!

THE UNIVERSITY OF WINNIPEG

PROFESSIONAL, APPLIED AND CONTINUING EDUCATION

Assignment 5 - Data Pipeline with Airflow

by

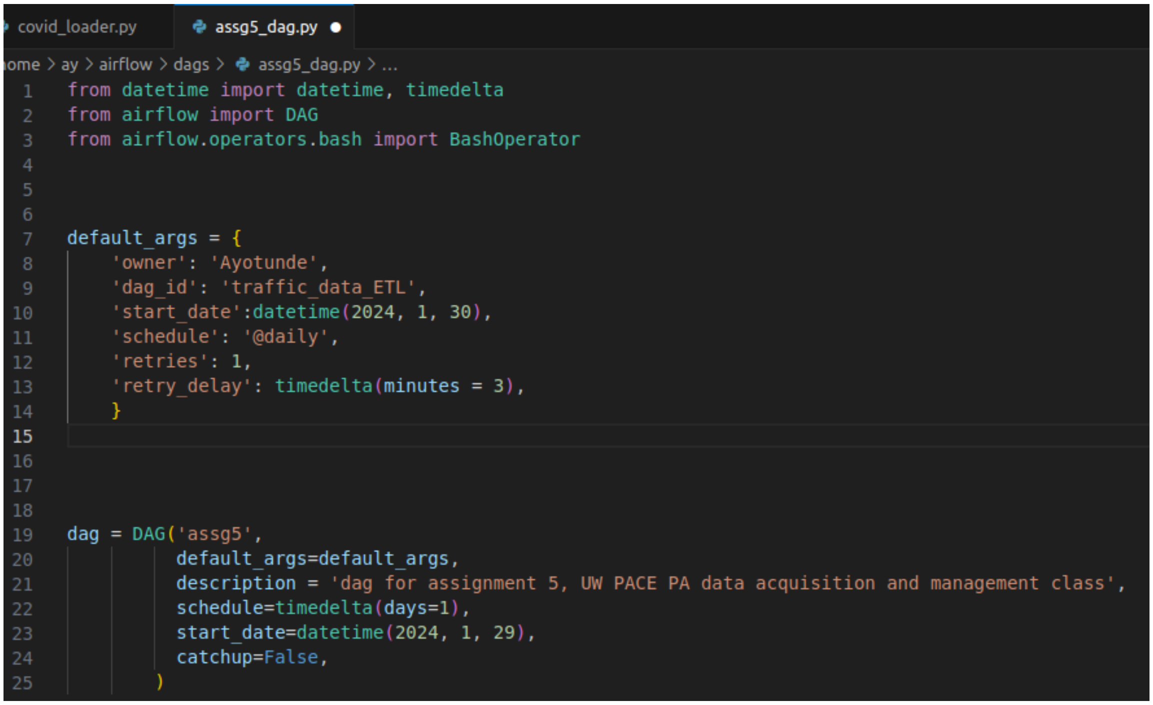
Ayotunde Oyewole

Executive Summary

Some methods of carrying out ETL tasks have been explored in previous projects. This project adds one for arrow the the ETL quiver by enabling the scheduling of various ETL tasks. A data pipeline will be built using airflow. The specific tasks to be carried out include downloading a zipped file from a given URL, extracting data of different formats from the zipped file, extracting only the needed subsets of the files and transforming part of the data as required. The scheduling of these tasks are automated with Apache Airflow and set to run daily. This report describes the whole process of creating and scheduling the task. The results of the ETL process is presented in screenshots. In line with proper data pipeline practice, all steps of the ETL process are carried out in their own seperate folders.

Preliminary Steps

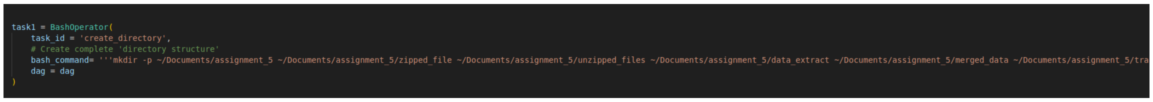
Apache Airflow has previously been installed in the Ubuntu environment. A folder named 'dags' is created in the Airflow and the python file for this project is created and saved in this folder. Afterwards, the dag is instantiated in the python file by defining the relevant variables including the dag_id, schedule, start date and so on. A rety delay is set for 3 minutes and it is set to retry only once. VS code is the editor used for this project. BashOperator was imported for this task, as I elected to use bash commands for all the tasks.



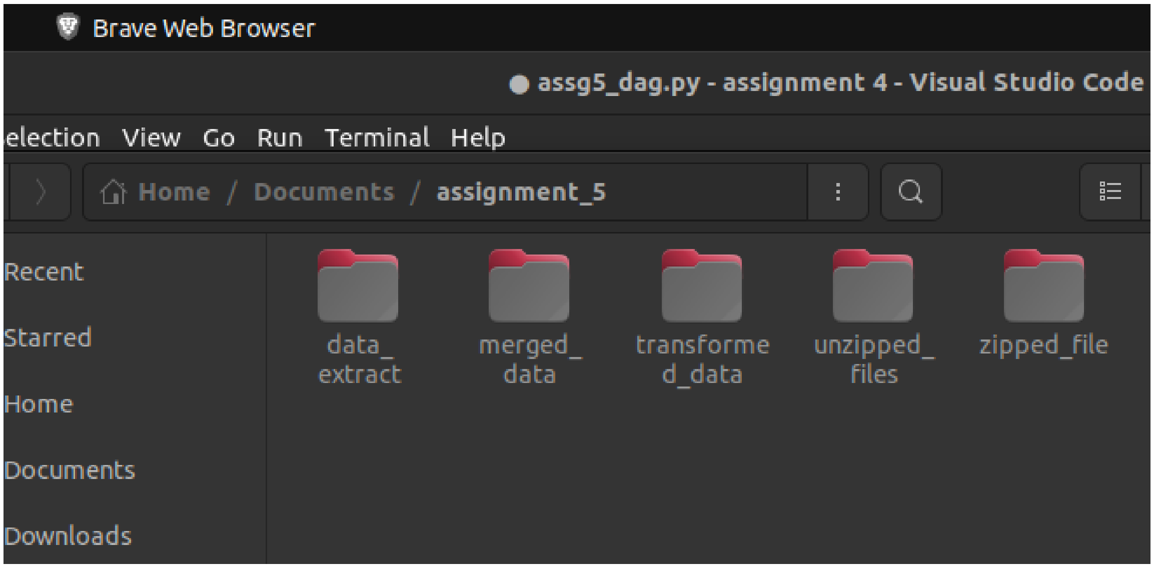
```
from datetime import datetime, timedelta
from airflow import DAG
from airflow.operators.bash import BashOperator



default_args = {
    'owner': 'Ayotunde',
    'dag_id': 'traffic_data_ETL',
    'start_date':datetime(2024, 1, 30),
    'schedule': '@daily',
    'retries': 1,
    'retry_delay': timedelta(minutes = 3),
    }




dag = DAG('assg5',
          default_args=default_args,
          description = 'dag for assignment 5, UW PACE PA data acquisition and management class',
          schedule=timedelta(days=1),
          start_date=datetime(2024, 1, 29),
          catchup=False,
        )
```

Task 1 - Directory Creation

This is the first task of the project. All directories used in the project were created in the assignment_5 parent directory. The folders created included zipped_file, unzipped_files, data_extract, merged_data, transformed_data. Bash command mkdir was used to create all the directories in one line of code. A challenge encountered at this step was the long line of code which could not be splitted into multiple lines. Three quotes ('") and escape characters were attempted, but not successfully. Eventually, the code was returned to a long single line.

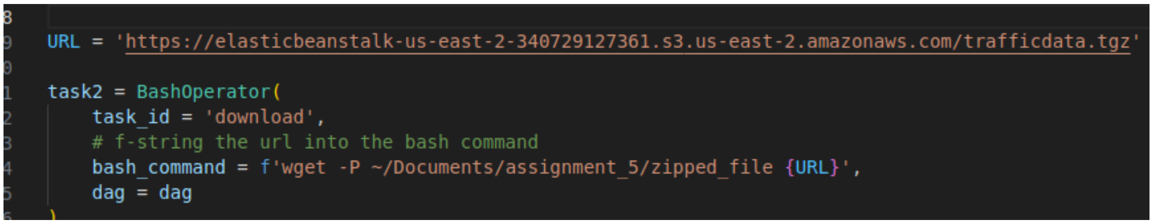


```
task1 = BashOperator(
    task_id = 'create_directory',
    # Create complete 'directory structure'
    bash_command= '''mkdir -p ~/Documents/assignment_5 ~/Documents/assignment_5/zipped_file ~/Documents/assignment_5/unzipped_files ~/Documents/assignment_5/data_extract ~/Documents/assignment_5/merged_data ~/Documents/assignment_5/tra
    dag = dag
)
```

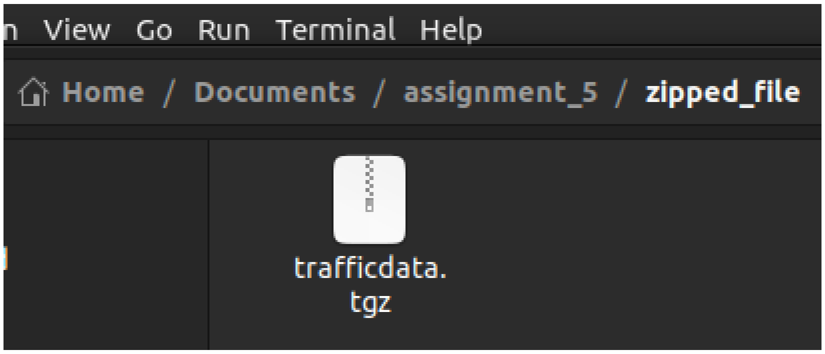


Task 2 - File Download

This task, with id 'download' in the dag script was used to download the zipped (.tgz) file from the url provided and save it to the zipped_file directory. The URL was saved as a global variable. The variable was then accessed from the bash command using a f-string.

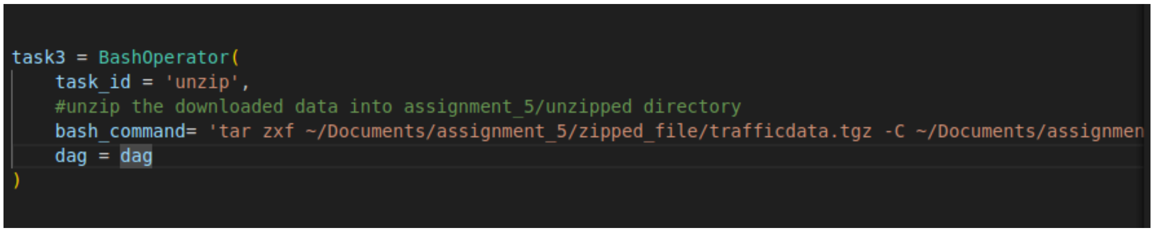


```
URL = 'https://elasticbeanstalk-us-east-2-340729127361.s3.us-east-2.amazonaws.com/trafficdata.tgz'

task2 = BashOperator(
    task_id = 'download',
    # f-string the url into the bash command
    bash_command = f'wget -P ~/Documents/assignment_5/zipped_file {URL}',
    dag = dag
)
```

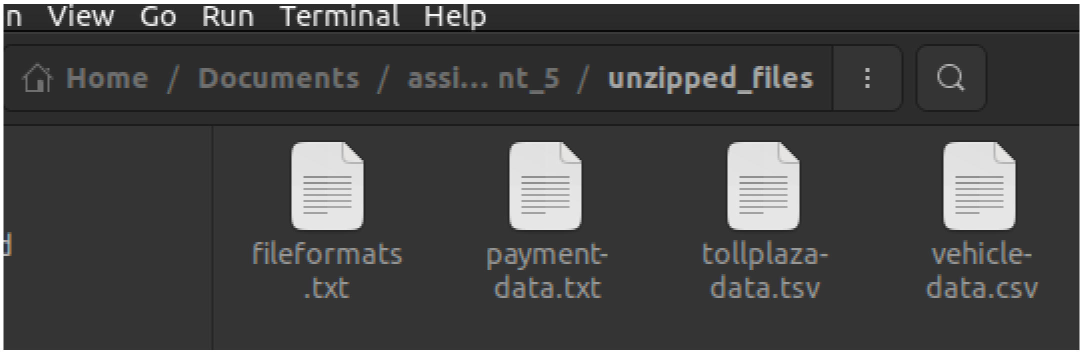

n View Go Run Terminal Help
Home / Documents / assignment_5 / zipped_file
trafficdata.tgz

Task 3 - File Extraction

Bash tar command was used to extract the content of the previously downloaded trafficdata.tgz. The contents of the extraction were saved in another directory called unzipped. The zipped file was found to contain payment-data.txt, tollplaza-data.tsv, vehicle-data.csv and a txt file that explains the file formats and headers of the other three files.

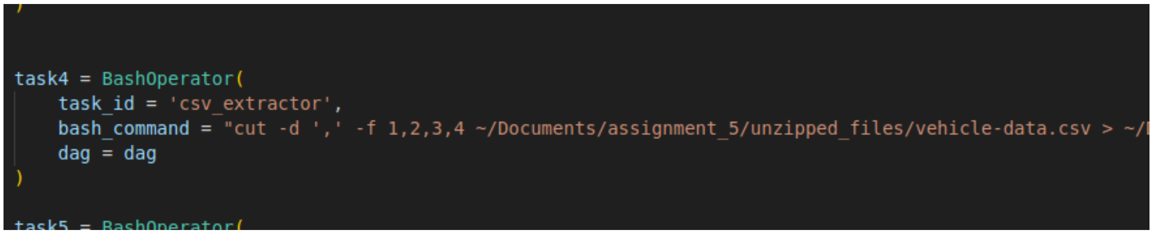


```
task3 = BashOperator(
    task_id = 'unzip',
    #unzip the downloaded data into assignment_5/unzipped directory
    bash_command= 'tar zxf ~/Documents/assignment_5/zipped_file/trafficdata.tgz -C ~/Documents/assignmen
    dag = dag
)
```

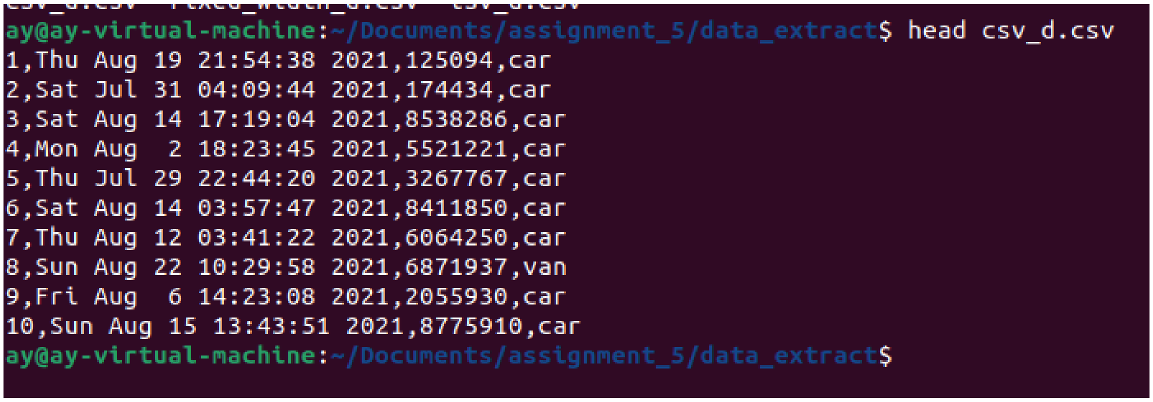

n View Go Run Terminal Help
Home / Documents / assi... nt_5 / unzipped_files
fileformats.txt
payment-data.txt
tollplaza-data.tsv
vehicle-data.csv

Task 4 - CSV Extraction

The csv file vehicle-data.csv was the easiest to work with. Since it is a comma seperated file, it was easy to pass the delimiter into the cut command and specify the columns to be extracted - 1,2,3,4 corresponding to the columns for Rowid, Timestamp, Anonymized Vehicle Number, and vehicle type. The extracted data was saved to a new csv file called csv_d.csv in a folder folder named data extract. The first ten rows of the csv_d is shown in the second screenshot below.



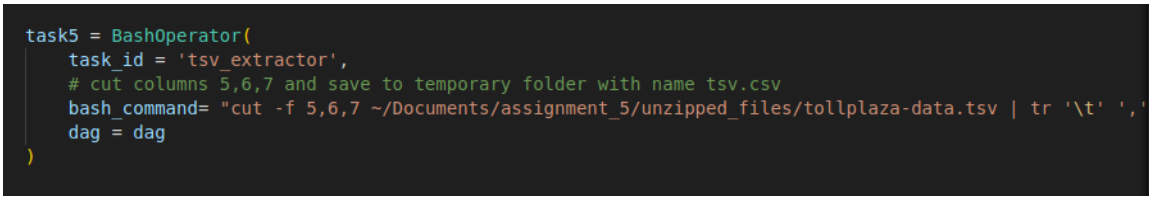
```
)


task4 = BashOperator(
    task_id = 'csv_extractor',
    bash_command = "cut -d ',' -f 1,2,3,4 ~/Documents/assignment_5/unzipped_files/vehicle-data.csv > ~/
    dag = dag
)

task5 = BashOperator(
```



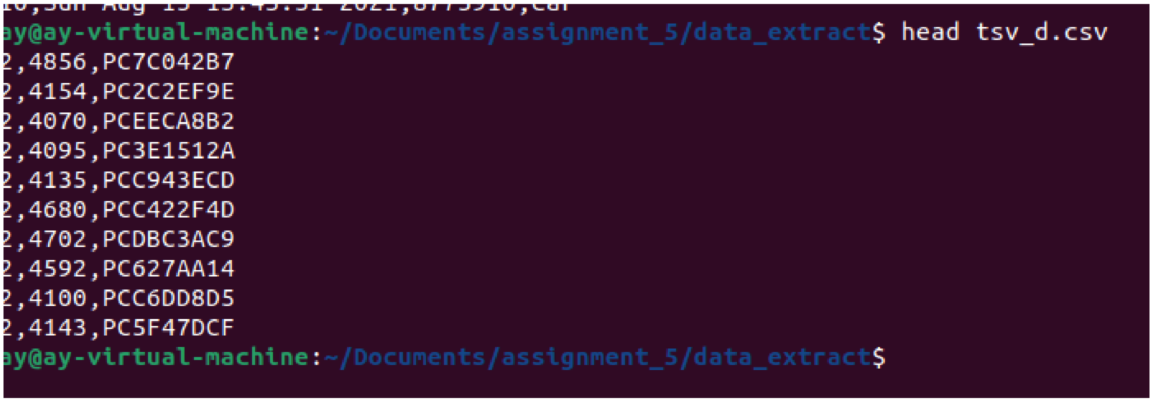
```
csv_d.csv  fixed_width_d.csv  tsv_d.csv
ay@ay-virtual-machine:~/Documents/assignment_5/data_extract$ head csv_d.csv
1,Thu Aug 19 21:54:38 2021,125094,car
2,Sat Jul 31 04:09:44 2021,174434,car
3,Sat Aug 14 17:19:04 2021,8538286,car
4,Mon Aug  2 18:23:45 2021,5521221,car
5,Thu Jul 29 22:44:20 2021,3267767,car
6,Sat Aug 14 03:57:47 2021,8411850,car
7,Thu Aug 12 03:41:22 2021,6064250,car
8,Sun Aug 22 10:29:58 2021,6871937,van
9,Fri Aug  6 14:23:08 2021,2055930,car
10,Sun Aug 15 13:43:51 2021,8775910,car
ay@ay-virtual-machine:~/Documents/assignment_5/data_extract$
```

Task 5 - TSV Extraction

Some required columns were extracted from the tab seperated data - tollplaza-data.tsv. Similar to extracting from a csv file, the cut command was used to extract the required columns - 5,6,7 representing Number of axles, Tollplaza id and Tollplaza code. The extracted data was saved to a csv file called tsv_d.csv. No delimiter was used in the code becuase the cut command had 'tab' as its default delimiter. The top ten records for the extracted file is shown in the second screenshot below.



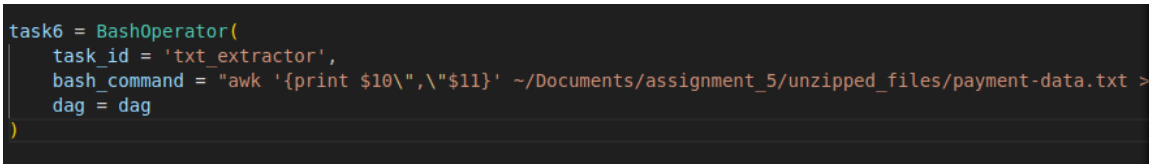
```
task5 = BashOperator(
    task_id = 'tsv_extractor',
    # cut columns 5,6,7 and save to temporary folder with name tsv.csv
    bash_command= "cut -f 5,6,7 ~/Documents/assignment_5/unzipped_files/tollplaza-data.tsv | tr '\t' ','
    dag = dag
)
```



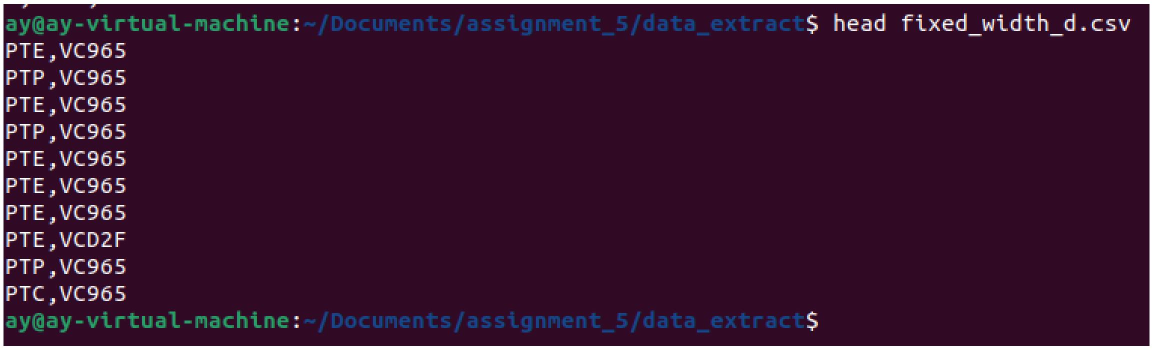
```
10,Sun Aug 15 15:45:51 2021,8773910,car
ay@ay-virtual-machine:~/Documents/assignment_5/data_extract$ head tsv_d.csv
2,4856,PC7C042B7
2,4154,PC2C2EF9E
2,4070,PCEECA8B2
2,4095,PC3E1512A
2,4135,PCC943ECD
2,4680,PCC422F4D
2,4702,PCDBC3AC9
2,4592,PC627AA14
2,4100,PCC6DD8D5
2,4143,PC5F47DCF
ay@ay-virtual-machine:~/Documents/assignment_5/data_extract$
```

Task 6 - TXT Extraction

A fixed width value data - payment-data.txt in txt format was part of the data to be extracted from. Type of payment code and vehicle code were the two columns extracted from this txt file. A different bash command - awk was used to identify the 10th and 11th columns in the file and then write their content to fixed_width_d.csv. To count of the 10/11 columns in both cases, every space represents the next column, since it is a fixed width seperated data. Screenshot of the top 10 records is shown below.



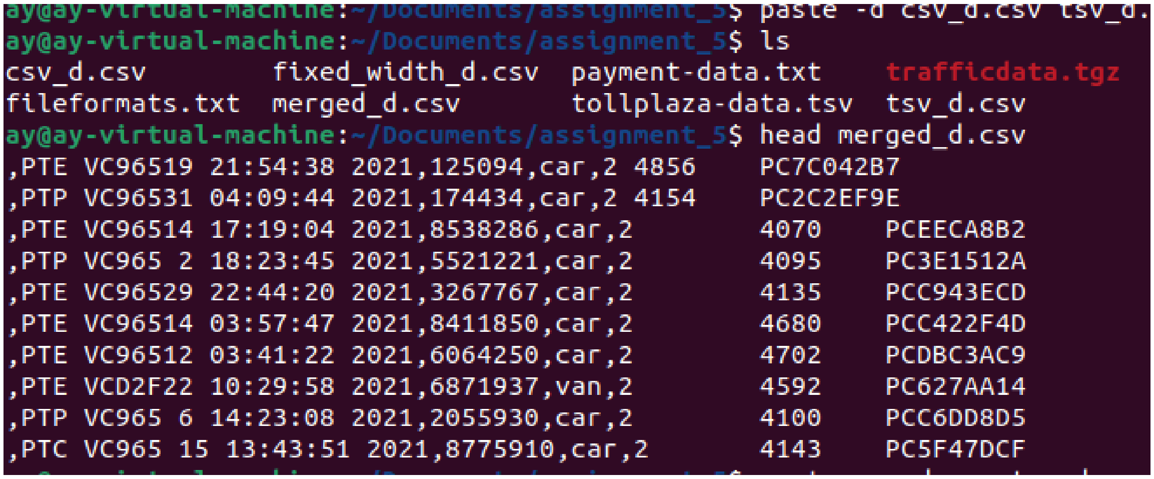
```
task6 = BashOperator(
    task_id = 'txt_extractor',
    bash_command = "awk '{print $10\",\"$11}' ~/Documents/assignment_5/unzipped_files/payment-data.txt >
    dag = dag
)
```



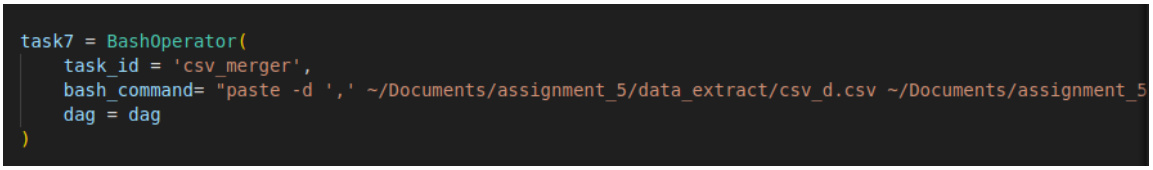
```
ay@ay-virtual-machine:~/Documents/assignment_5/data_extract$ head fixed_width_d.csv
PTE,VC965
PTP,VC965
PTE,VC965
PTP,VC965
PTE,VC965
PTE,VC965
PTE,VC965
PTE,VCD2F
PTP,VC965
PTC,VC965
ay@ay-virtual-machine:~/Documents/assignment_5/data_extract$
```

Task 7 - Document Merger

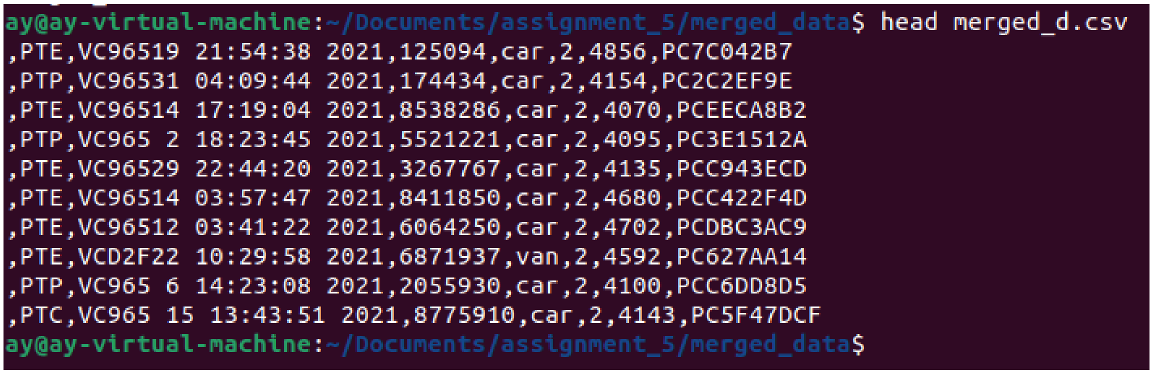
This step, with id 'csv_merger' was a task to merge all the csv files into one, along their columns. Bash paste command was used for this task. At first attempt, the file was not well formated with the right delimiter (Screenshot 1 below). To resolve this, the tr command was included to the bash command for task 5 to trim off the tab and replace with comma. The delimiter argument was then introduced to the paste command for this task and that resulted in a properly formatted document (Screenshot3 below).



```
ay@ay-virtual-machine:~/Documents/assignment_5$ paste -d csv_d.csv tsv_d.
ay@ay-virtual-machine:~/Documents/assignment_5$ ls
csv_d.csv        fixed_width_d.csv  payment-data.txt     trafficdata.tgz
fileformats.txt  merged_d.csv       tollplaza-data.tsv  tsv_d.csv
ay@ay-virtual-machine:~/Documents/assignment_5$ head merged_d.csv
,PTE VC96519 21:54:38 2021,125094,car,2 4856    PC7C042B7
,PTP VC96531 04:09:44 2021,174434,car,2 4154    PC2C2EF9E
,PTE VC96514 17:19:04 2021,8538286,car,2        4070    PCEECA8B2
,PTP VC965 2 18:23:45 2021,5521221,car,2        4095    PC3E1512A
,PTE VC96529 22:44:20 2021,3267767,car,2        4135    PCC943ECD
,PTE VC96514 03:57:47 2021,8411850,car,2        4680    PCC422F4D
,PTE VC96512 03:41:22 2021,6064250,car,2        4702    PCDBC3AC9
,PTE VCD2F22 10:29:58 2021,6871937,van,2        4592    PC627AA14
,PTP VC965 6 14:23:08 2021,2055930,car,2        4100    PCC6DD8D5
,PTC VC965 15 13:43:51 2021,8775910,car,2       4143    PC5F47DCF
```



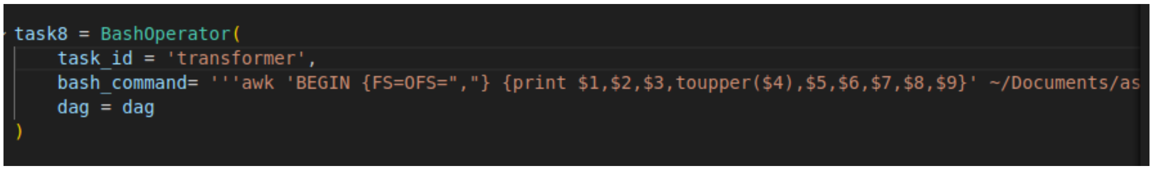
```
task7 = BashOperator(
    task_id = 'csv_merger',
    bash_command= "paste -d ',' ~/Documents/assignment_5/data_extract/csv_d.csv ~/Documents/assignment_5
    dag = dag
)
```



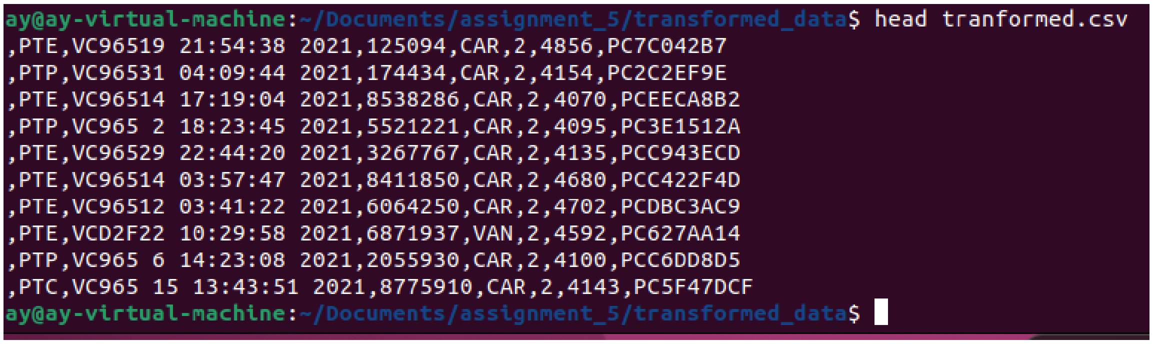
```
ay@ay-virtual-machine:~/Documents/assignment_5/merged_data$ head merged_d.csv
,PTE,VC96519 21:54:38 2021,125094,car,2,4856,PC7C042B7
,PTP,VC96531 04:09:44 2021,174434,car,2,4154,PC2C2EF9E
,PTE,VC96514 17:19:04 2021,8538286,car,2,4070,PCEECA8B2
,PTP,VC965 2 18:23:45 2021,5521221,car,2,4095,PC3E1512A
,PTE,VC96529 22:44:20 2021,3267767,car,2,4135,PCC943ECD
,PTE,VC96514 03:57:47 2021,8411850,car,2,4680,PCC422F4D
,PTE,VC96512 03:41:22 2021,6064250,car,2,4702,PCDBC3AC9
,PTE,VCD2F22 10:29:58 2021,6871937,van,2,4592,PC627AA14
,PTP,VC965 6 14:23:08 2021,2055930,car,2,4100,PCC6DD8D5
,PTC,VC965 15 13:43:51 2021,8775910,car,2,4143,PC5F47DCF
ay@ay-virtual-machine:~/Documents/assignment_5/merged_data$
```

Task 8 - Transformer

The final task on this project was to transform the fourth column (vehicle type) to uppercase. This was accomplished by a mix of awk command and toupper command. The data with one column converted to uppercase was written to transformed_data directory and named transformed.csv.



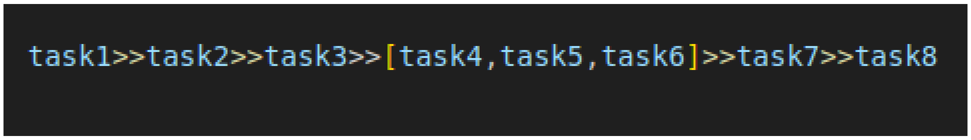
```
task8 = BashOperator(
    task_id = 'transformer',
    bash_command= '''awk 'BEGIN {FS=OFS=","} {print $1,$2,$3,toupper($4),$5,$6,$7,$8,$9}' ~/Documents/as
    dag = dag
)
```



```
ay@ay-virtual-machine:~/Documents/assignment_5/transformed_data$ head tranformed.csv
,PTE,VC96519 21:54:38 2021,125094,CAR,2,4856,PC7C042B7
,PTP,VC96531 04:09:44 2021,174434,CAR,2,4154,PC2C2EF9E
,PTE,VC96514 17:19:04 2021,8538286,CAR,2,4070,PCEECA8B2
,PTP,VC965 2 18:23:45 2021,5521221,CAR,2,4095,PC3E1512A
,PTE,VC96529 22:44:20 2021,3267767,CAR,2,4135,PCC943ECD
,PTE,VC96514 03:57:47 2021,8411850,CAR,2,4680,PCC422F4D
,PTE,VC96512 03:41:22 2021,6064250,CAR,2,4702,PCDBC3AC9
,PTE,VCD2F22 10:29:58 2021,6871937,VAN,2,4592,PC627AA14
,PTP,VC965 6 14:23:08 2021,2055930,CAR,2,4100,PCC6DD8D5
,PTC,VC965 15 13:43:51 2021,8775910,CAR,2,4143,PC5F47DCF
ay@ay-virtual-machine:~/Documents/assignment_5/transformed_data$
```

Rounding Up

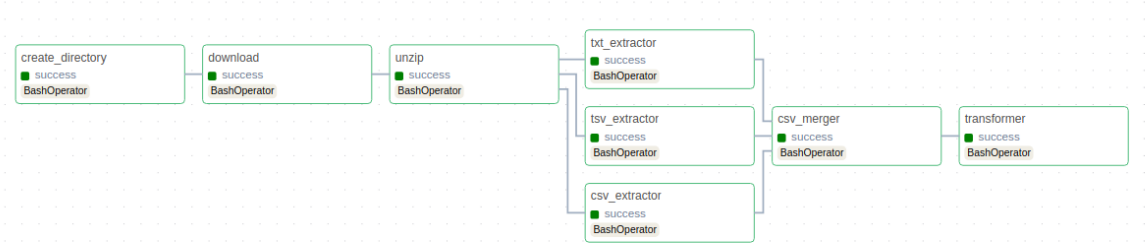
The pipeline dependencies were defined and tasks 4, 5, and 6 were set to run in parallel. After this, the modifications on the python file were saved and executed from vscode to identify and amend errors, if any exists.



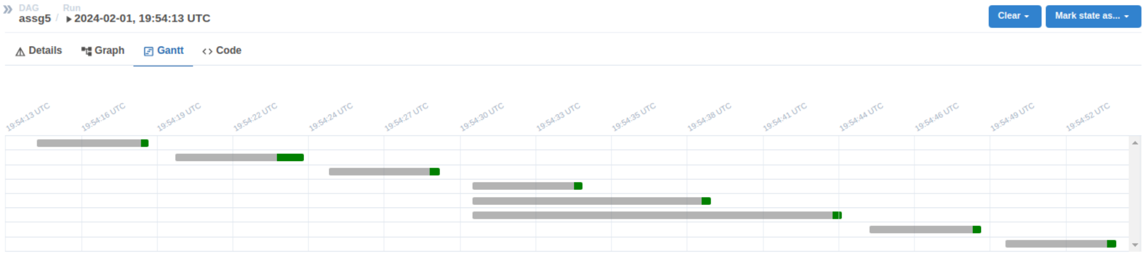
```
task1>>task2>>task3>>[task4,task5,task6]>>task7>>task8
```

Airflow was started using 'airflow standalone' and the from the local host URL, the DAG was triggered. It is noteworthy that there were some unsuccessful trigger attempts (failed at certain points in the pipeline). These failures were painstakingly addressed through a review of the logs. Additional screenshots from the airflow interface showing various aspects of the triggered dag are shown below.

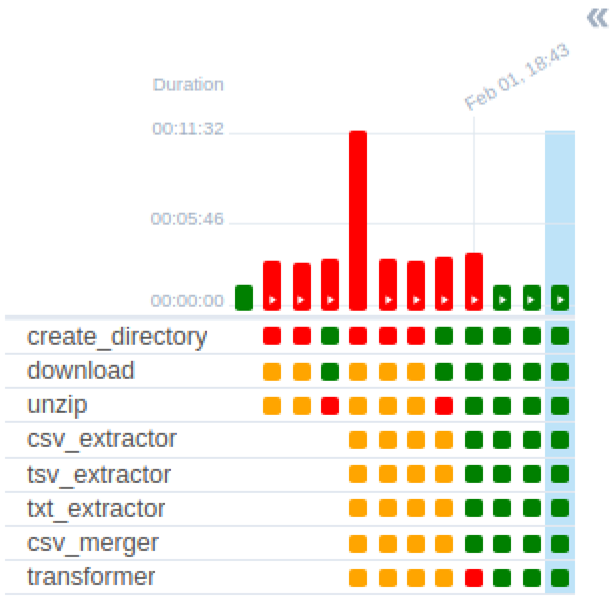
Graph of the DAG's dependencies from Airflow



Gantt chart of the most recent execution of the DAG

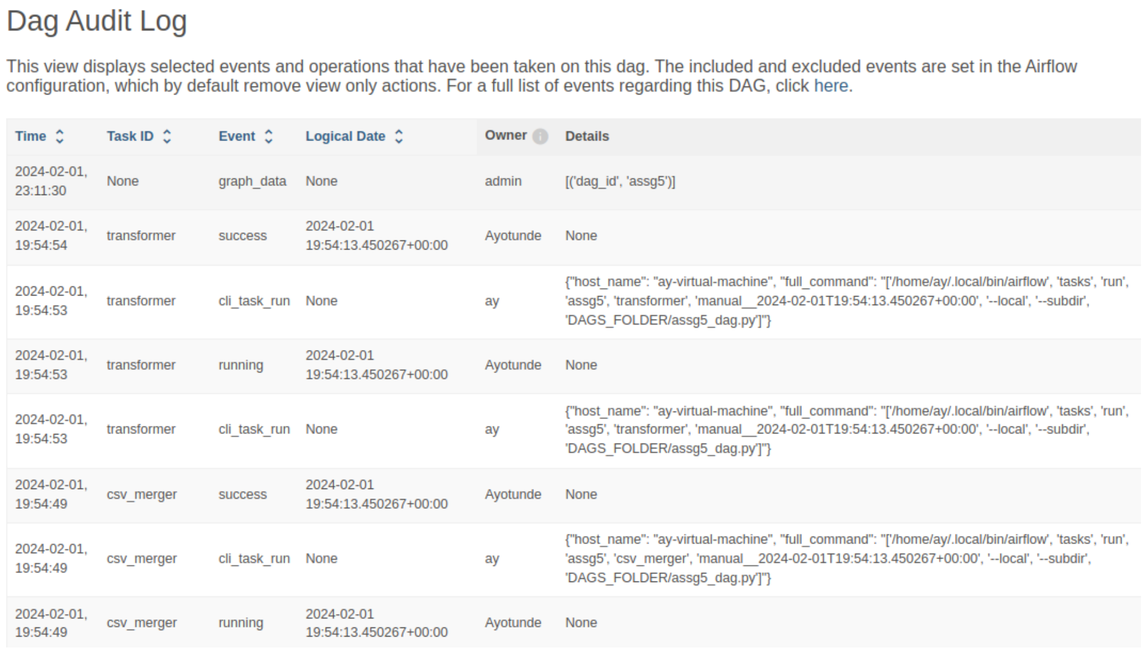


Some failed executions before successful activations



create_directory	🟥	🟥	🟩	🟥	🟥	🟥	🟩	🟩	🟩	🟩	🟩
download	🟧	🟧	🟩	🟧	🟧	🟧	🟩	🟩	🟩	🟩	🟩
unzip	🟧	🟧	🟥	🟧	🟧	🟧	🟥	🟩	🟩	🟩	🟩
csv_extractor				🟧	🟧	🟧	🟧	🟩	🟩	🟩	🟩
tsv_extractor				🟧	🟧	🟧	🟧	🟩	🟩	🟩	🟩
txt_extractor				🟧	🟧	🟧	🟧	🟩	🟩	🟩	🟩
csv_merger				🟧	🟧	🟧	🟧	🟩	🟩	🟩	🟩
transformer				🟧	🟧	🟧	🟧	🟥	🟩	🟩	🟩

A few logs

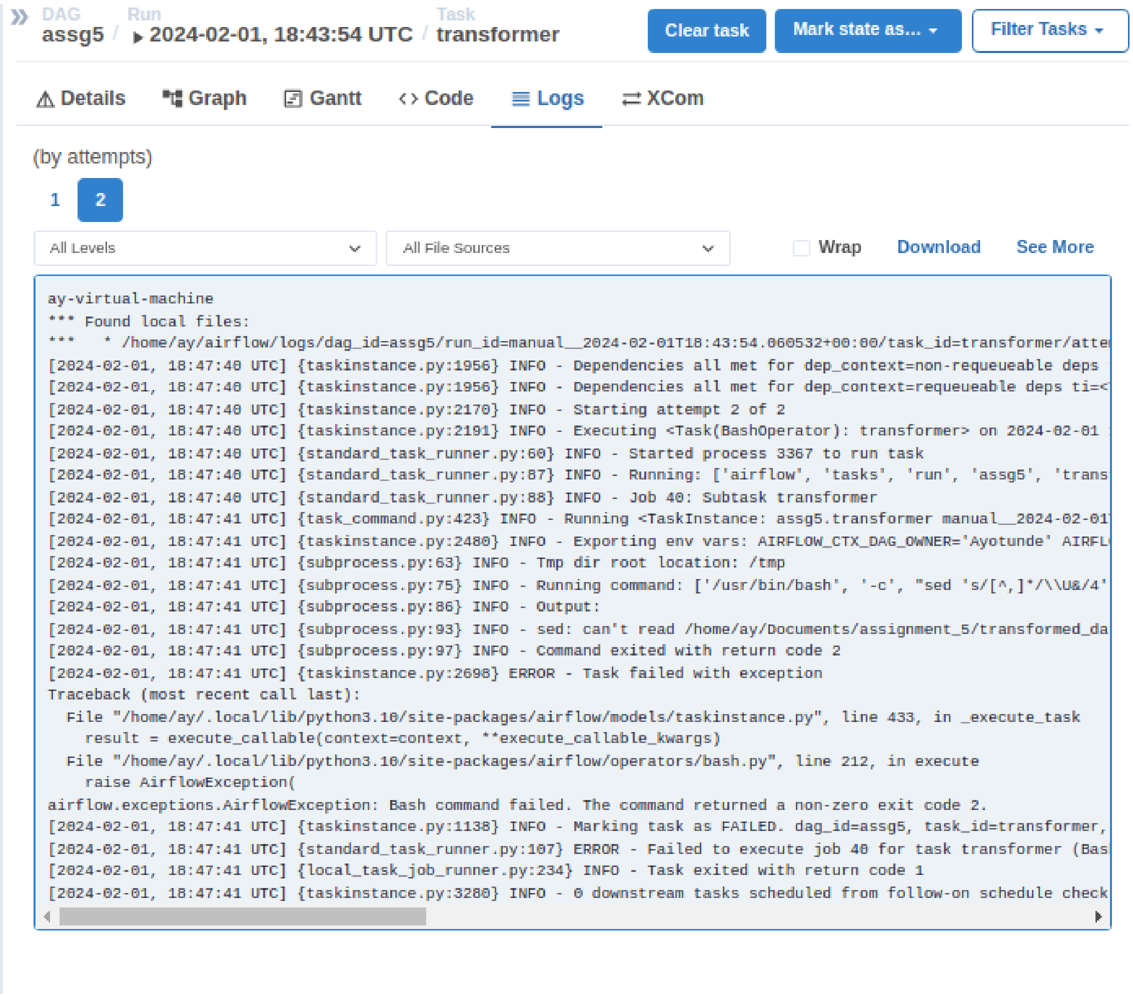


Dag Audit Log

This view displays selected events and operations that have been taken on this dag. The included and excluded events are set in the Airflow configuration, which by default remove view only actions. For a full list of events regarding this DAG, click here.

Time	Task ID	Event	Logical Date	Owner	Details
2024-02-01, 23:11:30	None	graph_data	None	admin	[('dag_id', 'assg5')]
2024-02-01, 19:54:54	transformer	success	2024-02-01 19:54:13.450267+00:00	Ayotunde	None
2024-02-01, 19:54:53	transformer	cli_task_run	None	ay	{"host_name": "ay-virtual-machine", "full_command": "['/home/ay/.local/bin/airflow', 'tasks', 'run', 'assg5', 'transformer', 'manual__2024-02-01T19:54:13.450267+00:00', '--local', '--subdir', 'DAGS_FOLDER/assg5_dag.py']"}
2024-02-01, 19:54:53	transformer	running	2024-02-01 19:54:13.450267+00:00	Ayotunde	None
2024-02-01, 19:54:53	transformer	cli_task_run	None	ay	{"host_name": "ay-virtual-machine", "full_command": "['/home/ay/.local/bin/airflow', 'tasks', 'run', 'assg5', 'transformer', 'manual__2024-02-01T19:54:13.450267+00:00', '--local', '--subdir', 'DAGS_FOLDER/assg5_dag.py']"}
2024-02-01, 19:54:49	csv_merger	success	2024-02-01 19:54:13.450267+00:00	Ayotunde	None
2024-02-01, 19:54:49	csv_merger	cli_task_run	None	ay	{"host_name": "ay-virtual-machine", "full_command": "['/home/ay/.local/bin/airflow', 'tasks', 'run', 'assg5', 'csv_merger', 'manual__2024-02-01T19:54:13.450267+00:00', '--local', '--subdir', 'DAGS_FOLDER/assg5_dag.py']"}
2024-02-01, 19:54:49	csv_merger	running	2024-02-01 19:54:13.450267+00:00	Ayotunde	None

Specific log of one of the failed tasks



```
ay-virtual-machine
*** Found local files:
***   * /home/ay/airflow/logs/dag_id=assg5/run_id=manual__2024-02-01T18:43:54.060532+00:00/task_id=transformer/atte
[2024-02-01, 18:47:40 UTC] {taskinstance.py:1956} INFO - Dependencies all met for dep_context=non-requeueable deps
[2024-02-01, 18:47:40 UTC] {taskinstance.py:1956} INFO - Dependencies all met for dep_context=requeueable deps ti=<
[2024-02-01, 18:47:40 UTC] {taskinstance.py:2170} INFO - Starting attempt 2 of 2
[2024-02-01, 18:47:40 UTC] {taskinstance.py:2191} INFO - Executing <Task(BashOperator): transformer> on 2024-02-01
[2024-02-01, 18:47:40 UTC] {standard_task_runner.py:60} INFO - Started process 3367 to run task
[2024-02-01, 18:47:40 UTC] {standard_task_runner.py:87} INFO - Running: ['airflow', 'tasks', 'run', 'assg5', 'trans
[2024-02-01, 18:47:40 UTC] {standard_task_runner.py:88} INFO - Job 40: Subtask transformer
[2024-02-01, 18:47:41 UTC] {task_command.py:423} INFO - Running <TaskInstance: assg5.transformer manual__2024-02-01
[2024-02-01, 18:47:41 UTC] {taskinstance.py:2480} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='Ayotunde' AIRFL
[2024-02-01, 18:47:41 UTC] {subprocess.py:63} INFO - Tmp dir root location: /tmp
[2024-02-01, 18:47:41 UTC] {subprocess.py:75} INFO - Running command: ['/usr/bin/bash', '-c', "sed 's/[^,]*/\\U&/4'
[2024-02-01, 18:47:41 UTC] {subprocess.py:86} INFO - Output:
[2024-02-01, 18:47:41 UTC] {subprocess.py:93} INFO - sed: can't read /home/ay/Documents/assignment_5/transformed_da
[2024-02-01, 18:47:41 UTC] {subprocess.py:97} INFO - Command exited with return code 2
[2024-02-01, 18:47:41 UTC] {taskinstance.py:2698} ERROR - Task failed with exception
Traceback (most recent call last):
  File "/home/ay/.local/lib/python3.10/site-packages/airflow/models/taskinstance.py", line 433, in _execute_task
    result = execute_callable(context=context, **execute_callable_kwargs)
  File "/home/ay/.local/lib/python3.10/site-packages/airflow/operators/bash.py", line 212, in execute
    raise AirflowException(
airflow.exceptions.AirflowException: Bash command failed. The command returned a non-zero exit code 2.
[2024-02-01, 18:47:41 UTC] {taskinstance.py:1138} INFO - Marking task as FAILED. dag_id=assg5, task_id=transformer,
[2024-02-01, 18:47:41 UTC] {standard_task_runner.py:107} ERROR - Failed to execute job 40 for task transformer (Bas
[2024-02-01, 18:47:41 UTC] {local_task_job_runner.py:234} INFO - Task exited with return code 1
[2024-02-01, 18:47:41 UTC] {taskinstance.py:3280} INFO - 0 downstream tasks scheduled from follow-on schedule check
```

Conclusion

Creating and maintaining data pipelines is a core skill of a Data Engineer. One of the ways of expressing this skill is in being able to create and deploy an ETL pipeline using Apache airflow. This project uses airflow to automate the scheduling of an ETL process consisting of eight tasks. It is scheduled to run daily and bash commands were used to carry out all the tasks in this project. The screenshots of all the steps from DAG creation to activation has been shown in this report.