**1. Data Pipeline for ETL (Extract, Transform, Load)**

**Objective**: Create a simple ETL pipeline that extracts data from a CSV file, performs transformations (e.g., data cleaning, handling missing values), and loads the processed data into another csv file database.

**Requirements**:

* Extract data from a CSV file.
* Perform data cleaning (handle missing values, duplicates).
* Transform data (e.g., change data types, add calculated columns).
* Load the data into a NEW csv FILE.
* Use Python libraries such as pandas, sqlalchemy, and psycopg2.

**2. Data Aggregation and Reporting**

**Objective**: Process and aggregate large datasets and generate summary reports.

**Requirements**:

* Use pandas to aggregate data (e.g., calculate mean, sum, median).
* Use SQL-like operations (group by, filter, etc.) on large datasets.
* Create a summary report, e.g., a time series analysis (monthly or yearly), top N performing products, etc.
* Output the results as a CSV file or Excel sheet.

**Example**: If you have a dataset containing sales data, group it by region, and calculate total sales, average sales per region, and number of transactions.

**3. Data Cleaning and Transformation**

**Objective**: Clean and preprocess raw data (e.g., handle missing values, remove duplicates, normalize columns).

**Requirements**:

* Create a function to clean data by removing duplicates and filling missing values (mean imputation, forward-fill, etc.).
* Normalize numerical columns so that they are within the same range.
* Standardize text data by converting it to lowercase, stripping extra spaces, and handling special characters.

**Example**: If you have a dataset with customer information, clean the data by removing incomplete or erroneous records, filling missing fields (e.g., address, email), and normalizing phone numbers.

**4. Data Transformation and Feature Engineering for Machine Learning**

**Objective**: Create features suitable for machine learning models by transforming the data.

**Requirements**:

* Use pandas and scikit-learn for data transformations like encoding categorical variables, scaling features, and splitting datasets.
* Create new features based on existing ones (e.g., extracting the year from a date column or calculating a customer lifetime value).
* Split the data into training and test sets.

**Example**: Take a dataset of users, create features like age group (under 18, 18-35, etc.), and encode categorical variables (gender, region) into numerical values.

**5. Building a Data Warehouse**

**Objective**: Create a simple data warehouse where data from various sources is processed and stored in a star schema format.

**Requirements**:

* Use Python to connect to multiple data sources (CSV files, API, databases).
* Transform the data into a star schema with fact tables and dimension tables.
* Load the data into a SQL-based database (e.g., MySQL or PostgreSQL).
* Write a Python script to automate the process.

**Example**: Create a warehouse for e-commerce sales, where the fact table contains sales transactions and the dimension tables contain information like product details, customers, and time.

**6. Log Parsing and Analysis**

**Objective**: Parse and analyze server logs to identify patterns (e.g., number of errors, traffic trends).

**Requirements**:

* Use regular expressions (re module) to extract useful data from server log files (e.g., timestamp, status codes).
* Analyze log data to identify trends, such as errors or peak traffic times.
* Visualize the results with matplotlib or seaborn.

**Example**: Parse web server logs to determine the frequency of 404 errors or identify the peak traffic times based on timestamps.

**7. Database Optimization: Query Performance Tuning**

**Objective**: Analyze and optimize SQL queries by detecting bottlenecks and improving performance.

**Requirements**:

* Write Python code to analyze SQL queries and identify inefficient queries.
* Optimize queries by rewriting them, adding indexes, or partitioning tables.
* Use Python libraries like psycopg2 to execute and profile SQL queries.

**Example**: Optimize a SQL query that joins multiple large tables and takes a long time to execute. This might involve creating indexes, changing the join order, or rewriting subqueries.

**8. Data Validation and Quality Checks**

**Objective**: Create a script that performs automated data validation and quality checks on datasets.

**Requirements**:

* Create Python functions to check for common data issues, such as missing values, outliers, or incorrect data types.
* Generate a summary report that lists any data quality issues found.
* Implement checks such as consistency checks between different tables (e.g., foreign key constraints) or checks for duplicate records.

**Example**: Check if any sales records have negative prices or if customer IDs are missing.

**9. Batch Data Processing with Dask or PySpark**

**Objective**: Process large datasets in parallel using distributed computing frameworks like Dask or PySpark.

**Requirements**:

* Use Dask or PySpark to load, process, and analyze large datasets that don’t fit into memory.
* Perform transformations like filtering, aggregating, and joining data.
* Use the distributed framework to scale the processing and speed up tasks.

**Example**: Process a large dataset of sensor data from IoT devices and generate real-time analytics.

**10. Data Serialization (JSON, Parquet, and CSV)**

**Objective**: Serialize and deserialize large datasets into efficient file formats (JSON, Parquet, CSV).

**Requirements**:

* Convert data from one format (e.g., JSON) to another (e.g., CSV or Parquet).
* Perform data compression when saving data (e.g., compress CSV files).
* Optimize the reading and writing process to handle large datasets efficiently.

**Example**: Convert a dataset from JSON format to Parquet and compress the resulting Parquet file for faster reading.