

# Week 1 Homework: Docker & SQL

Georgios Grigoriou

[georgiosvgrigoriou@gmail.com](mailto:georgiosvgrigoriou@gmail.com)

Deadline: 26 January (Thursday), 22:00 CET

In this homework, I prepared the environment and practice with Docker and SQL as part of the [Data Engineering Zoomcamp course](#). Solutions' headings are in italic and highlighted in yellow under each Question's instruction.

## Question 1. Knowing docker tags

Run the command to get information on Docker

```
docker --help
```

Now run the command to get help on the "docker build" command

Which tag has the following text? - *Write the image ID to the file*

- --imageid string
- --iidfile string
- --idimage string
- --idfile string

### **Question 1 Solution**

*The correct answer is --iidfile string. By running in the terminal the following command*

```
docker --help build
```

*We can see see that the answer corresponding to Write the image ID to the file is*

```
--iidfile string
```

```
Last login: Mon Jan 23 18:45:47 on console
[(base) georgiosgrigoriou@Georgioss-MacBook-Pro ~ % docker --help build

Usage: docker build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile

Options:
  --add-host list          Add a custom host-to-IP mapping (host:ip)
  --build-arg list          Set build-time variables
  --cache-from strings      Images to consider as cache sources
  --cgroup-parent string    Optional parent cgroup for the container
  --compress                 Compress the build context using gzip
  --cpu-period int          Limit the CPU CFS (Completely Fair
                           Scheduler) period
  --cpu-quota int           Limit the CPU CFS (Completely Fair
                           Scheduler) quota
  -c, --cpu-shares int      CPU shares (relative weight)
  --cpuset-cpus string      CPUs in which to allow execution (0-3, 0,1)
  --cpuset-mems string      MEMs in which to allow execution (0-3, 0,1)
  --disable-content-trust   Skip image verification (default true)
  -f, --file string          Name of the Dockerfile (Default is
                           'PATH/Dockerfile')
  --force-rm                  Always remove intermediate containers
  --iidfile string            Write the image ID to the file
  --isolation string          Container isolation technology
  --label list                 Set metadata for an image
  -m, --memory bytes          Memory limit
  --memory-swap bytes         Swap limit equal to memory plus swap:
                           '-1' to enable unlimited swap
  --network string             Set the networking mode for the RUN
                           instructions during build (default "default")
  --no-cache                   Do not use cache when building the image
  --pull                      Always attempt to pull a newer version of
                           the image
  -q, --quiet                  Suppress the build output and print image
                           ID on success
  --rm                         Remove intermediate containers after a
                           successful build (default true)
  --security-opt strings       Security options
  --shm-size bytes              Size of /dev/shm
  --squash                     Squash newly built layers into a single
                           new layer
  -t, --tag list                Name and optionally a tag in the
                           'name:tag' format
  --target string                Set the target build stage to build.
  --ulimit ulimit                Ulimit options (default [])

```

Figure 1: running `docker -- help build` in terminal

## Question 2. Understanding docker first run

Run docker with the python:3.9 image in an interactive mode and the entrypoint of bash. Now check the python modules that are installed ( use pip list). How many python packages/modules are installed?

- 1
- 6
- 3
- 7

## Question 2 Solution

The correct answer is 3. By running in the terminal the following commands

```
docker run -it --entrypoint=bash python:3.9
pip list
```

We can see that we have 3 packages/modules

```
(base) georgiosgrigoriou@Georgioss-MacBook-Pro data_engineering_zoomcamp % docker run -it --entrypoint=bash python:3.9
[root@efaf559ca4134:/# pip list
Package          Version
-----
pip            22.0.4
setuptools      58.1.0
wheel           0.38.4
WARNING: You are using pip version 22.0.4; however, version 22.3.1 is available.
You should consider upgrading via the 'curl https://bootstrap.pypa.io/get-pip.py | python' command.
root@efaf559ca4134:/#
```

# Prepare Postgres

Run Postgres and load data as shown in the videos We'll use the green taxi trips from January 2019:

```
wget
https://github.com/DataTalksClub/nyc-tlc-data/releases/download/green/green_tripdata_2019-01.csv.gz
```

You will also need the dataset with zones:

```
wget https://s3.amazonaws.com/nyc-tlc/misc/taxi+_zone_lookup.csv
```

Download this data and put it into Postgres (with jupyter notebooks or with a pipeline)

## Prepare Postgres

*Using the pipeline shown in [the sessions of week 1 \(DE Zoomcamp 1.2.1-1.2.6\)](#) and the instructions [here](#), the datasets about the green taxi trips and taxi zone lookups were stored in the Postgresql database*

Folder, files and code can be found in this [link](#)

## Prepare Postgres and pgAdmin

### [docker-compose.yml file](#)

```
services:

pgdatabase:
  image: postgres:13
  environment:
    - POSTGRES_USER=root
    - POSTGRES_PASSWORD=root
    - POSTGRES_DB=ny_taxi
```

```

volumes:
  - "./ny_taxi_postgres_data:/var/lib/postgresql/data:rw"

ports:
  - "5432:5432"

pgadmin:
  image: dpage/pgadmin4

  environment:
    - PGADMIN_DEFAULT_EMAIL=admin@admin.com
    - PGADMIN_DEFAULT_PASSWORD=root

  ports:
    - "8080:80"

```

## Running docker-compose.yml file

```

Last login: Tue Jan 24 08:38:13 on ttys000
(base) georgiosgrigoriou@Georgioss-MacBook-Pro ~ % cd Desktop
(base) georgiosgrigoriou@Georgioss-MacBook-Pro Desktop % cd homework_week1_docker_sql
(base) georgiosgrigoriou@Georgioss-MacBook-Pro homework_week1_docker_sql % docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
(base) georgiosgrigoriou@Georgioss-MacBook-Pro homework_week1_docker_sql % docker-compose up
[+] Running 3/3
[ 1] Network homework_week1_docker_sql_default          Created          0.1s
[ 2] Container homework_week1_docker_sql-pgadmin-1      Created          0.2s
[ 3] Container homework_week1_docker_sql-pgdatabase-1   Created          0.2s
Attaching to homework_week1_docker_sql-pgadmin-1, homework_week1_docker_sql-pgdatabase-1
homework_week1_docker_sql-pgdatabase-1  The files belonging to this database system will be owned by user "postgres".
homework_week1_docker_sql-pgdatabase-1  This user must also own the server process.
The database cluster will be initialized with locale "en_US.utf8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".
Data page checksums are disabled.
fixing permissions on existing directory /var/lib/postgresql/data ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Etc/UTC
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
NOTE: Configuring authentication for SERVER mode.
pgAdmin 4 - Application Initialisation
=====
syncing data to disk ... ok
initdb: warning: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.
Success. You can now start the database server using:
  pg_ctl -D /var/lib/postgresql/data -l logfile start
waiting for server to start....2023-01-24 08:40:16.047 UTC [49] LOG:  starting PostgreSQL 13.9 (Debian 13.9-1.pgdg110+1) on x86_64-debian-linux-gnu (Ubuntu 18.04.6 LTS)
2023-01-24 08:40:16.050 UTC [49] LOG:  listening on Unix socket "/var/run/postgresql/.PGSQL.5432"
2023-01-24 08:40:16.075 UTC [50] LOG:  database system was shut down at 2023-01-24 08:40:12 UTC
2023-01-24 08:40:16.104 UTC [49] LOG:  database system is ready to accept connections
done
server started
CREATE DATABASE
/usr/local/bin/docker-entrypoint.sh: ignoring /docker-entrypoint-initdb.d/*
2023-01-24 08:40:19.312 UTC [49] LOG:  received fast shutdown request
waiting for server to shut down...2023-01-24 08:40:19.315 UTC [49] LOG:  aborting any active transactions
2023-01-24 08:40:19.318 UTC [49] LOG:  background worker "logical replication launcher" (PID 56) exited with exit code 1
2023-01-24 08:40:19.322 UTC [51] LOG:  shutting down
2023-01-24 08:40:19.400 UTC [49] LOG:  database system is shut down
done
server stopped

```

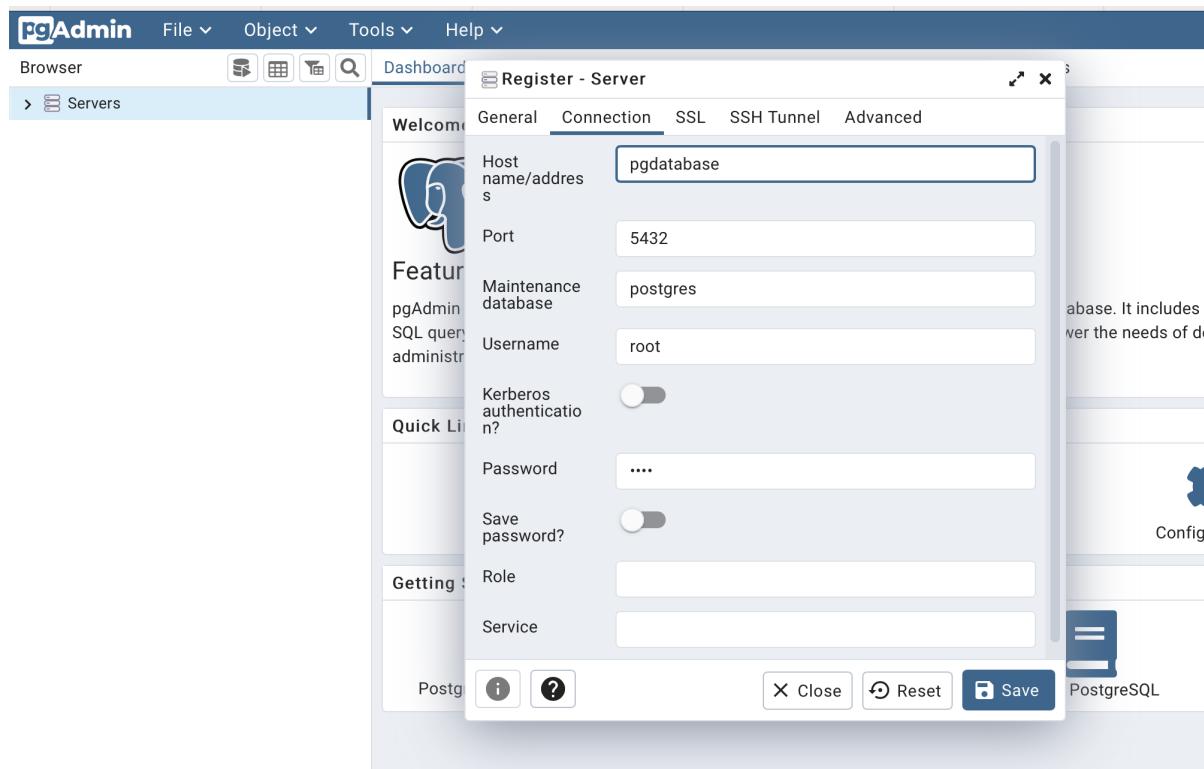
**Figure 2:** running docker composer in terminal

## Download taxi\_zone\_looker.csv

```
(base) georgiosgrigoriou@Georgioss-MacBook-Pro homework_week1_docker_sql % wget https://s3.amazonaws.com/nyc-tlc/misc/taxi+_zone_lookup.csv
--2023-01-24 08:44:44--  https://s3.amazonaws.com/nyc-tlc/misc/taxi+_zone_lookup.csv
Resolving s3.amazonaws.com (s3.amazonaws.com)... 52.216.10.173, 54.231.201.8, 52.217.41.150, ...
Connecting to s3.amazonaws.com (s3.amazonaws.com)|52.216.10.173|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12322 (12K) [application/octet-stream]
Saving to: 'taxi+_zone_lookup.csv'

taxi+_zone_lookup.c 100%[=====] 12.03K  --.-KB/s   in 0s
2023-01-24 08:44:44 (52.7 MB/s) - 'taxi+_zone_lookup.csv' saved [12322/12322]
(base) georgiosgrigoriou@Georgioss-MacBook-Pro homework_week1_docker_sql %
```

## Register Server



# Building and running ingest\_data.py script via docker image taxi\_green\_ingest

```
Last login: Tue Jan 24 09:51:08 on ttys002
(base) georgiosgrigoriou@Georgioss-MacBook-Pro ~ % cd Desktop/homework_week1_docker_sql
(base) georgiosgrigoriou@Georgioss-MacBook-Pro homework_week1_docker_sql % docker network ls
[NODE] NAME DRIVER SCOPE
a167628bbb02 bridge bridge local
7e7827ab3b33 data_engineering_zoomcamp_default bridge local
df950443c706 homework_week1_docker_sql_default bridge local
f73aba4dac50 host host local
371281e23d6b none null local
3762f02dd876 pg-network bridge local
(base) georgiosgrigoriou@Georgioss-MacBook-Pro homework_week1_docker_sql % docker build -t taxi_green_ingest:v01 .
[+] Building 21.8s (11/11) FINISHED
[=> [internal] load configuration from Dockerfile
=> => transferring dockerfile: 264B
=> [internal] load .dockerrcignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3.9.1
=> [auth] library/python:pull token for registry1.docker.io
=> [1/5] FROM docker.io/library/python:3.9.1@sha256:ca8bd3c91af8b12c2d042ade99f7c8f578a9f80a0dbbd12ed261eeba96dd632f
=> [internal] load build context
=> => transferring context: 120B
=> CACHED [2/5] RUN apt-get install wget
=> [3/5] RUN pip install pandas sqlalchemy psycopg2-binary
=> [4/5] WORKDIR /app2
=> [5/5] COPY ingest_data.py ingest_data.py
=> exporting to image
=> => exporting layers
=> => writing image sha256:1ccb1eee6e6db69096e5d078660249c662e65238cc1e0407b8d4e1e0442ab611
=> => naming to docker.io/library/taxi_green_ingest:v01
```

## Running data\_ingest.py and store in Postgresql Database

```
(base) georgiosgrigoriou@Georgioss-MacBook-Pro homework_week1_docker_sql % docker run -it \
--network=homework_week1_docker_sql_default \
taxi_green_ingest:v01 \
--user-root \
--password-root \
--host=pgsqlDatabase \
--port=5432 \
--dbn=y_taxi \
--table_name=green_taxi_trips \
-u https://data.talksclub.nyc-tlc-data/releases/download/green/green_tripdata_2019-01.csv.gz' \
--2023-01-24T09:58:10Z https://github.com/DataTalksClub/nyc-tlc-data/releases/download/green/green_tripdata_2019-01.csv.gz
Resolving github.com (github.com) [149.82.121.4]:443... connected.
Connecting to github.com (github.com) [149.82.121.4]:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github.com/github-production-release-asset-2e65be/513814948/d3984232-1a2b-431b-803d-8ee802cd14fc?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A
%2F20230124%2Fus-east-1%2F53%2Faws4_request&X-Amz-Date=20230124T095810Z&X-Amz-Expires=3000X-Amz-Signature=48cae116e0984611a6676e488e3382ae3668e971fd2bdfe882da11a38X-Z-SignedHeaders=host&actor_id
=88key_id@88repo_id513814948&response-content-disposition=attachment;filename=green_tripdata_2019-01.csv.gz&response-type=application/x-gzip;stream=1
--2023-01-24T09:58:10Z https://objects.githubusercontent.com/github/production/release-asset-2e65be/513814948/d3984232-1a2b-431b-803d-8ee802cd14fc?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIW
WHJYAX4CSVEH53A%2F20230124%2Fus-east-1%2F53%2Faws4_request&X-Amz-Expires=3000X-Amz-Signature=8cc81216e098804611a676e488e3382ae3668e971fd2bdfe882da11a38X-Amz-SignedHeader
s=host&actor_id=88key_id@88repo_id513814948&response-content-disposition=attachment;filename=green_tripdata_2019-01.csv.gz&response-content-type=application/x-gzip;stream=1
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.111.133, 185.199.109.133, 185.199.108.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11099245 (11M) [application/octet-stream]
Saving to: 'output.csv.gz'

output.csv.gz    100%[=====] 10.58M  16.6MB/s  in 0.6s

2023-01-24 09:58:11 (16.6 MB/s) - 'output.csv.gz' saved [11099245/11099245]

inserted another chunk, took 18.159 second
inserted another chunk, took 19.084 second
inserted another chunk, took 19.885 second
inserted another chunk, took 18.738 second
inserted another chunk, took 20.016 second
inserted another chunk, took 7.866 second
Finished ingesting data into the postgres database
```

## Running taxi\_zone\_lookup.py and store data from taxi+zone\_lookup.csv to Postgresql Database

```
In [1]: import pandas as pd
In [6]: url="https://d37ci6vzurychx.cloudfront.net/misc/taxi+zone_lookup.csv"
In [7]: df=pd.read_csv(url)
In [8]: df.head()
Out[8]:
   LocationID Borough          Zone service_zone
0            1      EWR  Newark Airport        EWR
1            2    Queens    Jamaica Bay    Boro Zone
2            3      Bronx Allerton/Pelham Gardens    Boro Zone
3            4  Manhattan    Alphabet City    Yellow Zone
4            5  Staten Island       Arden Heights    Boro Zone

In [10]: from sqlalchemy import create_engine
engine=create_engine("postgresql://root:root@localhost:5432/ny_taxi")
engine.connect()
Out[10]: <sqlalchemy.engine.base.Connection at 0x7fe2c4cef580>
In [11]: df.to_sql(name='taxi_zone_lookup',con=engine,if_exists='replace')
Out[11]: 265
```

## pgAdmin and pgcli UI

The screenshot shows the pgAdmin interface. The left sidebar displays a tree view of database objects under the 'Schemas' node, including 'public' and 'Tables'. The 'Tables' node is expanded, showing 'green\_taxi\_trips' and 'taxi\_zone\_lookup'. The main panel is titled 'Statistics' and contains a table with the following data:

Table name	Tuples inserted	Tuples updated	Tuples deleted	Tuples HOT updated	Live tuples	Dead
green_taxi_trips	630918	0	0	0	630918	0
taxi_zone_lookup	265	0	0	0	265	0

```

Last login: Tue Jan 24 09:53:20 on ttys002
(base) georgiosgrigoriou@Georgioss-MacBook-Pro ~ % pgcli -h localhost -p 5432 -u root -d ny_taxi
Password for root:
[Server: PostgreSQL 13.9 (Debian 13.9-1.pgdg110+1)
Version: 3.5.0
Home: http://pgcli.com
root@localhost:ny_taxi> \dt
+-----+-----+-----+
| Schema | Name      | Type   | Owner |
+-----+-----+-----+
| public | green_taxi_trips | table  | root   |
| public | taxi_zone_lookup | table  | root   |
+-----+-----+-----+
SELECT 2
Time: 0.038s
root@localhost:ny_taxi>
root@localhost:ny_taxi> █

```

## Question 3. Count records

How many taxi trips were totally made on January 15?

Tip: started and finished on 2019-01-15.

Remember that `lpep_pickup_datetime` and `lpep_dropoff_datetime` columns are in the format timestamp (date and hour+min+sec) and not in date.

- 20689
- 20530
- 17630
- 21090

### **Question 3 Count records Solution**

***The correct answer is 20530. Running the following snippet in Postgresql***

```

SELECT COUNT(*) from green_taxi_trips
where date_trunc('day', lpep_pickup_datetime)='2019-01-15'
and date_trunc('day', lpep_dropoff_datetime)='2019-01-15';

```

We find out that the correct answer is 20530

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane lists database objects: Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, Schemas (1), public (Aggregates, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, Tables (3)), and green\_taxi\_trips. The 'Query' tab in the center contains the following SQL code:

```

1 SELECT COUNT(1) FROM green_taxi_trips
2 WHERE date_trunc('day', lpep_pickup_datetime)='2019-01-15'
3 AND date_trunc('day', lpep_dropoff_datetime)='2019-01-15';
4

```

The 'Data Output' pane at the bottom shows the result of the query:

	count	bigint
1	20530	

**Figure 3:** running the SQL command in pgAdmin

## Question 4. Largest trip for each day

Which was the day with the largest trip distance

Use the pick up time for your calculations.

- 2019-01-18
- 2019-01-28
- 2019-01-15
- 2019-01-10

## **Question 4 Largest trip for each day**

***The correct answer is 2019-01-15. Running the following snippet in Postgresql***

```

SELECT
    CAST(lpep_pickup_datetime AS DATE) AS "day",
    trip_distance
FROM
    green_taxi_trips
GROUP BY
    CAST(lpep_pickup_datetime AS DATE),
    trip_distance
ORDER BY
    trip_distance DESC

```

***We find that the correct answer is 2019-01-15***

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under the 'Browser' section, there is a tree view of database objects. One node, 'green\_taxi\_trips', has its 'Columns (21)' expanded, showing a list of columns including 'index', 'VendorID', 'lpep\_pickup\_datetime', 'lpep\_dropoff\_datetime', 'store\_and\_fwd\_flag', 'RatecodeID', 'PULocationID', 'DOLocationID', 'passenger\_count', 'trip\_distance', 'fare\_amount', 'extra', 'mta\_tax', 'tip\_amount', 'tolls\_amount', 'ehail\_fee', 'improvement\_surcharge', 'total\_amount', and 'payment\_type'. The main area contains a SQL query window with the following code:

```

1 select
2     CAST(lpep_pickup_datetime AS date) AS "day",
3     trip_distance
4 from
5     green_taxi_trips
6 GROUP BY
7     CAST(lpep_pickup_datetime AS date),
8     trip_distance
9 ORDER BY
10    trip_distance DESC
11

```

Below the query window is a 'Data Output' tab showing the results of the query. The results are presented in a table with two columns: 'day' (date) and 'trip\_distance' (double precision). The data consists of 9 rows:

day	trip_distance
2019-01-15	117.99
2019-01-18	80.96
2019-01-28	64.27
2019-01-10	64.2
2019-01-06	60.91
2019-01-07	60.08
2019-01-01	59.13
2019-01-05	51.79
2019-01-22	51.36

Total rows: 1000 of 56265    Query complete 00:00:04.185    Ln 5, Col 19

**Figure 4:** running the SQL command in pgAdmin

## Question 5. The number of passengers

In 2019-01-01 how many trips had 2 and 3 passengers?

- 2: 1282 ; 3: 266
- 2: 1532 ; 3: 126
- 2: 1282 ; 3: 254
- 2: 1282 ; 3: 274

### **Question 5 number of passengers Solution**

***The correct answer is 2: 1282 ; 3: 254 .***

*Running the following snippet in Postgresql*

**For passenger:2**

```

SELECT DATE_TRUNC('DAY',lpep_pickup_datetime),
       count(passenger_count)
FROM
      green_taxi_trips
WHERE DATE_TRUNC('DAY',lpep_pickup_datetime)='2019-01-01' and
      passenger_count=2
GROUP BY
      DATE_TRUNC('DAY',lpep_pickup_datetime)

```

### For passenger:3

```
SELECT DATE_TRUNC('DAY',lpep_pickup_datetime),
       count(passenger_count)
FROM
    green_taxi_trips
WHERE DATE_TRUNC('DAY',lpep_pickup_datetime)='2019-01-01' and
      passenger_count=3
GROUP BY
    DATE_TRUNC('DAY',lpep_pickup_datetime)
```

We find out that the correct answer is **1282** and **254** respectively

The screenshot shows the pgAdmin interface. On the left, the 'Browser' pane displays a database schema with various objects like Domains, FTS Configurations, and Tables. The 'Tables' section is expanded, showing 'green\_taxi\_trips' and other tables. The 'green\_taxi\_trips' table is selected, and its 'Columns (21)' are listed. The main workspace shows the SQL query being run:

```
1 SELECT DATE_TRUNC('DAY',lpep_pickup_datetime),
2        count(passenger_count)
3 FROM
4    green_taxi_trips
5 WHERE DATE_TRUNC('DAY',lpep_pickup_datetime)='2019-01-01' and
6      passenger_count=2
7 GROUP BY
8    DATE_TRUNC('DAY',lpep_pickup_datetime)
```

Below the query, the 'Data Output' tab is active, showing the result of the query:

date_trunc	count
2019-01-01 00:00:00	1282

Figure 5: running the SQL command in pgAdmin

This screenshot shows the pgAdmin interface again, with the same schema and table selection as Figure 5. The main workspace shows the same SQL query being run:

```
1 SELECT DATE_TRUNC('DAY',lpep_pickup_datetime),
2        count(passenger_count)
3 FROM
4    green_taxi_trips
5 WHERE DATE_TRUNC('DAY',lpep_pickup_datetime)='2019-01-01' and
6      passenger_count=3
7 GROUP BY
8    DATE_TRUNC('DAY',lpep_pickup_datetime)
```

Below the query, the 'Data Output' tab is active, showing the result of the query:

date_trunc	count
2019-01-01 00:00:00	254

Figure 6: running the SQL command in pgAdmin

## Question 6. Largest tip

For the passengers picked up in the Astoria Zone which was the drop off zone that had the largest tip? We want the name of the zone, not the id.

Note: it's not a typo, it's `tip` , not `trip`

- Central Park
- Jamaica
- South Ozone Park
- Long Island City/Queens Plaza

### **Question 6 Largest tip Solution**

*The correct answer is Long Island City/Queens Plaza. Running the following snippet in Postgresql*

```
select
    lpep_pickup_datetime,
    lpep_dropoff_datetime,
    zpu."Zone" as pickupZone,
    zdo."Zone" as dropoffZone,
    t.tip_amount
from
    green_taxi_trips t join taxi_zone_lookup zpu
        on t."PUlocationID"=zpu."LocationID"
join taxi_zone_lookup zdo
    on      t."DOLocationID"=zdo."LocationID"
where
    zpu."Zone"='Astoria'
order by
    t.tip_amount desc
```

pgAdmin File Object Tools Help

Browser Properties SQL Statistics Dependencies Dependents Processes ny\_taxi/root@Docker localhost\*

Query History Scratch

```

1 select
2     lpep_pickup_datetime,
3     lpep_dropoff_datetime,
4     zpu."Zone" as pickupZone,
5     zdo."Zone" as dropoffZone,
6     t.tip_amount
7 from
8     green_taxi_trips t join taxi_zone_lookup zpu
9         on t."PULocationID"=zpu."LocationID"
10    join taxi_zone_lookup zdo
11        on t."DOLocationID"=zdo."LocationID"
12 where
13     zpu."Zone"='Astoria'
14 order by
15     tip_amount desc

```

Data Output Messages Notifications

	lpep_pickup_datetime timestamp without time zone	lpep_dropoff_datetime timestamp without time zone	pickupzone text	dropoffzone text	tip_amount double precision
1	2019-01-26 00:46:06	2019-01-26 00:50:10	Astoria	Long Island City/Queens Plaza	88
2	2019-01-24 12:58:02	2019-01-24 13:29:59	Astoria	Central Park	30
3	2019-01-17 08:21:52	2019-01-17 08:56:59	Astoria	[null]	25
4	2019-01-20 22:48:21	2019-01-20 23:18:05	Astoria	Jamaica	25
5	2019-01-02 04:17:39	2019-01-02 05:09:07	Astoria	[null]	19.16
6	2019-01-01 03:21:12	2019-01-01 04:23:07	Astoria	Astoria	18.16
7	2019-01-01 14:09:40	2019-01-01 14:54:11	Astoria	Coney Island	16.95
8	2019-01-20 07:20:49	2019-01-20 07:28:37	Astoria	Long Island City/Queens Plaza	16.47
9	2019-01-24 23:17:35	2019-01-24 23:17:40	Astoria	Astoria	15

**Figure 7:** running the SQL command in pgAdmin