

Project

MAANG Stock Prices Analysis

Milestone 6

Cloud Problem Definition

Indra Kumar Chandaka

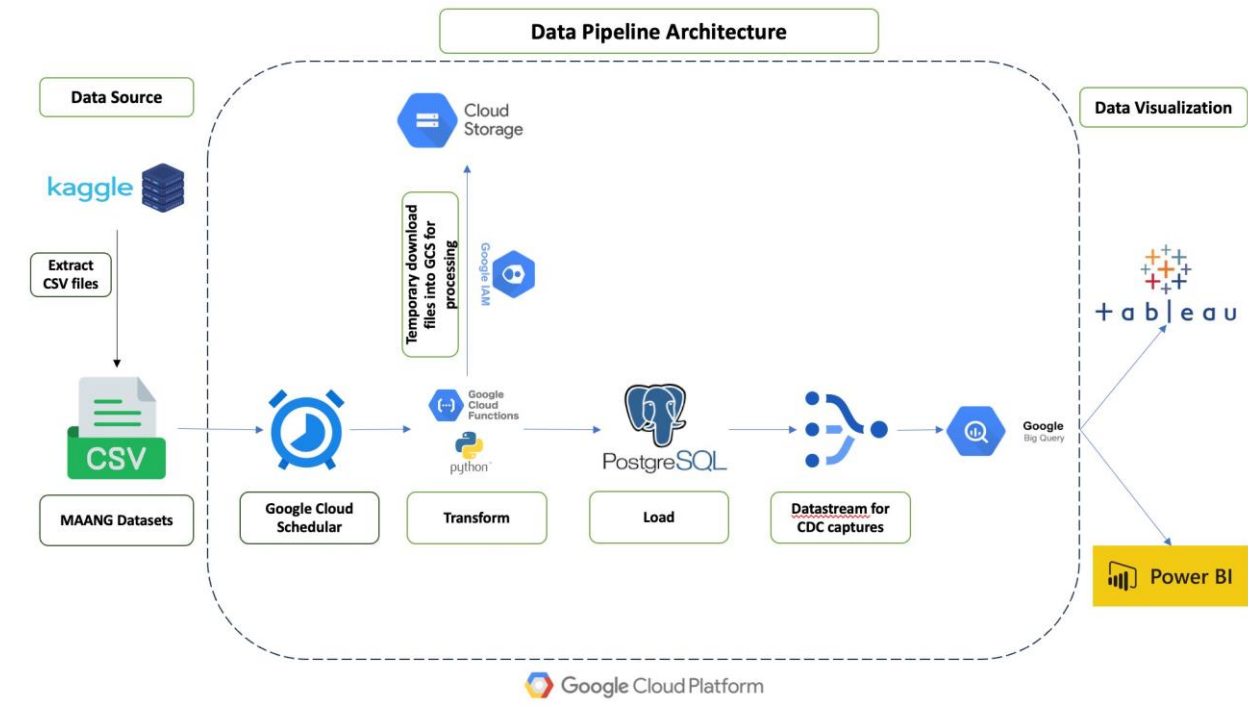
ichandak@gitam.in

Submission Date: Dec 1st, 2023

Objective:

This project seeks to develop a data driven solution to address the complexities of analyzing the historical stock prices of MAANG companies and identify the market conditions and external factors that could help make informed investment decisions. With this, the project aims to contribute to a broader understanding of MAANG stock prices, providing investors with niche perspectives of market dynamics and risks involved. The project aims to predict the effectiveness of trading strategies by back tracing with historical data insights and provide actionable insights to make data driven decisions about the trends, opportunities, and volatility.

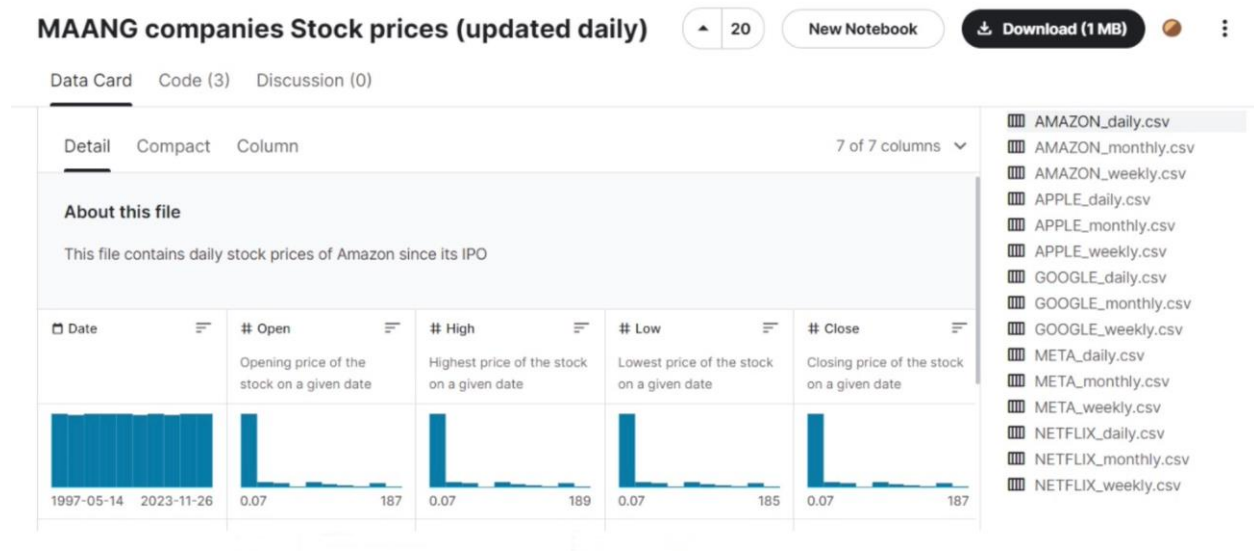
Furthermore, by developing an efficient cloud-based data engineering solution which can handle large volumes of data for real time analysis.

Data Pipeline Architecture:

The MAANG stocks dataset from Kaggle is updated on a daily, weekly, and monthly basis. Here our aim is to retrieve the data daily and transform it to the database (Postgres) using the Kaggle API. After this, we read the data in cloud functions using python and this is triggered by Google cloud scheduler based on the requirements. As the cloud function is serverless, we need a temporary storage which can be used to hold the extracted files in Google Cloud Storage. Then we transform the data (data types, renaming columns, adding columns) and then files are ingested to three tables based on daily, weekly, and monthly into the Postgres DB. Then we use Datastream to capture for the changes (Auto incremental load) and move it into BigQuery.

Data source:

<https://www.kaggle.com/datasets/nikhil1e9/ne7lix-stock-price>



Data Inges&on:

We have adopted py scripts for daily, weekly, and monthly to transform and insert the 15 files into cloud postgres. Here, we used the Kaggle API instead of downloading the 15 files into the local or cloud bucket, as the data is very dynamic and changes daily.

We are cloud scheduler to trigger the cloud func4ons.

Google Cloud My First Project sche Search

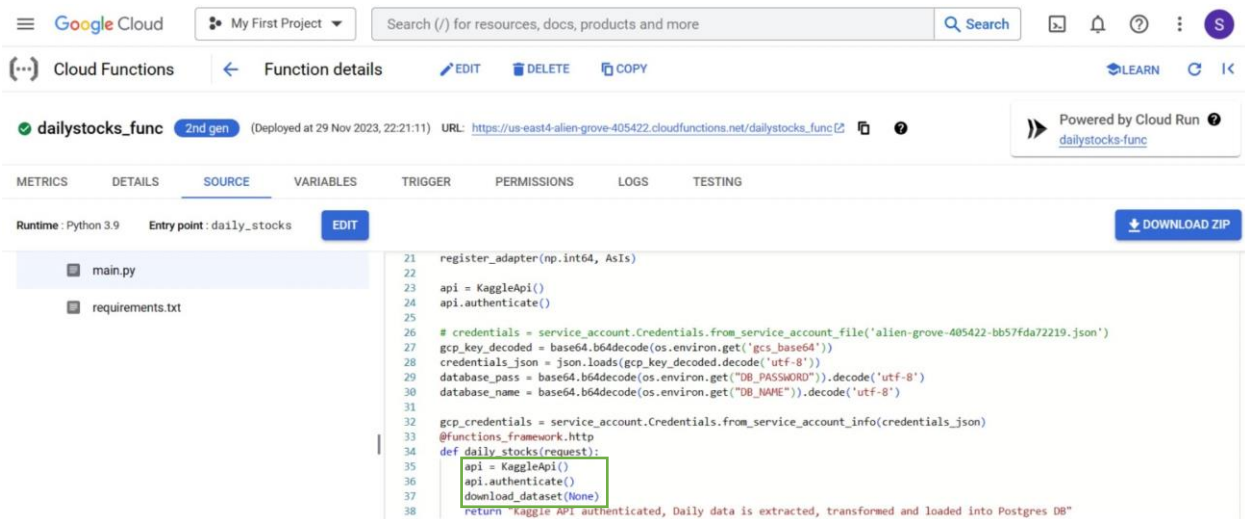
Cloud Scheduler Jobs CREATE JOB REFRESH FORCE RUN EDIT COPY PAUSE RESUME DELETE

SCHEDULER JOBS APP ENGINE CRON JOBS

Name	Status of last execution	Region	State	Description	Frequency	Target	Last run	Next run	Actions
dailyscheduler	Has not run yet.	us-central1	Enabled	Daily Scheduler to run cloud functions	0 23 * * * (America/New_York)	URL : https://us-east4-alien-grove-405422.cloudfunctions.net/dailystocks_func		30 Nov 2023, 23:00:00	⋮
monthlyscheduler	Has not run yet.	us-east4	Enabled	Monthly Scheduler for stocks	0 23 1 * * (America/New_York)	URL : https://us-central1-alien-grove-405422.cloudfunctions.net/monthlystocks_func		1 Dec 2023, 23:00:00	⋮
weeklyscheduler	Has not run yet.	us-east1	Enabled	Weekly Scheduler for stocks data	0 23 * * 6 (America/New_York)	URL : https://us-east1-alien-grove-405422.cloudfunctions.net/weeklystocks_func		2 Dec 2023, 23:00:00	⋮

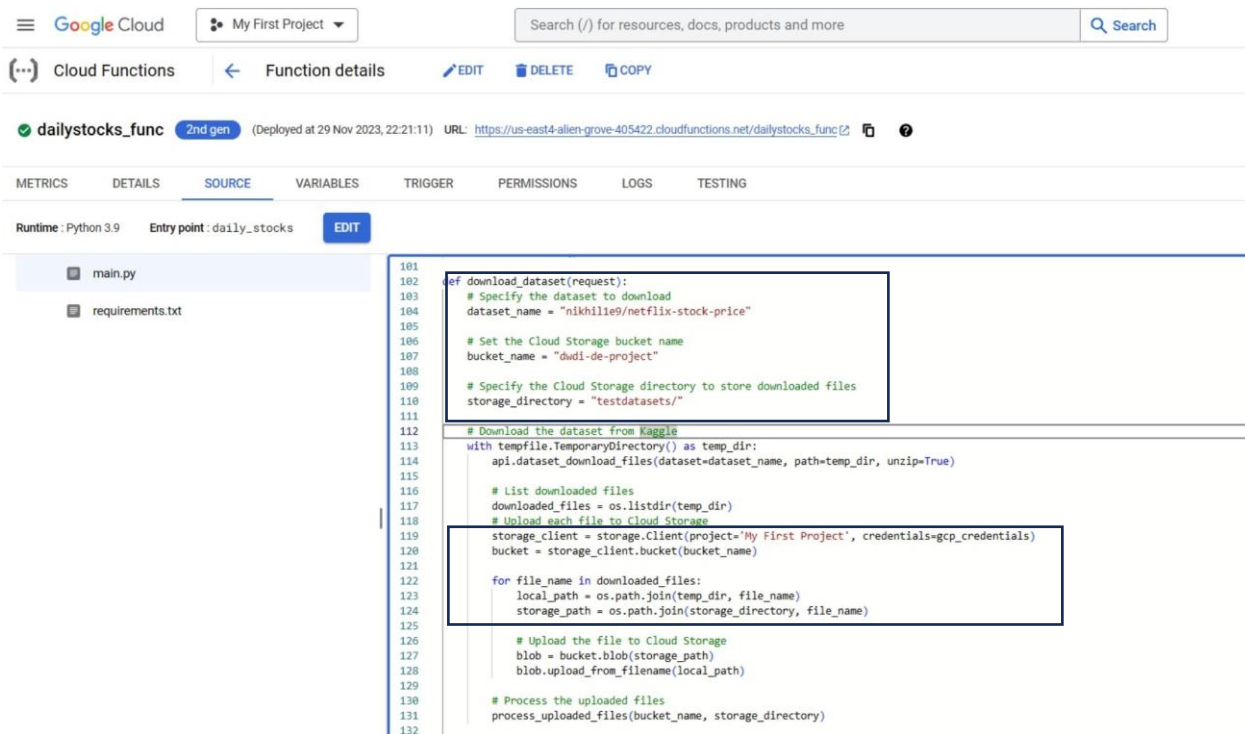
Authen4ca4on of Kaggle API by providing creden4als:

IE7374 Data Warehousing and Data Integra5on



```
21 register_adapter(np.int64, AsIs)
22
23 api = KaggleApi()
24 api.authenticate()
25
26 # credentials = service_account.Credentials.from_service_account_file('alien-grove-405422-bb57fda72219.json')
27 gcp_key_decoded = base64.b64decode(os.environ.get('gcs_base64'))
28 credentials_json = json.loads(gcp_key_decoded.decode('utf-8'))
29 database_pass = base64.b64decode(os.environ.get('DB_PASSWORD')).decode('utf-8')
30 database_name = base64.b64decode(os.environ.get('DB_NAME')).decode('utf-8')
31
32 gcp_credentials = service_account.Credentials.from_service_account_info(credentials_json)
33 @functions_framework.http
34 def daily_stocks(request):
35     api = KaggleApi()
36     api.authenticate()
37     download_dataset(None)
38     return "Kaggle API authenticated, Daily data is extracted, transformed and loaded into Postgres DB"
```

Providing the dataset link to download from Kaggle. Furthermore, the details of temporary storage bucket are also provided. It automa4cally authen4cates using GCP default creden4als to get the bucket access:



```
101
102 def download_dataset(request):
103     # Specify the dataset to download
104     dataset_name = "nikhill9/netflix-stock-price"
105
106     # Set the Cloud Storage bucket name
107     bucket_name = "dudi-de-project"
108
109     # Specify the Cloud Storage directory to store downloaded files
110     storage_directory = "testdatasets/"
111
112     # Download the dataset from Kaggle
113     with tempfile.TemporaryDirectory() as temp_dir:
114         api.dataset_download_files(dataset=dataset_name, path=temp_dir, unzip=True)
115
116     # List downloaded files
117     downloaded_files = os.listdir(temp_dir)
118     # Upload each file to Cloud Storage
119     storage_client = storage.Client(project='My First Project', credentials=gcp_credentials)
120     bucket = storage_client.bucket(bucket_name)
121
122     for file_name in downloaded_files:
123         local_path = os.path.join(temp_dir, file_name)
124         storage_path = os.path.join(storage_directory, file_name)
125
126         # Upload the file to Cloud Storage
127         blob = bucket.blob(storage_path)
128         blob.upload_from_filename(local_path)
129
130     # Process the uploaded files
131     process_uploaded_files(bucket_name, storage_directory)
132
```

Now we call the process to upload files into cloud storage by providing the temporary path. Below is the screenshot for Daily, weekly, and monthly. Here the data is read and transformed. Now here we deleted the data stored in the bucket i.e., blobs.

Daily -

Google Cloud My First Project Search (/) for resources, docs, products and more

Cloud Functions Function details EDIT DELETE COPY

dailystocks_func 2nd gen (Deployed at 29 Nov 2023, 22:21:11) URL: https://us-east4-alien-grove-405422.cloudfunctions.net/dailystocks_func

METRICS DETAILS SOURCE VARIABLES TRIGGER PERMISSIONS LOGS TESTING

Runtime: Python 3.9 Entry point: daily_stocks EDIT

main.py requirements.txt

```

149 def process_uploaded_files(bucket_name, storage_directory):
150     data_dict = {"daily": []}
151     storage_client = storage.Client()
152     bucket = storage_client.bucket(bucket_name)
153
154     # List files in the Cloud Storage directory
155     blobs = list(bucket.list_blobs(prefix=storage_directory))
156     daily_blobs = [blob for blob in blobs if 'daily' in blob.name]
157     print("Daily blobs: ", daily_blobs)
158     for blob in daily_blobs:
159         # Process only CSV files
160         if blob.name.endswith('.csv'):
161             # Download the file content
162             content = blob.download_as_text()
163             # Process the content (example: convert to DataFrame)
164             df = pd.read_csv(StringIO(content))
165             company = blob.name.split('.')[0]
166             company_updated = company.split('/')[1]
167             # Append the DataFrame to the 'weekly' list
168             data_dict["daily"].append(df.assign(company=company_updated))
169
170     delete_objects(bucket_name, storage_directory)
171     # Concatenate DataFrames
172     daily_data = pd.concat(data_dict["daily"], ignore_index=True)
173     column_mapping = {
174         'Date': 'date',
175         'Open': 'open',
176         'High': 'high',
177         'Low': 'low',
178         'Close': 'close',
179         'Adj Close': 'adj_close',
180         'Volume': 'volume',
181         'Company': 'company'
182     }

```

Weekly -

Google Cloud My First Project Search (/) for resources, docs, products and more

Cloud Functions Function details EDIT DELETE COPY

weeklystocks_func 2nd gen (Deployed at 29 Nov 2023, 20:48:53) URL: https://us-east1-alien-grove-405422.cloudfunctions.net/weeklystocks_func

METRICS DETAILS SOURCE VARIABLES TRIGGER PERMISSIONS LOGS TESTING

Runtime: Python 3.9 Entry point: weekly_stocks EDIT

main.py requirements.txt

```

160 def process_uploaded_files(bucket_name, storage_directory):
161     data_dict = {"weekly": []}
162     storage_client = storage.Client()
163     bucket = storage_client.bucket(bucket_name)
164
165     # List files in the Cloud Storage directory
166     blobs = list(bucket.list_blobs(prefix=storage_directory))
167     weekly_blobs = [blob for blob in blobs if 'weekly' in blob.name]
168     print("Weekly blobs: ", weekly_blobs)
169     for blob in weekly_blobs:
170         # Process only CSV files
171         if blob.name.endswith('.csv'):
172             # Download the file content
173             content = blob.download_as_text()
174             # Process the content (example: convert to DataFrame)
175             df = pd.read_csv(StringIO(content))
176             company = blob.name.split('.')[0]
177             company_updated = company.split('/')[1]
178             # Append the DataFrame to the 'weekly' list
179             data_dict["weekly"].append(df.assign(company=company_updated))
180
181     delete_objects(bucket_name, storage_directory)
182     # Concatenate DataFrames
183     weekly_data = pd.concat(data_dict["weekly"], ignore_index=True)
184     column_mapping = {
185         'Date': 'date',
186         'Open': 'open',
187         'High': 'high',
188         'Low': 'low',
189         'Close': 'close',
190         'Adj Close': 'adj_close',
191         'Volume': 'volume',
192         'Company': 'company'
193     }

```

Monthly -

```

173 def process_uploaded_files(bucket_name, storage_directory):
174     data_dict = {"monthly": []}
175     storage_client = storage.Client()
176     bucket = storage_client.bucket(bucket_name)
177
178     # List files in the Cloud Storage directory
179     blobs = list(bucket.list_blobs(prefix=storage_directory))
180     monthly_blobs = [blob for blob in blobs if 'monthly' in blob.name]
181     print("Monthly blobs: ", monthly_blobs)
182     for blob in monthly_blobs:
183         # Process only CSV files
184         if blob.name.endswith('.csv'):
185             # Download the file content
186             content = blob.download_as_text()
187             # Process the content (example: convert to DataFrame)
188             df = pd.read_csv(StringIO(content))
189             company = blob.name.split('/')[0]
190             company_updated = company.split('/')[1]
191             # Append the DataFrame to the 'monthly' list
192             data_dict["monthly"].append(df.assign(company=company_updated))
193
194     delete_objects(bucket_name, storage_directory)
195     # Concatenate DataFrames
196     monthly_data = pd.concat(data_dict["monthly"], ignore_index=True)
197     column_mapping = {
198         'Date': 'date',
199         'Open': 'open',
200         'High': 'high',
201         'Low': 'low',
202         'Close': 'close',
203         'Adj Close': 'adj_close',
204         'Volume': 'volume',
205         'Company': 'company'
206     }

```

Now, after this the data gets downloaded into the cloud storage bucket for temporary purpose. We read the extracted files using the py script and read the files based on the files name and then read it as dataframes to transform such as renaming columns, adding column (Company name) and modifying the datatypes. Then we check now if the data is already present in Postgres Cloud using the py script. In case it is not present, we insert the data required to avoid data redundancy.

After reading the data, we call the connectDB function to insert it into Cloud Postgres. Now the transformed data is fed into cloud postgres.

```

182     }
183     # Map DataFrame columns to PostgreSQL columns
184     df_daily_mapped = daily_data.rename(columns=column_mapping)
185     connect_db_bulk(df_daily_mapped, 'daily')
186
187     return len(df_daily_mapped)
188
189

```

Now we provide the DB connections to connect to postgres DB and the password and DB name has been encoded in Base 64 format to ensure security.

IE7374 Data Warehousing and Data Integra5on

Google Cloud | My First Project | Search (/) for resources, docs, products and more

Cloud Functions | Function details | EDIT | DELETE | COPY

🟢 dailystocks_func 2nd gen (Deployed at 29 Nov 2023, 22:21:11) URL: https://us-east4-alien-grove-405422.cloudfunctions.net/dailystocks_func

METRICS | DETAILS | SOURCE | VARIABLES | TRIGGER | PERMISSIONS | LOGS | TESTING

Runtime: Python 3.9 | Entry point: daily_stocks | EDIT

main.py | requirements.txt

```
34 def daily_stocks(request):
35     api = KaggleApi()
36     api.authenticate()
37     download_dataset(None)
38     return "Kaggle API authenticated, Daily data is extracted, transformed and loaded into Postgres DB"
39
40 def connect_db_bulk(df, tbname):
41     #Get Credentials
42     db_host = os.environ.get("DB_HOST")
43     db_port = os.environ.get("DB_PORT")
44     db_name = database_name
45     db_user = os.environ.get("DB_USER")
46     db_password = database_pass
47
48     # Use the variables in your database connection logic
49     connection_params = {
50         'host': db_host,
51         'port': db_port,
52         'database': db_name,
53         'user': db_user,
54         'password': db_password,
55     }
56     # Establish a connection to your PostgreSQL database
57     connection = psycopg2.connect(**connection_params)
```

We are now checking of the data is present or not, if not we insert the data in order to avoid data redundancy.

Google Cloud | My First Project | Search (/) for resources, docs, products and more

Cloud Functions | Function details | EDIT | DELETE | COPY

🟢 dailystocks_func 2nd gen (Deployed at 29 Nov 2023, 22:21:11) URL: https://us-east4-alien-grove-405422.cloudfunctions.net/dailystocks_func

METRICS | DETAILS | SOURCE | VARIABLES | TRIGGER | PERMISSIONS | LOGS | TESTING

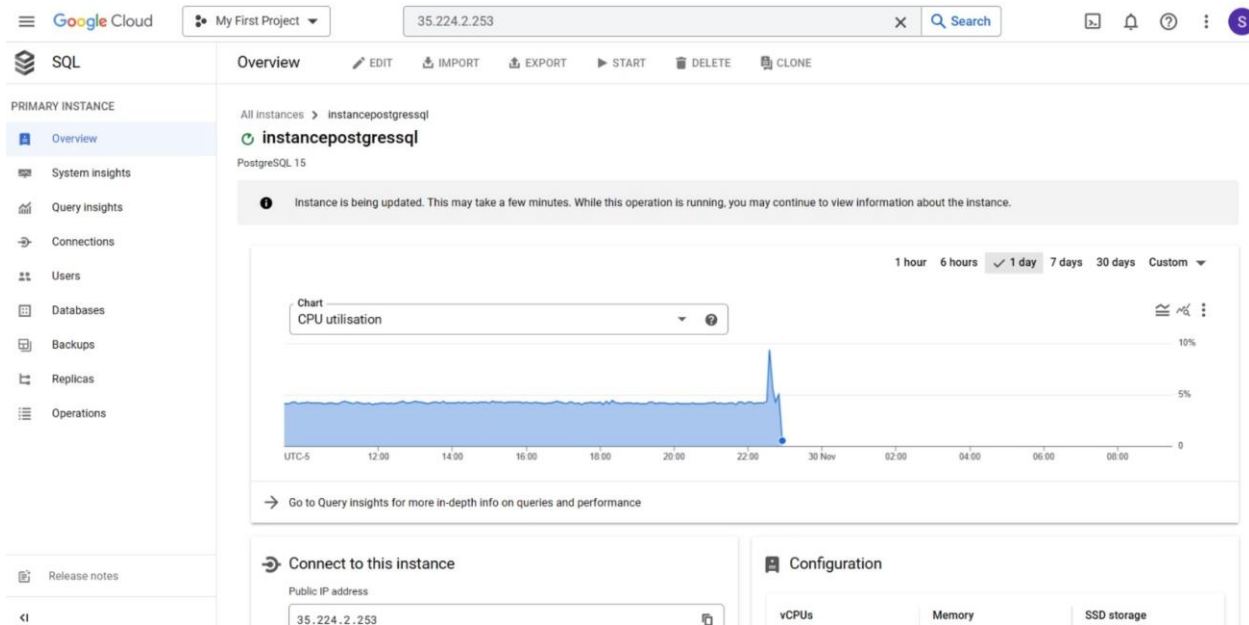
Runtime: Python 3.9 | Entry point: daily_stocks | EDIT

main.py | requirements.txt

```
60 cursor = connection.cursor()
61 table_name = tbname
62 schema_name = 'public'
63 df['date'] = pd.to_datetime(df['date']).dt.date
64 max_dates_by_company = df.groupby('company')['date'].max()
65 # max_dates_by_company_df = max_dates_by_company.reset_index()
66 max_dates_by_company = max_dates_by_company.reset_index().sort_values(by='company')
67 # print(max_dates_by_company)
68 sql_query = "SELECT company, MAX(date) as date FROM public.daily GROUP BY company ORDER BY company;"
69 # Execute the query and store the result in a DataFrame
70 db_result = pd.read_sql_query(sql_query, connection)
71 db_result['date'] = pd.to_datetime(db_result['date']).dt.date
72 res_cd = pd.DataFrame()
73
74 if len(db_result) == 0:
75     # res_cd = max_dates_by_company
76     insert_query = f"INSERT INTO {schema_name}.{table_name} VALUES (%s, %s, %s, %s, %s, %s, %s)"
77     records = df.to_records(index=False)
78     values = [tuple(record) for record in records]
79     cursor.executemany(insert_query, values)
80 else:
81     db_result['date'] = pd.to_datetime(db_result['date']).dt.date
82     for i in range(len(db_result)):
83         if db_result.loc[i, "date"] < max_dates_by_company.loc[i, "date"]:
84             res_cd = res_cd.append(db_result.loc[i], ignore_index=True)
85
86     res = pd.DataFrame()
87     for i in range(len(res_cd)):
88         company_res = df[(df['company'] == res_cd.loc[i, 'company']) & (df['date'] > res_cd.loc[i, 'date'])]
89         res = res.append(company_res, ignore_index=True)
90
91     print("result: \n", res)
92     if(len(res)!=0):
93         insert_query = f"INSERT INTO {schema_name}.{table_name} VALUES (%s, %s, %s, %s, %s, %s, %s)"
94         records = res.to_records(index=False)
95         values = [tuple(record) for record in records]
96         cursor.executemany(insert_query, values)
97
98     connection.commit()
99     cursor.close()
100     connection.close()
```

Loading into Postgres:

IE7374 Data Warehousing and Data Integra5on



Daily data:

The screenshot shows the DBeaver SQL Editor with a query result for 'daily d'. The query is 'select * from daily d;'. The result is a table with columns: date, open, high, low, close, adj_close, volume, and company. The data is sorted by date in descending order, showing daily stock price data for various companies including Apple, Meta, Google, Netflix, and Amazon.

	date	open	high	low	close	adj_close	volume	company
1	2023-11-27	189.92	190.67	188.9	189.79	189.79	40,500,500	APPLE
2	2023-11-27	336.18	339.9	334.2	334.7	334.7	15,646,300	META
3	2023-11-27	137.57	139.63	137.54	138.05	138.05	17,868,000	GOOGLE
4	2023-11-27	479.03	482	475.35	479.17	479.17	3,623,800	NETFLIX
5	2023-11-27	147.53	149.26	146.88	147.73	147.73	53,666,700	AMAZON
6	2023-11-24	139.54	139.677	137.47	138.22	138.22	8,828,600	GOOGLE
7	2023-11-24	340.13	341.86	336.77	338.23	338.23	5,467,500	META
8	2023-11-24	146.7	147.2	145.32	146.74	146.74	22,378,400	AMAZON
9	2023-11-24	190.87	190.9	189.25	189.97	189.97	24,048,300	APPLE
10	2023-11-24	477.11	480.4	475.2	479.56	479.56	1,404,700	NETFLIX
11	2023-11-22	139.1	141.1	139	140.02	140.02	17,306,400	GOOGLE
12	2023-11-22	144.57	147.74	144.57	146.71	146.71	45,669,100	AMAZON
13	2023-11-22	339.21	342.92	338.58	341.49	341.49	10,702,700	META
14	2023-11-22	476.8	482.7	476.56	478	478	2,841,600	NETFLIX
15	2023-11-22	191.49	192.93	190.83	191.31	191.31	39,617,700	APPLE
16	2023-11-21	143.91	144.05	141.5	143.9	143.9	71,226,000	AMAZON
17	2023-11-21	472.63	477.02	471.21	474.95	474.95	2,997,700	NETFLIX
18	2023-11-21	338.33	339.9	335.9	336.98	336.98	12,027,900	META
19	2023-11-21	137.94	138.965	137.705	138.62	138.62	17,648,100	GOOGLE
20	2023-11-21	191.41	191.52	189.74	190.64	190.64	38,134,500	APPLE
21	2023-11-20	189.89	189.89	191.91	189.88	191.45	46,505,100	APPLE
22	2023-11-20	135.5	138.425	135.49	137.92	137.92	19,569,400	GOOGLE
23	2023-11-20	334.89	341.87	334.19	339.97	339.97	16,960,500	META
24	2023-11-20	465.4	476.76	465.4	474.47	474.47	3,617,600	NETFLIX
25	2023-11-20	145.13	146.63	144.73	146.13	146.13	41,951,200	AMAZON
26	2023-11-17	190.25	190.38	188.57	189.69	189.69	50,922,700	APPLE
27	2023-11-17	142.66	145.23	142.54	145.18	145.18	49,636,700	AMAZON

Weekly data:

IE7374 Data Warehousing and Data Integration

Database Navigator: dwdi-cloud-db, public, Tables, weekly

SQL Editor: select * from weekly w;

weekly 1 x

	date	open	high	low	close	adj_close	volume	company
1	2023-11-27	137.57	139.63	137.54	138.05	138.05	17,868,000	GOOGLE
2	2023-11-27	479.03	482	475.35	479.17	479.17	3,623,800	NETFLIX
3	2023-11-27	189.92	190.67	188.9	189.79	189.79	40,500,500	APPLE
4	2023-11-27	147.53	149.26	146.88	147.73	147.73	53,666,700	AMAZON
5	2023-11-27	336.18	339.9	334.2	334.7	334.7	15,646,300	META
6	2023-11-20	465.4	482.7	465.4	479.56	479.56	10,861,600	NETFLIX
7	2023-11-20	334.89	342.92	334.19	338.23	338.23	45,158,600	META
8	2023-11-20	135.5	141.1	135.49	138.22	138.22	63,352,500	GOOGLE
9	2023-11-20	189.89	192.93	189.25	189.97	189.97	148,305,600	APPLE
10	2023-11-20	145.13	147.74	141.5	146.74	146.74	181,224,700	AMAZON
11	2023-11-13	326.2	338.4	325.7	334.19	334.19	67,552,100	META
12	2023-11-13	133.36	138.88	132.77	138.7	138.7	72,183,200	GOOGLE
13	2023-11-13	447.25	467.28	442.6	466.95	466.95	15,648,300	NETFLIX
14	2023-11-13	185.82	190.96	184.21	189.71	189.71	211,939,300	APPLE
15	2023-11-13	142.08	147.29	139.52	142.83	142.83	205,884,400	AMAZON
16	2023-11-06	130.22	134.27	129.93	134.06	134.06	88,527,200	GOOGLE
17	2023-11-06	315.98	329.1	314.45	328.77	328.77	75,752,300	META
18	2023-11-06	176.38	186.57	176.21	186.4	186.155	303,608,500	APPLE
19	2023-11-06	434.38	447.48	429.61	447.24	447.24	15,827,200	NETFLIX
20	2023-11-06	138.76	143.65	138.36	143.56	143.56	228,568,800	AMAZON
21	2023-10-30	299.09	318.82	296.86	314.6	314.6	106,689,800	META
22	2023-10-30	169.02	177.78	167.9	176.65	176.418	310,010,400	APPLE
23	2023-10-30	129.72	139.49	128.56	138.6	138.6	281,848,200	AMAZON
24	2023-10-30	402.35	434.82	399.41	432.36	432.36	22,141,600	NETFLIX
25	2023-10-30	124.46	130.73	123.88	130.37	130.37	115,435,200	GOOGLE
26	2023-10-23	309.5	318.35	279.4	296.73	296.73	175,795,200	META

Monthly data:

Database Navigator: dwdi-cloud-db, public, Tables, monthly

SQL Editor: select * from monthly m;

monthly 1 x

	date	open	high	low	close	adj_close	volume	company
1	2023-11-01	414.77	482.7	414.18	479.17	479.17	61,706,300	NETFLIX
2	2023-11-01	125.34	141.1	124.925	138.05	138.05	337,642,400	GOOGLE
3	2023-11-01	301.85	342.92	301.85	334.7	334.7	277,424,200	META
4	2023-11-01	133.96	149.26	133.71	147.73	147.73	876,754,600	AMAZON
5	2023-11-01	171	192.93	170.12	189.79	189.54	969,310,000	APPLE
6	2023-10-01	377.48	418.84	344.73	411.69	411.69	164,021,900	NETFLIX
7	2023-10-01	132.155	142.38	121.46	125.3	125.3	514,877,100	GOOGLE
8	2023-10-01	302.74	330.54	279.4	301.27	301.27	511,307,900	META
9	2023-10-01	171.22	182.34	165.67	170.77	170.545	1,172,719,600	APPLE
10	2023-10-01	127.28	134.48	118.35	133.09	133.09	1,224,564,700	AMAZON
11	2023-09-01	139.46	145.86	123.04	127.12	127.12	1,120,271,900	AMAZON
12	2023-09-01	189.49	189.98	167.62	171.21	170.985	1,337,586,600	APPLE
13	2023-09-01	437.73	453.45	371.1	377.6	377.6	100,278,600	NETFLIX
14	2023-09-01	299.37	312.87	286.79	300.21	300.21	406,686,600	META
15	2023-09-01	138.43	139.93	128.19	131.85	131.85	389,593,900	GOOGLE
16	2023-08-01	133.55	143.63	126.41	138.01	138.01	1,210,426,200	AMAZON
17	2023-08-01	437.37	445.25	398.15	433.68	433.68	107,298,900	NETFLIX
18	2023-08-01	317.54	324.14	274.38	295.89	295.89	423,147,800	META
19	2023-08-01	130.855	138.4	127	137.35	137.35	463,482,000	GOOGLE
20	2023-08-01	196.24	196.73	171.96	187.87	187.37	1,322,439,400	APPLE
21	2023-07-01	193.78	198.23	186.6	196.45	195.927	996,066,400	APPLE
22	2023-07-01	120.32	134.07	115.83	133.11	133.11	525,456,900	GOOGLE
23	2023-07-01	286.7	326.2	284.85	318.6	318.6	624,605,100	META
24	2023-07-01	130.82	136.65	125.92	133.68	133.68	1,058,754,800	AMAZON
25	2023-07-01	439.76	485	411.88	438.97	438.97	168,720,200	NETFLIX

Data Ingestion into BigQuery:

Now the updated data in postgres will be inserted into BigQuery using DataStream.

The screenshot shows the 'Stream details' page for a Datastream named 'postgrestobq'. The stream is of type 'PostgreSQL/BigQuery'. A notification at the top indicates that 1,512 events were unsupported and not loaded within the last 7 days. The 'Properties' section lists various configuration details.

Property	Value
Stream ID	postgrestobq
Source profile	pgconnection
Destination profile	dsbigqueryconnection
Created	22 Nov 2023, 17:09:10
Updated	30 Nov 2023, 18:37:58

Property	Value
Region	us-central1 (Iowa)
Labels	No labels set
Objects to include	1 schema
Objects to exclude	None
Backfill mode	Automatic
Destination data set	MAANGdata
Staleness limit	5 minutes
Encryption	Google-managed
Tags	None

Below is the connection profile in DataStream:

The screenshot shows the 'Connection profiles' page in Datastream. It lists two connection profiles: 'dsbigqueryconnection' and 'pgconnection'. The table below provides details for each profile.

Profile name	Activity	Connection profile ID	Type	Connection details	Created	Region	Labels
dsbigqueryconnection	1 stream	dsbigqueryconnection	BigQuery	—	22 Nov 2023	us-central1 (Iowa)	
pgconnection	1 stream	pgconnection	PostgreSQL	35.224.2.253 : dwdi-cloud-db	22 Nov 2023	us-central1 (Iowa)	

In order to capture the changes in the postgres database, we implemented DataStream to capture the incremental load in data in BigQuery.

Daily:

IE7374 Data Warehousing and Data Integra5on

Google Cloud | My First Project | Search (/) for resources, docs, products and more

BigQuery Explorer

public_daily

VIEWING RESOURCES. SHOW STARRED ONLY

alien-grove-405422

public_daily

Summary

public_daily

alien-grove-405422.MAANGdata

Last modified: 29 Nov 2023, 23:32:21 UTC-5

Data location: us-central1

Row	date	open	high	low	close
1	2013-12-18	19.632143020629883	19.694643020629883	19.24285697937012	19.67035675048828
2	2014-07-31	28.950515747070312	29.102598190307617	28.42196655273437	28.501747131347656
3	2013-12-19	19.625	19.64285659790039	19.418928146362305	19.44499969482422
4	1997-05-15	0.1218750029802322	0.125	0.0963540002703666	0.0979169979691505
5	2014-08-01	28.441913604736328	28.719152450561523	28.06544685363769	28.22600555419922
6	1994-02-09	0.3191959857940674	0.3258930146694183	0.3147319853305816	0.3236609995365143
7	2017-03-29	42.95249938964844	43.821998596191406	42.95100021362305	43.71599960327149
8	2022-12-09	115.3000030517578	117.5400091552734	113.87000274658205	115.9000015258789
9	2013-12-20	19.479642868041992	19.70035743713379	19.457857131958008	19.60785675048828
10	1997-05-16	0.0984380021691322	0.0989580005407333	0.0854170024394989	0.0864579975605011
11	2014-08-04	28.374099731445312	28.68873405456543	28.127775192260746	28.579036712646484
12	1994-02-10	0.3236609995365143	0.3348209857940674	0.3214290142059326	0.3258930146694183
13	2017-03-30	43.747501373291016	43.85300064086914	43.58300018310547	43.81700134277344
14	2022-12-12	115.18000030517578	115.72000122070312	113.13999938964844	114.70999908447266
15	2013-12-23	20.28571319580078	20.382856369018555	20.09857177734375	20.3603572845459
16	1997-05-19	0.0880210027098655	0.0885419994592666	0.0812499970197677	0.0854170024394989
17	2014-08-05	28.424461364746094	28.52069664001465	28.0534782409668	28.1761417388916
18	1994-02-11	0.3236609995365143	0.3348209857940674	0.3236609995365143	0.330356985305816

Results per page: 50 | 1 - 50 of 30676

Job history

Weekly:

Google Cloud | My First Project | Search (/) for resources, docs, products and more

BigQuery Explorer

public_weekly

VIEWING RESOURCES. SHOW STARRED ONLY

alien-grove-405422

public_weekly

Summary

public_weekly

alien-grove-405422.MAANGdata

Last modified: 29 Nov 2023, 23:31:57 UTC-5

Data location: us-central1

Row	date	open	high	low	close
1	1997-05-12	0.1218750029802322	0.125	0.0854170024394989	0.0864579975605011
2	1997-05-19	0.0880210027098655	0.0885419994592666	0.0656249970197677	0.0750000029802322
3	1997-05-26	0.075520997296333	0.0822919979691505	0.0729169994592666	0.0750000029802322
4	1997-06-02	0.075520997296333	0.0854170024394989	0.0687500014901161	0.0828130021691322
5	1997-06-09	0.0828130021691322	0.0854170024394989	0.0765630006790161	0.0791670009493827
6	1997-06-16	0.0802080035209655	0.0802080035209655	0.074740000691413	0.0763019993901252
7	1997-06-23	0.0770829990506172	0.0770829990506172	0.0739580020308494	0.0744789987802505
8	1997-06-30	0.075520997296333	0.0958330035209655	0.0739580020308494	0.0955730006098747
9	1997-07-07	0.0916669964790344	0.1286460012197494	0.0916669964790344	0.1145830005407333
10	1997-07-14	0.1161459982395172	0.1247399970889091	0.10572899878025051	0.1078130006790161
11	1997-07-21	0.1088540032509589	0.1166670024394989	0.1031249985098838	0.1114580035209655
12	1997-07-28	0.1114580035209655	0.1252599954605102	0.1109379976987838	0.12083300203084939
13	1997-08-04	0.1187499985098838	0.1205729991197586	0.10520800203084939	0.1145830005407333
14	1997-08-11	0.1145830005407333	0.1166670024394989	0.0968749970197677	0.10572899878025051
15	1997-08-18	0.10260400176048272	0.1104170009493827	0.09843800216913219	0.1062500029802322
16	1997-08-25	0.10520800203084939	0.11979199945926661	0.10520800203084939	0.1169269979000091
17	1997-09-01	0.11718799918890001	0.1333329975605011	0.11562500149011609	0.125
18	1997-09-08	0.1265629976987838	0.1848960071802139	0.125	0.1843750029802322

Results per page: 50 | 1 - 50 of 6363

Job history

Monthly:

Google Cloud BigQuery Explorer interface showing the 'public_monthly' table. The table is located under the 'MAANGdata' dataset. The table schema and data are as follows:

Row	volume	company	datastream_metadata.uid	data_source_timestamp	is_deleted
18	13841858	AMAZON	77f4cdc0-896f-4a7f-b586-35a9...	1700691122375	false
19	54418945	AMAZON	77f4cdc0-896f-4a7f-b586-35a9...	1700691122375	false
20	220459	AMAZON	77f4cdc0-896f-4a7f-b586-35a9...	1700691122375	false
21	6362920000	AMAZON	77f4cdc0-896f-4a7f-b586-35a9...	1700691122375	false
22	6837158	AMAZON	77f4cdc0-896f-4a7f-b586-35a9...	1700691122375	false
23	16102295	AMAZON	77f4cdc0-896f-4a7f-b586-35a9...	1700691122375	false
24	6075356000	AMAZON	77f4cdc0-896f-4a7f-b586-35a9...	1700691122375	false
25	12316284	AMAZON	77f4cdc0-896f-4a7f-b586-35a9...	1700691122375	false
26	220459	AMAZON	77f4cdc0-896f-4a7f-b586-35a9...	1700691122375	false
27	9181716000	AMAZON	77f4cdc0-896f-4a7f-b586-35a9...	1700691122375	false
28	17683716	AMAZON	77f4cdc0-896f-4a7f-b586-35a9...	1700691122375	false
29	5086230000	AMAZON	77f4cdc0-896f-4a7f-b586-35a9...	1700691122375	false
30	10734863	AMAZON	77f4cdc0-896f-4a7f-b586-35a9...	1700691122375	false
31	15367432	AMAZON	77f4cdc0-896f-4a7f-b586-35a9...	1700691122375	false
32	15367432	AMAZON	77f4cdc0-896f-4a7f-b586-35a9...	1700691122375	false
33	146325684	AMAZON	77f4cdc0-896f-4a7f-b586-35a9...	1700691122375	false
34	146325684	AMAZON	77f4cdc0-896f-4a7f-b586-35a9...	1700691122375	false

Anal&cal Queries:

1. Calculate the average closing price for last week for Apple:

Google Cloud BigQuery Explorer interface showing a query to calculate the average closing price for Apple for the last week. The query is executed, and the results are displayed below.

```

1 SELECT company, AVG(CAST(close AS FLOAT64)) AS average_closing_price
2 FROM `alien-grove-405422.MAANGdata.public_weekly`
3 WHERE company = 'APPLE'
4 AND date >= DATE_SUB(CURRENT_DATE(), INTERVAL EXTRACT(DAYOFWEEK FROM CURRENT_DATE()) + 6 DAY)
5 AND date < DATE_SUB(CURRENT_DATE(), INTERVAL EXTRACT(DAYOFWEEK FROM CURRENT_DATE()) - 1 DAY)
6 group by 1;
7

```

Query results:


Row	company	average_closing_price
1	APPLE	189.9700012207...

2. What is the opening stock price for Meta in the month of October?


```

16
17 SELECT company, open
18 FROM `alien-grove-405422.MAANGdata.public_monthly`
19 WHERE company = 'META'
20 AND EXTRACT(MONTH FROM date) = 10
21 AND EXTRACT(YEAR FROM date) = EXTRACT(YEAR FROM CURRENT_DATE());
22

```

Query results 

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUTION
Row	company	open				
1	META	302.739990234375				

3. Which is the lowest trading price and company in the month of November?

```


14
15 WITH MonthlyLowestPrices AS (
16   SELECT company, MIN(low) AS lowest_trading_price
17   FROM `alien-grove-405422.MAANGdata.public_monthly`
18   WHERE EXTRACT(MONTH FROM date) = 11
19   AND EXTRACT(YEAR FROM date) = EXTRACT(YEAR FROM CURRENT_DATE())
20   GROUP BY company
21 )
22
23 SELECT company, lowest_trading_price
24 FROM MonthlyLowestPrices
25 ORDER BY lowest_trading_price ASC
26 LIMIT 1;
27
28

```

Query results

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON	EXECUT
Row	company	lowest_trading_price				
1	GOOGLE	124.9250030517578				

4. On the 27th of November, what were the top 3 stocks sold and by which company?



Query Editor Interface showing a SQL query to find the top 3 stocks sold on November 27, 2023. The query is as follows:

```
30
31 SELECT company, SUM(volume) AS total_volume
32 FROM `alien-grove-405422.MAANGdata.public_daily`
33 WHERE DATE(date) = DATE('2023-11-27')
34 GROUP BY company
35 ORDER BY total_volume DESC
36 LIMIT 3;
37
38
39
40
41
42
43
44
```

Query results

JOB INFORMATION		RESULTS	CHART	PREVIEW	JSON
Row	company	total_volume			
1	AMAZON	53666700			
2	APPLE	40500500			
3	GOOGLE	17868000			