Data Pipelines with Python Project

Project Deliverable

- A GitHub repository with a python file (.py) or notebook (.ipynb) with your solution.

Project Deliverable

Telecom companies often have to extract billing data from multiple CSV files generated from various systems and transform it into a structured format for analysis and revenue reporting. This process can be time-consuming, error-prone, and hinder decision-making. Manually analyzing and reconciling billing data from different sources is a tedious task and often leads to delays in generating revenue reports. Thus, there is a need for an automated data pipeline that can extract billing data from multiple sources and transform it into a structured format for efficient analysis and revenue reporting.

Guidelines

Here are some guidelines and hints to help you create the data pipeline:

- **Determine the requirements:** First, you need to define the requirements of the data pipeline, including the source and destination of the data, the type of data that needs to be processed, the transformations that need to be applied, and the output format.
- **Extract the data:** Use Python to read the CSV files and extract the data.
- **Clean the data:** Perform data cleaning on the extracted data to remove any missing values and outliers. For example, you can replace missing values with an appropriate value or remove them altogether.
- **Transform the data:** Apply any necessary transformations on the data, such as data type conversion, data aggregation, and data filtering, to prepare the data for analysis.
- Merge the datasets: Join the different datasets into a single dataset that can be used for analysis.
- **Load the data:** Load the transformed data into a database or a file, such as a CSV file, that can be easily analyzed.
- **Automate the process:** Automate the data pipeline by scheduling it to run at a specific time, such as daily or weekly so that it can update the analysis data automatically.
- **Test the pipeline:** Test the data pipeline to ensure it produces the correct results. This can be done by comparing the results with the expected output or using a test dataset.

- **Optimize the pipeline:** Optimize the data pipeline to improve performance and reduce errors. This can be done by optimizing the code, parallel processing, and reducing the data size.
- **Monitor the pipeline:** Monitor the data pipeline to ensure that it runs smoothly and that there are no errors or issues.

Datasets

Here are three sample datasets (https://bit.ly/416WE1X) with billing data that can be joined. The datasets contain some missing values and outliers:

Dataset 1:
- Customer ID (numeric)
- Date of purchase (MM/DD/YYYY)
- Total amount billed (numeric)
- Payment status (categorical - paid, overdue, disputed)
- Payment method (categorical - credit card, bank transfer, e-wallet)
- Promo code (text)
- Country of purchase (categorical)

Dataset 2:
- Customer ID (numeric)
- Date of payment (MM/DD/YYYY)
- Amount paid (numeric)
- Payment method (categorical - credit card, bank transfer, e-wallet)
- Payment status (categorical - paid, overdue, disputed)
- Late payment fee (numeric)
- Country of payment (categorical)

Dataset 3:
- Customer ID (numeric)
- Date of refund (MM/DD/YYYY)
- Refund amount (numeric)
- Reason for refund (text)
- Country of refund (categorical)

Notes:
1. The datasets can be joined using Customer ID, Date of purchase/payment/refund, and country of purchase/payment/refund as keys.
2. The datasets may contain missing values and outliers for some fields, such as the total amount billed or refund amount.
3. The payment status may be missing or incomplete for some of the transactions.
4. The promo code field may be empty for some of the purchases.
5. The reason for the refund may be missing for some of the refund transactions.