Documentation: (Prompt 1)

How to Run the project(https://github.com/surebhandari/data) locally:

Prerequisites:

- Docker/Docker-composer installed and running before starting the steps below
- Sometimes starting with a clean slate helps and we can run the following command **docker system prune -a** to clean up.

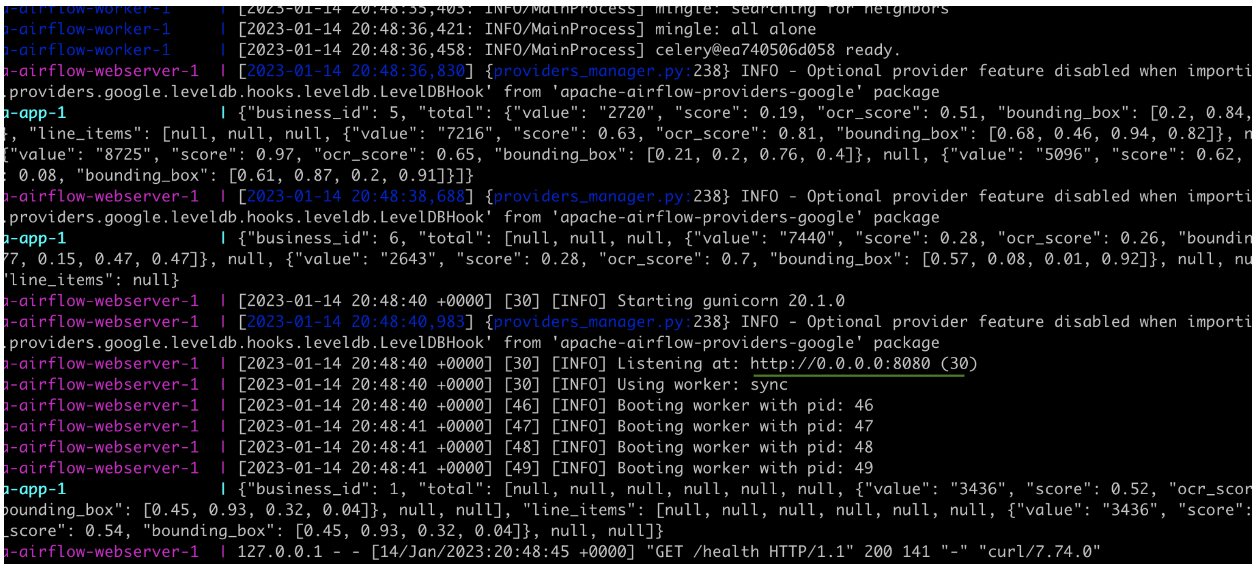
Steps:

- Clone the Repository: https://github.com/surebhandari/data
- It has the following structure(under data directory)
    - **Dags** -> Directory where the airflow job(postgres_db_dag.py) exists
        - It has the following four tasks

```
task_create_output_table >>
task_get_veryfi_data >>
task_process_veryfi_data >>
task_write_veryfi_data
```

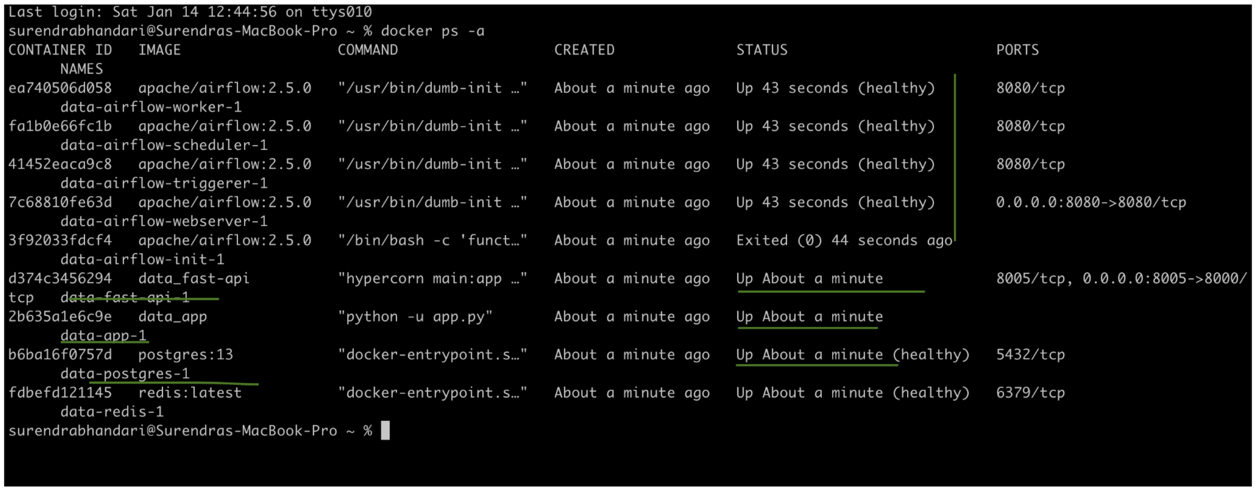
        - They are implemented using PostgresOperator/PythonOperator
        - First we create the output table(name = **total**) where we will write the total values at the end.
        - Next we get the receipt data in batches(Batch of Max 2000 as of now) from the documents table.
        - Next we process the data received above in batches.
        - And at the end we update the output table(total) with the total value corresponding to a business_id.
        - We keep on updating the total value of business_id on each job run.
    - **App ->** This simulates the receipt data generation mentioned in the assignment using a docker container
        - This is a running container whose job is to insert the receipt data to the documents table every 3 seconds.
    - **Db** -> database directory with scripts for database initialization tasks used during docker build/compose
        - Initialization scripts to create **documents** and **tracker** tables as soon as the app container boots.
    - **Fast-api** -> container to host a simple api
        - Simple implementation of two get endpoints, one for the simple index and other for finding total value of a given business_id. This can be extended to other endpoints easily.

        - **Docker-compose.yaml**
            - This is a single yaml file that is used for
                - Airflow startup
                - Single postgres instance (used by airflow as well as the receipt data generation containerized app (inside /app)
                - Building and starting the app for receipt data generation
                - Building and starting fast-api container
                - This yaml mentioned the dependency of apps on postgres, so we are covered on that they should start after postgres successfully started.
    - After the clone please run the following command from **inside the data directory**
        - **docker compose up --build** -> This command will build/start all the components for airflow, postgres instance, app to load receipt data and start the fast-api container.
    - It will take some time to start all the components and towards the end the screen should look something similar to this one



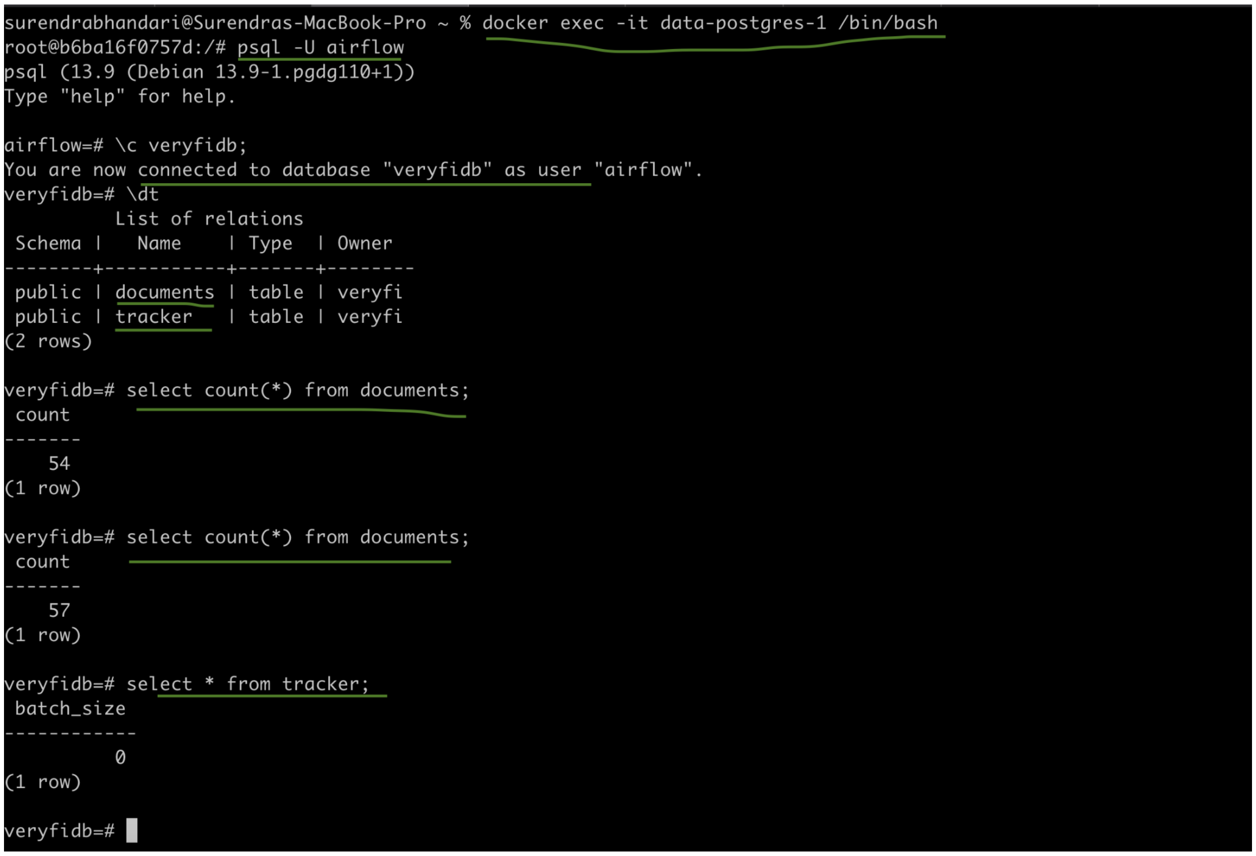
```
a-airflow-worker-1     | [2023-01-14 20:48:35,403: INFO/MainProcess] mingle: searching for neighbors
a-airflow-worker-1     | [2023-01-14 20:48:36,421: INFO/MainProcess] mingle: all alone
a-airflow-worker-1     | [2023-01-14 20:48:36,458: INFO/MainProcess] celery@ea740506d058 ready.
a-airflow-webserver-1  | [2023-01-14 20:48:36,830] {providers_manager.py:238} INFO - Optional provider feature disabled when importi
.providers.google.leveldb.hooks.leveldb.LevelDBHook' from 'apache-airflow-providers-google' package
a-app-1                | {"business_id": 5, "total": {"value": "2720", "score": 0.19, "ocr_score": 0.51, "bounding_box": [0.2, 0.84,
}, "line_items": [null, null, null, {"value": "7216", "score": 0.63, "ocr_score": 0.81, "bounding_box": [0.68, 0.46, 0.94, 0.82]}, n
{"value": "8725", "score": 0.97, "ocr_score": 0.65, "bounding_box": [0.21, 0.2, 0.76, 0.4]}, null, {"value": "5096", "score": 0.62,
: 0.08, "bounding_box": [0.61, 0.87, 0.2, 0.91]}]}
a-airflow-webserver-1  | [2023-01-14 20:48:38,688] {providers_manager.py:238} INFO - Optional provider feature disabled when importi
.providers.google.leveldb.hooks.leveldb.LevelDBHook' from 'apache-airflow-providers-google' package
a-app-1                | {"business_id": 6, "total": [null, null, null, {"value": "7440", "score": 0.28, "ocr_score": 0.26, "boundin
77, 0.15, 0.47, 0.47]}, null, {"value": "2643", "score": 0.28, "ocr_score": 0.7, "bounding_box": [0.57, 0.08, 0.01, 0.92]}, null, nu
'line_items": null}
a-airflow-webserver-1  | [2023-01-14 20:48:40 +0000] [30] [INFO] Starting gunicorn 20.1.0
a-airflow-webserver-1  | [2023-01-14 20:48:40,983] {providers_manager.py:238} INFO - Optional provider feature disabled when importi
.providers.google.leveldb.hooks.leveldb.LevelDBHook' from 'apache-airflow-providers-google' package
a-airflow-webserver-1  | [2023-01-14 20:48:40 +0000] [30] [INFO] Listening at: http://0.0.0.0:8080 (30)
a-airflow-webserver-1  | [2023-01-14 20:48:40 +0000] [30] [INFO] Using worker: sync
a-airflow-webserver-1  | [2023-01-14 20:48:40 +0000] [46] [INFO] Booting worker with pid: 46
a-airflow-webserver-1  | [2023-01-14 20:48:41 +0000] [47] [INFO] Booting worker with pid: 47
a-airflow-webserver-1  | [2023-01-14 20:48:41 +0000] [48] [INFO] Booting worker with pid: 48
a-airflow-webserver-1  | [2023-01-14 20:48:41 +0000] [49] [INFO] Booting worker with pid: 49
a-app-1                | {"business_id": 1, "total": [null, null, null, null, null, null, {"value": "3436", "score": 0.52, "ocr_scor
bounding_box": [0.45, 0.93, 0.32, 0.04]}, null, null], "line_items": [null, null, null, null, null, null, {"value": "3436", "score":
_score": 0.54, "bounding_box": [0.45, 0.93, 0.32, 0.04]}, null, null]}
a-airflow-webserver-1  | 127.0.0.1 - - [14/Jan/2023:20:48:45 +0000] "GET /health HTTP/1.1" 200 141 "-" "curl/7.74.0"
```

- The same can be verified by looking at the docker processes and we should see something similar to the screenshot below: (**docker ps -a**)
    - It shows airflow, fast-api, data-app, postgres instance all up and running



```
Last login: Sat Jan 14 12:44:56 on ttys010
surendrabhandari@Surendras-MacBook-Pro ~ % docker ps -a
CONTAINER ID   IMAGE                  COMMAND                  CREATED              STATUS                        PORTS
      NAMES
ea740506d058   apache/airflow:2.5.0   "/usr/bin/dumb-init …"   About a minute ago   Up 43 seconds (healthy)       8080/tcp
      data-airflow-worker-1
fa1b0e66fc1b   apache/airflow:2.5.0   "/usr/bin/dumb-init …"   About a minute ago   Up 43 seconds (healthy)       8080/tcp
      data-airflow-scheduler-1
41452eaca9c8   apache/airflow:2.5.0   "/usr/bin/dumb-init …"   About a minute ago   Up 43 seconds (healthy)       8080/tcp
      data-airflow-triggerer-1
7c68810fe63d   apache/airflow:2.5.0   "/usr/bin/dumb-init …"   About a minute ago   Up 43 seconds (healthy)       0.0.0.0:8080->8080/tcp
      data-airflow-webserver-1
3f92033fdcf4   apache/airflow:2.5.0   "/bin/bash -c 'funct…"   About a minute ago   Exited (0) 44 seconds ago
      data-airflow-init-1
d374c3456294   data_fast-api          "hypercorn main:app …"   About a minute ago   Up About a minute             8005/tcp, 0.0.0.0:8005->8000/
tcp   data-fast-api-1
2b635a1e6c9e   data_app               "python -u app.py"       About a minute ago   Up About a minute
      data-app-1
b6ba16f0757d   postgres:13            "docker-entrypoint.s…"   About a minute ago   Up About a minute (healthy)   5432/tcp
      data-postgres-1
fdbefd121145   redis:latest           "docker-entrypoint.s…"   About a minute ago   Up About a minute (healthy)   6379/tcp
      data-redis-1
surendrabhandari@Surendras-MacBook-Pro ~ %
```

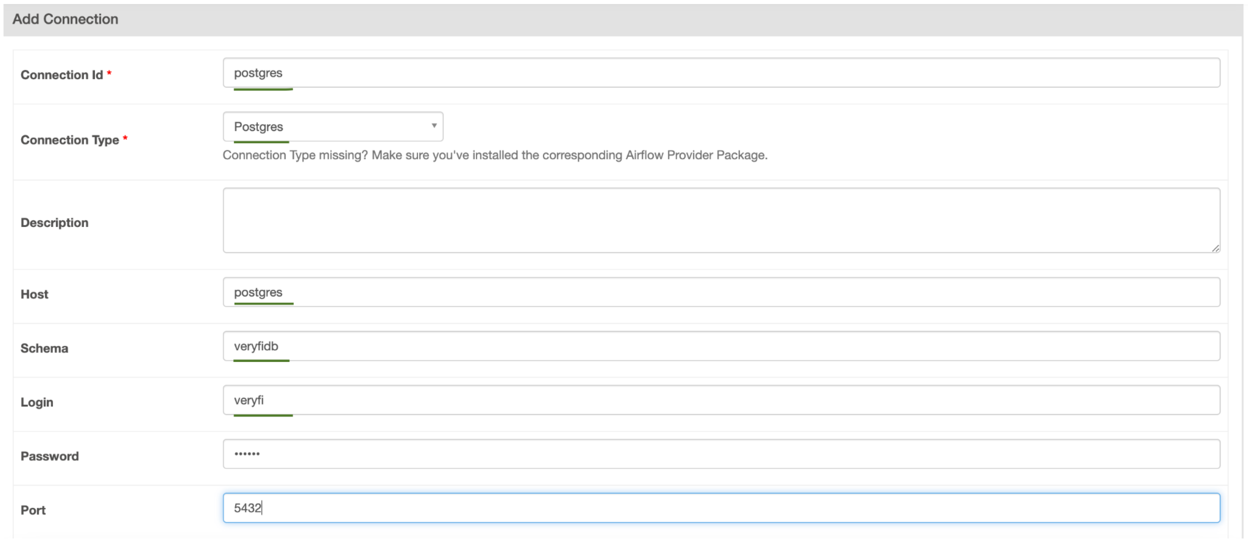
- Now verify that the data-app has already started inserting the data in the documents table. Please use the command below as shown in the screen to verify.



```
surendrabhandari@Surendras-MacBook-Pro ~ % docker exec -it data-postgres-1 /bin/bash
root@b6ba16f0757d:/# psql -U airflow
psql (13.9 (Debian 13.9-1.pgdg110+1))
Type "help" for help.

airflow=# \c veryfidb;
You are now connected to database "veryfidb" as user "airflow".
veryfidb=# \dt
          List of relations
 Schema |   Name    | Type  | Owner
--------+-----------+-------+--------
 public | documents | table | veryfi
 public | tracker   | table | veryfi
(2 rows)

veryfidb=# select count(*) from documents;
 count
-------
    54
(1 row)

veryfidb=# select count(*) from documents;
 count
-------
    57
(1 row)

veryfidb=# select * from tracker;
 batch_size
------------
          0
(1 row)

veryfidb=#
```

- First login to the data-postgres-1 docker container
- Login to postgres using **psql -U airflow**
- Verify the list of tables (documents and tracker)
- Documents table has the data for each receipt (document_id, response) format
- Tracker table is to maintain the next batch index for processing the data in a batch. It is initialized with 0 which means that we will start processing with the first document.

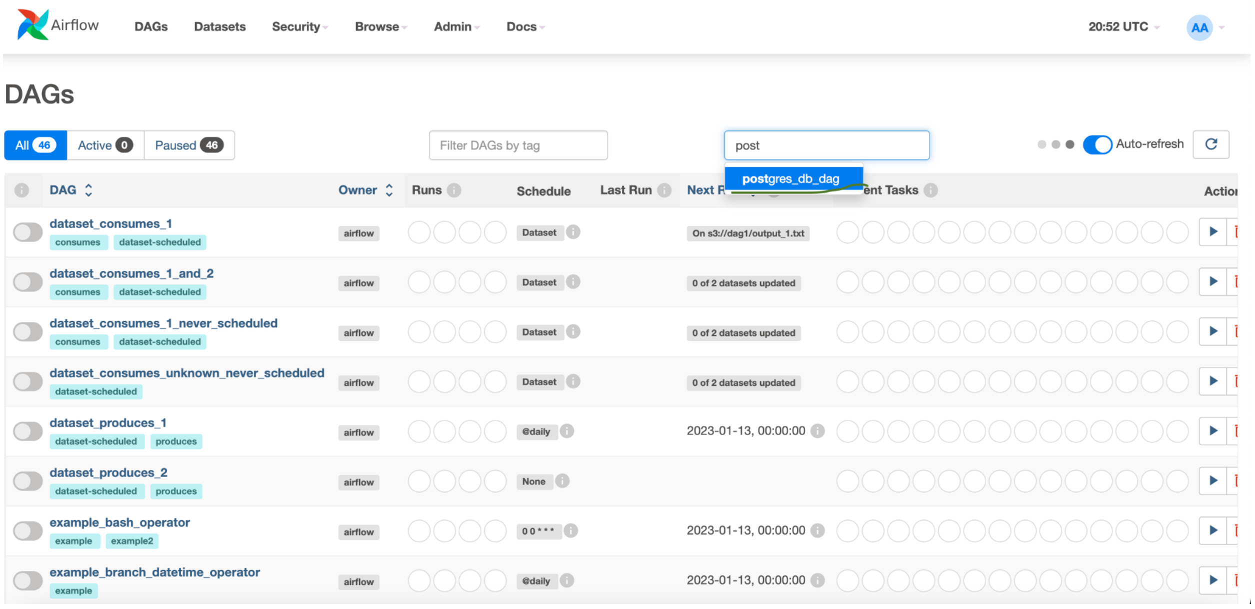
- Now we need to login to the airflow using airflow/airflow as username/password:
    - http://0.0.0.0:8080/
- Create a connection which will be used by the dag job to connect to the database.



Add Connection

Connection Id *	postgres
Connection Type *	Postgres Connection Type missing? Make sure you've installed the corresponding Airflow Provider Package.
Description	
Host	postgres
Schema	veryfidb
Login	veryfi
Password	••••••
Port	5432

- Connection Id = postgres
- Connection Type = Postgres
- Host = postgres
- Schema = veryfidb
- Login = veryfi
- Password = veryfi
- Save the connection by clicking the save button. You can test the connection using the test button too before saving it.
- Now verify the dag is present by following the dags screen and search “postgres_db_dag”, the dag that is in the dags directory in the codebase.



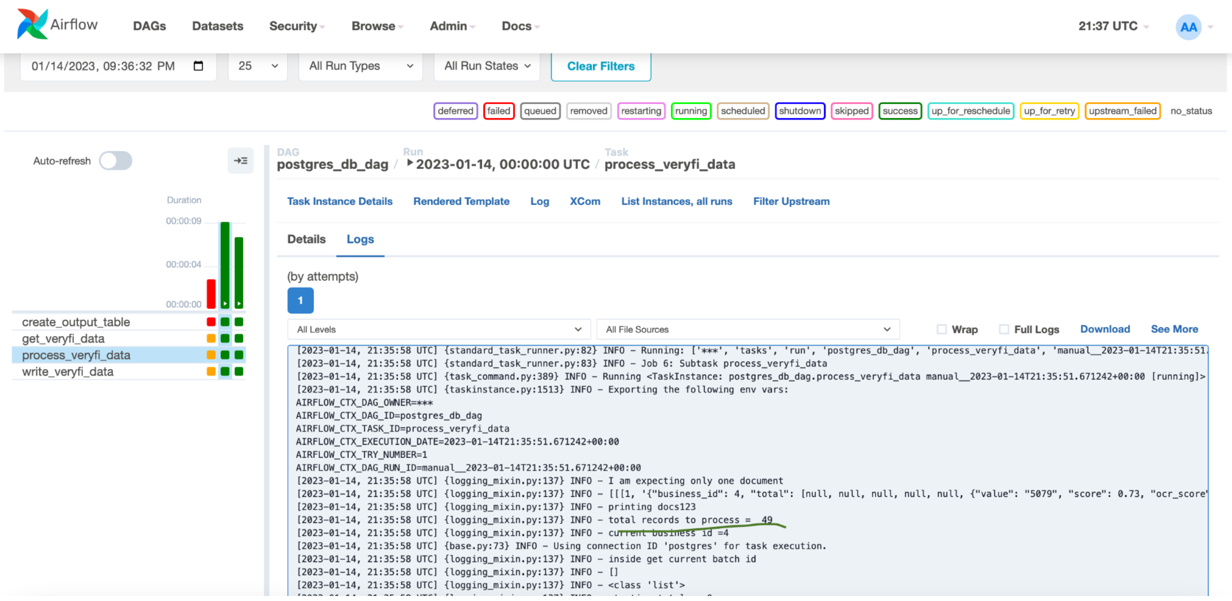
DAGs

All 46 | Active 0 | Paused 46

Filter DAGs by tag | post → postgres_db_dag

DAG	Owner	Schedule	Next Run
dataset_consumes_1 (consumes, dataset-scheduled)	airflow	Dataset	On s3://dag1/output_1.txt
dataset_consumes_1_and_2 (consumes, dataset-scheduled)	airflow	Dataset	0 of 2 datasets updated
dataset_consumes_1_never_scheduled (consumes, dataset-scheduled)	airflow	Dataset	0 of 2 datasets updated
dataset_consumes_unknown_never_scheduled (dataset-scheduled)	airflow	Dataset	0 of 2 datasets updated
dataset_produces_1 (dataset-scheduled, produces)	airflow	@daily	2023-01-13, 00:00:00
dataset_produces_2 (dataset-scheduled, produces)	airflow	None	
example_bash_operator (example, example2)	airflow	0 0 * * *	2023-01-13, 00:00:00
example_branch_datetime_operator (example)	airflow	@daily	2023-01-13, 00:00:00

- Select this job and the screen will appear like this:

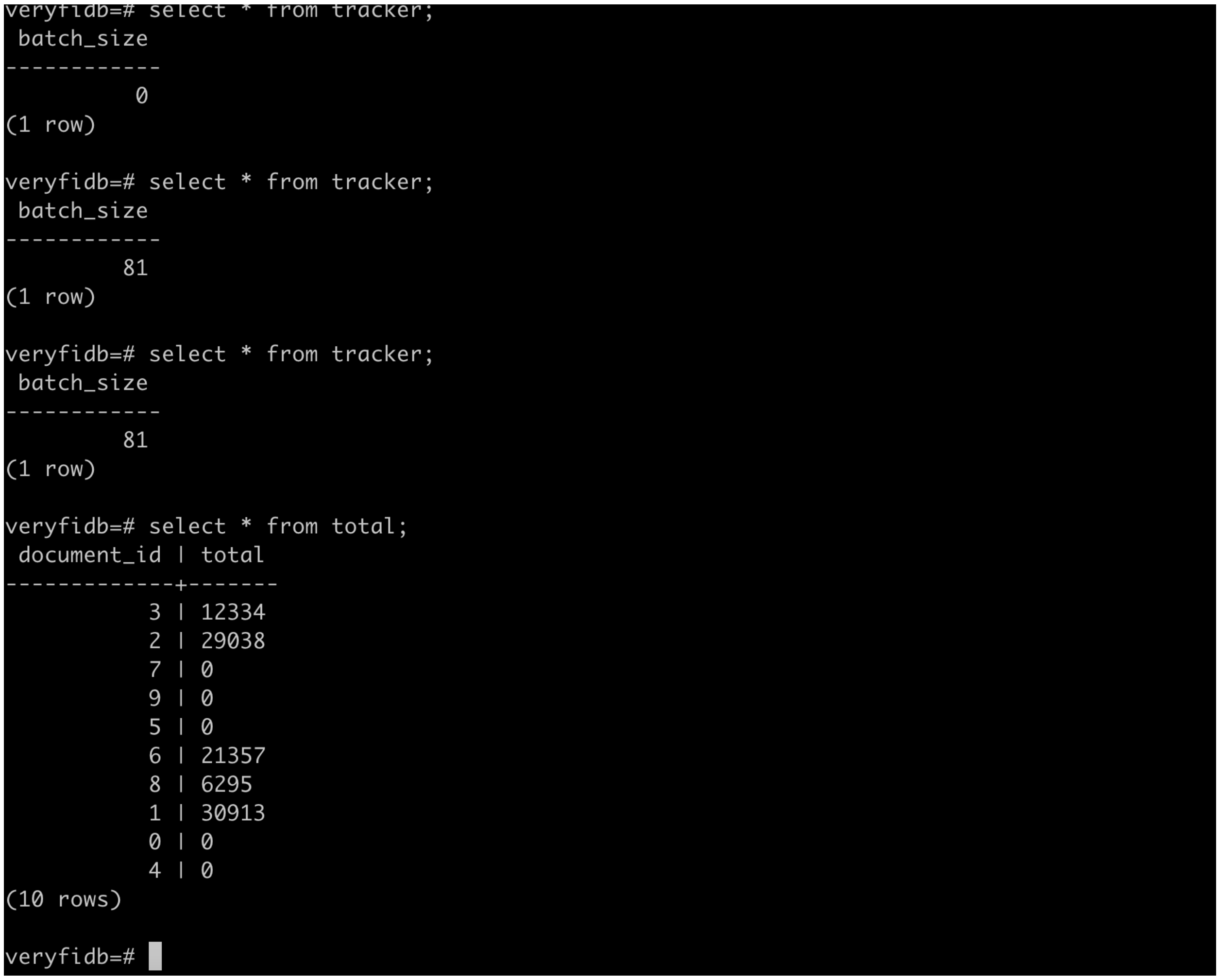


DAG postgres_db_dag / Run 2023-01-14, 00:00:00 UTC / Task process_veryfi_data

Tasks: create_output_table, get_veryfi_data, process_veryfi_data, write_veryfi_data

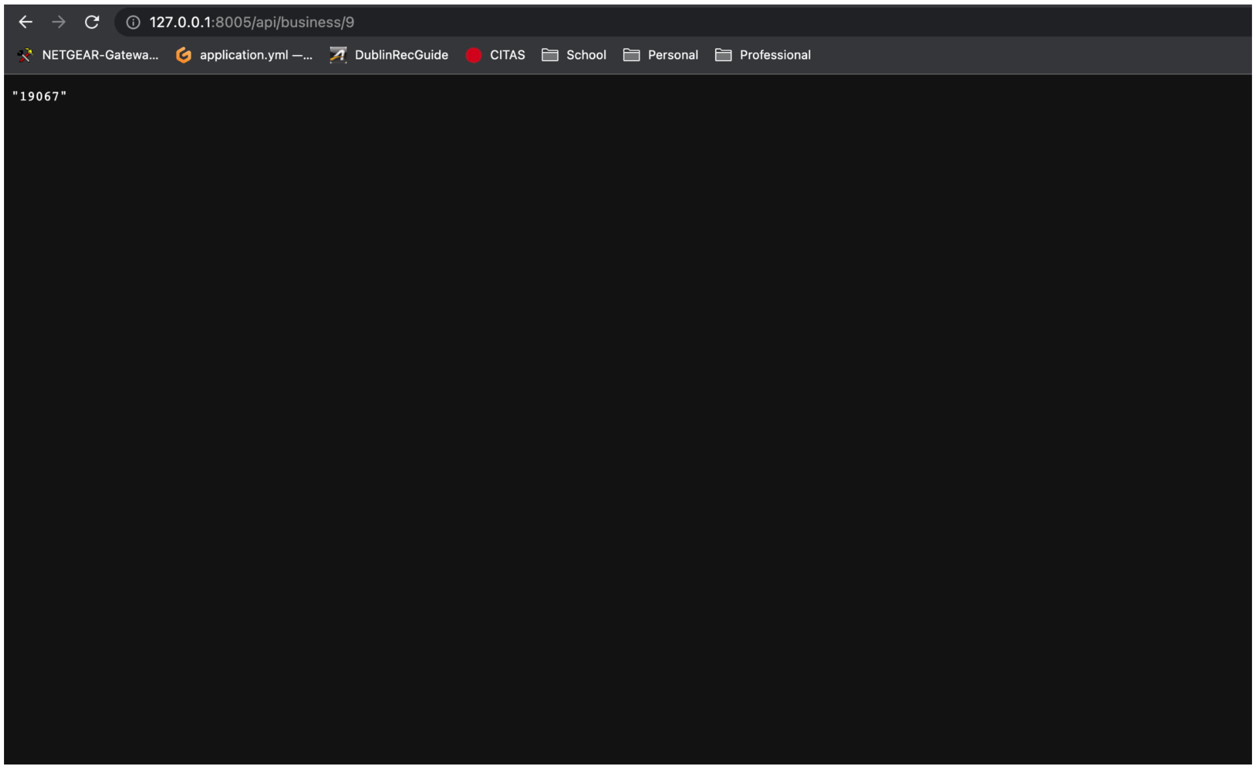
```
[2023-01-14, 21:35:58 UTC] {standard_task_runner.py:82} INFO - Running: ['***', 'tasks', 'run', 'postgres_db_dag', 'process_veryfi_data', 'manual__2023-01-14T21:35:51.
[2023-01-14, 21:35:58 UTC] {standard_task_runner.py:83} INFO - Job 6: Subtask process_veryfi_data
[2023-01-14, 21:35:58 UTC] {task_command.py:389} INFO - Running <TaskInstance: postgres_db_dag.process_veryfi_data manual__2023-01-14T21:35:51.671242+00:00 [running]>
[2023-01-14, 21:35:58 UTC] {taskinstance.py:1513} INFO - Exporting the following env vars:
AIRFLOW_CTX_DAG_OWNER=***
AIRFLOW_CTX_DAG_ID=postgres_db_dag
AIRFLOW_CTX_TASK_ID=process_veryfi_data
AIRFLOW_CTX_EXECUTION_DATE=2023-01-14T21:35:51.671242+00:00
AIRFLOW_CTX_TRY_NUMBER=1
AIRFLOW_CTX_DAG_RUN_ID=manual__2023-01-14T21:35:51.671242+00:00
[2023-01-14, 21:35:58 UTC] {logging_mixin.py:137} INFO - I am expecting only one document
[2023-01-14, 21:35:58 UTC] {logging_mixin.py:137} INFO - [[[1, '{"business_id": 4, "total": [null, null, null, null, null, {"value": "5079", "score": 0.73, "ocr_score"
[2023-01-14, 21:35:58 UTC] {logging_mixin.py:137} INFO - printing docs123
[2023-01-14, 21:35:58 UTC] {logging_mixin.py:137} INFO - total records to process =  49
[2023-01-14, 21:35:58 UTC] {logging_mixin.py:137} INFO - current business id =4
[2023-01-14, 21:35:58 UTC] {base.py:73} INFO - Using connection ID 'postgres' for task execution.
[2023-01-14, 21:35:58 UTC] {logging_mixin.py:137} INFO - inside get current batch id
[2023-01-14, 21:35:58 UTC] {logging_mixin.py:137} INFO - []
[2023-01-14, 21:35:58 UTC] {logging_mixin.py:137} INFO - <class 'list'>
```

- It shows the tasks which are implemented in the "**postgres_db_dag.py**" file.
- Now click on run and it will run the job and process the records.
- After it process the data, we can verify the following tables in the database



```
veryfidb=# select * from tracker;
 batch_size
------------
          0
(1 row)

veryfidb=# select * from tracker;
 batch_size
------------
         81
(1 row)

veryfidb=# select * from tracker;
 batch_size
------------
         81
(1 row)

veryfidb=# select * from total;
 document_id | total
-------------+-------
           3 | 12334
           2 | 29038
           7 | 0
           9 | 0
           5 | 0
           6 | 21357
           8 | 6295
           1 | 30913
           0 | 0
           4 | 0
(10 rows)

veryfidb=# 
```

- It is clear from the above screenshot that now the tracker is pointing to 81 as the next document to start processing from.
- Total table has the data for all business_id(s) and their total value
- If you run the job again, you will make a change in both the tracker as well as the total table because we would have processed more records.
- At this point we can verify the same information from the fast-api endpoint too.
  - If we run http://127.0.0.1:8005/ we will get {“Hello” : “Veryfi!!”} sample response
  - And if we run the endpoint as http://127.0.0.1:8005/api/business/9 it will print the total value of the business_id which will be exactly equal to one present in the total table above.



```
"19067"
```

After this setup and implementation, I think we can add more features, improve what is already there, and experience many more exciting things. If you would like to go into more detail about the strategy and solution, perhaps we can talk about those. If you have any questions, feel free to ask for clarification, and I would be pleased to respond.