Docker and SQL

Notes I used for preparing the videos: link

Commands

All the commands from the video

Downloading the data

```
wget https://github.com/DataTalksClub/nyc-tlc-
data/releases/download/yellow/yellow_tripdata_2021-01.csv.gz
```

> Note: now the CSV data is stored in the `csv_backup` folder, not `trip+date` like previously

Running Postgres with Docker

Windows

Running Postgres on Windows (note the full path)

```
docker run -it \
  -e POSTGRES_USER="root" \
  -e POSTGRES_PASSWORD="root" \
  -e POSTGRES_DB="ny_taxi" \
  -v c:/Users/alexe/git/data-engineering-
zoomcamp/week_1_basics_n_setup/2_docker_sql/ny_taxi_postgres_data:/var/lib/postgresql/
data \
  -p 5432:5432 \
  postgres:13
```

If you have the following error:

```
docker run -it \
  -e POSTGRES_USER="root" \
  -e POSTGRES_PASSWORD="root" \
  -e POSTGRES_DB="ny_taxi" \
  -v
e:/zoomcamp/data_engineer/week_1_fundamentals/2_docker_sql/ny_taxi_postgres_data:/var/
lib/postgresql/data  \
  -p 5432:5432 \
  postgres:13

docker: Error response from daemon: invalid mode: \Program
Files\Git\var\lib\postgresql\data.
See 'docker run --help'.
```

Change the mounting path. Replace it with the following:

```
-v /e/zoomcamp/...:/var/lib/postgresql/data
```

Linux and MacOS

```
docker run -it \
  -e POSTGRES_USER="root" \
  -e POSTGRES_PASSWORD="root" \
  -e POSTGRES_DB="ny_taxi" \
  -v $(pwd)/ny_taxi_postgres_data:/var/lib/postgresql/data \
  -p 5432:5432 \
  postgres:13
```

If you see that `ny_taxi_postgres_data` is empty after running the container, try these:

- Deleting the folder and running Docker again (Docker will re-create the folder)
- Adjust the permissions of the folder by running `sudo chmod a+rwx ny_taxi_postgres_data`

CLI for Postgres

Installing `pgcli`

```
pip install pgcli
```

If you have problems installing `pgcli` with the command above, try this:

```
conda install -c conda-forge pgcli
pip install -U mycli
```

Using `pgcli` to connect to Postgres

```
pgcli -h localhost -p 5432 -u root -d ny_taxi
```

NY Trips Dataset

Dataset:

- https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
- https://www1.nyc.gov/assets/tlc/downloads/pdf/data_dictionary_trip_records_yellow.pdf

> According to the TLC data website, from 05/13/2022, the data will be in `.parquet` format instead of `.csv` The website has provided a useful link with sample steps to read `.parquet` file and convert it to Pandas data frame.
>
> You can use the csv backup located here, https://github.com/DataTalksClub/nyc-tlc-data/releases/download/yellow/yellow_tripdata_2021-01.csv.gz, to follow along with the video.

```
$ aws s3 ls s3://nyc-tlc
                           PRE csv_backup/
                           PRE misc/
                           PRE trip data/
```

You can refer the `data-loading-parquet.ipynb` and `data-loading-parquet.py` for code to handle both csv and parquet files. (The lookup zones table which is needed later in this course is a csv file)

> Note: You will need to install the `pyarrow` library. (add it to your Dockerfile)

pgAdmin

Running pgAdmin

```
docker run -it \
  -e PGADMIN_DEFAULT_EMAIL="admin@admin.com" \
  -e PGADMIN_DEFAULT_PASSWORD="root" \
  -p 8080:80 \
  dpage/pgadmin4
```

Running Postgres and pgAdmin together

Create a network

```
docker network create pg-network
```

Run Postgres (change the path)

```
docker run -it \
  -e POSTGRES_USER="root" \
  -e POSTGRES_PASSWORD="root" \
  -e POSTGRES_DB="ny_taxi" \
  -v c:/Users/alexe/git/data-engineering-
zoomcamp/week_1_basics_n_setup/2_docker_sql/ny_taxi_postgres_data:/var/lib/postgresql/
data \
  -p 5432:5432 \
  --network=pg-network \
  --name pg-database \
  postgres:13
```

Run pgAdmin

```
docker run -it \
  -e PGADMIN_DEFAULT_EMAIL="admin@admin.com" \
  -e PGADMIN_DEFAULT_PASSWORD="root" \
  -p 8080:80 \
  --network=pg-network \
  --name pgadmin-2 \
  dpage/pgadmin4
```

Data ingestion

Running locally

```
URL="https://github.com/DataTalksClub/nyc-tlc-
data/releases/download/yellow/yellow_tripdata_2021-01.csv.gz"

python ingest_data.py \
  --user=root \
  --password=root \
  --host=localhost \
  --port=5432 \
  --db=ny_taxi \
  --table_name=yellow_taxi_trips \
  --url=${URL}
```

Build the image

```
docker build -t taxi_ingest:v001 .
```

On Linux you may have a problem building it:

```
error checking context: 'can't stat
'/home/name/data_engineering/ny_taxi_postgres_data''.
```

You can solve it with `.dockerignore`:

- Create a folder `data`
- Move `ny_taxi_postgres_data` to `data` (you might need to use `sudo` for that)
- Map `-v $(pwd)/data/ny_taxi_postgres_data:/var/lib/postgresql/data`
- Create a file `.dockerignore` and add `data` there
- Check this video (the middle) for more details

Run the script with Docker

```
URL="https://github.com/DataTalksClub/nyc-tlc-
data/releases/download/yellow/yellow_tripdata_2021-01.csv.gz"

docker run -it \
  --network=pg-network \
  taxi_ingest:v001 \
    --user=root \
    --password=root \
    --host=pg-database \
    --port=5432 \
    --db=ny_taxi \
    --table_name=yellow_taxi_trips \
    --url=${URL}
```

Docker-Compose

Run it:

```
docker-compose up
```

Run in detached mode:

```
docker-compose up -d
```

Shutting it down:

```
docker-compose down
```

Note: to make pgAdmin configuration persistent, create a folder `data_pgadmin`. Change its permission via

```
sudo chown 5050:5050 data_pgadmin
```

and mount it to the `/var/lib/pgadmin` folder:

```
services:
  pgadmin:
    image: dpage/pgadmin4
    volumes:
      - ./data_pgadmin:/var/lib/pgadmin
    ...
```

SQL Refresher

Pre-Requisites: If you followed the course in the given order, Docker Compose should already be running with pgdatabase and pgAdmin.

You can run the following code using Jupyter Notebook to ingest the data for Taxi Zones:

```
import pandas as pd
from sqlalchemy import create_engine

engine = create_engine('postgresql://root:root@localhost:5432/ny_taxi')
engine.connect()

!wget https://github.com/DataTalksClub/nyc-tlc-
data/releases/download/misc/taxi_zone_lookup.csv

df_zones = pd.read_csv("taxi_zone_lookup.csv")
df_zones.to_sql(name='zones', con=engine, if_exists='replace')
```

Once done, you can go to http://localhost:8080/browser/ to access pgAdmin. Don't forget to Right Click on the server or database to refresh it in case you don't see the new table.

Now start querying!

Joining Yellow Taxi table with Zones Lookup table (implicit INNER JOIN)

```sql
SELECT
    tpep_pickup_datetime,
    tpep_dropoff_datetime,
    total_amount,
    CONCAT(zpu."Borough", ' | ', zpu."Zone") AS "pickup_loc",
    CONCAT(zdo."Borough", ' | ', zdo."Zone") AS "dropoff_loc"
FROM
    yellow_taxi_trips t,
    zones zpu,
    zones zdo
WHERE
    t."PULocationID" = zpu."LocationID"
    AND t."DOLocationID" = zdo."LocationID"
LIMIT 100;
```

Joining Yellow Taxi table with Zones Lookup table (Explicit INNER JOIN)

```sql
SELECT
    tpep_pickup_datetime,
    tpep_dropoff_datetime,
    total_amount,
    CONCAT(zpu."Borough", ' | ', zpu."Zone") AS "pickup_loc",
    CONCAT(zdo."Borough", ' | ', zdo."Zone") AS "dropoff_loc"
FROM
    yellow_taxi_trips t
JOIN
-- or INNER JOIN but it's less used, when writing JOIN, postgreSQL understands
implicitly that we want to use an INNER JOIN
    zones zpu ON t."PULocationID" = zpu."LocationID"
JOIN
    zones zdo ON t."DOLocationID" = zdo."LocationID"
LIMIT 100;
```

Checking for records with NULL Location IDs in the Yellow Taxi table

```sql
SELECT
    tpep_pickup_datetime,
    tpep_dropoff_datetime,
    total_amount,
    "PULocationID",
    "DOLocationID"
FROM
    yellow_taxi_trips
WHERE
    "PULocationID" IS NULL
    OR "DOLocationID" IS NULL
LIMIT 100;
```

Checking for Location IDs in the Zones table NOT IN the Yellow Taxi table

```sql
SELECT
    tpep_pickup_datetime,
    tpep_dropoff_datetime,
    total_amount,
    "PULocationID",
    "DOLocationID"
FROM
    yellow_taxi_trips
WHERE
    "DOLocationID" NOT IN (SELECT "LocationID" from zones)
    OR "PULocationID" NOT IN (SELECT "LocationID" from zones)
LIMIT 100;
```

Using LEFT, RIGHT, and OUTER JOINS when some Location IDs are not in either Tables

```sql
DELETE FROM zones WHERE "LocationID" = 142;

SELECT
    tpep_pickup_datetime,
    tpep_dropoff_datetime,
    total_amount,
    CONCAT(zpu."Borough", ' | ', zpu."Zone") AS "pickup_loc",
    CONCAT(zdo."Borough", ' | ', zdo."Zone") AS "dropoff_loc"
FROM
    yellow_taxi_trips t
LEFT JOIN
    zones zpu ON t."PULocationID" = zpu."LocationID"
JOIN
    zones zdo ON t."DOLocationID" = zdo."LocationID"
LIMIT 100;
```

```
SELECT
    tpep_pickup_datetime,
    tpep_dropoff_datetime,
    total_amount,
    CONCAT(zpu."Borough", ' | ', zpu."Zone") AS "pickup_loc",
    CONCAT(zdo."Borough", ' | ', zdo."Zone") AS "dropoff_loc"
FROM
    yellow_taxi_trips t
RIGHT JOIN
    zones zpu ON t."PULocationID" = zpu."LocationID"
JOIN
    zones zdo ON t."DOLocationID" = zdo."LocationID"
LIMIT 100;
```

```
SELECT
    tpep_pickup_datetime,
    tpep_dropoff_datetime,
    total_amount,
    CONCAT(zpu."Borough", ' | ', zpu."Zone") AS "pickup_loc",
    CONCAT(zdo."Borough", ' | ', zdo."Zone") AS "dropoff_loc"
FROM
    yellow_taxi_trips t
OUTER JOIN
    zones zpu ON t."PULocationID" = zpu."LocationID"
JOIN
    zones zdo ON t."DOLocationID" = zdo."LocationID"
LIMIT 100;
```

Using GROUP BY to calculate number of trips per day

```
SELECT
    CAST(tpep_dropoff_datetime AS DATE) AS "day",
    COUNT(1)
FROM
    yellow_taxi_trips
GROUP BY
    CAST(tpep_dropoff_datetime AS DATE)
LIMIT 100;
```

Using ORDER BY to order the results of your query

```sql
-- Ordering by day

SELECT
    CAST(tpep_dropoff_datetime AS DATE) AS "day",
    COUNT(1)
FROM
    yellow_taxi_trips
GROUP BY
    CAST(tpep_dropoff_datetime AS DATE)
ORDER BY
    "day" ASC
LIMIT 100;

-- Ordering by count

SELECT
    CAST(tpep_dropoff_datetime AS DATE) AS "day",
    COUNT(1) AS "count"
FROM
    yellow_taxi_trips
GROUP BY
    CAST(tpep_dropoff_datetime AS DATE)
ORDER BY
    "count" DESC
LIMIT 100;
```

Other kinds of aggregations

```sql
SELECT
    CAST(tpep_dropoff_datetime AS DATE) AS "day",
    COUNT(1) AS "count",
    MAX(total_amount) AS "total_amount",
    MAX(passenger_count) AS "passenger_count"
FROM
    yellow_taxi_trips
GROUP BY
    CAST(tpep_dropoff_datetime AS DATE)
ORDER BY
    "count" DESC
LIMIT 100;
```

Grouping by multiple fields

```sql
SELECT
    CAST(tpep_dropoff_datetime AS DATE) AS "day",
    "DOLocationID",
    COUNT(1) AS "count",
    MAX(total_amount) AS "total_amount",
    MAX(passenger_count) AS "passenger_count"
FROM
    yellow_taxi_trips
GROUP BY
    1, 2
ORDER BY
    "day" ASC,
    "DOLocationID" ASC
LIMIT 100;
```