Module 1 Homework

Docker & SQL

In this homework we'll prepare the environment and practice with Docker and SQL

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:

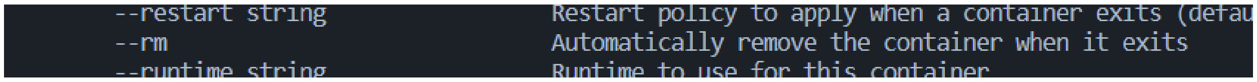
```
docker build --help
```

Do the same for "docker run".

Which tag has the following text? - *Automatically remove the container when it exits*

- `--delete`
- `--rc`
- `--rmc`
- `--rm`

ANSWER: -RM = automatically remove the container when it exists



```
      --restart string                 Restart policy to apply when a container exits (defau
      --rm                             Automatically remove the container when it exits
      --runtime string                 Runtime to use for this container
```

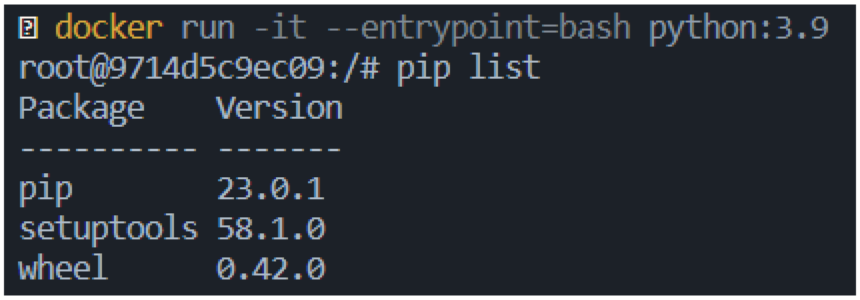
Question 2. Understanding docker first run

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`).

What is version of the package *wheel* ?

- 0.42.0
- 1.0.0
- 23.0.1
- 58.1.0

ANSWER: wheel version = 0.42.0



```
docker run -it --entrypoint=bash python:3.9
root@9714d5c9ec09:/# pip list
Package    Version
---------- -------
pip        23.0.1
setuptools 58.1.0
wheel      0.42.0
```

Prepare Postgres

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

```
wget
https://github.com/DataTalksClub/nyc-tlc-data/releases/download/green/
green_tripdata_2019-09.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)

Question 3. Count records

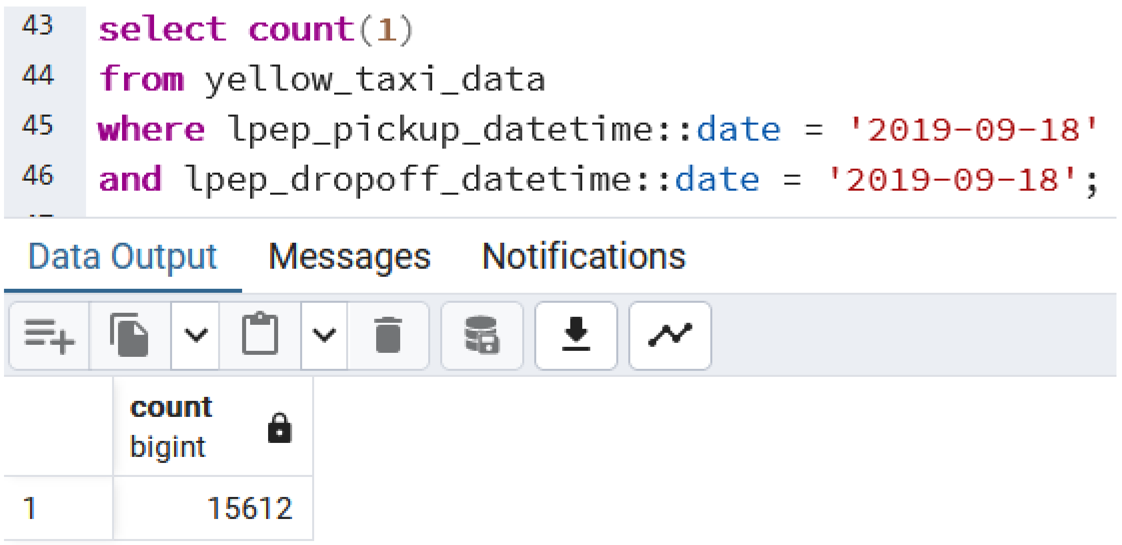
How many taxi trips were totally made on September 18th 2019?

Tip: started and finished on 2019-09-18.

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

- 15767
- 15612
- 15859
- 89009

ANSWER: 15612



```sql
select count(1)
from yellow_taxi_data
where lpep_pickup_datetime::date = '2019-09-18'
and lpep_dropoff_datetime::date = '2019-09-18';
```

Data Output Messages Notifications

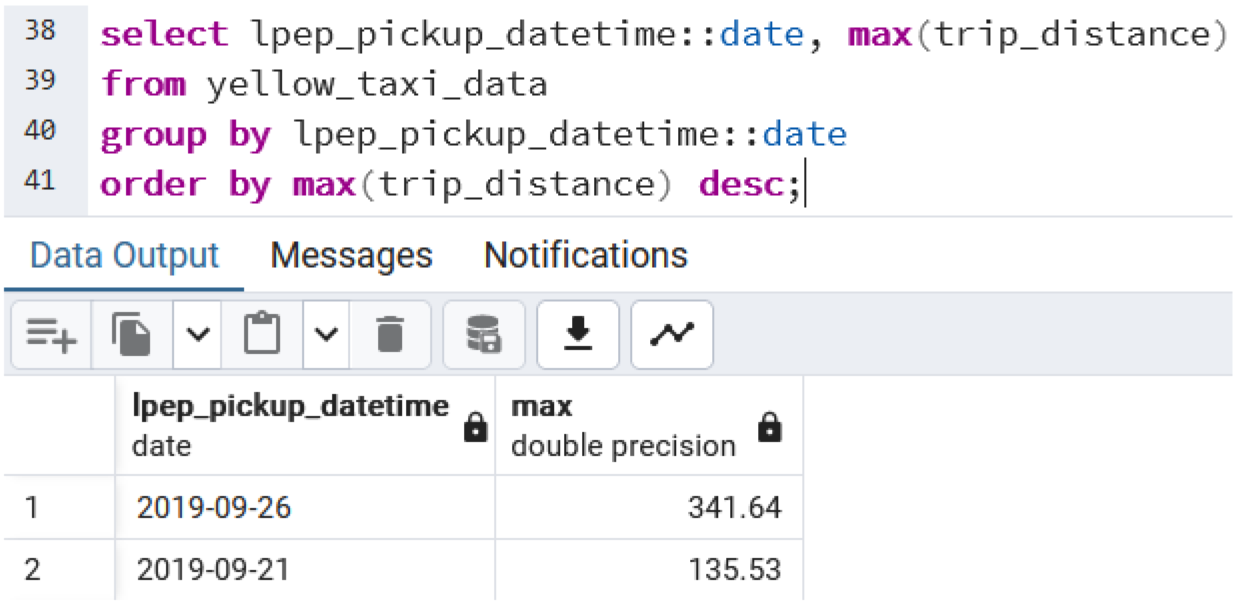
	count bigint
1	15612

Question 4. Largest trip for each day

Which was the pick up day with the largest trip distance Use the pick up time for your calculations.

- 2019-09-18
- 2019-09-16
- 2019-09-26
- 2019-09-21

ANSWER: 2019-09-26



```sql
select lpep_pickup_datetime::date, max(trip_distance)
from yellow_taxi_data
group by lpep_pickup_datetime::date
order by max(trip_distance) desc;
```

Data Output Messages Notifications

	lpep_pickup_datetime date	max double precision
1	2019-09-26	341.64
2	2019-09-21	135.53

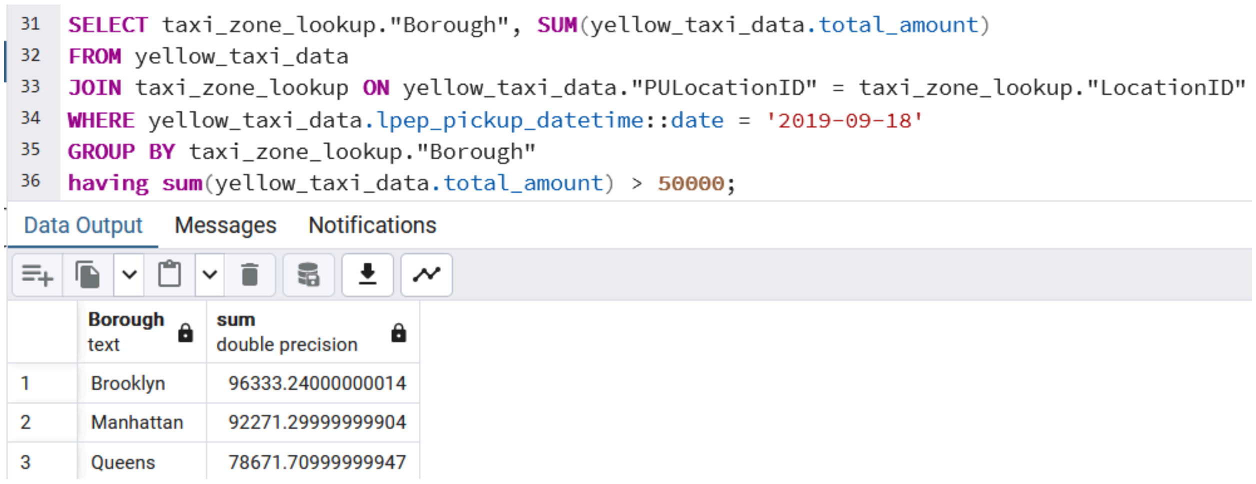
Question 5. Three biggest pick up Boroughs

Consider lpep_pickup_datetime in '2019-09-18' and ignoring Borough has Unknown

Which were the 3 pick up Boroughs that had a sum of total_amount superior to 50000?

- "Brooklyn" "Manhattan" "Queens"
- "Bronx" "Brooklyn" "Manhattan"
- "Bronx" "Manhattan" "Queens"
- "Brooklyn" "Queens" "Staten Island"

Answer: Brooklyn, Manhattan, Queens



```sql
SELECT taxi_zone_lookup."Borough", SUM(yellow_taxi_data.total_amount)
FROM yellow_taxi_data
JOIN taxi_zone_lookup ON yellow_taxi_data."PULocationID" = taxi_zone_lookup."LocationID"
WHERE yellow_taxi_data.lpep_pickup_datetime::date = '2019-09-18'
GROUP BY taxi_zone_lookup."Borough"
having sum(yellow_taxi_data.total_amount) > 50000;
```

	Borough text	sum double precision
1	Brooklyn	96333.24000000014
2	Manhattan	92271.29999999904
3	Queens	78671.70999999947

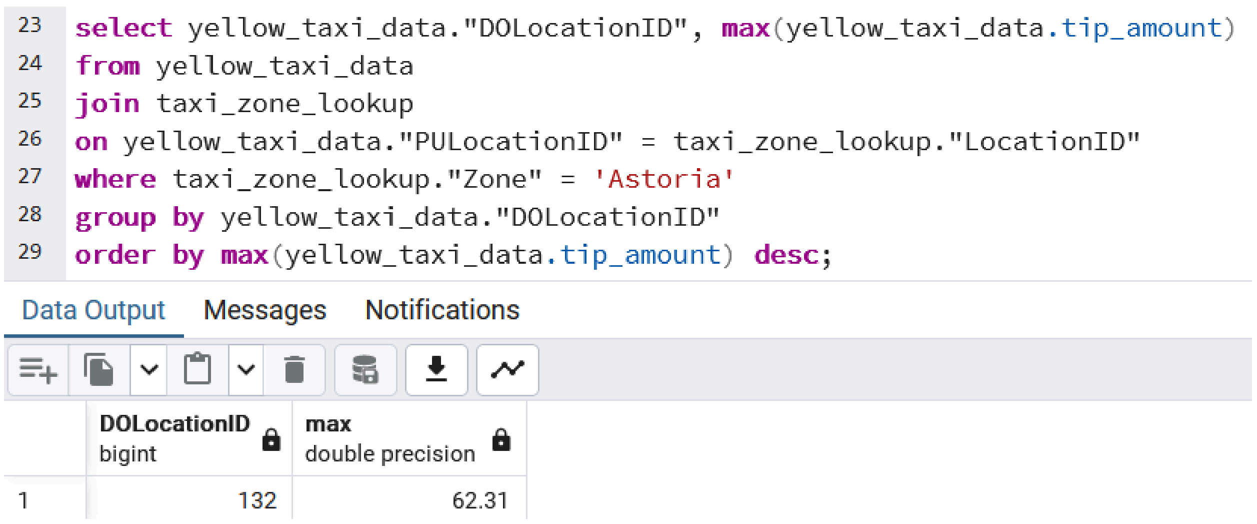
Question 6. Largest tip

For the passengers picked up in September 2019 in the zone name Astoria 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
- JFK Airport
- Long Island City/Queens Plaza

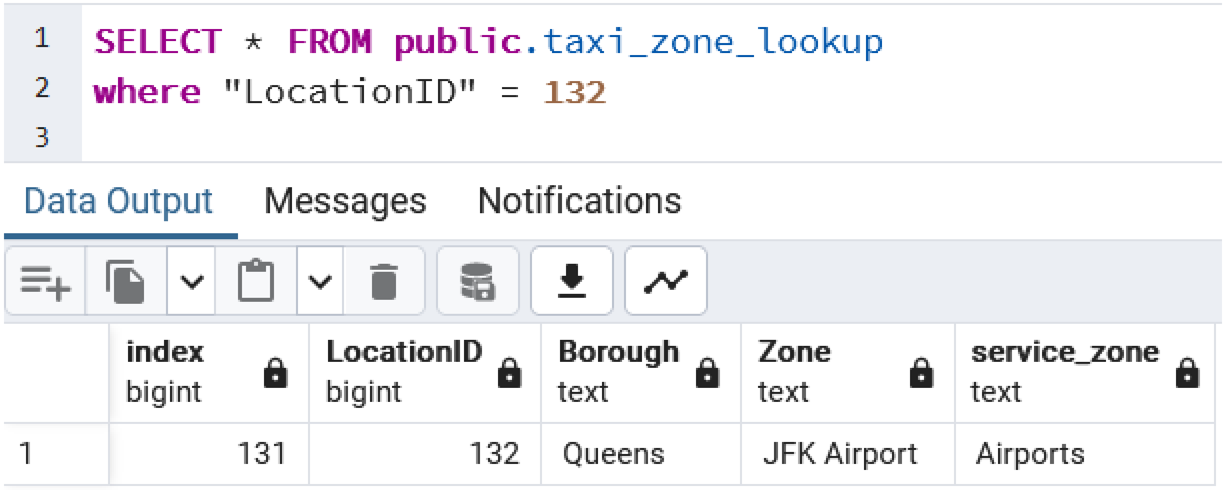
ANSWER: JFK airport



```
select yellow_taxi_data."DOLocationID", max(yellow_taxi_data.tip_amount)
from yellow_taxi_data
join taxi_zone_lookup
on yellow_taxi_data."PULocationID" = taxi_zone_lookup."LocationID"
where taxi_zone_lookup."Zone" = 'Astoria'
group by yellow_taxi_data."DOLocationID"
order by max(yellow_taxi_data.tip_amount) desc;
```

	DOLocationID bigint	max double precision
1	132	62.31

Largest tip_amount came from DOLocationID of 132.



```
SELECT * FROM public.taxi_zone_lookup
where "LocationID" = 132
```

	index bigint	LocationID bigint	Borough text	Zone text	service_zone text
1	131	132	Queens	JFK Airport	Airports

Terraform

In this section homework we'll prepare the environment by creating resources in GCP with Terraform.

In your VM on GCP/Laptop/GitHub Codespace install Terraform. Copy the files from the course repo here to your VM/Laptop/GitHub Codespace.

Modify the files as necessary to create a GCP Bucket and Big Query Dataset.

Question 7. Creating Resources

After updating the main.tf and variable.tf files run:

terraform apply

Paste the output of this command into the homework submission form.

```
Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:

  + create



Terraform will perform the following actions:



  # google_bigquery_dataset.demo_dataset will be created

  + resource "google_bigquery_dataset" "demo_dataset" {

        + creation_time              = (known after apply)

        + dataset_id         = "demo_dataset"

        + default_collation          = (known after apply)

        + delete_contents_on_destroy = false

        + effective_labels           = (known after apply)

        + etag               = (known after apply)

        + id                 = (known after apply)

        + is_case_insensitive        = (known after apply)

        + last_modified_time         = (known after apply)

        + location           = "US"

        + max_time_travel_hours      = (known after apply)

        + project            = "rapid-will-412408"

        + self_link          = (known after apply)
        + storage_billing_model      = (known after apply)

        + terraform_labels           = (known after apply)

        }


# google_storage_bucket.demo-bucket will be created

+ resource "google_storage_bucket" "demo-bucket" {

        + effective_labels           = (known after apply)

        + force_destroy              = true

        + id                 = (known after apply)

        + location           = "US"

        + name               = "terraform-demo-terra-bucket"

        + project            = (known after apply)

        + public_access_prevention        = (known after apply)

        + self_link          = (known after apply)

        + storage_class              = "STANDARD"

        + terraform_labels           = (known after apply)

        + uniform_bucket_level_access = (known after apply)

        + url                = (known after apply)


        + lifecycle_rule {

        + action {

        + type = "AbortIncompleteMultipartUpload"

        }

        + condition {
          + age                   = 1
          + matches_prefix        = []
          + matches_storage_class = []
          + matches_suffix        = []
          + with_state            = (known after apply)
        }
      }
    }

Plan: 2 to add, 0 to change, 0 to destroy.

─────────────────────────────────────────────────────────────────────────────

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to
take exactly these actions if you run "terraform apply"

now.
```

Submitting the solutions

- Form for submitting: https://courses.datatalks.club/de-zoomcamp-2024/homework/hw01
- You can submit your homework multiple times. In this case, only the last submission will be used.

Deadline: 29 January, 23:00 CET