**Project Roadmap: PySpark + PostgreSQL Data Warehouse**

**🔹 Phase 1: Environment & Setup (Foundation)**

1. Install and configure PostgreSQL.
2. Create 4 schemas: source, stage, core, reports.
3. Install Apache Spark (or PySpark local).
4. Set up JDBC driver to connect Spark ↔ PostgreSQL.
5. Create a Python virtual environment for project dependencies.
6. Create project folder structure (src/, sql/, data/, docs/).
7. Initialize GitHub repo and push initial setup.

**🔹 Phase 2: Data Ingestion (Source Layer)**

1. Choose dataset (E-commerce / Retail transactions).
2. Explore dataset with Pandas (basic stats, nulls, duplicates).
3. Define raw table structures in source schema.
4. Write PySpark job to read raw CSV/JSON into DataFrames.
5. Write DataFrame → PostgreSQL (df.write.jdbc) into source tables.
6. Automate ingestion for multiple files (loop through folder).
7. Validate ingestion by row counts (PySpark vs PostgreSQL).

**🔹 Phase 3: Staging Layer (Incremental Loads)**

1. Create staging tables (stage.orders, stage.customers).
2. Implement logic for **full load** (initial load).
3. Implement logic for **incremental load** (using timestamp or surrogate key).
4. Create a control table stage.load\_tracker to store last load timestamp.
5. Write PySpark function to load only new rows → stage schema.
6. Handle **duplicate rows** (drop duplicates by ID + timestamp).
7. Add data quality checks (not null, valid ranges).
8. Document differences between full load vs incremental load.

**🔹 Phase 4: Core Layer (Business Transformations)**

1. Design a star schema (fact + dimension model).

* dim\_customer, dim\_product, dim\_date
* fact\_sales

1. Create SQL scripts for core schema tables.
2. Use PySpark transformations to populate dimension tables.
3. Use PySpark joins to build fact\_sales.
4. Implement surrogate keys (e.g., customer\_id, product\_id).
5. Handle slowly changing dimensions (SCD Type 1: overwrite).
6. Add data quality checks: foreign key consistency, duplicate facts.
7. Test queries (e.g., join fact\_sales + dim\_customer).

**🔹 Phase 5: Reports Layer (Analytics Views)**

1. Create SQL views in reports schema:

* monthly\_sales\_summary
* top\_customers
* sales\_by\_region
* product\_category\_performance

1. Test queries with sample results.
2. Optimize queries with indexes (on date, customer\_id).
3. Create PySpark SQL queries for quick validation.
4. Export report query results as CSV/Parquet.

**🔹 Phase 7: Advanced Features**

1. Connect BI tool (Power BI / Tableau / Streamlit dashboard) to reports schema.
2. Create visualization (sales trend, top categories).