Week 2 Homework: Workflow Orchestration and Observation

Georgios Grigoriou
georgiosvgrigoriou@gmail.com
Deadline: 06 February (Monday), 22:00 CET

In this homework, I prepared the environment and practice with Prefect, a workflow orchestration tool as part of the Data Engineering Zoomcamp course. Solutions' headings are in italic and highlighted in yellow under each Question's instruction.

The code that was used can be found here

Week 2 Homework

The goal of this homework is to familiarise myself with workflow orchestration and observation.

Question 1. Load January 2020 data

Using the `etl_web_to_gcs.py` flow that loads taxi data into GCS as a guide, create a flow that loads the green taxi CSV dataset for January 2020 into GCS and run it. Look at the logs to find out how many rows the dataset has.

How many rows does that dataset have?

- **447,770**
- 766,792
- 299,234
- 822,132

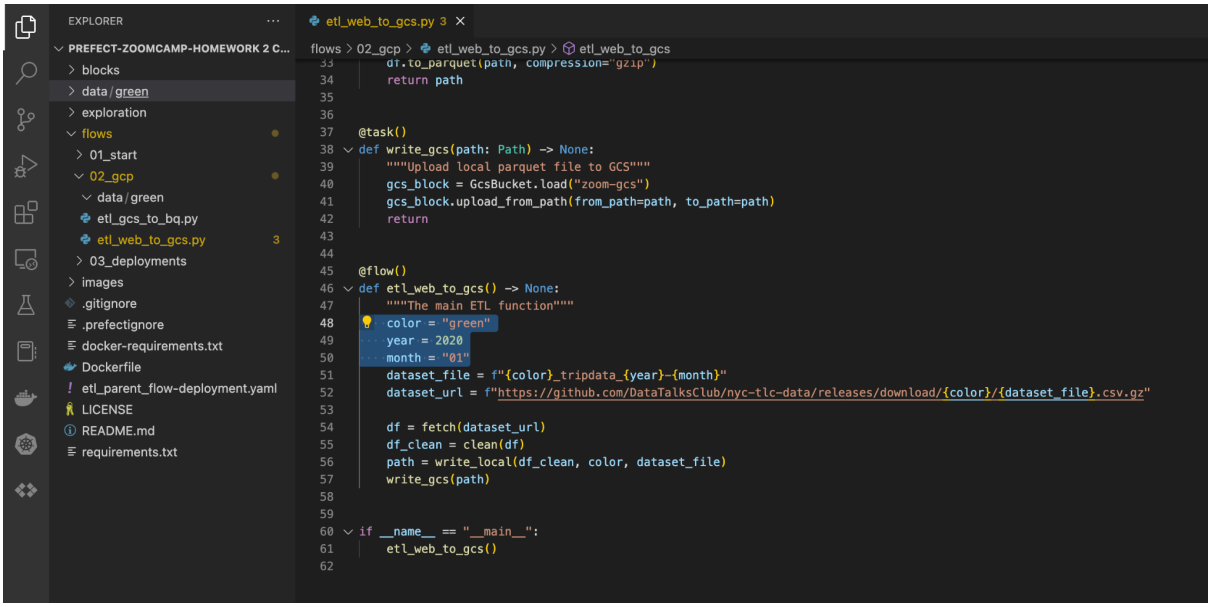
Question 1. Load January 2020 data Prep and Solution

Using the etl_web_to_gcs.py file we configure the flow as follows:

color="green"

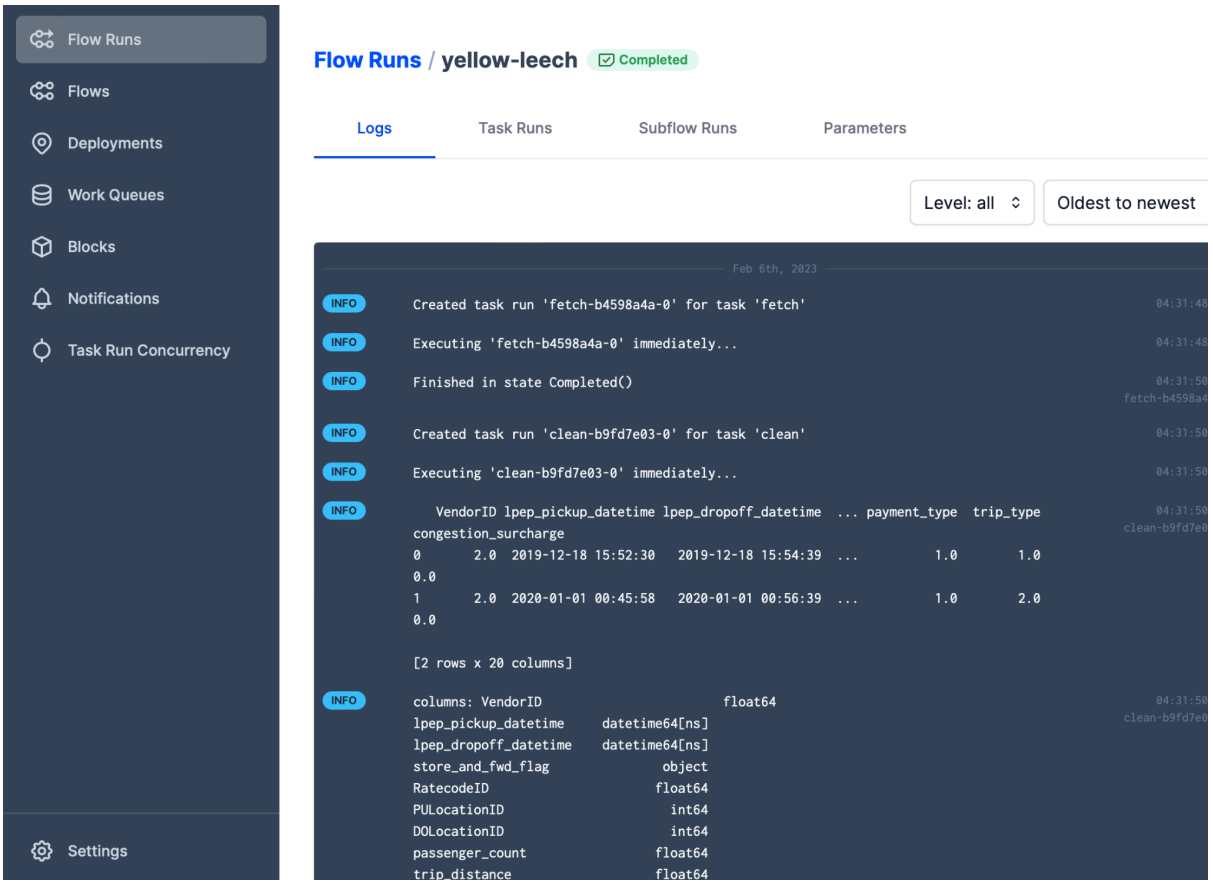
year="2020"

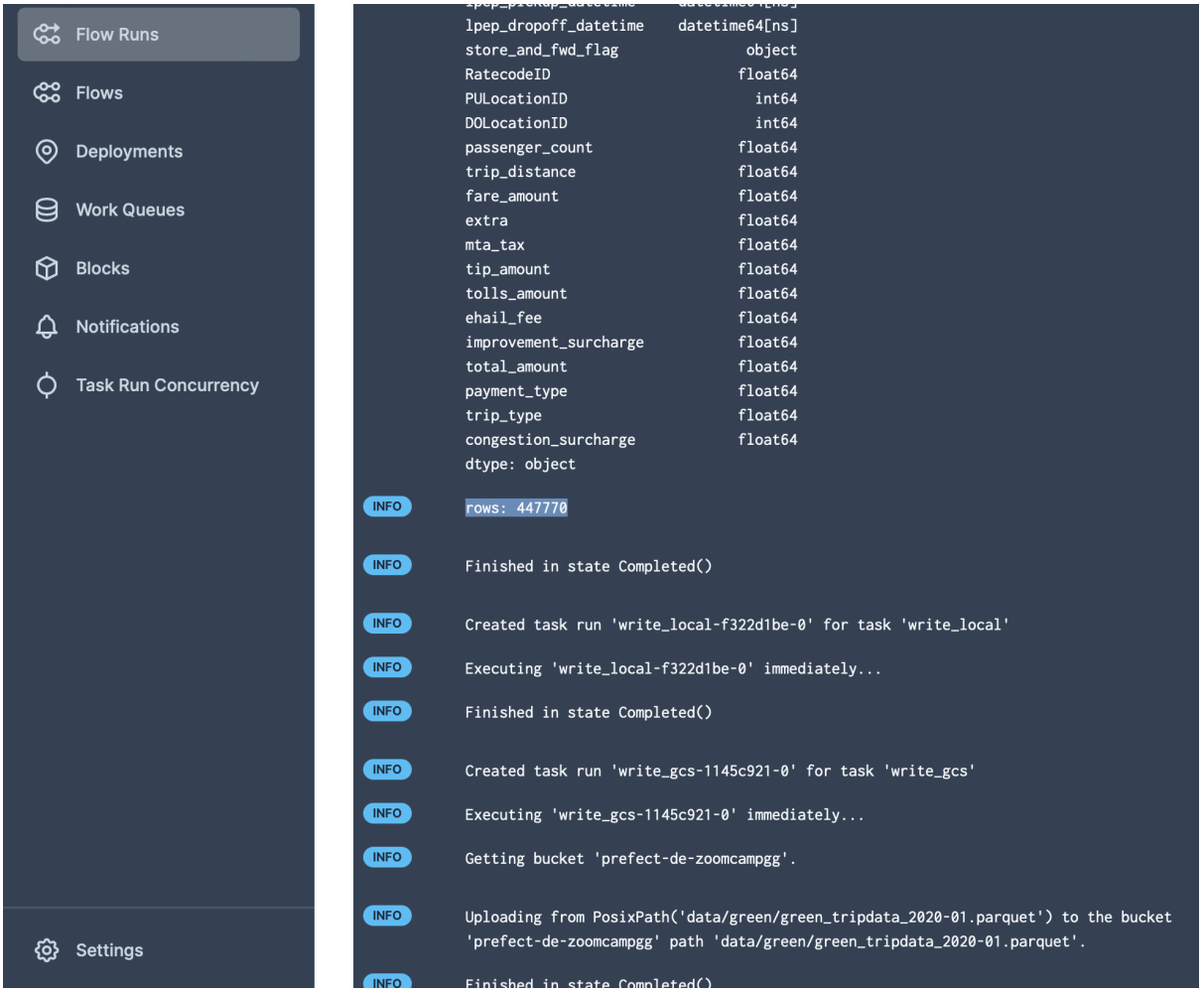
month="01"

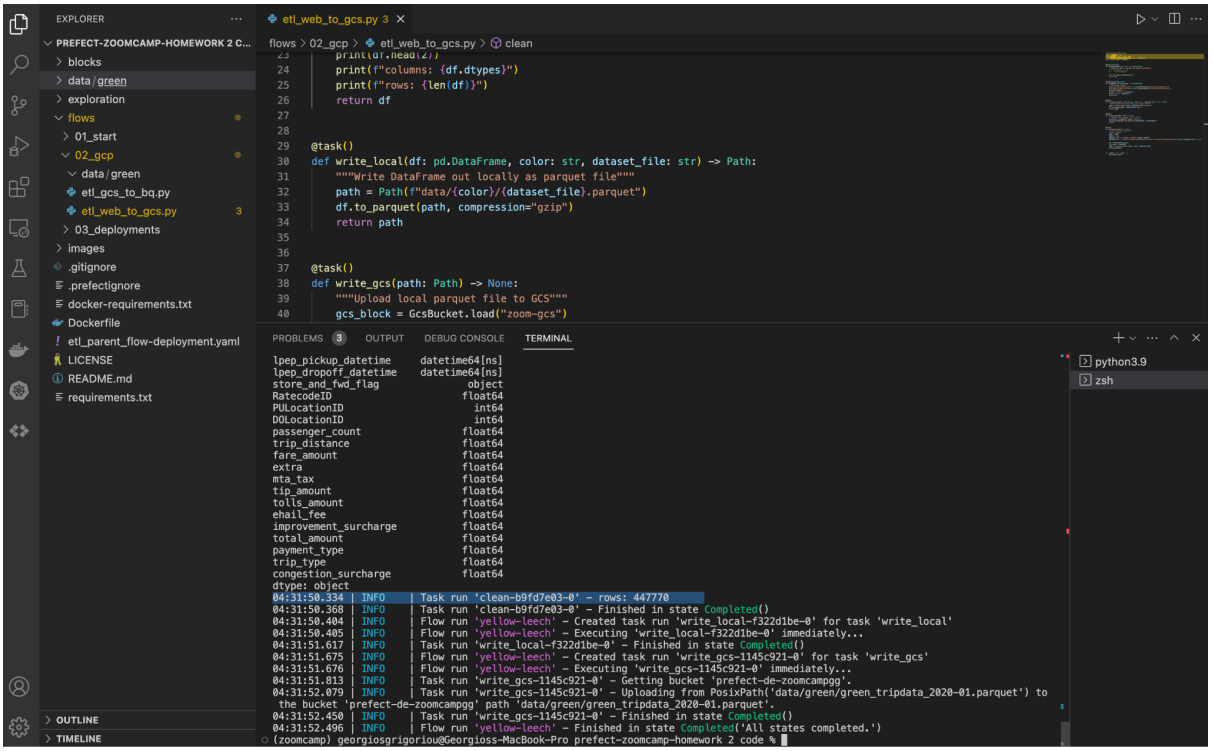


I found out that the correct answer is **447,770**

The figures below show the required information about the number of rows from the Logs Section of Prefect UI and the terminal of Visual Studio Code respectively.




Flow Runs
Flows
Deployments
Work Queues
Blocks
Notifications
Task Run Concurrency
Settings
lpep_dropoff_datetime datetime64[ns]
store_and_fwd_flag object
RatecodeID float64
PULocationID int64
DOLocationID int64
passenger_count float64
trip_distance float64
fare_amount float64
extra float64
mta_tax float64
tip_amount float64
tolls_amount float64
ehail_fee float64
improvement_surcharge float64
total_amount float64
payment_type float64
trip_type float64
congestion_surcharge float64
dtype: object
INFO rows: 447770
INFO Finished in state Completed()
INFO Created task run 'write_local-f322d1be-0' for task 'write_local'
INFO Executing 'write_local-f322d1be-0' immediately...
INFO Finished in state Completed()
INFO Created task run 'write_gcs-1145c921-0' for task 'write_gcs'
INFO Executing 'write_gcs-1145c921-0' immediately...
INFO Getting bucket 'prefect-de-zoomcampgg'.
INFO Uploading from PosixPath('data/green/green_tripdata_2020-01.parquet') to the bucket 'prefect-de-zoomcampgg' path 'data/green/green_tripdata_2020-01.parquet'.


EXPLORER
etl_web_to_gcs.py 3
PREFECT-ZOOMCAMP-HOMEWORK 2 C...
blocks
data / green
exploration
flows
01_start
02_gcp
data / green
etl_gcs_to_bq.py
etl_web_to_gcs.py 3
03_deployments
images
.gitignore
.prefectignore
docker-requirements.txt
Dockerfile
etl_parent_flow-deployment.yaml
LICENSE
README.md
requirements.txt
flows > 02_gcp > etl_web_to_gcs.py > clean
print(f"columns: {df.dtypes}")
print(f"rows: {len(df)}")
return df
@task()
def write_local(df: pd.DataFrame, color: str, dataset_file: str) -> Path:
"""Write DataFrame out locally as parquet file"""
path = Path(f"data/{color}/{dataset_file}.parquet")
df.to_parquet(path, compression="gzip")
return path
@task()
def write_gcs(path: Path) -> None:
"""Upload local parquet file to GCS"""
gcs_block = GcsBucket.load("zoom-gcs")
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL
python3.9
zsh
lpep_pickup_datetime datetime64[ns]
lpep_dropoff_datetime datetime64[ns]
store_and_fwd_flag object
RatecodeID float64
PULocationID int64
DOLocationID int64
passenger_count float64
trip_distance float64
fare_amount float64
extra float64
mta_tax float64
tip_amount float64
tolls_amount float64
ehail_fee float64
improvement_surcharge float64
total_amount float64
payment_type float64
trip_type float64
congestion_surcharge float64
dtype: object
04:31:50.334 | INFO | Task run 'clean-b9fd7e03-0' - rows: 447770
04:31:50.368 | INFO | Task run 'clean-b9fd7e03-0' - Finished in state Completed()
04:31:50.404 | INFO | Flow run 'yellow-leech' - Created task run 'write_local-f322d1be-0' for task 'write_local'
04:31:50.405 | INFO | Flow run 'yellow-leech' - Executing 'write_local-f322d1be-0' immediately...
04:31:51.617 | INFO | Task run 'write_local-f322d1be-0' - Finished in state Completed()
04:31:51.675 | INFO | Flow run 'yellow-leech' - Created task run 'write_gcs-1145c921-0' for task 'write_gcs'
04:31:51.676 | INFO | Flow run 'yellow-leech' - Executing 'write_gcs-1145c921-0' immediately...
04:31:51.813 | INFO | Task run 'write_gcs-1145c921-0' - Getting bucket 'prefect-de-zoomcampgg'.
04:31:52.079 | INFO | Task run 'write_gcs-1145c921-0' - Uploading from PosixPath('data/green/green_tripdata_2020-01.parquet') to the bucket 'prefect-de-zoomcampgg' path 'data/green/green_tripdata_2020-01.parquet'.
04:31:52.450 | INFO | Task run 'write_gcs-1145c921-0' - Finished in state Completed()
04:31:52.496 | INFO | Flow run 'yellow-leech' - Finished in state Completed('All states completed.')
(zoomcamp) georgiosgrigoriou@Georgioss-MacBook-Pro prefect-zoomcamp-homework 2 code %
OUTLINE
TIMELINE

Question 2. Scheduling with Cron

Cron is a common scheduling specification for workflows.

Using the flow in `etl_web_to_gcs.py`, create a deployment to run on the first of every month at 5am UTC. What's the cron schedule for that?

- 0 5 1 * *
- 0 0 5 1 *
- 5 * 1 0 *
- * * 5 1 0

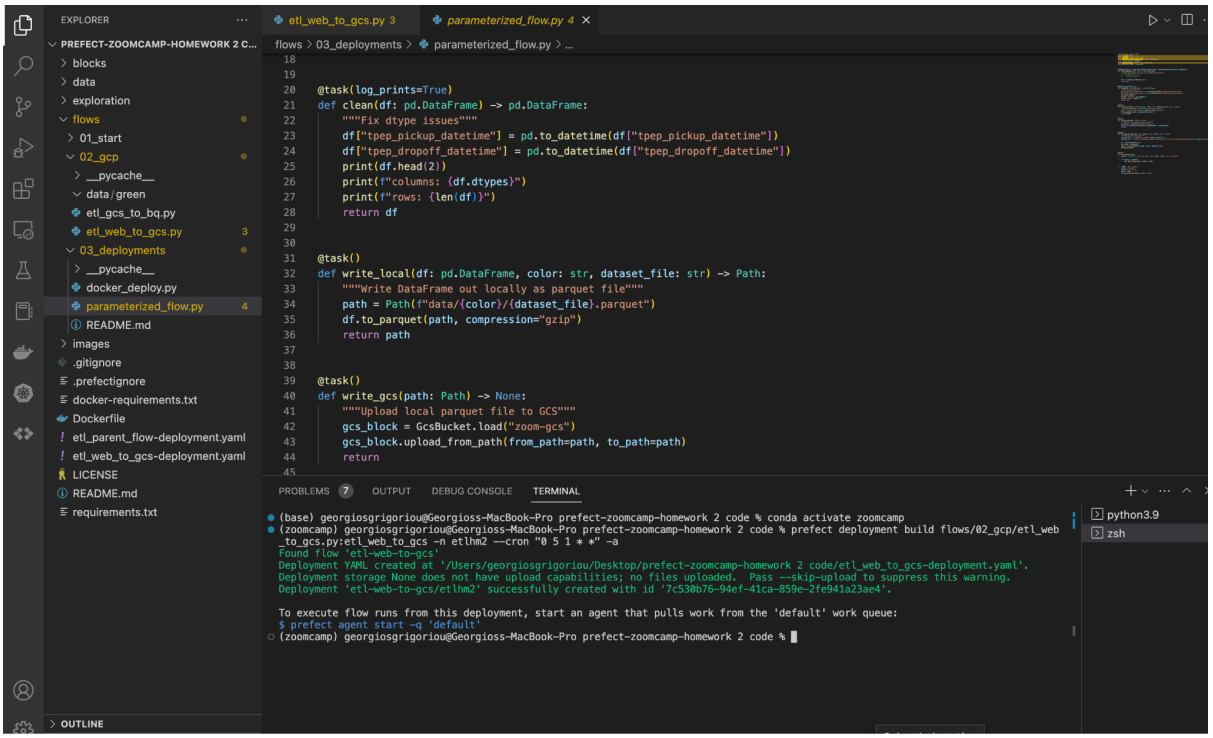
Question 2. Scheduling with Cron Solution

The correct answer is 0 5 1 * *.

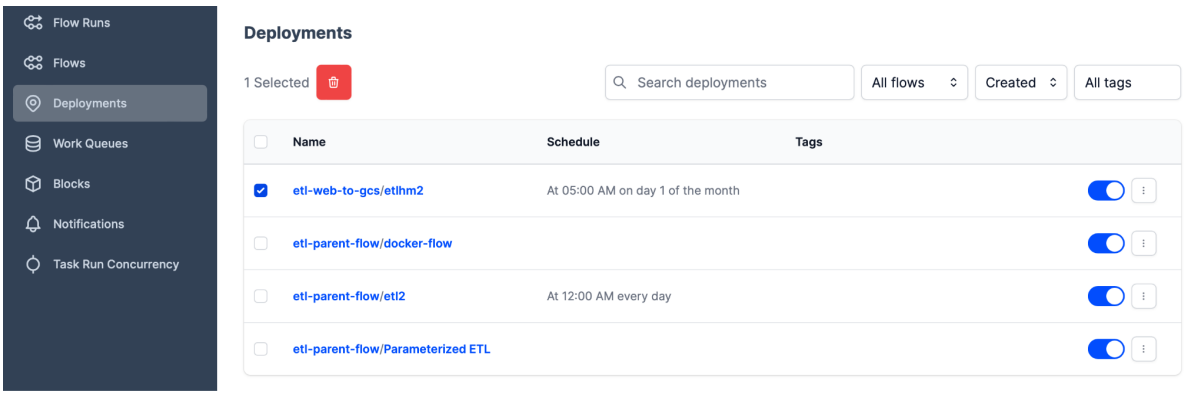
To get that, I ran the following command

*prefect deployment build flows/02_gcp/etl_web_to_gcs.py:etl_web_to_gcs -n etlhm2 --cron "0 5 1 * *" -a*

From the terminal in my virtual environment at Visual Studio Code, as illustrated in the figure below



Now, if we go to my local Prefect UI, I can see the deployment named ethhm2, which was scheduled to run at 05:00 AM on day 1 of the month



Question 3. Loading data to BigQuery

Using `etl_gcs_to_bq.py` as a starting point, modify the script for extracting data from GCS and loading it into BigQuery. This new script should not fill or remove rows with missing values. (The script is really just doing the E and L parts of ETL).

The main flow should print the total number of rows processed by the script. Set the flow decorator to log the print statement.

Parametrize the entrypoint flow to accept a list of months, a year, and a taxi color.

Make any other necessary changes to the code for it to function as required.

Create a deployment for this flow to run in a local subprocess with local flow code storage (the defaults).

Make sure you have the parquet data files for Yellow taxi data for Feb. 2019 and March 2019 loaded in GCS. Run your deployment to append this data to your BiqQuery table. How many rows did your flow code process?

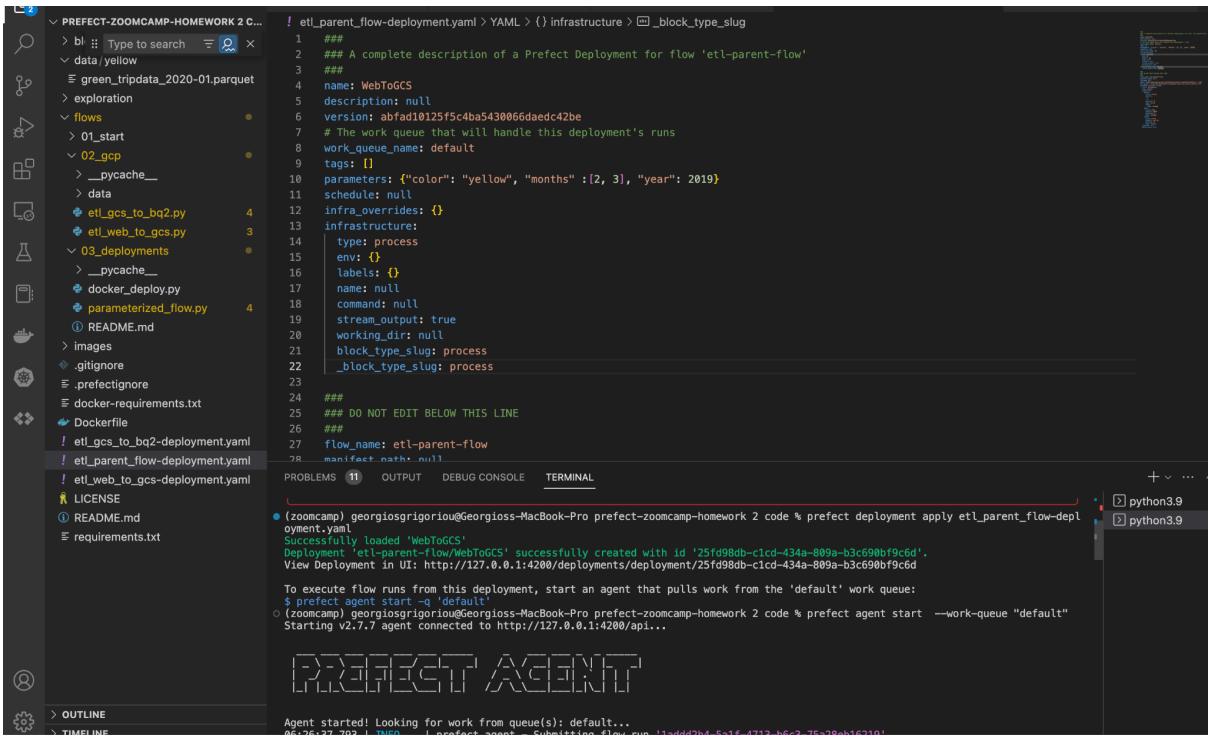
- **14,851,920**
- 12,282,990
- 27,235,753
- 11,338,483

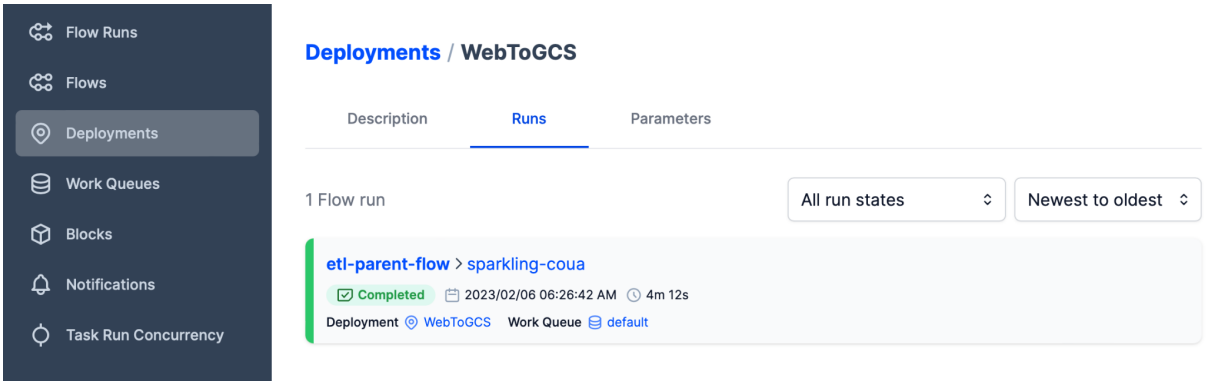
Question 3. Loading data to BigQuery Solution

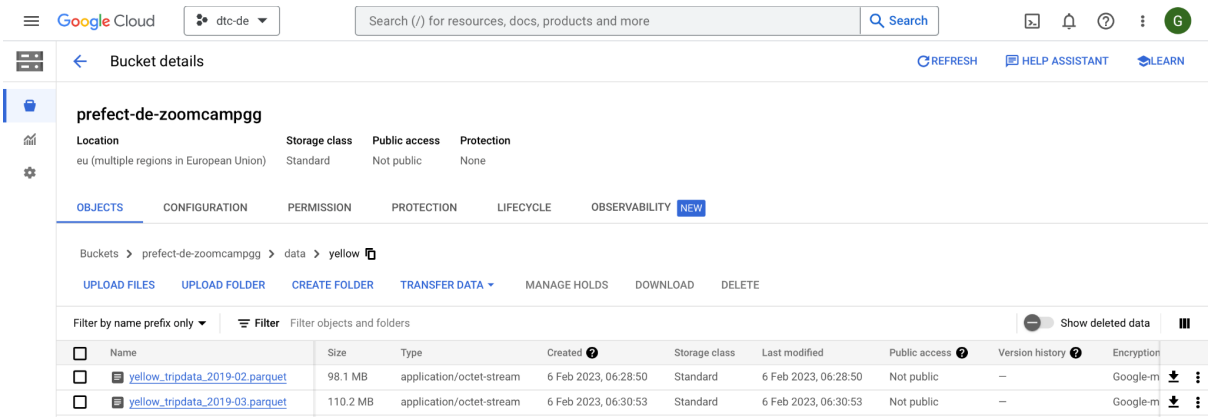
The correct answer is 14,851,920. Here are the Steps that were followed:

Step 1: Store parquet data files for Yellow taxi data for Feb. 2019 and March 2019 loaded in GCS.

To do that I ran a deployment called WebToGCS and apply the appropriate changes in the parameter key of the yaml file







Terminal commands (in *Italic)*

(zoomcamp) georgiosgrigoriou@Georgioss-MacBook-Pro prefect-zoomcamp-homework 2 code % *prefect deployment build flows/03_deployments/parameterized_flow.py:etl_parent_flow -n "WebToGCS"*

Found flow 'etl-parent-flow'

Deployment YAML created at '/Users/georgiosgrigoriou/Desktop/prefect-zoomcamp-homework 2 code/etl_parent_flow-deployment.yaml'.

Deployment storage None does not have upload capabilities; no files uploaded. Pass --skip-upload to suppress this warning.

(zoomcamp) georgiosgrigoriou@Georgioss-MacBook-Pro prefect-zoomcamp-homework 2 code % prefect deployment apply etl_parent_flow-deployment.yaml

Successfully loaded 'WebToGCS'

Deployment 'etl-parent-flow/WebToGCS' successfully created with id '25fd98db-c1cd-434a-809a-b3c690bf9c6d'.

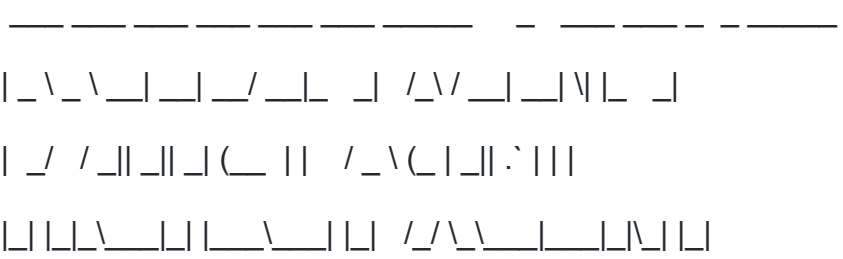
View Deployment in UI: http://127.0.0.1:4200/deployments/deployment/25fd98db-c1cd-434a-809a-b3c690bf9c6d

To execute flow runs from this deployment, start an agent that pulls work from the 'default' work queue:

$ prefect agent start -q 'default'

(zoomcamp) georgiosgrigoriou@Georgioss-MacBook-Pro prefect-zoomcamp-homework 2 code % **prefect agent start --work-queue "default"**

Starting v2.7.7 agent connected to http://127.0.0.1:4200/api...



Agent started! Looking for work from queue(s): default...

06:26:37.793 | INFO | prefect.agent - Submitting flow run '1addd2b4-5a1f-4713-b6c3-75a28eb16219'

06:26:37.872 | INFO | prefect.infrastructure.process - Opening process 'sparkling-coua'...

06:26:37.902 | INFO | prefect.agent - Completed submission of flow run '1addd2b4-5a1f-4713-b6c3-75a28eb16219'

/opt/anaconda3/envs/zoomcamp/lib/python3.9/runpy.py:127: RuntimeWarning: 'prefect.engine' found in sys.modules after import of package 'prefect', but prior to execution of 'prefect.engine'; this may result in unpredictable behaviour

  warn(RuntimeWarning(msg))

06:26:42.603 | INFO | Flow run 'sparkling-coua' - Downloading flow code from storage at '/Users/georgiosgrigoriou/Desktop/prefect-zoomcamp-homework 2 code'

06:26:42.972 | INFO | Flow run 'sparkling-coua' - Created subflow run 'practical-armadillo' for flow 'etl-web-to-gcs'

06:26:43.050 | INFO | Flow run 'practical-armadillo' - Created task run 'fetch-ba00c645-0' for task 'fetch'

06:26:43.051 | INFO | Flow run 'practical-armadillo' - Executing 'fetch-ba00c645-0' immediately...

06:28:22.554 | INFO | Task run 'fetch-ba00c645-0' - Finished in state Completed()

06:28:22.621 | INFO | Flow run 'practical-armadillo' - Created task run 'clean-2c6af9f6-0' for task 'clean'

06:28:22.621 | INFO | Flow run 'practical-armadillo' - Executing 'clean-2c6af9f6-0' immediately...

06:28:27.401 | INFO | Task run 'clean-2c6af9f6-0' - VendorID tpep_pickup_datetime ... total_amount congestion_surcharge

0 1 2019-02-01 00:59:04 ... 12.3 0.0

1 1 2019-02-01 00:33:09 ... 33.3 0.0

[2 rows x 18 columns]

06:28:27.404 | INFO | Task run 'clean-2c6af9f6-0' - columns: VendorID int64

tpep_pickup_datetime datetime64[ns]

tpep_dropoff_datetime datetime64[ns]

passenger_count int64

trip_distance float64

RatecodeID int64

store_and_fwd_flag object

PULocationID int64

DOLocationID int64

payment_type int64

fare_amount float64

extra float64

mta_tax float64

tip_amount float64

tolls_amount float64

improvement_surcharge float64

total_amount float64

congestion_surcharge float64

dtype: object

06:28:27.405 | INFO | Task run 'clean-2c6af9f6-0' - rows: 7019375

06:28:27.430 | INFO | Task run 'clean-2c6af9f6-0' - Finished in state Completed()

06:28:27.457 | INFO | Flow run 'practical-armadillo' - Created task run 'write_local-09e9d2b8-0' for task 'write_local'

06:28:27.457 | INFO | Flow run 'practical-armadillo' - Executing 'write_local-09e9d2b8-0' immediately...

06:28:48.256 | INFO | Task run 'write_local-09e9d2b8-0' - Finished in state Completed()

06:28:48.281 | INFO | Flow run 'practical-armadillo' - Created task run 'write_gcs-67f8f48e-0' for task 'write_gcs'

06:28:48.282 | INFO | Flow run 'practical-armadillo' - Executing 'write_gcs-67f8f48e-0' immediately...

06:28:48.411 | INFO | Task run 'write_gcs-67f8f48e-0' - Getting bucket 'prefect-de-zoomcampgg'.

06:28:48.676 | INFO | Task run 'write_gcs-67f8f48e-0' - Uploading from PosixPath('data/yellow/yellow_tripdata_2019-02.parquet') to the bucket 'prefect-de-zoomcampgg' path 'data/yellow/yellow_tripdata_2019-02.parquet'.

06:28:50.585 | INFO | Task run 'write_gcs-67f8f48e-0' - Finished in state Completed()

06:28:50.632 | INFO | Flow run 'practical-armadillo' - Finished in state Completed('All states completed.')

06:28:50.749 | INFO | Flow run 'sparkling-coua' - Created subflow run 'poetic-leech' for flow 'etl-web-to-gcs'

06:28:50.826 | INFO | Flow run 'poetic-leech' - Created task run 'fetch-ba00c645-0' for task 'fetch'

06:28:50.827 | INFO | Flow run 'poetic-leech' - Executing 'fetch-ba00c645-0' immediately...

06:30:23.751 | INFO | Task run 'fetch-ba00c645-0' - Finished in state Completed()

06:30:23.823 | INFO | Flow run 'poetic-leech' - Created task run 'clean-2c6af9f6-0' for task 'clean'

06:30:23.824 | INFO | Flow run 'poetic-leech' - Executing 'clean-2c6af9f6-0' immediately...

06:30:28.670 | INFO | Task run 'clean-2c6af9f6-0' - VendorID tpep_pickup_datetime ... total_amount congestion_surcharge

0 1 2019-03-01 00:24:41 ... 3.8 0.0

1 1 2019-03-01 00:25:27 ... 15.0 0.0

[2 rows x 18 columns]

06:30:28.672 | INFO | Task run 'clean-2c6af9f6-0' - columns: VendorID int64

tpep_pickup_datetime datetime64[ns]

tpep_dropoff_datetime datetime64[ns]

passenger_count int64

trip_distance float64

RatecodeID int64

store_and_fwd_flag object

PULocationID int64

DOLocationID int64

payment_type int64

fare_amount float64

extra float64

mta_tax float64

tip_amount float64

tolls_amount float64

improvement_surcharge float64

total_amount float64

congestion_surcharge float64

dtype: object

06:30:28.673 | INFO | Task run 'clean-2c6af9f6-0' - rows: 7832545

06:30:28.696 | INFO | Task run 'clean-2c6af9f6-0' - Finished in state Completed()

06:30:28.724 | INFO | Flow run 'poetic-leech' - Created task run 'write_local-09e9d2b8-0' for task 'write_local'

06:30:28.725 | INFO | Flow run 'poetic-leech' - Executing 'write_local-09e9d2b8-0' immediately...

06:30:49.796 | INFO | Task run 'write_local-09e9d2b8-0' - Finished in state Completed()

06:30:49.821 | INFO | Flow run 'poetic-leech' - Created task run 'write_gcs-67f8f48e-0' for task 'write_gcs'

06:30:49.822 | INFO | Flow run 'poetic-leech' - Executing 'write_gcs-67f8f48e-0' immediately...

06:30:49.954 | INFO | Task run 'write_gcs-67f8f48e-0' - Getting bucket 'prefect-de-zoomcampgg'.

06:30:50.185 | INFO | Task run 'write_gcs-67f8f48e-0' - Uploading from PosixPath('data/yellow/yellow_tripdata_2019-03.parquet') to the bucket 'prefect-de-zoomcampgg' path 'data/yellow/yellow_tripdata_2019-03.parquet'.

06:30:53.882 | INFO | Task run 'write_gcs-67f8f48e-0' - Finished in state Completed()

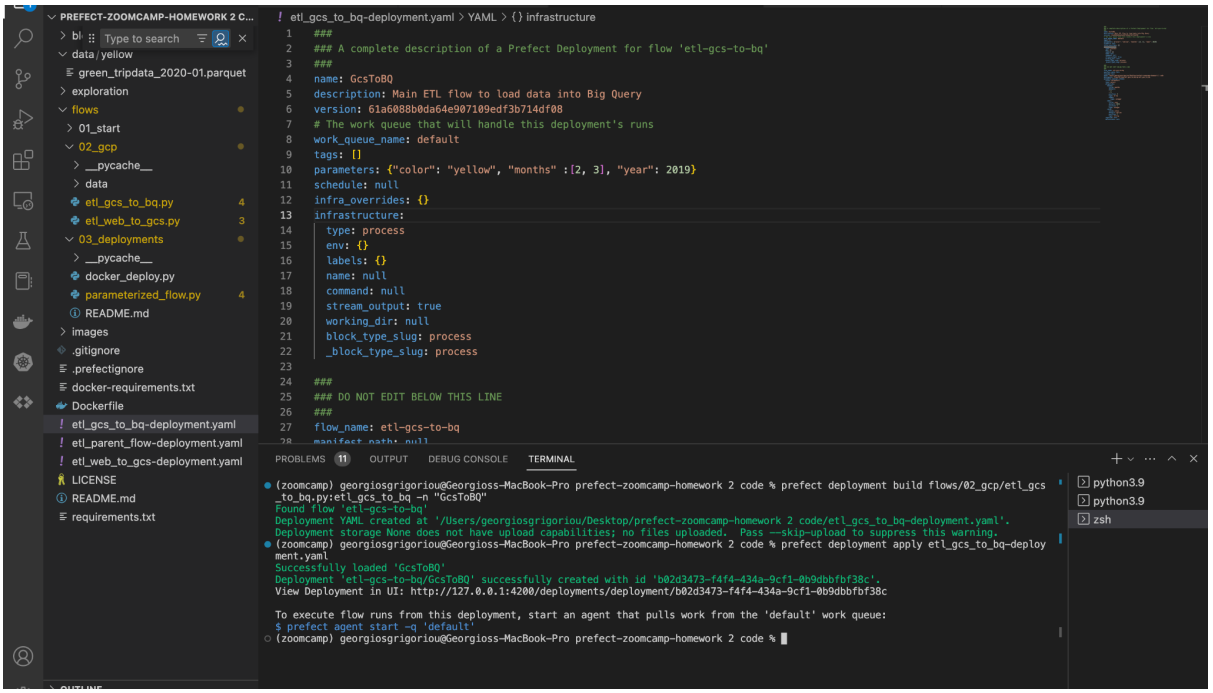
06:30:53.929 | INFO | Flow run 'poetic-leech' - Finished in state Completed('All states completed.')

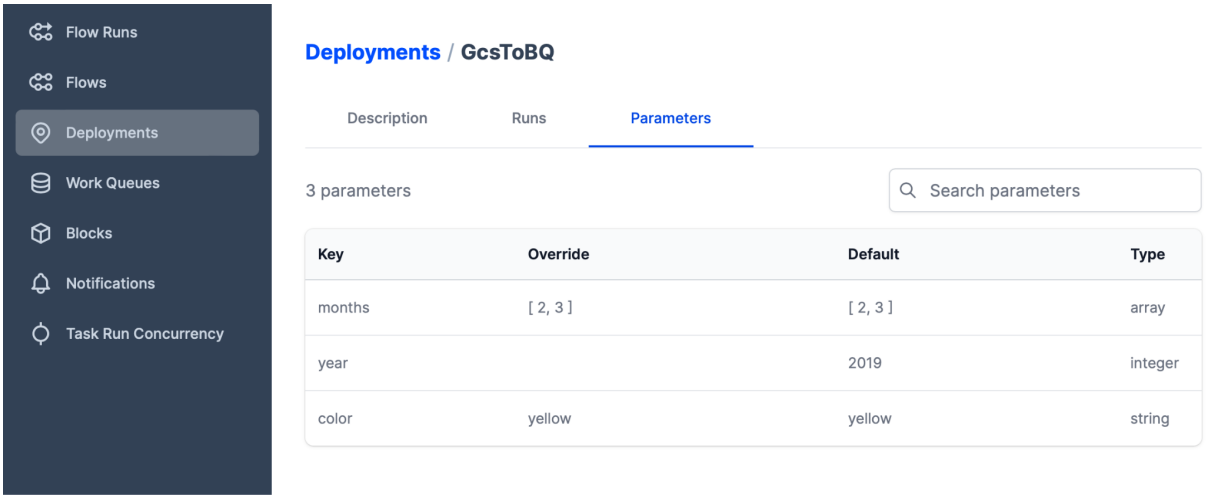
06:30:53.958 | INFO | Flow run 'sparkling-coua' - Finished in state Completed('All states completed.')

06:30:54.888 | INFO | prefect.infrastructure.process - Process 'sparkling-coua' exited cleanly.

Step 2: gcp to bq.py

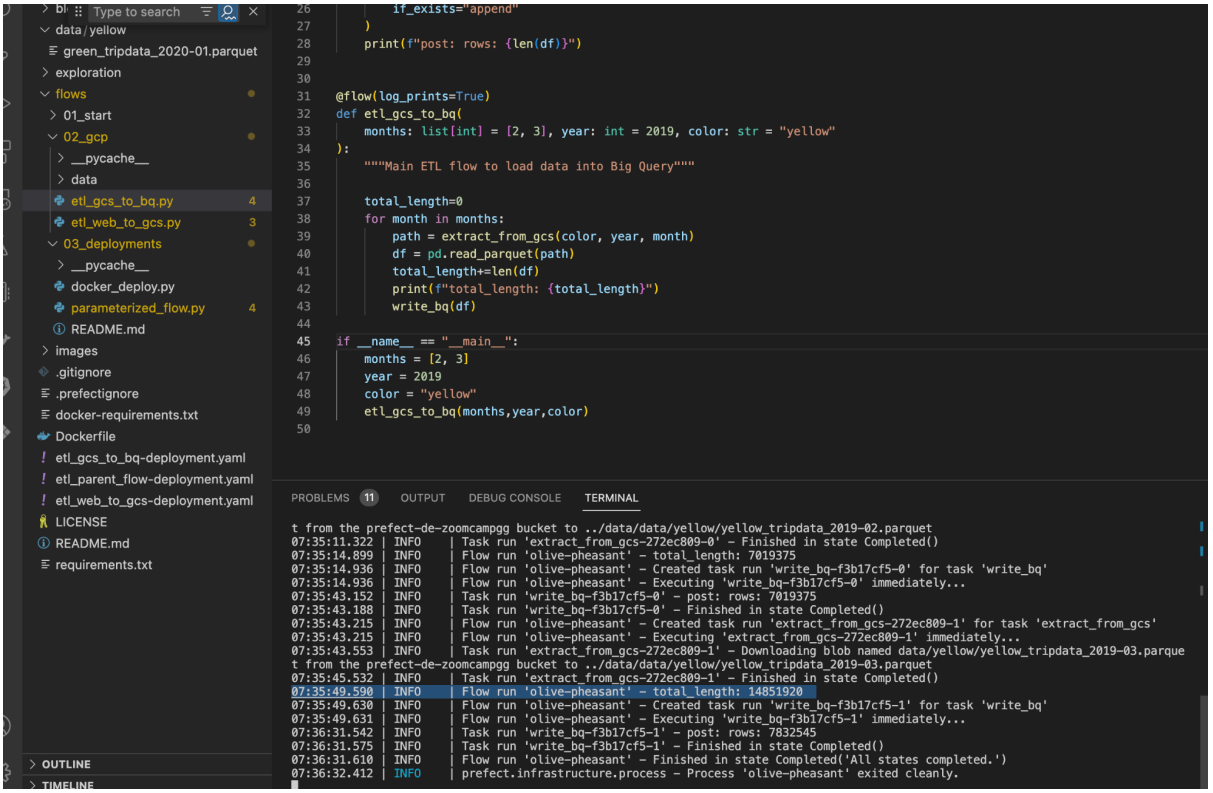
With the same logic as Step 1, I built and applied deployment by modifying the etl_gcs_to_bq.py file in order to store data from Google Cloud Storage to Big Query Modified script can be found below and in the github repo by clicking here.





Feb 6th, 2023

INFO	Downloading flow code from storage at '/Users/georgiosgrigoriou/Desktop/prefect-zoomcamp-homework 2 code'	07:35:08 AM
INFO	Created task run 'extract_from_gcs-272ec809-0' for task 'extract_from_gcs'	07:35:09 AM
INFO	Executing 'extract_from_gcs-272ec809-0' immediately...	07:35:09 AM
INFO	Downloading blob named data/yellow/yellow_tripdata_2019-02.parquet from the prefect-de-zoomcampgg bucket to ../data/data/yellow/yellow_tripdata_2019-02.parquet	07:35:09 AM extract_from_gcs-272ec809-0
INFO	Finished in state Completed()	07:35:11 AM extract_from_gcs-272ec809-0
INFO	total_length: 7019375	07:35:14 AM
INFO	Created task run 'write_bq-f3b17cf5-0' for task 'write_bq'	07:35:14 AM
INFO	Executing 'write_bq-f3b17cf5-0' immediately...	07:35:14 AM
INFO	post: rows: 7019375	07:35:43 AM write_bq-f3b17cf5-0
INFO	Finished in state Completed()	07:35:43 AM write_bq-f3b17cf5-0
INFO	Created task run 'extract_from_gcs-272ec809-1' for task 'extract_from_gcs'	07:35:43 AM
INFO	Executing 'extract_from_gcs-272ec809-1' immediately...	07:35:43 AM
INFO	Downloading blob named data/yellow/yellow_tripdata_2019-03.parquet from the prefect-de-zoomcampgg bucket to ../data/data/yellow/yellow_tripdata_2019-03.parquet	07:35:43 AM extract_from_gcs-272ec809-1
INFO	Finished in state Completed()	07:35:45 AM extract_from_gcs-272ec809-1
INFO	total_length: 14851920	07:35:49 AM
INFO	Created task run 'write_bq-f3b17cf5-1' for task 'write_bq'	07:35:49 AM
INFO	Executing 'write_bq-f3b17cf5-1' immediately...	07:35:49 AM



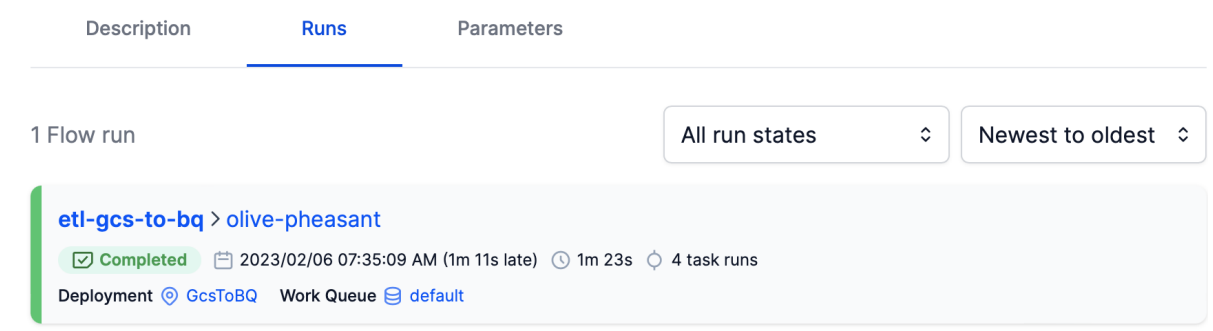
Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH PREVIEW

Row	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_coun	trip_distance	RatecodeID	store_and_fwd_flag
1	2.0	2019-03-14 20:56:08 UTC	2019-03-14 21:07:53 UTC	1.0	2.1	1.0	N
2	2.0	2019-03-14 20:02:11 UTC	2019-03-14 20:16:42 UTC	1.0	2.24	1.0	N
3	1.0	2019-03-14 20:43:11 UTC	2019-03-14 20:50:19 UTC	1.0	1.4	1.0	N
4	2.0	2019-03-14 20:39:25 UTC	2019-03-14 21:07:21 UTC	1.0	8.83	1.0	N
5	1.0	2019-03-14 20:26:31 UTC	2019-03-14 20:46:41 UTC	1.0	3.9	1.0	N
6	1.0	2019-03-14 20:00:35 UTC	2019-03-14 20:08:00 UTC	2.0	1.4	1.0	N
7	2.0	2019-03-14 20:03:46 UTC	2019-03-14 20:20:19 UTC	3.0	1.46	1.0	N

Results per page: 50 1 – 50 of 14851920



Modified Script etl_gcp_to_bq.py:

```
from pathlib import Path
import pandas as pd
from prefect import flow, task
```

```python
from prefect_gcp.cloud_storage import GcsBucket
from prefect_gcp import GcpCredentials
@task(retries=3)
def extract_from_gcs(color: str, year: int, month: int) -> Path:
    """Download trip data from GCS"""
    gcs_path = f"data/{color}/{color}_tripdata_{year}-{month:02}.parquet"
    gcs_block = GcsBucket.load("zoom-gcs")
    gcs_block.get_directory(from_path=gcs_path, local_path=f"../data/")
    return Path(f"../data/{gcs_path}")

@task(log_prints=True)
def write_bq(df: pd.DataFrame) -> None:
    """Write DataFrame to BiqQuery"""
    gcp_credentials_block = GcpCredentials.load("zoom-gcs-creds")

    df.to_gbq(
        destination_table="dezoomcamp.rides",
        project_id="<CONFIDENTIAL>",
        credentials=gcp_credentials_block.get_credentials_from_service_account(),
        chunksize=500_000,
        if_exists="append"
    )
    print(f"post: rows: {len(df)}")


@flow(log_prints=True)
def etl_gcs_to_bq(
    months: list[int] = [2, 3], year: int = 2019, color: str = "yellow"
):
    """Main ETL flow to load data into Big Query"""

    total_length=0
    for month in months:
        path = extract_from_gcs(color, year, month)
        df = pd.read_parquet(path)
        total_length+=len(df)
        print(f"total_length: {total_length}")
        write_bq(df)

if __name__ == "__main__":
    months = [2, 3]
    year = 2019
    color = "yellow"
    etl_gcs_to_bq(months,year,color)
```

Question 4. Github Storage Block

Using the `web_to_gcs` script from the videos as a guide, you want to store your flow code in a GitHub repository for collaboration with your team. Prefect can look in the GitHub repo to find your flow code and read it. Create a GitHub storage block from the UI or in Python code and use that in your Deployment instead of storing your flow code locally or baking your flow code into a Docker image.

Note that you will have to push your code to GitHub, Prefect will not push it for you.

Run your deployment in a local subprocess (the default if you don't specify an infrastructure). Use the Green taxi data for the month of November 2020.

How many rows were processed by the script?

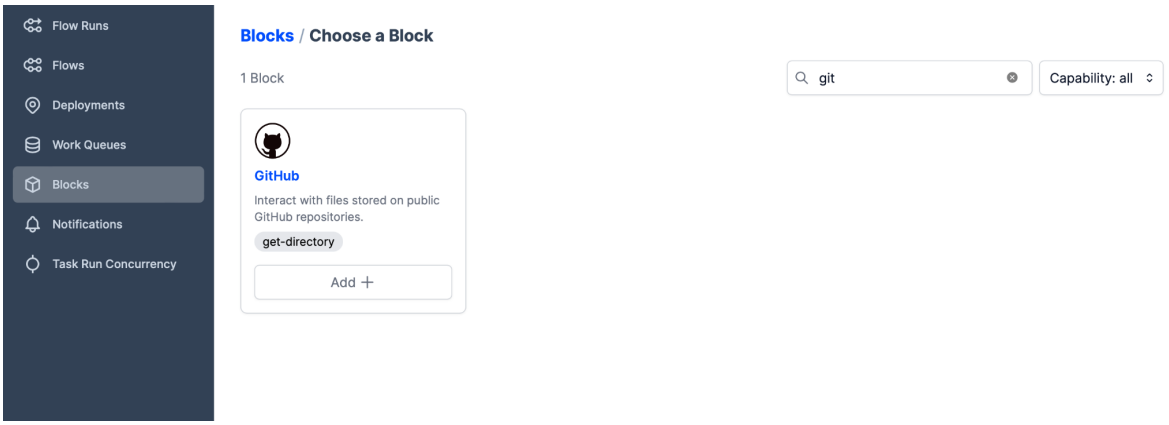
- 88,019
- 192,297
- **88,605**
- 190,225

Question 4. Github Storage Block Solution

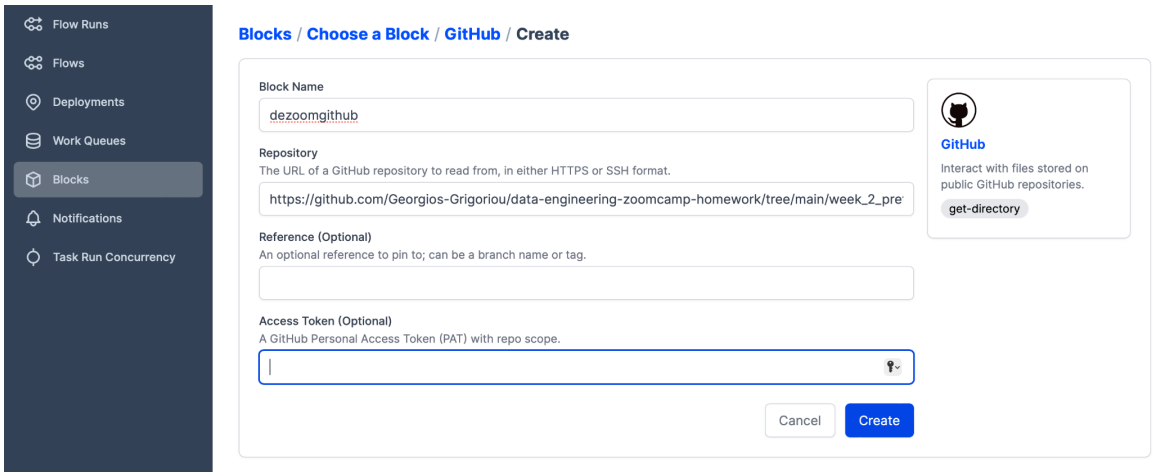
The correct answer is 88605. Change the flow from etl_web_to_gcs.py accordingly
Firstly, data from green taxi 2020 for the month of November (11)

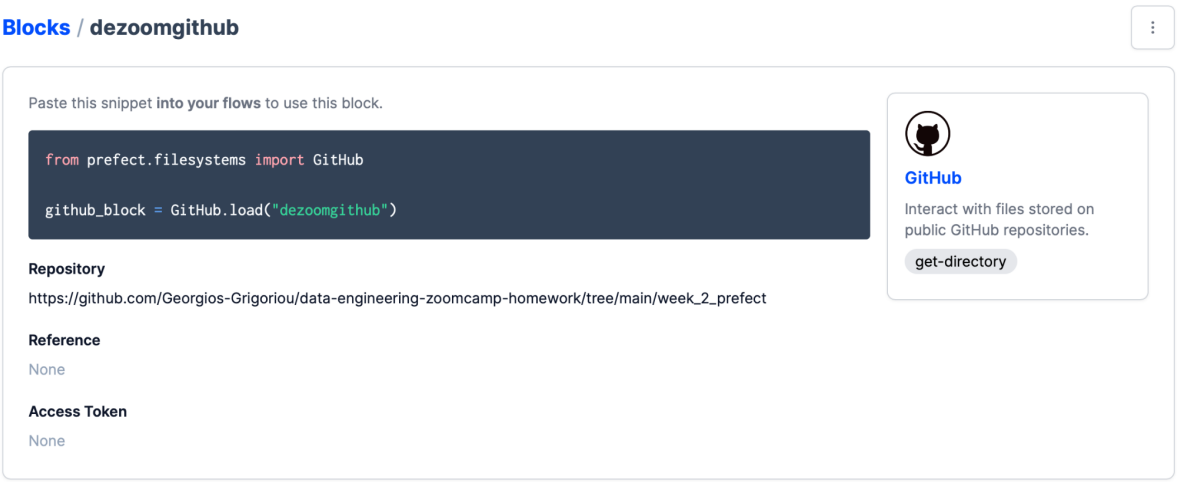
```python
@flow()
def etl_web_to_gcs() -> None:
    """The main ETL function"""
    color = "green"
    year = 2020
    month = "11"
    dataset_file = f"{color}_tripdata_{year}-{month}"
    dataset_url = f"https://github.com/DataTalksClub/nyc-tlc-data/releases/download/{color}/{dataset_file}.csv.gz"

    df = fetch(dataset_url)
    df_clean = clean(df)
    path = write_local(df_clean, color, dataset_file)
    write_gcs(path)
```

Block in Prefect



Follow the instruction to create the github block



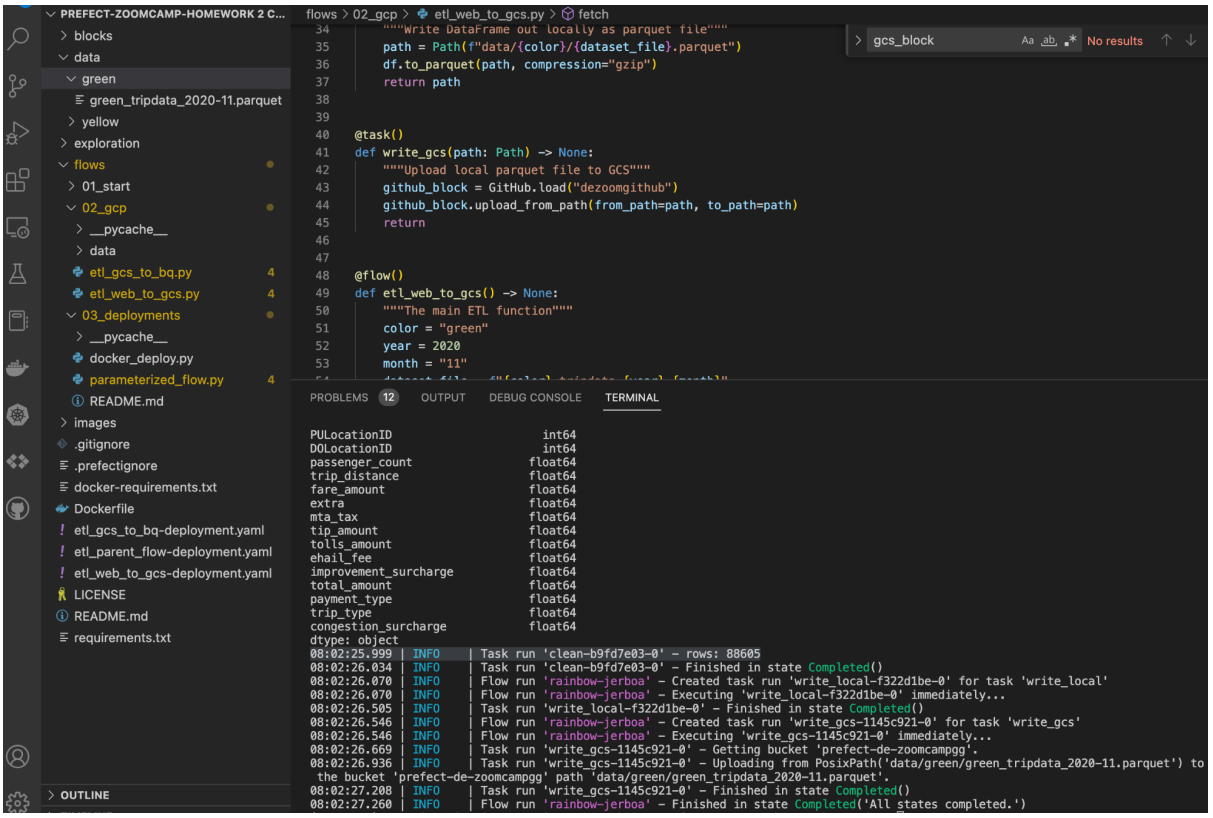


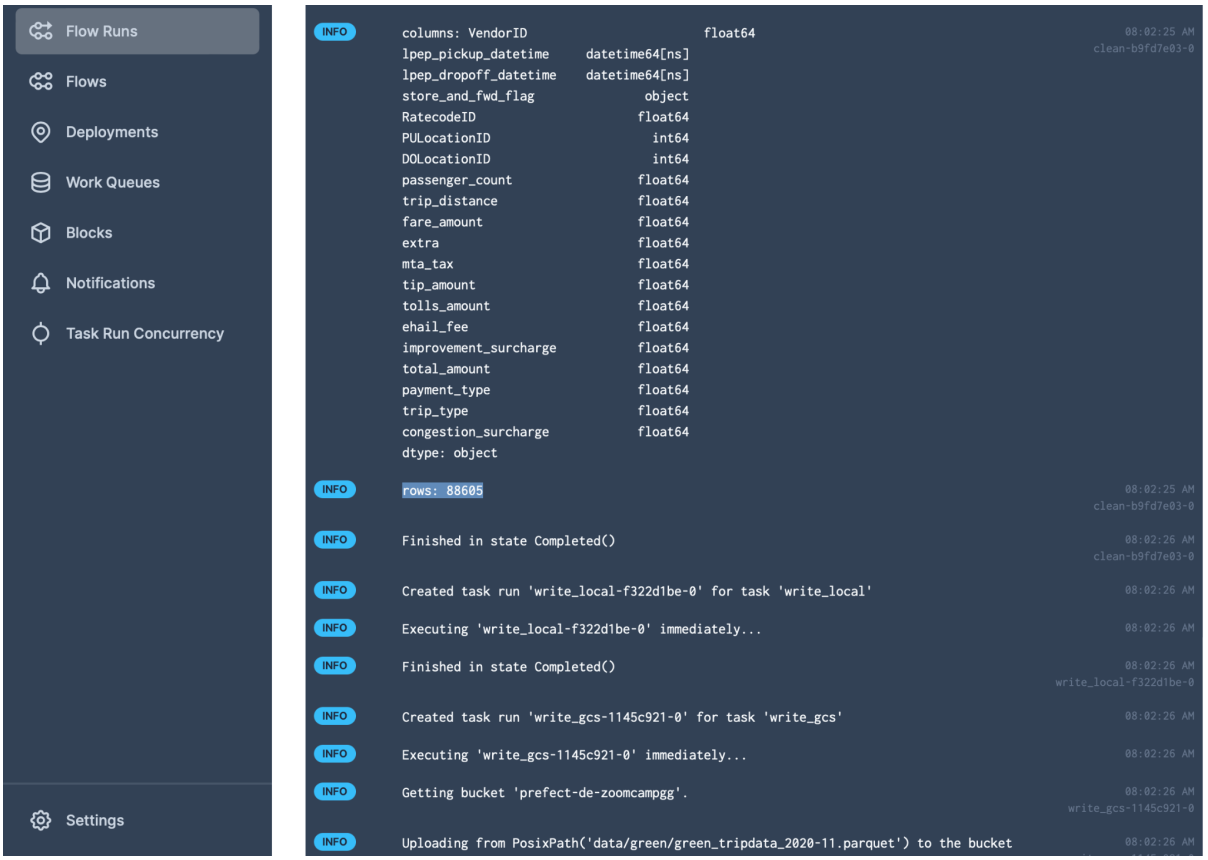
Import GitHuv and modify the task accordingly

```
from prefect.filesystems import GitHub
```

```
@task()
def write_gcs(path: Path) -> None:
    """Upload local parquet file to GCS"""
    github_block = GitHub.load("dezoomgithub")
    github_block.upload_from_path(from_path=path, to_path=path)
    return
```

Process ran locally and show the result in the terminal and Prefect UI respectively





Push on github can be found here

Question 5. Email or Slack notifications

Q5. It's often helpful to be notified when something with your dataflow doesn't work as planned. Choose one of the options below for creating email or slack notifications.

The hosted Prefect Cloud lets you avoid running your own server and has Automations that allow you to get notifications when certain events occur or don't occur.

Create a free forever Prefect Cloud account at app.prefect.cloud and connect your workspace to it following the steps in the UI when you sign up.

Set up an Automation that will send yourself an email when a flow run completes. Run the deployment used in Q4 for the Green taxi data for April 2019. Check your email to see the notification.

Alternatively, use a Prefect Cloud Automation or a self-hosted Orion server Notification to get notifications in a Slack workspace via an incoming webhook.

Join my temporary Slack workspace with this link. 400 people can use this link and it expires in 90 days.

In the Prefect Cloud UI create an Automation or in the Prefect Orion UI create a Notification to send a Slack message when a flow run enters a Completed state. Here is the Webhook URL to use: https://hooks.slack.com/services/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

Test the functionality.

Alternatively, you can grab the webhook URL from your own Slack workspace and Slack App that you create.

How many rows were processed by the script?

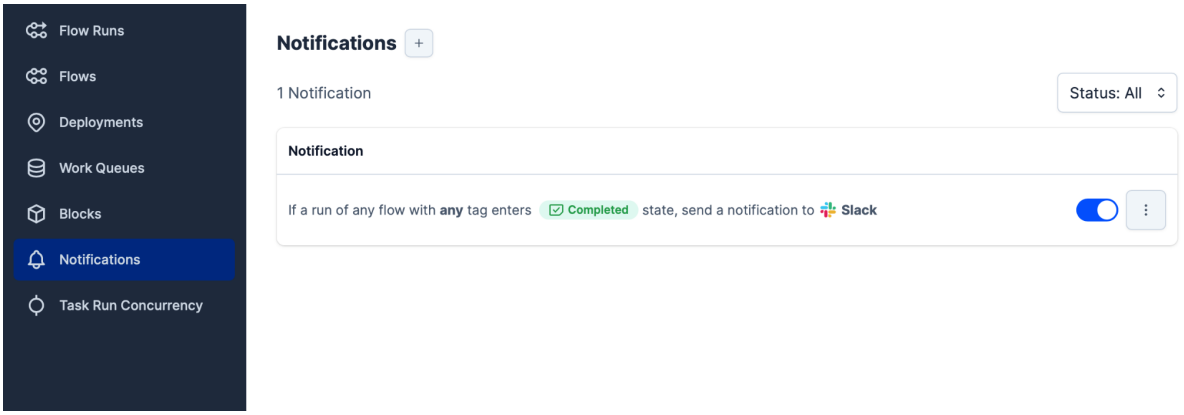
- `125,268`
- `377,922`
- `728,390`
- **`514,392`**

Question 5. Email or Slack notifications Solution

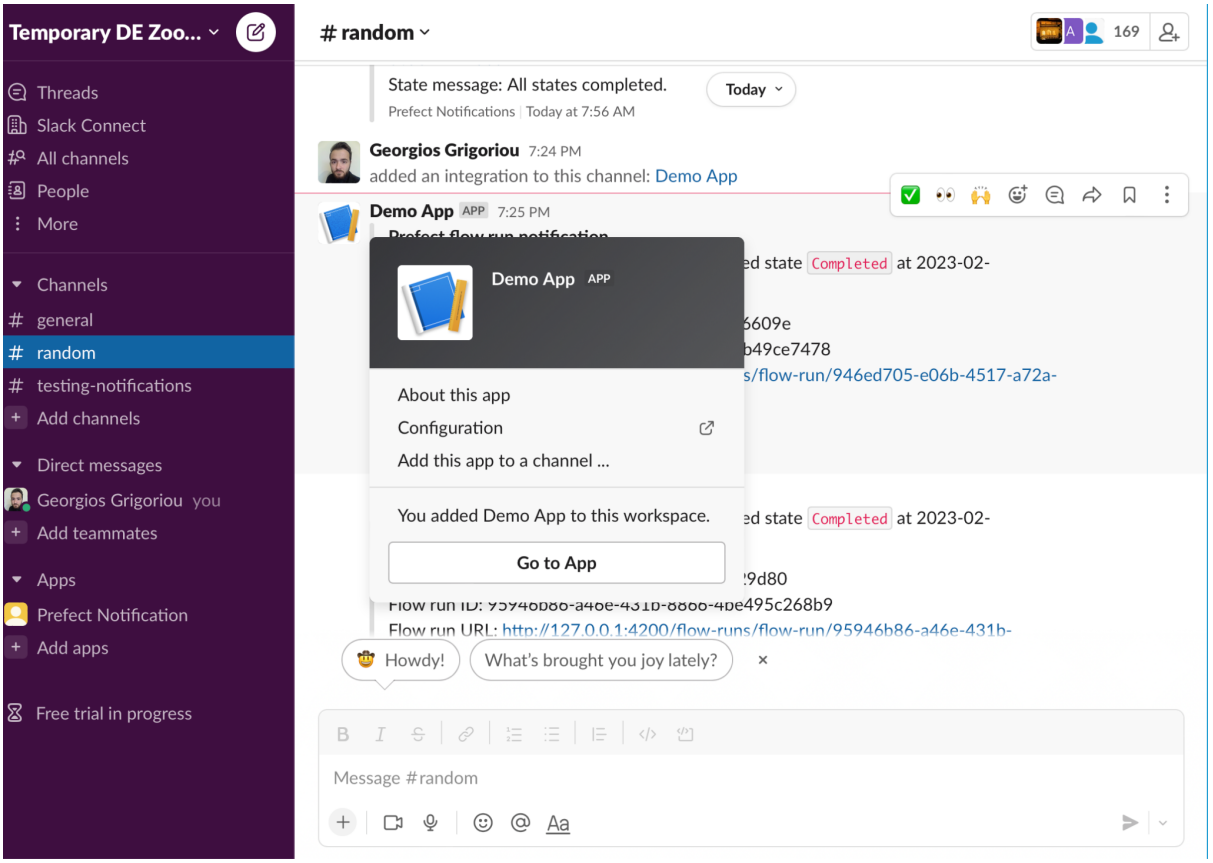
```
The correct answer is 514,392.
```

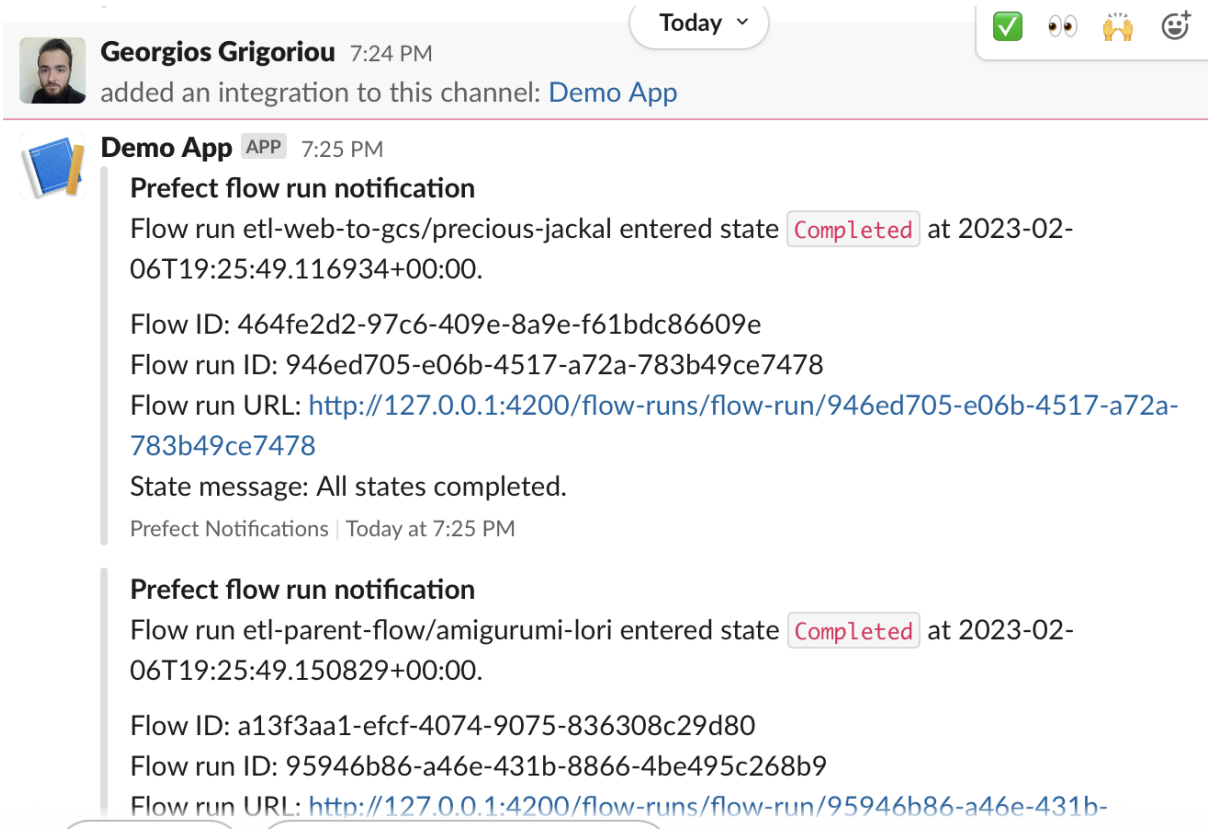
Steps:

Create Slack Notification

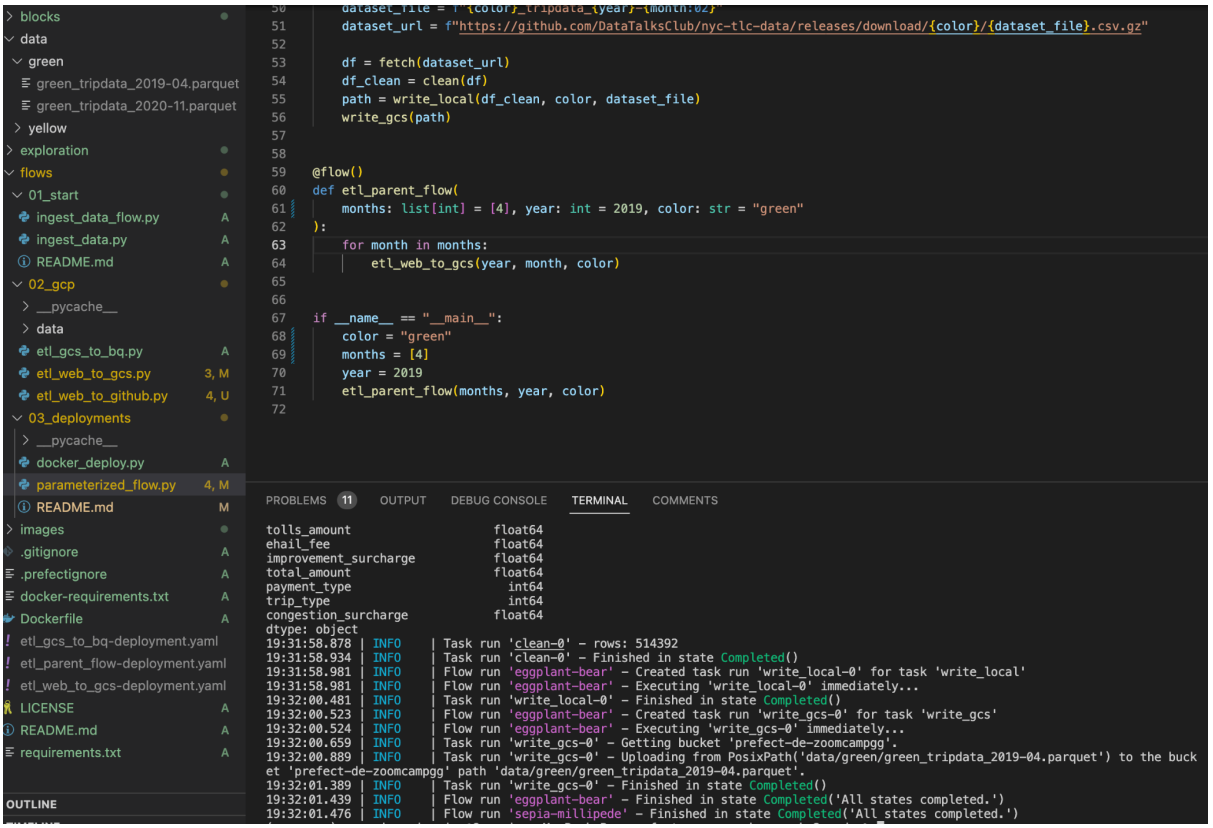


Using the Incoming Webhooks Slack API, I created a custom app and a webhook url following the instructions https://api.slack.com/messaging/webhooks.





By running the flow etl_web_to_gcs for April 2019 green, I found that the processed rows are 514,392



Question 6. Secrets

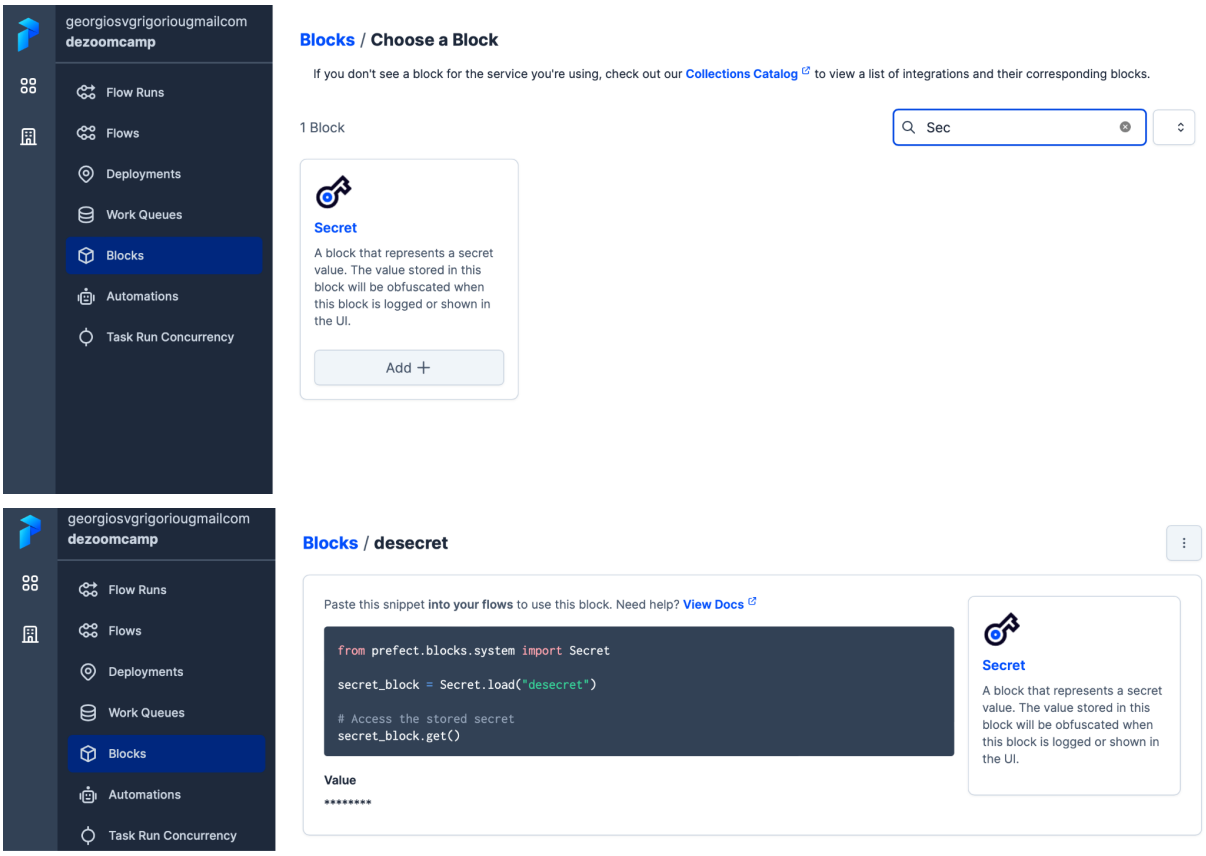
Prefect Secret blocks provide secure, encrypted storage in the database and obfuscation in the UI. Create a secret block in the UI that stores a fake 10-digit password to connect to a third-party service. Once you've created your block in the UI, how many characters are shown as asterisks (*) on the next page of the UI?

- 5
- 6
- 8
- 10

Question 6. Secrets Solution

The correct answer is 8.

On Prefect Cloud, I searched on the secret block



We can see that the asterisks are 8.

Submitting the solutions

- Form for submitting: https://forms.gle/PY8mBEGXJ1RvmTM97
- You can submit your homework multiple times. In this case, only the last submission will be used.