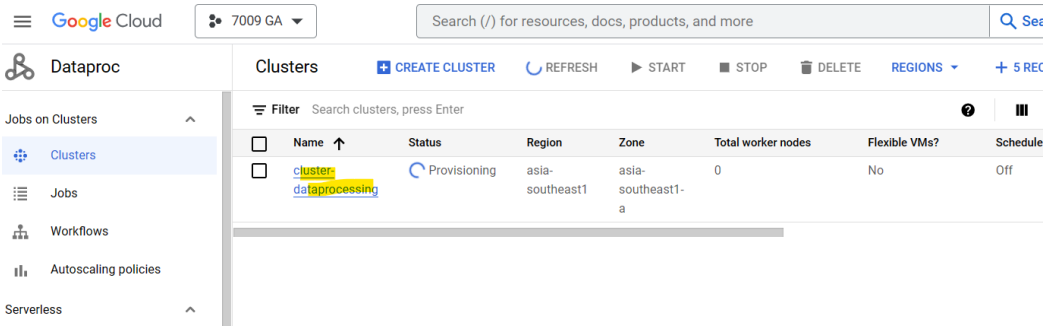
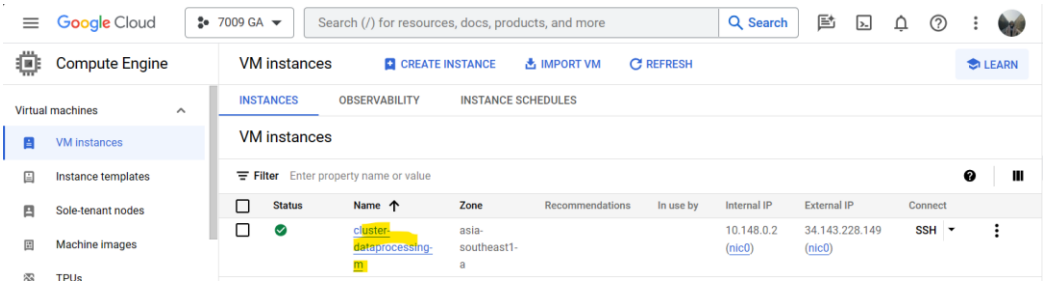
Data Processing

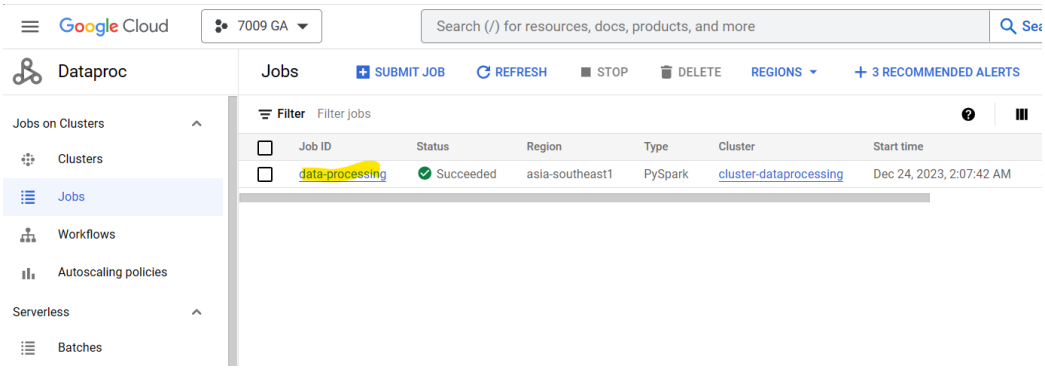
Figure 5: Data Processing Workflow

The implementation of the data processing part involves using Google Cloud Dataproc to manage Spark and Hadoop services and submitting a Pyspark job to perform specific data processing tasks. Below is a step-by-step outline of the data processing implementation:

Tools	**Explanation**
Dataproc	Setup Spark and Hadoop Cluster – create a datarpoc cluster with the necessary Spark and Hadoop configurations to handle our data processing tasks. "cluster-dataprocessing" has been created in Dataproc.  VMinstance has been created automatically once created dataproc cluster. 

PySpark

It is a language for writing data processing tasks on Dataproc. In this case, after PySpark script are completed and uploaded to Cloud Storage, we submit PySpark jobs to Dataproc clusters for distributed data processing.



Before we submit the PySpark job in dataproc cluster, we have to write script and upload PySpark script into Cloud Storage.

The following data processing tasks have been implemented in PySpark:

i. Data Cleaning (Drop Missing Values)

```
# Remove leading and trailing whitespaces and empty strings
df = df.withColumn('Date', col('Date').cast(StringType()))  #
df = df.na.drop()
```

ii. Data Transformation (convert the date into date format)

```
# Convert to date format
date_format = 'd-MMM-yy'
df = df.withColumn('Date', to_date(col('Date'), date_format))
```

iii. Data Integration & Aggregation (calculate average for each Month)

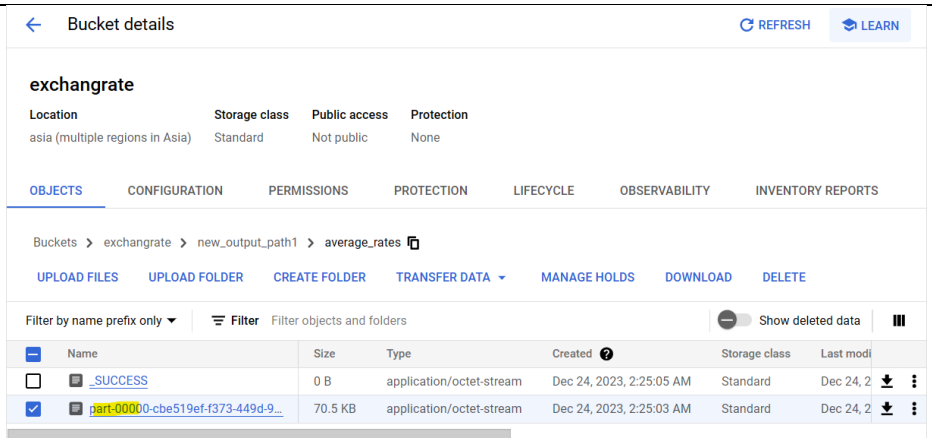
```
# Add 'Month' and 'Year' columns to the DataFrame
df = df.withColumn('Month', month(col('Date')))
df = df.withColumn('Year', year(col('Date')))

# Group the data by Year, Month, and calculate the average for each currency
average_rates = df.groupBy('Year', 'Month').agg(
    {'USD': 'avg', 'GBP': 'avg', 'EUR': 'avg', 'JPY100': 'avg', 'CHF': 'avg',
     'AUD': 'avg', 'CAD': 'avg', 'SGD': 'avg', 'HKD100': 'avg', 'THB100': 'avg',
     'PHP100': 'avg', 'TWD100': 'avg', 'KRW100': 'avg', 'IDR100': 'avg',
     'SAR100': 'avg', 'SDR': 'avg', 'CNY': 'avg', 'BND': 'avg'}
)
```

iv. Stored output in Cloud Storage:

```
# Save the results to GCS
average_rates.write.csv(output_path, header=True, mode='overwrite')
```

Cloud Storage

After PySpark job is succeeded, we can see that the output of the data processing is stored in Cloud Storage.



This implementation leverages Dataproc to manage Spark and Hadoop clusters efficiently, and PySpark for data processing tasks. The processed data is stored back in Cloud Storage, providing a scalable and cloud-native solution for large-scale data processing.