HW2

1. The code for this Streaming Analytics is:

```
import finnhub
import time
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, DoubleType, LongType, TimestampType, IntegerType
from pyspark.sql import Row
from pyspark.sql.functions import mean
from pyspark.sql.functions import date_format
import datetime
```

```
spark = SparkSession.builder.appName("StockData").getOrCreate()
finnhub_client = finnhub.Client(api_key="ckmap1hr01qu6et565b0ckmap1hr01qu6et565bg")
# define datafrme
schema = StructType([
    StructField("Stock Name", StringType(), True),
    StructField("UTC Timestamp", TimestampType(), True),
    StructField("c", DoubleType(), True),
    StructField("l", DoubleType(), True),
    StructField("h", DoubleType(), True),
    StructField("o", DoubleType(), True),
    StructField("v", DoubleType(), True)
])
schema_MA = StructType([
    StructField("Stock", StringType(), True),
    StructField("Datetime", TimestampType(), True),
    StructField("c_MA", DoubleType(), True),
    StructField("l_MA", DoubleType(), True),
    StructField("h_MA", DoubleType(), True),
    StructField("o_MA", DoubleType(), True),
    StructField("v_MA", DoubleType(), True)
])


total_runtime = 30 * 60

api_call_interval = 5 * 60

total_api_calls = total_runtime // api_call_interval

end_time = int(time.time())
start_time = end_time - 3600
```

```
for m in range(total_api_calls):

    data = finnhub_client.stock_candles('AAPL', '1', start_time, end_time)
    stock_name = "APPL"
    timestamps = data["t"]
    closes = data["c"]
    lows = data["l"]
    highs = data["h"]
    opens = data["o"]
    volumes = data["v"]
    rows=[]
    for i in range(len(timestamps)):
        row = Row("stock_name", "timestamp", "c", "l", "h", "o", "v")(stock_name, datetime.datetime.utcfromtimestamp(timestamps[i]), :
        rows.append(row)

    if m==0:
        df = spark.createDataFrame(rows, schema)
    else:
        df1=spark.createDataFrame(rows, schema)
        difer=df1.join(df, df1["UTC Timestamp"]==df["UTC Timestamp"], "left_anti")
        df=df.union(difer)

    sorted_df = df.orderBy("UTC Timestamp")
    sorted_df.show(10)
    lastrows = sorted_df.tail(10)
    lastdf=spark.createDataFrame(lastrows)
    lastdf.show()

    mean_values = df.agg(mean("c").alias("avg_c"),
                mean("l").alias("avg_l"),
                mean("h").alias("avg_h"),
                mean("o").alias("avg_o"),
                mean("v").alias("avg_v")).collect()[0]

    row_MA = Row(stock_name="APPL",
             timestamp=datetime.datetime.utcfromtimestamp(end_time),
             c=mean_values.avg_c,
             l=mean_values.avg_l,
             h=mean_values.avg_h,
             o=mean_values.avg_o,
             v=mean_values.avg_v)
```

```
    if m==0:
        df_MA = spark.createDataFrame([row_MA], schema_MA)
    else:
        df1_MA=spark.createDataFrame([row_MA], schema_MA)
        df_MA=df_MA.union(df1_MA)

    df_MA.show()
    time.sleep(api_call_interval)

    end_time = int(time.time())
    start_time = end_time - 3600
```

For every read from finnhub, df1 stores the newest data. df stores the processed data. By using the “leftanti join” can find the differences between df and df1, i.e., the data generated in the past 5 minutes. After that, these data are unionid to df to finish processing. df_MA stores the averages.

The output of this program is (the second dataframe displays the last 10 rows of the first dataframe:

Stock Name	UTC Timestamp	c	l	h	o	v
APPL	2023-10-17 17:40:00	177.645	177.5801	177.7	177.68	57960.0
APPL	2023-10-17 17:41:00	177.65	177.645	177.71	177.65	46743.0
APPL	2023-10-17 17:42:00	177.65	177.63	177.71	177.65	54681.0
APPL	2023-10-17 17:43:00	177.6199	177.5225	177.66	177.65	101147.0
APPL	2023-10-17 17:44:00	177.7499	177.61	177.7503	177.6199	72581.0
APPL	2023-10-17 17:45:00	177.7601	177.72	177.7868	177.75	57735.0
APPL	2023-10-17 17:46:00	177.64	177.635	177.77	177.7603	63343.0
APPL	2023-10-17 17:47:00	177.51	177.48	177.65	177.64	88912.0
APPL	2023-10-17 17:48:00	177.57	177.481	177.57	177.51	132862.0
APPL	2023-10-17 17:49:00	177.54	177.5	177.58	177.5601	63155.0

only showing top 10 rows

Stock Name	UTC Timestamp	c	l	h	o	v
APPL	2023-10-17 18:30:00	177.0518	177.0	177.1	177.09	84338.0
APPL	2023-10-17 18:31:00	177.0411	177.0	177.08	177.055	64495.0
APPL	2023-10-17 18:32:00	177.12	176.9329	177.16	177.0582	163783.0
APPL	2023-10-17 18:33:00	177.2489	177.1071	177.265	177.1272	119272.0
APPL	2023-10-17 18:34:00	177.2	177.195	177.2462	177.2218	41299.0
APPL	2023-10-17 18:35:00	177.26	177.19	177.29	177.1913	50780.0
APPL	2023-10-17 18:36:00	177.1	177.085	177.26	177.26	71394.0
APPL	2023-10-17 18:37:00	177.25	177.095	177.28	177.095	65026.0
APPL	2023-10-17 18:38:00	177.33	177.2201	177.34	177.24	81648.0
APPL	2023-10-17 18:39:00	177.3	177.26	177.34	177.3205	54224.0

Stock	Datetime	c_MA	l_MA	h_MA	o_MA	v_MA
APPL	2023-10-17 18:39:07	177.31362500000003	177.2449566666667	177.37991666666665	177.31804833333337	92189.33333333333

Stock Name	UTC Timestamp	c	l	h	o	v
APPL	2023-10-17 17:40:00	177.645	177.5801	177.7	177.68	57960.0
APPL	2023-10-17 17:41:00	177.65	177.645	177.71	177.65	46743.0
APPL	2023-10-17 17:42:00	177.65	177.63	177.71	177.65	54681.0
APPL	2023-10-17 17:43:00	177.6199	177.5225	177.66	177.65	101147.0
APPL	2023-10-17 17:44:00	177.7499	177.61	177.7503	177.6199	72581.0
APPL	2023-10-17 17:45:00	177.7601	177.72	177.7868	177.75	57735.0
APPL	2023-10-17 17:46:00	177.64	177.635	177.77	177.7603	63343.0
APPL	2023-10-17 17:47:00	177.51	177.48	177.65	177.64	88912.0
APPL	2023-10-17 17:48:00	177.57	177.481	177.57	177.51	132862.0
APPL	2023-10-17 17:49:00	177.54	177.5	177.58	177.5601	63155.0

only showing top 10 rows

Stock Name	UTC Timestamp	c	l	h	o	v
APPL	2023-10-17 18:35:00	177.26	177.19	177.29	177.1913	50780.0
APPL	2023-10-17 18:36:00	177.1	177.085	177.26	177.26	71394.0
APPL	2023-10-17 18:37:00	177.25	177.095	177.28	177.095	65026.0
APPL	2023-10-17 18:38:00	177.33	177.2201	177.34	177.24	81648.0
APPL	2023-10-17 18:39:00	177.3	177.26	177.34	177.3205	54224.0
APPL	2023-10-17 18:40:00	177.29	177.26	177.3383	177.3	57686.0
APPL	2023-10-17 18:41:00	177.2581	177.25	177.3191	177.29	64454.0
APPL	2023-10-17 18:42:00	177.1151	177.04	177.28	177.245	130520.0
APPL	2023-10-17 18:43:00	177.04	177.01	177.1299	177.11	79474.0
APPL	2023-10-17 18:44:00	177.05	177.0245	177.09	177.05	49045.0

Stock	Datetime	c_MA	l_MA	h_MA	o_MA	v_MA
APPL	2023-10-17 18:39:07	177.31362500000003	177.2449566666667	177.37991666666665	177.31804833333337	92189.33333333333
APPL	2023-10-17 18:44:17	177.3010876923077	177.23510615384617	177.36849692307692	177.3088907692308	90962.13846153846

Stock Name	UTC Timestamp	c	l	h	o	v
APPL	2023-10-17 17:40:00	177.645	177.5801	177.7	177.68	57960.0
APPL	2023-10-17 17:41:00	177.65	177.645	177.71	177.65	46743.0
APPL	2023-10-17 17:42:00	177.65	177.63	177.71	177.65	54681.0
APPL	2023-10-17 17:43:00	177.6199	177.5225	177.66	177.65	101147.0
APPL	2023-10-17 17:44:00	177.7499	177.61	177.7503	177.6199	72581.0
APPL	2023-10-17 17:45:00	177.7601	177.72	177.7868	177.75	57735.0
APPL	2023-10-17 17:46:00	177.64	177.635	177.77	177.7603	63343.0
APPL	2023-10-17 17:47:00	177.51	177.48	177.65	177.64	88912.0
APPL	2023-10-17 17:48:00	177.57	177.481	177.57	177.51	132862.0
APPL	2023-10-17 17:49:00	177.54	177.5	177.58	177.5601	63155.0

only showing top 10 rows

Stock Name	UTC Timestamp	c	l	h	o	v
APPL	2023-10-17 18:40:00	177.29	177.26	177.3383	177.3	57686.0
APPL	2023-10-17 18:41:00	177.2581	177.25	177.3191	177.29	64454.0
APPL	2023-10-17 18:42:00	177.1151	177.04	177.28	177.245	130520.0
APPL	2023-10-17 18:43:00	177.04	177.01	177.1299	177.11	79474.0
APPL	2023-10-17 18:44:00	177.05	177.0245	177.09	177.05	49045.0
APPL	2023-10-17 18:45:00	177.04	176.971	177.128	177.05	73572.0
APPL	2023-10-17 18:46:00	177.05	177.0199	177.08	177.05	44040.0
APPL	2023-10-17 18:47:00	177.09	177.01	177.14	177.04	103628.0
APPL	2023-10-17 18:48:00	177.05	177.02	177.13	177.08	96503.0
APPL	2023-10-17 18:49:00	177.0789	177.03	177.13	177.06	42903.0

Stock	Datetime	c_MA	l_MA	h_MA	o_MA	v_MA
APPL	2023-10-17 18:39:07	177.31362500000003	177.2449566666667	177.37991666666665	177.31804833333337	92189.33333333333
APPL	2023-10-17 18:44:17	177.3010876923077	177.23510615384617	177.36849692307692	177.3088907692308	90962.13846153846
APPL	2023-10-17 18:49:24	177.2839942857143	177.21904000000004	177.35086142857142	177.29082714285718	89616.92857142857

Stock Name	UTC Timestamp	c	l	h	o	v
APPL	2023-10-17 17:40:00	177.645	177.5801	177.7	177.68	57960.0
APPL	2023-10-17 17:41:00	177.65	177.645	177.71	177.65	46743.0
APPL	2023-10-17 17:42:00	177.65	177.63	177.71	177.65	54681.0
APPL	2023-10-17 17:43:00	177.6199	177.5225	177.66	177.65	101147.0
APPL	2023-10-17 17:44:00	177.7499	177.61	177.7503	177.6199	72581.0
APPL	2023-10-17 17:45:00	177.7601	177.72	177.7868	177.75	57735.0
APPL	2023-10-17 17:46:00	177.64	177.635	177.77	177.7603	63343.0
APPL	2023-10-17 17:47:00	177.51	177.48	177.65	177.64	88912.0
APPL	2023-10-17 17:48:00	177.57	177.481	177.57	177.51	132862.0
APPL	2023-10-17 17:49:00	177.54	177.5	177.58	177.5601	63155.0

only showing top 10 rows

Stock Name	UTC Timestamp	c	l	h	o	v
APPL	2023-10-17 18:45:00	177.04	176.971	177.128	177.05	73572.0
APPL	2023-10-17 18:46:00	177.05	177.0199	177.08	177.05	44040.0
APPL	2023-10-17 18:47:00	177.09	177.01	177.14	177.04	103628.0
APPL	2023-10-17 18:48:00	177.05	177.02	177.13	177.08	96503.0
APPL	2023-10-17 18:49:00	177.0789	177.03	177.13	177.06	42903.0
APPL	2023-10-17 18:50:00	176.81	176.56	177.08	177.08	501015.0
APPL	2023-10-17 18:51:00	176.8	176.7301	176.825	176.8	141101.0
APPL	2023-10-17 18:52:00	176.73	176.64	176.8143	176.809	107578.0
APPL	2023-10-17 18:53:00	176.755	176.69	176.8	176.7299	78776.0
APPL	2023-10-17 18:54:00	176.49	176.45	176.76	176.75	210169.0

Stock	Datetime	c_MA	l_MA	h_MA	o_MA	v_MA
APPL	2023-10-17 18:39:07	177.31362500000003	177.2449566666667	177.37991666666665	177.31804833333337	92189.33333333333
APPL	2023-10-17 18:44:17	177.3010876923077	177.23510615384617	177.36849692307692	177.3088907692308	90962.13846153846
APPL	2023-10-17 18:49:24	177.2839942857143	177.21904000000004	177.35086142857142	177.29082714285718	89616.92857142857
APPL	2023-10-17 18:54:32	177.24619466666667	177.17870533333334	177.31786133333333	177.2603573333334	97490.98666666666

Stock Name	UTC Timestamp	c	l	h	o	v
APPL	2023-10-17 17:40:00	177.645	177.5801	177.7	177.68	57960.0
APPL	2023-10-17 17:41:00	177.65	177.645	177.71	177.65	46743.0
APPL	2023-10-17 17:42:00	177.65	177.63	177.71	177.65	54681.0
APPL	2023-10-17 17:43:00	177.6199	177.5225	177.66	177.65	101147.0
APPL	2023-10-17 17:44:00	177.7499	177.61	177.7503	177.6199	72581.0
APPL	2023-10-17 17:45:00	177.7601	177.72	177.7868	177.75	57735.0
APPL	2023-10-17 17:46:00	177.64	177.635	177.77	177.7603	63343.0
APPL	2023-10-17 17:47:00	177.51	177.48	177.65	177.64	88912.0
APPL	2023-10-17 17:48:00	177.57	177.481	177.57	177.51	132862.0
APPL	2023-10-17 17:49:00	177.54	177.5	177.58	177.5601	63155.0

only showing top 10 rows

Stock Name	UTC Timestamp	c	l	h	o	v
APPL	2023-10-17 18:50:00	176.81	176.56	177.08	177.08	501015.0
APPL	2023-10-17 18:51:00	176.8	176.7301	176.825	176.8	141101.0
APPL	2023-10-17 18:52:00	176.73	176.64	176.8143	176.809	107578.0
APPL	2023-10-17 18:53:00	176.755	176.69	176.8	176.7299	78776.0
APPL	2023-10-17 18:54:00	176.49	176.45	176.76	176.75	210169.0
APPL	2023-10-17 18:55:00	176.455	176.365	176.48	176.48	175835.0
APPL	2023-10-17 18:56:00	176.65	176.46	176.66	176.46	155712.0
APPL	2023-10-17 18:57:00	176.58	176.5057	176.66	176.66	107299.0
APPL	2023-10-17 18:58:00	176.6	176.52	176.62	176.565	123753.0
APPL	2023-10-17 18:59:00	176.62	176.55	176.67	176.59	78594.0

Stock	Datetime	c_MA	l_MA	h_MA	o_MA	v_MA
APPL	2023-10-17 18:39:07	177.31362500000003	177.2449566666667	177.37991666666665	177.31804833333337	92189.33333333333
APPL	2023-10-17 18:44:17	177.3010876923077	177.23510615384617	177.36849692307692	177.3088907692308	90962.13846153846
APPL	2023-10-17 18:49:24	177.2839942857143	177.21904000000004	177.35086142857142	177.29082714285718	89616.92857142857
APPL	2023-10-17 18:54:32	177.24619466666667	177.17870533333334	177.31786133333333	177.2603573333334	97490.98666666666
APPL	2023-10-17 18:59:42	177.20462	177.13504500000002	177.27411999999998	177.21602250000007	99412.7125

2. Airflow Data Pipelining

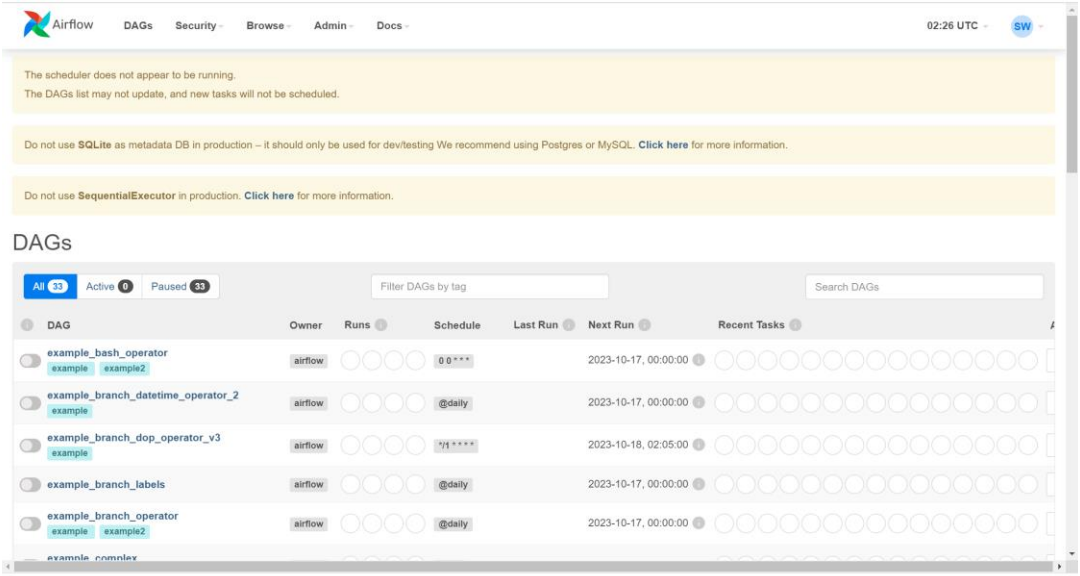
Task1

Q1.1 (1)

```
(airflow) sw3828@instance-test:~$ airflow webserver --port 8080
/home/sw3828/miniconda3/envs/airflow/lib/python3.8/site-packages/airflow/configuration.py:276: DeprecationWarning: d
istutils Version classes are deprecated. Use packaging.version instead.
  if StrictVersion(sqlite3.sqlite_version) < StrictVersion(min_sqlite_version):
  ____________       _____________
 ____    |__( )_________  __/__  /________      __
____  /| |_  /__  ___/_  /_ __  /_  __ \_ | /| / /
___  ___ |  / _  /   _  __/ _  / / /_/ /_ |/ |/ /
 _/_/  |_/_/  /_/    /_/    /_/  \____/____/|__/
[2023-10-18 02:15:52,219] {dagbag.py:500} INFO - Filling up the DagBag from /dev/null
[2023-10-18 02:15:52,416] {manager.py:512} WARNING - Refused to delete permission view, assoc with role exists DAG R
uns.can_create Admin
Running the Gunicorn Server with:
Workers: 4 sync
Host: 0.0.0.0:8080
Timeout: 120
Logfiles: - -
Access Logformat:
=================================================================
/home/sw3828/miniconda3/envs/airflow/lib/python3.8/site-packages/airflow/configuration.py:276: DeprecationWarning: d
istutils Version classes are deprecated. Use packaging.version instead.
  if StrictVersion(sqlite3.sqlite_version) < StrictVersion(min_sqlite_version):
[2023-10-18 02:15:54 +0000] [16158] [INFO] Starting gunicorn 20.1.0
[2023-10-18 02:15:54 +0000] [16158] [INFO] Listening at: http://0.0.0.0:8080 (16158)
[2023-10-18 02:15:54 +0000] [16158] [INFO] Using worker: sync
[2023-10-18 02:15:54 +0000] [16160] [INFO] Booting worker with pid: 16160
[2023-10-18 02:15:54 +0000] [16161] [INFO] Booting worker with pid: 16161
[2023-10-18 02:15:54 +0000] [16162] [INFO] Booting worker with pid: 16162
[2023-10-18 02:15:54 +0000] [16163] [INFO] Booting worker with pid: 16163
[2023-10-18 02:15:56,478] {manager.py:512} WARNING - Refused to delete permission view, assoc with role exists DAG R
uns.can_create Admin
[2023-10-18 02:15:56,823] {manager.py:512} WARNING - Refused to delete permission view, assoc with role exists DAG R
uns.can_create Admin
[2023-10-18 02:15:56,898] {manager.py:512} WARNING - Refused to delete permission view, assoc with role exists DAG R
uns.can_create Admin
[2023-10-18 02:15:56,941] {manager.py:512} WARNING - Refused to delete permission view, assoc with role exists DAG R
uns.can_create Admin
151.205.164.190 - - [18/Oct/2023:02:18:55 +0000] "GET / HTTP/1.1" 302 217 "-" "Mozilla/5.0 (Windows NT 10.0; Win64;
```

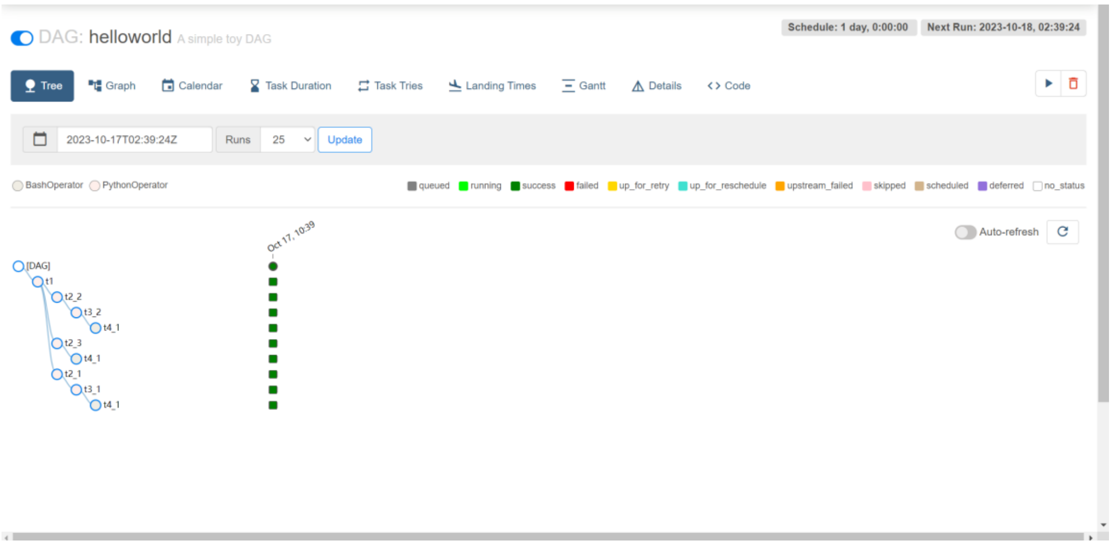
```
New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


*** System restart required ***
Last login: Wed Oct 18 01:57:35 2023 from 35.235.244.2
(base) sw3828@instance-test:~$ conda activate airflow
(airflow) sw3828@instance-test:~$ airflow db init
/home/sw3828/miniconda3/envs/airflow/lib/python3.8/site-packages/airflow/configuration.py:276: DeprecationWarnin
g: distutils Version classes are deprecated. Use packaging.version instead.
  if StrictVersion(sqlite3.sqlite_version) < StrictVersion(min_sqlite_version):
DB: sqlite:////home/sw3828/airflow/airflow.db
[2023-10-18 02:24:13,082] {db.py:867} INFO - Creating tables
INFO  [alembic.runtime.migration] Context impl SQLiteImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
WARNI [airflow.models.crypto] empty cryptography key - values will not be stored encrypted.
Initialization done
(airflow) sw3828@instance-test:~$ airflow scheduler
/home/sw3828/miniconda3/envs/airflow/lib/python3.8/site-packages/airflow/configuration.py:276: DeprecationWarnin
g: distutils Version classes are deprecated. Use packaging.version instead.
  if StrictVersion(sqlite3.sqlite_version) < StrictVersion(min_sqlite_version):
  ____________       _____________
 ____    |__( )_________  __/__  /________      __
____  /| |_  /__  ___/_  /_ __  /_  __ \_ | /| / /
___  ___ |  / _  /   _  __/ _  / / /_/ /_ |/ |/ /
 _/_/  |_/_/  /_/    /_/    /_/  \____/____/|__/
[2023-10-18 02:24:21,667] {scheduler_job.py:596} INFO - Starting the scheduler
[2023-10-18 02:24:21,668] {scheduler_job.py:601} INFO - Processing each file at most -1 times
[2023-10-18 02:24:21,672] {manager.py:163} INFO - Launched DagFileProcessorManager with pid: 16350
[2023-10-18 02:24:21,673] {scheduler_job.py:1115} INFO - Resetting orphaned tasks for active dag runs
[2023-10-18 02:24:21 +0000] [16349] [INFO] Starting gunicorn 20.1.0
[2023-10-18 02:24:21 +0000] [16349] [INFO] Listening at: http://0.0.0.0:8793 (16349)
[2023-10-18 02:24:21 +0000] [16349] [INFO] Using worker: sync
[2023-10-18 02:24:21,680] {settings.py:52} INFO - Configured default timezone Timezone('UTC')
[2023-10-18 02:24:21 +0000] [16351] [INFO] Booting worker with pid: 16351
[2023-10-18 02:24:21,701] {manager.py:431} WARNING - Because we cannot use more than 1 thread (parsing_processes
 = 2 ) when using sqlite. So we set parallelism to 1.
[2023-10-18 02:24:21 +0000] [16353] [INFO] Booting worker with pid: 16353
```

Q1.1(2)

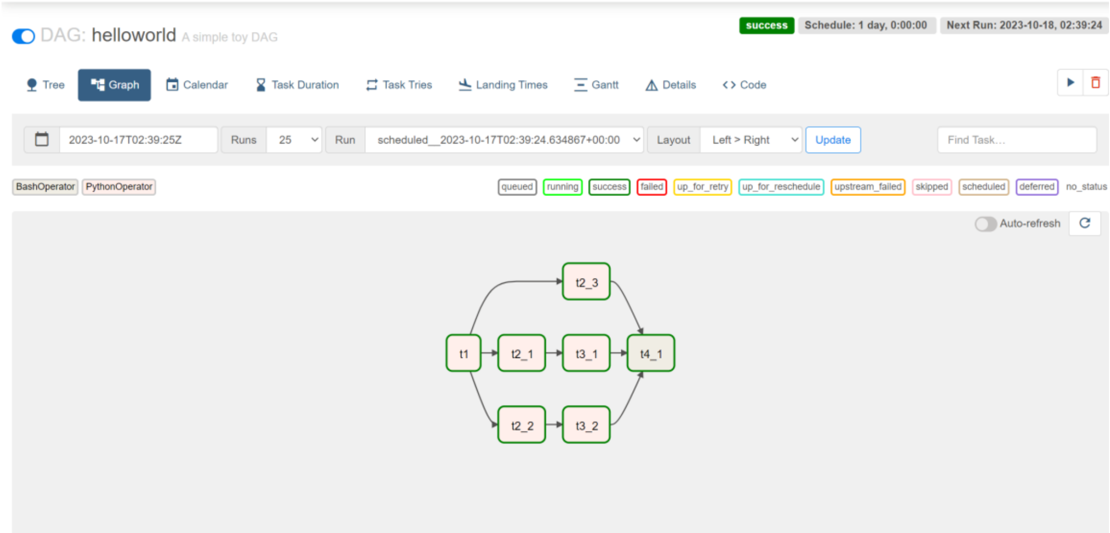


Airflow
DAGs
Security
Browse
Admin
Docs
02:26 UTC
SW
The scheduler does not appear to be running.
The DAGs list may not update, and new tasks will not be scheduled.
Do not use SQLite as metadata DB in production – it should only be used for dev/testing We recommend using Postgres or MySQL. Click here for more information.
Do not use SequentialExecutor in production. Click here for more information.
DAGs
All 33
Active 0
Paused 33
Filter DAGs by tag
Search DAGs
DAG
Owner
Runs
Schedule
Last Run
Next Run
Recent Tasks
example_bash_operator
example
example2
airflow
0 0 * * *
2023-10-17, 00:00:00
example_branch_datetime_operator_2
example
airflow
@daily
2023-10-17, 00:00:00
example_branch_dop_operator_v3
example
airflow
*/1 * * * *
2023-10-18, 02:05:00
example_branch_labels
airflow
@daily
2023-10-17, 00:00:00
example_branch_operator
example
example2
airflow
@daily
2023-10-17, 00:00:00

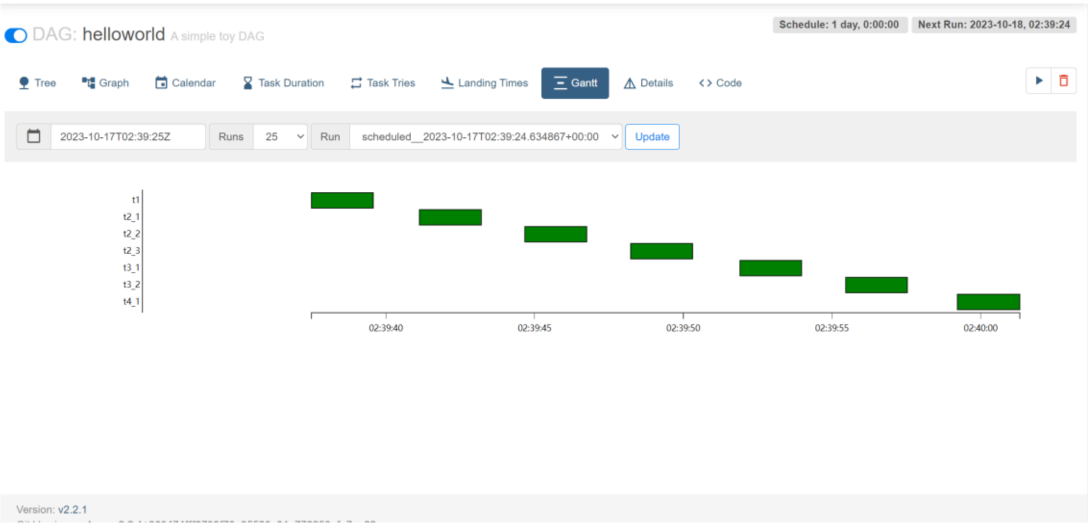
Q1.2(1)



DAG: helloworld
A simple toy DAG
Schedule: 1 day, 0:00:00
Next Run: 2023-10-18, 02:39:24
Tree
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
2023-10-17T02:39:24Z
Runs
25
Update
BashOperator
PythonOperator
queued
running
success
failed
up_for_retry
up_for_reschedule
upstream_failed
skipped
scheduled
deferred
no_status
Auto-refresh
Oct 17, 10:39
[DAG]
t1
t2_2
t3_2
t4_1
t2_3
t4_1
t2_1
t3_1
t4_1



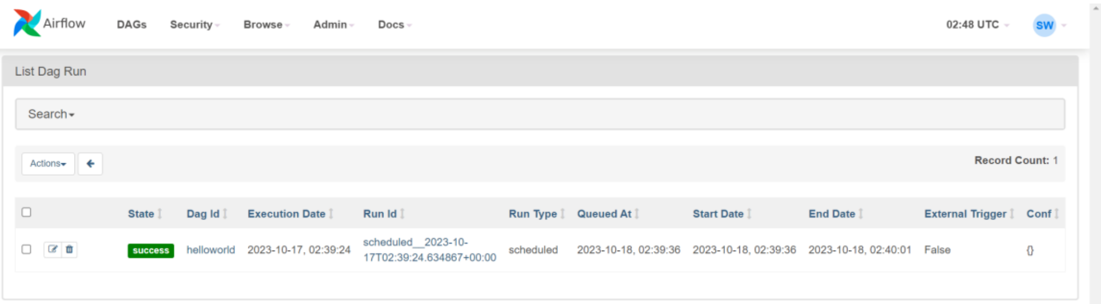
DAG: helloworld
A simple toy DAG
success
Schedule: 1 day, 0:00:00
Next Run: 2023-10-18, 02:39:24
Tree
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
2023-10-17T02:39:25Z
Runs
25
Run
scheduled__2023-10-17T02:39:24.634867+00:00
Layout
Left > Right
Update
Find Task...
BashOperator
PythonOperator
queued
running
success
failed
up_for_retry
up_for_reschedule
upstream_failed
skipped
scheduled
deferred
no_status
Auto-refresh
t1
t2_1
t2_2
t2_3
t3_1
t3_2
t4_1



Q1.2 (2)

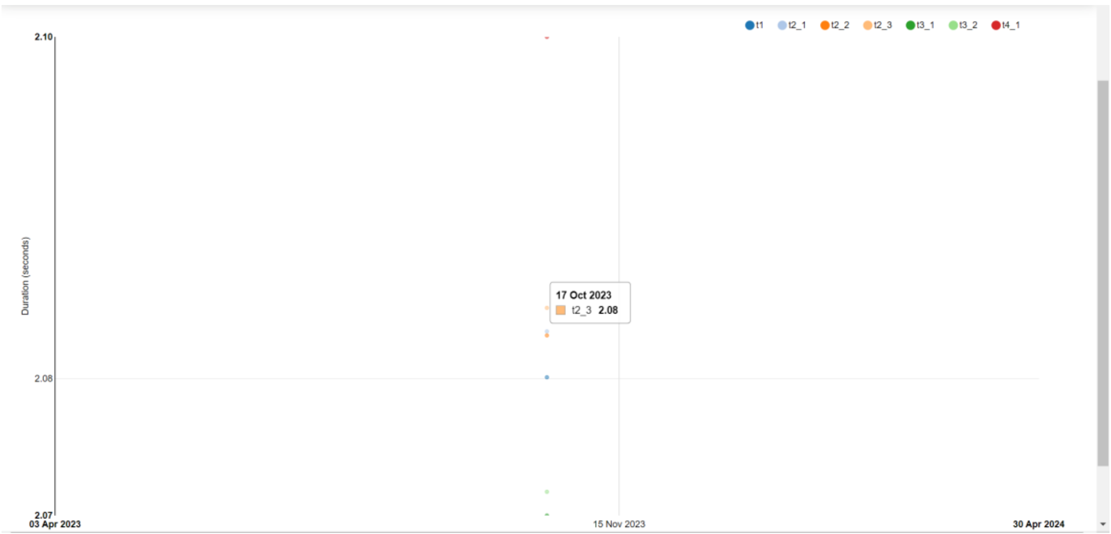
a) Dag Runs

This can monitor the DAG executions. You can view the status, duration, and start time of each run.



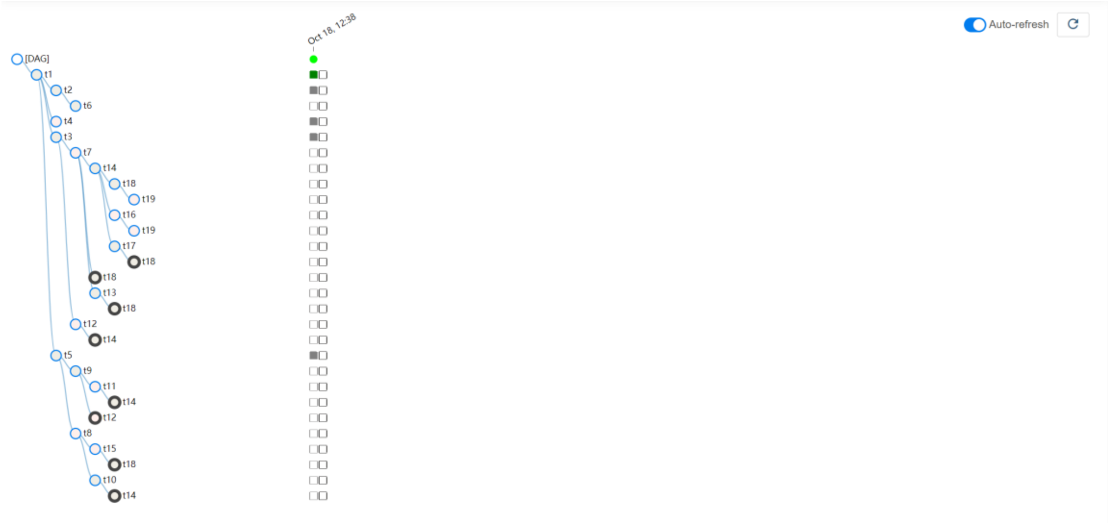
b) Task Duration

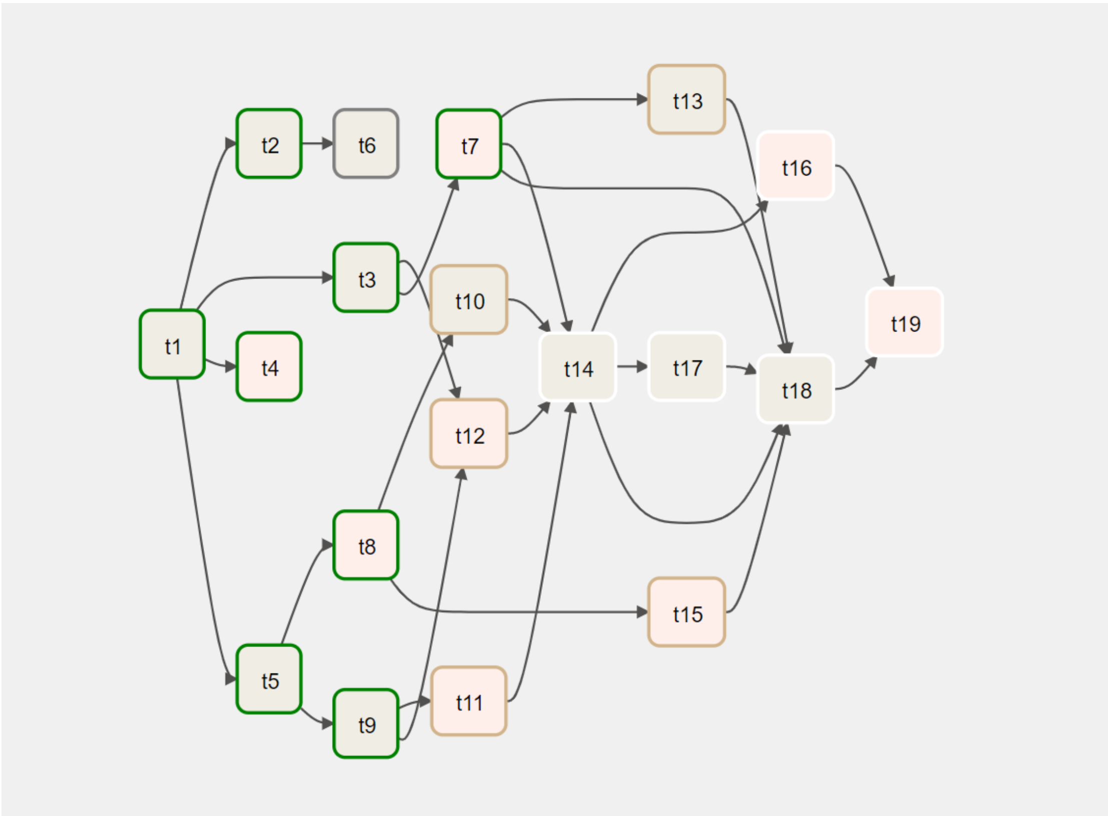
Task Duration indicates how long different tasks have been executed every day in the past. You can find out how long the same task takes to execute by comparing the daily execution times.



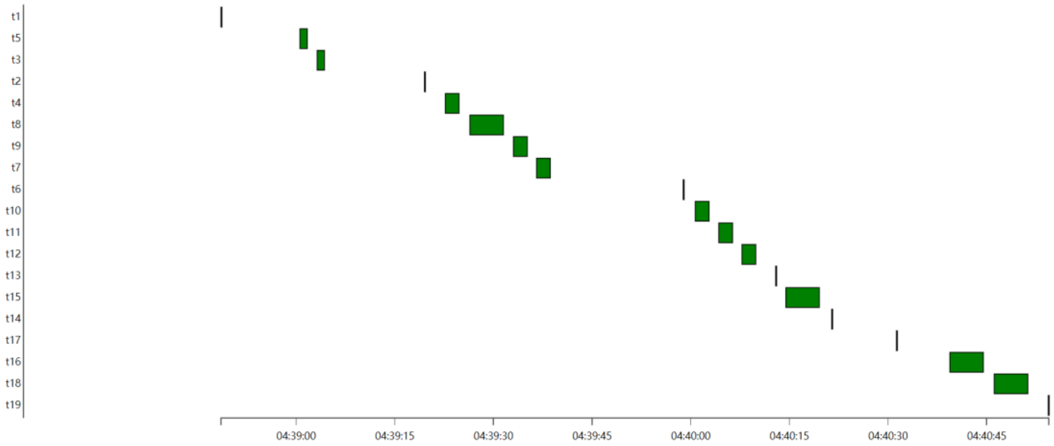
Q2.1

(1) The tree and the graph is:

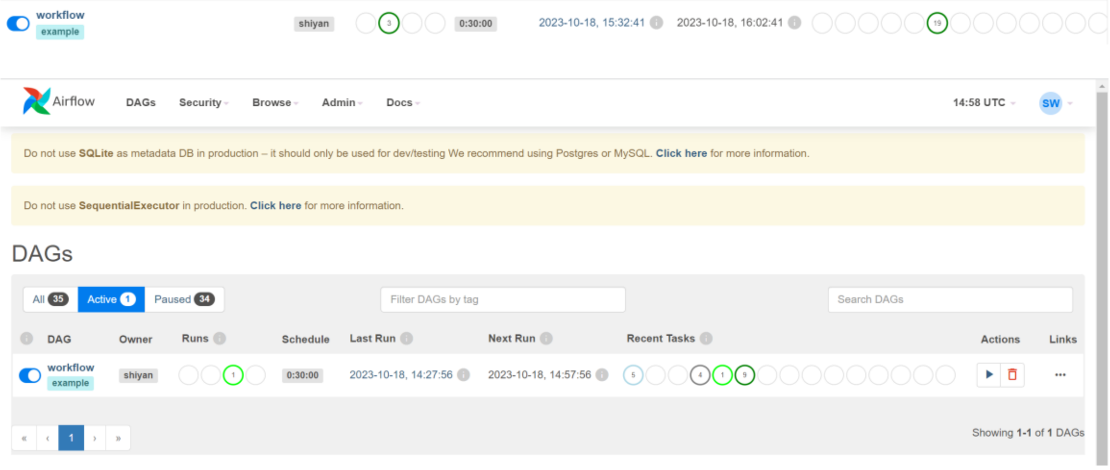




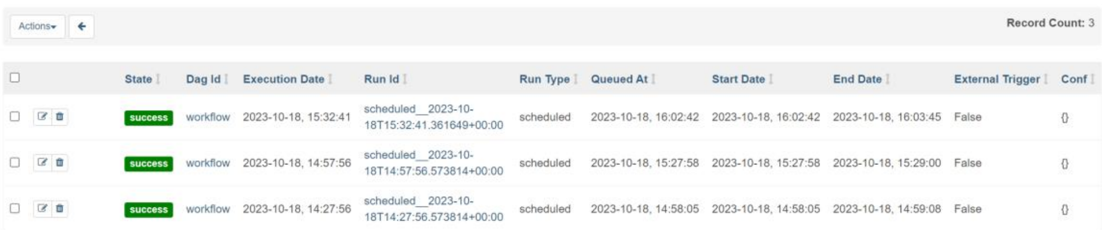
(2) The Gantt Chart is:



(3) For dags, the scheduler runs your job one schedule_interval after the start date, at the end of the period. Therefore, as the interval is set to 30 minutes. If we want the dag to execute immediately, the start time should be 30 minutes earlier than current time.



These are records of 3 executions:


Actions▾

Record Count: 3

State	Dag Id	Execution Date	Run Id	Run Type	Queued At	Start Date	End Date	External Trigger	Conf
success	workflow	2023-10-18, 15:32:41	scheduled__2023-10-18T15:32:41.361649+00:00	scheduled	2023-10-18, 16:02:42	2023-10-18, 16:02:42	2023-10-18, 16:03:45	False	{}
success	workflow	2023-10-18, 14:57:56	scheduled__2023-10-18T14:57:56.573814+00:00	scheduled	2023-10-18, 15:27:58	2023-10-18, 15:27:58	2023-10-18, 15:29:00	False	{}
success	workflow	2023-10-18, 14:27:56	scheduled__2023-10-18T14:27:56.573814+00:00	scheduled	2023-10-18, 14:58:05	2023-10-18, 14:58:05	2023-10-18, 14:59:08	False	{}

Because of a connection interruption, the time between the third and second times was not exactly 30 minutes.

Q2.2

Train model on the data from 2023-1-1 to 2023-10-9 and predict the highest value basing on the dates: 2023-10-10 to 2023-10-16. The code is:

```python
from airflow import DAG
from datetime import datetime, timedelta
from textwrap import dedent
import time
import os
from airflow.operators.python_operator import PythonOperator
from airflow.operators.bash import BashOperator
import yfinance as yf
import pandas as pd
from sklearn import preprocessing
import numpy as np
import math
from sklearn.linear_model import LinearRegression


def download_stock_data(ticker_symbol, start_date, end_date):
    data = yf.download(ticker_symbol, start=start_date, end=end_date)
    data.to_csv(f'{ticker_symbol}_stock_data.csv')


def read_data(ticker_symbol):
    relative_error=[]
    path=f'{ticker_symbol}_stock_data.csv'
    data = pd.read_csv(path)
    data.set_index('Date', inplace=True)
    X = data[['Open', 'High', 'Low', 'Close', 'Volume']]
    data['next_High'] = data['High'].shift(1)
    data.fillna(-99999, inplace=True)
    y = data['next_High']
    current_date = '2023-10-09'
    date = ['2023-10-10', '2023-10-11', '2023-10-12', '2023-10-13', '2023-10-16']
    # train data
    X_train = X[:current_date]
    y_train = y[:current_date]

```

```python
    # train model
    model = LinearRegression()
    model.fit(X_train, y_train)

    for i in range(5):
        # predict next_high
        slice=X.loc[date[i]]
        X_test = pd.DataFrame(slice).T
        y_true = y[date[i]]
        y_pred = model.predict(X_test)

        error = (y_pred - y_true) / y_true
        print(error)
        num_array = np.array(error)
        err = float(num_array[0])
        relative_error.append(err)

    error_df = pd.DataFrame(date, columns=['Date'])
    error_df[ticker_symbol] = relative_error
    error_df['Date'] = date
    print(error_df)

    csv_file = f'{ticker_symbol}_predict_data.csv'

    error_df.to_csv(csv_file, index=False)
    print(f"DataFrame written to {csv_file}")

def combine():
    # read and set index
    df1 = pd.read_csv('AAPL_predict_data.csv')
    df1.set_index('Date', inplace=True)

    df2 = pd.read_csv('GOOGL_predict_data.csv')
    df2.set_index('Date', inplace=True)
```

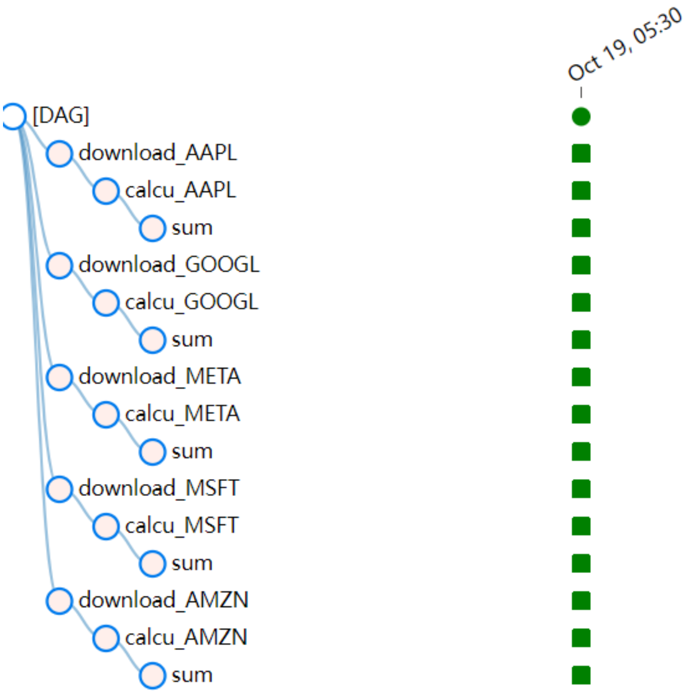
```python
    df2 = pd.read_csv('GOOGL_predict_data.csv')
    df2.set_index('Date', inplace=True)

    df3 = pd.read_csv('META_predict_data.csv')
    df3.set_index('Date', inplace=True)
    df4 = pd.read_csv('MSFT_predict_data.csv')
    df4.set_index('Date', inplace=True)
    df5 = pd.read_csv('AMZN_predict_data.csv')
    df5.set_index('Date', inplace=True)

    # join according to index
    result = df1.join(df2)
    result = result.join(df3)
    result = result.join(df4)
    result = result.join(df5)
    csv_file = 'relative_errors .csv'

    result.to_csv(csv_file)
    print(f"DataFrame written to {csv_file}")



############################################
# DEFINE AIRFLOW DAG (SETTINGS + SCHEDULE)
############################################

default_args = {
    'owner': 'shiyan',
    'depends_on_past': False,
    'email': ['sw3828@columbia.edu'],
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(seconds=30),
    # 'queue': 'bash_queue',
```

```python
    'retry_delay': timedelta(seconds=30),
    # 'queue': 'bash_queue',
    # 'pool': 'backfill',
    # 'priority_weight': 10,
    # 'end_date': datetime(2016, 1, 1),
    # 'wait_for_downstream': False,
    # 'dag': dag,
    # 'sla': timedelta(hours=2),
    # 'execution_timeout': timedelta(seconds=300),
    # 'on_failure_callback': some_function,
    # 'on_success_callback': some_other_function,
    # 'on_retry_callback': another_function,
    # 'sla_miss_callback': yet_another_function,
    # 'trigger_rule': 'all_success'
}

with DAG(
        'Q2.2',
        default_args=default_args,
        description='A simple toy DAG',
        start_date= datetime(2023, 10, 15),
        schedule_interval=None,
        catchup=False,
        tags=['example'],
) as dag:

    download_AAPL = PythonOperator(
        task_id='download_AAPL',
        python_callable=download_stock_data,
        op_args=['AAPL', '2023-01-01', '2023-10-17'],
        dag=dag
    )
```

```
download_GOOGL = PythonOperator(
    task_id='download_GOOGL',
    python_callable=download_stock_data,
    op_args=['GOOGL', '2023-01-01', '2023-10-17'],
    dag=dag
)

download_META = PythonOperator(
    task_id='download_META',
    python_callable=download_stock_data,
    op_args=['META', '2023-01-01', '2023-10-17'],
    dag=dag
)

download_MSFT = PythonOperator(
    task_id='download_MSFT',
    python_callable=download_stock_data,
    op_args=['MSFT', '2023-01-01', '2023-10-17'],
    dag=dag
)

download_AMZN = PythonOperator(
    task_id='download_AMZN',
    python_callable=download_stock_data,
    op_args=['AMZN', '2023-01-01', '2023-10-17'],
    dag=dag
)

calcu_AAPL = PythonOperator(
    task_id='calcu_AAPL',
    python_callable=read_data,
    op_args=['AAPL'],
    dag=dag
)
```

```
calcu_GOOGL = PythonOperator(
    task_id='calcu_GOOGL',
    python_callable=read_data,
    op_args=['GOOGL'],
    dag=dag
)

calcu_META = PythonOperator(
    task_id='calcu_META',
    python_callable=read_data,
    op_args=['META'],
    dag=dag
)

calcu_MSFT = PythonOperator(
    task_id='calcu_MSFT',
    python_callable=read_data,
    op_args=['MSFT'],
    dag=dag
)


calcu_AMZN = PythonOperator(
    task_id='calcu_AMZN',
    python_callable=read_data,
    op_args=['AMZN'],
    dag=dag
)

sum = PythonOperator(
    task_id='sum',
    python_callable=combine,
    dag=dag
)
```

```
###########################################
# DEFINE TASKS HIERARCHY
###########################################

    # task dependencies

    download_AAPL >> calcu_AAPL
    download_GOOGL >> calcu_GOOGL
    download_META >> calcu_META
    download_MSFT >> calcu_MSFT
    download_AMZN >> calcu_AMZN
    [calcu_AAPL, calcu_AMZN, calcu_MSFT,calcu_META,calcu_GOOGL]>>sum
```



Every company needs one task to download the data, and then one task to train the model and calculate the errors. After that, a final task is going to combine the errors to form a final cvs file of errors. Calculation tasks (calcu_AAPL, calcu_GOOGL, calcu_META, calcu_MSFT, calcu_AMZN) depend on the corresponding download tasks. This means that the computing task will not be triggered until the download task is completed. The 'sum' task depends on all calculation tasks, which means that the sum task will not be triggered until all calculation tasks are completed. The data transformation between tasks is through writing and reading cvs files.

The final result is:

```
(airflow) sw3828@instance-test:~$ cat 'relative_errors .csv'
Date,AAPL,GOOGL,META,MSFT,AMZN
2023-10-10,12.515876242321266,4.698446938344299,2.3841779702389148,2.136836366486057,2.9091628503266898
2023-10-11,11.26553713543798,7.606749084238896,2.4782918240969365,3.5751471563743022,3.178366054606628
2023-10-12,3.0711053488877544,4.906651282886682,3.0285911115532103,0.8975600172927406,3.52198474246744
2023-10-13,-15.9566563972384,0.6997353260082676,3.3500157185289865,-5.2295451415346506,6.725207936308758
2023-10-16,11.681035928077325,9.71548670134589,2.902225092085563,1.5147924294957302,3.580743764666707
```