Self-documenting Code for CPF ETL Batch Pipeline

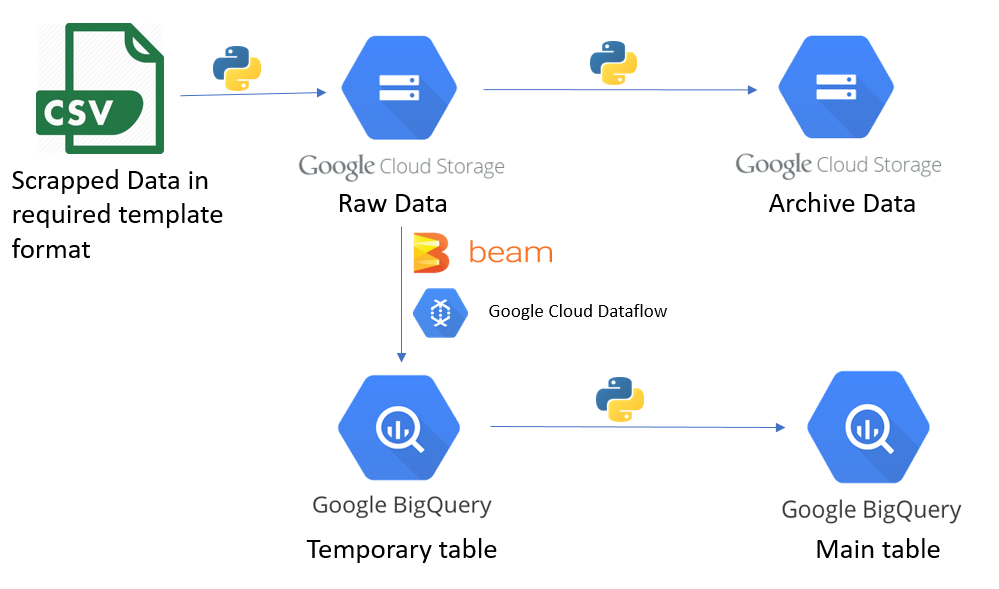
* Language used
* Directory and files

Introduction

Core of BI lies in data warehouse that gathers data from multiple sources and consolidate into a optimized model so they can be queried rapidly. The process of loading data into warehouse is otherwise called ETL. When data size is vast and voluminous the processing also needs to be robust. Distributed processing and big data solutions comes into rescue to provide agility for data loading and querying.

Big Data usually runs in cloud because of greater computational power available due to network of remote servers rather than local computer that bounded with fixed memory and speed. Google Cloud is a growing competitor in cloud computing and runs in same infrastructure that Google uses for its end-user products. It provides products for computing, storage, networking, artificial intelligence and big data.

Here we have created an ETL pipeline for Big Data using Google Cloud resources like Dataflow for processing, CloudStorage for storing and BigQuery as Data Warehouse. We have used GCPs AI platform notebook for managing the whole process by interacting with Dataflow, CloudStorage and BigQuery. Our pipeline uses Apache Beam model to batch process the data files and load into BigQuery.

Flow****

Source Files: Data files copied into our local hard drive.

Initialize BigQuery: Check whether Dataset, Stage and Main table exists, if not then create them using schema from our property files.

Initialize CloudStorage: Check whether cloud storage bucket is available, if not then create it.

Data Ingestion: Copy data files from local hard drive into input folder of Cloud Storage.

Run Pipeline: Run the Apache Beam pipeline. This will insert data into stage table from cloud storage.

Upsert: Perform upsert from stage table to main table. Also handles calculated columns.

Archiving: Move the data files from input folder of cloud storage into archive folder.

## Code Structure

CPF

├── cred

│ └── **useful-aquifer-293804-647d400a8cf9**.json

├── properties

│ ├── **properties\_cpf\_commodity\_daily\_data.json**

│ └── sql.py

├── resource

│ ├── BigQuery.py

│ └── CloudStorage.py

├── schemastore

│ └── **schema\_cpf\_itg\_commodity\_daily\_data.json**

├── wrapper

│ └── wrapper.py

├── testpipefinal.py

**cred/useful-aquifer-293804-647d400a8cf9.json** is the downloaded service account key file.

**properties/properties\_cpf\_commodity\_daily\_data.json** holds holds credentials and parameters/settings for the cloud resources.

**properties/sql.py** parses properties and schema to generate dynamic sql queries to be executed for upsert operation.

**resource/BigQuery.py** has functions to get/create bigquery dataset, tables and perform upsert.

**resource/CloudStorage.py** has functions for creating bucket, archiving files.

**schemastore/schema\_cpf\_itg\_commodity\_daily\_data.json** defines schema for temporary table and main table.

**wrapper/wrapper.py** holds wrapper functions for getting cloud resources and parsing bigquery table schema.

**testpipefinal.py** is our main file with pipeline code.

## Creating Property Files

We have two types of property files for each table i.e for cpf\_itg\_commodity\_daily\_data,

**source/schemastore/schema\_cpf\_itg\_commodity\_daily\_data.json** holds the schema of temporary table and main table.

**source/properties/properties\_cpf\_commodity\_daily\_data.json** holds GCP parameters, pipeline parameters.