## horizontal line



Tupande Data Engineer Doc

22.05.2023

**─**

By Constantine Ingumba

# Overview

To design and implement a data pipeline to process the given datasets and generate a dataset for the Tupande field team, I followed the below steps

**Step 1: Data Loading**

* Read the four CSV files (contract\_offers.csv, contract\_payments.csv, contracts.csv, leads.csv) into separate pandas DataFrames.
* The datasets were loaded into a database, Oracle.

**Step 2: Data Cleaning and Transformation**

* Used SQL to merge the relevant columns from different DataFrames to create a consolidated dataset that combines the information from all four datasets.
* Used the unique identifiers (e.g., offer\_id, contract\_reference, lead\_id) to join the datasets appropriately.

**Step 3: Dataset Generation**

* Created a new DataFrame containing the processed and analyzed data.
* Selected the relevant fields that are required by the Tupande field team.
* Saved the final dataset in the desired format i.e CSV.

**Step 4: Documentation**

* Documented the data pipeline, including any assumptions and recommendations.

# 

# Data Pipeline for Processing Loan Data

**Introduction**:

The goal of this data pipeline was to process loan data from multiple CSV files and generate a final dataset for the Tupande field team.

The pipeline was to be designed to be scalable, fault-tolerant, and cost-efficient. The following tools and technologies were selected based on their suitability for the task.

**Data Extraction, Transformation, and Loading (ETL):**

For the ETL process, Python was used due to its versatility and extensive data processing libraries such as pandas and ability to connect to multiple databases. Python allows us to efficiently read and manipulate CSV files, perform data transformations, and load the data into a database or data warehouse. The script can be executed locally or on a cloud-hosted environment.

Modern ELT tools like airbyte could have been used to load the data to a data warehouse and have DBT setup for performing transformations using SQL.

**Database**:

To store and manage the loan data, Oracle database was used. Oracle is a robust and scalable relational database management system (RDBMS) that provides ACID compliance and supports complex queries and aggregations. It is well-suited for handling structured data like loan information.

Oracle was readily available and thus chosen.

Another option could have been PostgreSQL that can be easily set up on a local machine or deployed on cloud platforms such as Amazon RDS, Azure Database for PostgreSQL, or Google Cloud SQL. Another good option could have been Amazon Redshift.

**Scalability and Fault Tolerance:**

To ensure scalability and fault tolerance, we can leverage cloud-based solutions. I used Oracle autonomous data warehouse on Oracle Cloud.

This offered automated backups, replication, and scalability features. By using a managed database service, we can easily scale the resources up or down based on the data processing requirements and handle high availability and fault tolerance without manual configuration.

**Orchestration**:

For orchestrating the data pipeline, we can use workflow management tools such as Apache Airflow,Prefect, AWS Step Functions, or Azure Data Factory. These tools enable us to define, schedule, and monitor the pipeline workflows, ensuring the timely execution of each step and handling dependencies between tasks.

**Data Validation and Quality:**

To ensure data quality and validate the generated dataset, we can incorporate data validation checks and quality assurance processes within the pipeline. This can include verifying data integrity, handling missing values, performing data quality checks, and incorporating data validation rules.

**Documentation and Collaboration:**

To facilitate collaboration and knowledge sharing, we can use collaborative platforms such as GitHub or GitLab to version control the pipeline code and document the pipeline design, including dependencies, configurations, and instructions for setting up and executing the pipeline.

**Conclusion:**

The proposed data pipeline design utilizes Python/Airbyte for ELT, Oracle/Redshift for data storage, and cloud-based services for scalability and fault tolerance. By leveraging these tools and technologies, we can efficiently process the loan data, generate the final dataset, and ensure scalability, fault tolerance, and cost efficiency in the pipeline's execution.