, Menus, Orders, Delivery Agents and Payments. For each entity we will identify what data fields exist, what the data sources are (CSV files, API exports, streaming logs) and how frequently the data arrives. We will also analyse compliance requirements such as PII masking and retention rules. A detailed **Software Requirement Specification (SRS)** will be drafted covering data flow diagrams, the planned star schema, slowly changing dimension strategy and KPI definitions. The Snowflake database environment will be planned, including Stage, Clean and Consumption schemas, warehouses, roles and policies. Draft ER diagrams and data lineage diagrams will be created for all major dimensions and facts. Finally, a GitHub repository will be created to version control all SQL scripts, ETL notebooks and documentation. The tech stack will be finalised—Snowflake for data warehousing, Python/DBT for orchestration, and Tableau/Power BI for reporting.

**Week 2 – Core Development: Ingestion & Dimension Loads**

In the second week we will build the foundation of the pipeline by creating **staging tables** in Snowflake for each entity. Using CREATE TABLE and COPY INTO scripts, raw CSV or API extracts will be loaded into the Stage schema with all columns as text plus audit fields. Snowflake Streams will be set up on each staging table to capture incremental changes such as new restaurants, updated menu prices or new customer records. Next, MERGE statements will populate the **Clean layer** tables where data types are converted, business rules applied and referential integrity enforced. All Clean tables will include audit columns and policy tags for sensitive data. By the end of Week 2 we will have fully functional Customer, Customer Address, Restaurant and Menu dimensions in the Clean layer, tested for both initial and incremental loads, and verified against SCD1/SCD2 requirements.

**Week 3 – Fact Tables, Delivery & Payment Tracking**

Week three shifts focus to building **Fact tables** and simulating the operational side of Swiggy. Using the Clean layer dimensions, we will design and implement Orders Fact and Delivery Fact tables capturing measures such as order amount, delivery time, delivery status and payment method. We will load sample “delta” CSV files to simulate new orders, changes in delivery status and refunds. These deltas will be processed through Streams and MERGE into the Clean and Consumption layers, demonstrating the real-time tracking capability. A separate Payment Fact table will be created to capture transaction status, amount, discounts and refunds. KPI SQL views will also be developed for common analytics such as “Order Status Transitions” (Ordered → Preparing → Out for Delivery → Delivered), “Average Delivery Time by City” and “Revenue by Restaurant.” This week ensures that all operational data can be transformed into analytics-ready facts in near real time.

**Week 4 – Finalisation: KPI Views, Testing & Documentation**

The last week is devoted to hardening the pipeline, adding historical tracking and preparing deliverables. All major dimensions (Customer, Address, Restaurant, Menu) will be extended to implement **SCD Type 2** logic in the Consumption schema so that historical changes are preserved. Additional KPI views will be created such as “Revenue by City and Month,” “Active Customers” and “Delivery Agent Performance.” Full pipeline testing will be performed by loading incremental data into staging, verifying Streams pick up changes, ensuring MERGE statements update Clean/Consumption correctly and validating SCD2 history records. Performance tests will check warehouse scaling and auto-suspend. Finally, all documentation will be polished: data lineage diagrams, schema descriptions, code walkthroughs and README. A short demo video will be recorded showing raw CSV files landing in the stage, being processed through the pipeline and feeding a Tableau dashboard automatically.

**SRS (Software Requirement Specification) for Swiggy Data Pipeline**

**1. Introduction**

This SRS document describes in detail the requirements, architecture and design principles for a Data Engineering pipeline that ingests, processes and manages data from a food delivery system similar to Swiggy. The pipeline transforms raw operational data—such as customers, restaurants, menus, orders, deliveries and payments—into an analytics-ready star schema on Snowflake. It is designed to support near real-time dashboards for stakeholders like business analysts, restaurant partners and delivery operations. By defining entities, data flow, security and performance expectations up front, this SRS serves as a blueprint for both development and future enhancements.

**2. Purpose**

The purpose of this pipeline is to provide a reliable, secure and scalable data foundation for business intelligence and regulatory reporting. Instead of building a transactional app, our focus is to design a backend data system that ingests raw feeds, applies business rules, tracks historical changes and exposes curated facts and dimensions to reporting tools. This enables stakeholders to make informed decisions on customer behaviour, restaurant performance, delivery efficiency and payment reconciliation while maintaining compliance with privacy laws.

**3. Scope**

The pipeline will cover end-to-end data flow from ingestion to analytics. It will load Customer, Customer Address, Restaurant, Menu, Order and Delivery data from CSV files or APIs into a **Stage layer**. From there data will be cleansed, typed and enriched in a **Clean layer** and finally modelled into a star schema in a **Consumption layer**. Incremental loads will be implemented using Snowflake Streams and MERGE statements. Slowly changing dimensions will be implemented using SCD Type 2 to maintain full history. KPI views will be created and exposed to Tableau/Power BI for interactive dashboards. Future scope includes integrating real streaming feeds, real payment gateways and additional facts such as refunds or promotions.

**4. Functional Requirements**

* **Data Ingestion:** All source files must be loaded into Snowflake staging tables using COPY INTO with appropriate file formats. Audit columns like file name, load timestamp and MD5 hash will be captured for traceability.
* **Data Transformation:** Clean layer tables will convert text columns into appropriate data types, enforce business rules, apply surrogate keys and apply Snowflake masking policies for sensitive fields such as mobile, email and DOB.
* **Dimensional Modelling:** Build Customer, Customer Address, Restaurant and Menu dimensions with SCD Type 2 handling to track historical changes.
* **Fact Tables:** Implement Orders Fact and Delivery Fact capturing measures like order amount, delivery status, payment type and timestamps, with foreign keys to the dimensions.
* **Incremental Loads:** Streams will capture inserts, updates and deletes from staging and apply them to Clean/Consumption tables via MERGE.
* **Audit & Traceability:** All layers will maintain \_stg\_file\_name, \_stg\_file\_load\_ts, \_stg\_file\_md5 and \_copy\_data\_ts to ensure complete lineage and compliance audits.

**5. Non-Functional Requirements**

The pipeline must scale to millions of records with minimal latency. It should be secure, employing Snowflake’s masking policies and policy tags to protect PII. It should be maintainable with modular SQL scripts, clear folder structures and GitHub version control. Performance should be optimised using appropriate warehouse sizes, auto-suspend and clustering where needed. It should be extensible for future facts and dimensions without major redesign.

**6. System Architecture**

The system is composed of four layers. **Stage Layer** stores raw files “as is” with audit columns. **Clean Layer** applies type casting, deduplication, referential integrity and business rules. **Consumption Layer** models the data into a star schema with dimensions (Customer, Address, Restaurant, Menu) and facts (Orders, Delivery, Payment) including SCD2 handling. Finally, a **BI Layer** connects Tableau/Power BI to the Consumption schema for dashboards. The architecture supports both batch and incremental processing and allows historical as well as current-state analysis.

**7. Entities & Attributes**

* **Customer:** ID, Name, Mobile, Email, Gender, DOB, Anniversary, Preferences, Created/Modified dates.
* **Customer Address:** Address ID, Customer ID, Flat No, House No, Locality, City, State, Pincode, Coordinates, Primary Flag, Address Type.
* **Restaurant:** ID, Name, Cuisine, Pricing for Two, Location ID, Phone, Operating Hours, Status.
* **Menu:** Menu ID, Restaurant ID, Item Name, Description, Price, Category, Availability, Item Type.
* **Orders Fact:** Order ID, Customer ID, Restaurant ID, Menu ID, Amount, Quantity, Status, Order Date/Time.
* **Delivery Fact:** Delivery ID, Order ID, Delivery Agent ID, Pickup Time, Delivery Time, Delivery Status, Refund Flag.  
  Each entity will be documented with its source, transformations, keys and relationships in the ER diagram.

**8. Data Flow**

Data flows from the external source (CSV/API) into the Snowflake Stage schema via COPY commands. Streams capture any changes in staging and feed them into the Clean schema through MERGE statements. The Clean layer ensures type integrity and enriches data. From Clean, data is merged into the Consumption schema’s dimensions and facts with SCD2 for history. BI tools connect directly to the Consumption schema to generate KPIs and dashboards. A diagram will show these arrows from Stage → Clean → Consumption → BI.

**9. Security & Privacy**

Sensitive columns such as Mobile, Email, Gender and DOB will be tagged with Snowflake policy tags and masked using masking policies. Audit columns in each layer provide full lineage and enable compliance checks. Only authorised roles will have access to PII columns. Warehouses will use secure connections and data at rest will be encrypted by Snowflake automatically.

**10. Testing & Validation**

Initial loads will be validated by comparing record counts between source files and staging tables. Delta files will be loaded to ensure Streams pick up changes and MERGE statements update the Clean and Consumption layers correctly. SCD2 tables will be validated for correct effective start and end dates and current record flags. KPI views will be checked against known sample outputs to verify correctness. Performance tests will measure load times and warehouse scaling behaviour.

**11. Deliverables**

* Snowflake SQL scripts for creating Stage, Clean and Consumption tables, Streams and MERGE logic.
* ER diagrams and data lineage diagrams showing relationships and flow.
* KPI SQL views for business metrics.
* SRS document hosted in GitHub.

.