Data Engineering Exercise: Building an ETL Pipeline with Apache Airflow

Objective

Develop a scalable and automated data pipeline using Apache Airflow to manage the ETL process of loading data from Google Cloud Storage (GCS) to BigQuery.

Setup Guidelines

Refer to the setup guide provided in `01_Cloud_Composer_Exercise_Setup.md` for instructions on setting up your GCP environment.

DAG Skeleton

Below is the DAG skeleton that you will complete as part of this exercise.

```
# Importing necessary libraries
from airflow import DAG
from airflow.providers.google.cloud.operators.bigquery import (
    BigQueryCreateEmptyDatasetOperator,
    BigQueryDeleteTableOperator,
    BigQueryCreateEmptyTableOperator,
)
from airflow.providers.google.cloud.transfers.gcs_to_bigquery import GCSToBigQueryOperator
from airflow.providers.google.cloud.transfers.gcs_to_variable import GoogleCloudStorageToVa
from airflow.utils.dates import days_ago

# Your code starts here

# Task 1: Define the Variables
# HINT: Define your variables (dataset_name, table_name, gcs_bucket, gcs_schema_object) usi

schema_fileds = [
    {"name": "INCIDENT_NUMBER", "type": "STRING", "mode": "NULLABLE"},
    {"name": "OFFENSE_CODE", "type": "INTEGER", "mode": "NULLABLE"},
    {"name": "OFFENSE_CODE_GROUP", "type": "STRING", "mode": "NULLABLE"},
    {"name": "OFFENSE_CODE_GROUP_No", "type": "INTEGER", "mode": "NULLABLE"},
    {"name": "OFFENSE_DESCRIPTION", "type": "STRING", "mode": "NULLABLE"},
    {"name": "DISTRICT", "type": "STRING", "mode": "NULLABLE"},
    {"name": "District_simple", "type": "STRING", "mode": "NULLABLE"},
    {"name": "District_simple_No", "type": "INTEGER", "mode": "NULLABLE"},
    {"name": "REPORTING_AREA", "type": "INTEGER", "mode": "NULLABLE"},
    {"name": "OCCURRED_ON_DATE", "type": "STRING", "mode": "NULLABLE"},
    {"name": "Hour1", "type": "STRING", "mode": "NULLABLE"},
    {"name": "Start_Night", "type": "STRING", "mode": "NULLABLE"},
    {"name": "Start_Day", "type": "STRING", "mode": "NULLABLE"},
    {"name": "Night_Day", "type": "STRING", "mode": "NULLABLE"},
    {"name": "YEAR", "type": "INTEGER", "mode": "NULLABLE"},
    {"name": "MONTH", "type": "INTEGER", "mode": "NULLABLE"},
    {"name": "DAY_OF_WEEK", "type": "STRING", "mode": "NULLABLE"},
    {"name": "WE_Workday", "type": "STRING", "mode": "NULLABLE"},
    {"name": "WE_Workday_No", "type": "INTEGER", "mode": "NULLABLE"},
    {"name": "HOUR", "type": "INTEGER", "mode": "NULLABLE"},
    {"name": "Counts_per_hour", "type": "INTEGER", "mode": "NULLABLE"},
    {"name": "STREET", "type": "STRING", "mode": "NULLABLE"},
    {"name": "Lat", "type": "FLOAT", "mode": "NULLABLE"},
    {"name": "Long", "type": "FLOAT", "mode": "NULLABLE"},
    {"name": "Location", "type": "STRING", "mode": "NULLABLE"}
]

# Define your DAG
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': days_ago(1),
    'retries': 1,
}
dag = DAG(
    'load_csv_to_bigquery',
    default_args=default_args,
    description='Load CSV data from GCS to BigQuery',
    schedule_interval='@once',
)

# Task 2: Load Data to BigQuery
load_data = GCSToBigQueryOperator(
    task_id='load_data',
    # Your code starts here
    # HINT: Define bucket, source_objects, destination_project_dataset_table, schema_fields
    # Your code ends here
    dag=dag,
)

# Task 3: Set Task Dependencies
# Your code starts here
# HINT: Ensure tasks run in the correct order by setting their dependencies using >> and <<
# Your code ends here
```

Task 1: Define the Variables

Hints (1) Define variables that will be used to specify dataset names, table names, GCS bucket names, and GCS schema object paths.

```
dataset_name = 'your_dataset_name'
table_name = 'your_table_name'
gcs_bucket = 'your_gcs_bucket_name'
gcs_schema_object = 'path_to_your_schema.json'
```

Task 2: Check/Create Dataset

Hints (2) Use `BigQueryCreateEmptyDatasetOperator` to check for or create the dataset in BigQuery.

```
create_dataset = BigQueryCreateEmptyDatasetOperator(
    task_id='create_dataset',
    dataset_id=dataset_name,  # use the variable defined in Task 1
    dag=dag,
)
```

Task 3: Load Data to BigQuery

Hints (3) Use `GCSToBigQueryOperator` to load data from GCS to BigQuery.

```
load_csv = GCSToBigQueryOperator(
    task_id='load_csv',
    bucket=gcs_bucket,  # use the variable defined in Task 1
    source_objects=['data_file1.csv', 'data_file2.csv'],  # specify your source data files
    destination_project_dataset_table=f"{dataset_name}.{table_name}",  # use variables defi
    skip_leading_rows=1,  # adjust as per your data
    write_disposition='WRITE_TRUNCATE',  # adjust as needed
    schema_fields=schema_fields,  # use the variable defined in Task 1
    field_delimiter=';',  # specify the delimiter used in your data files
    dag=dag,
)
```

Task 4: Set Task Dependencies

Hints (4) Use the bitshift operators (», «) or `set_downstream` and `set_upstream` methods to set the task dependencies.

```
# Using bitshift operators
create_dataset >> load_schema >> delete_table >> create_table >> load_data

# OR using set_downstream and set_upstream methods
create_dataset.set_downstream(load_schema)
load_schema.set_downstream(delete_table)
```

```
delete_table.set_downstream(create_table)
create_table.set_downstream(load_data)
```