Docker + SQL

- What docker is, why we need it
- Running postgres locally with docker
- Putting some data for testing to local postgres with Python
- Packaging this script in docker
- Running postgres and the script in one network
- Docker compose and running pgadmin and postres together with docker-compose
- SQL: group by, joins, window function, union

Docker

- Docker allows you to put everything an application needs inside a container - sort of a box that contains everything: OS, system-level libraries, python, etc.
- You run this box on a host machine. The container is completely isolated from the host machine env.
- In the container you can have Ubuntu 18.04, while your host is running on Windows.
- You can run multiple containers on one host and they won't have any conflict.
- An image = set of instructions that were executed + state. All saved in "image"
- Installing docker: https://docs.docker.com/get-docker/

Why should data engineers care about containerization and docker?

- Setting up things locally for experiments
- Integration tests, CI/CD
- Batch jobs (AWS Batch, Kubernetes jobs, etc — outside of the scope)
- Spark
- Serverless (AWS Lambda)
- So containers are everywhere

Simple example

- Python 3.8
- Extend it - install some libraries like pandas
- More - next

Postgres

- We can set it up locally to practice SQL and test things before doing this in the cloud

Running postgres with Docker (taken from Airflow's environment - you'll see it later in the next week)

```
docker run -it \
  -e POSTGRES_USER="root" \
  -e POSTGRES_PASSWORD="root" \
  -e POSTGRES_DB="ny_taxi" \
  -v "./ny-taxi-
volume:/var/lib/postgresql/data" \
  -p 5432:5432 \
  postgres:13
```

It's running, now let's test it

I assume you have Python. If you don't, I recommend Anaconda - https://www.anaconda.com/products/individual. It's simpler to install than other Python distributions + it already has a lot of useful libraries

```
pip install pgcli
```

This installs a client for postgres that we'll use

Let's connect to it:

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

And check what's there

```
\dt
```

Not much. Let's put some data

Taxi Rides dataset + putting data to Postgres

Download the dataset: https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2021-01.csv

- "Data dictionary" - https://www1.nyc.gov/assets/tlc/downloads/pdf/data_dictionary_trip_records_yellow
- Zones - https://s3.amazonaws.com/nyc-tlc/misc/taxi+_zone_lookup.csv

We'll have a simple Python script for doing that (based on https://stackoverflow.com/questions/23103962/how-to-write-dataframe-to-postgres-table)

```
from sqlalchemy import create_engine
engine =
create_engine('postgresql://root:root@localhost:5432/ny_taxi')
```

Looking at types + DDL

```
print(pd.io.sql.get_schema(df, 'yellow_taxi',
con=engine))
```

```
CREATE TABLE yellow_taxi (
    "VendorID" BIGINT,
    tpep_pickup_datetime TEXT,
    tpep_dropoff_datetime TEXT,
    passenger_count BIGINT,
    trip_distance FLOAT(53),
    "RatecodeID" BIGINT,
    store_and_fwd_flag TEXT,
    "PULocationID" BIGINT,
    "DOLocationID" BIGINT,
    payment_type BIGINT,
    fare_amount FLOAT(53),
    extra FLOAT(53),
    mta_tax FLOAT(53),
    tip_amount FLOAT(53),
    tolls_amount FLOAT(53),
    improvement_surcharge FLOAT(53),
    total_amount FLOAT(53),
    congestion_surcharge FLOAT(53)
)
```

- Types - not optimal
- Also, dates as text (pd.to_datetime)
- Better types: https://www.singlestore.com/blog/nyc-taxi-data-ingested-into-memsql/

Create table:

```
df.head(0).to_sql('yellow_taxi', engine,
if_exists='replace', index=False)
```

Insert data:

```
df.to_sql('yellow_taxi', engine, if_exists='append',
index=False)
```

- exploration and querying
- pgAdmin - the standard graphical tool for postgres for that - https://www.pgadmin.org/
- Let's also run it with docker

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

However, this docker container can't access the postgres container. We need to link them

Docker network

docker network create pg

```
docker run -it \
  -e POSTGRES_USER="root" \
  -e POSTGRES_PASSWORD="root" \
  -e POSTGRES_DB="ny_taxi" \
  -v "./ny-taxi-
volume:/var/lib/postgresql/data" \
  -p 5432:5432 \
  --name pgdatabase \
  --net pg \
  postgres:13
```

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

It works, but we need to keep two terminal tabs running, manually create a network - and a bunch of other things. Let's use compose that will take care of that.

If you have docker desktop (windows and mac), you already have docker compose
On Linux, you need to install it
https://docs.docker.com/compose/install/

Let's create docker-compose.yaml

```
services:
  pgdatabase:
    image: postgres:13
    restart: always
    environment:
      POSTGRES_USER: root
      POSTGRES_PASSWORD: root
      POSTGRES_DB: ny_taxi
    volumes:
      - "./ny-taxi-
volume:/var/lib/postgresql/data:rw"
    ports:
      - "5432:5432"
  pgadmin:
    image: dpage/pgadmin4
    restart: always
    environment:
      PGADMIN_DEFAULT_EMAIL:
admin@admin.com
      PGADMIN_DEFAULT_PASSWORD: root
    ports:
      - "8080:80"
```

And then do docker-compose up

And then create a database connection using pgdatabase

SQL

Finally let's do SQL

(To be updated)

Vic's queries:

Docker + SQL


```
-- Revenue calculation
sum(fare_amount) as revenue_monthly_fare,
sum(extra) as revenue_monthly_extra,
sum(mta_tax) as revenue_monthly_mta_tax,
sum(tip_amount) as revenue_monthly_tip_amount,
sum(tolls_amount) as revenue_monthly_tolls_amount,
sum(ehail_fee) as revenue_monthly_ehail_fee,
sum(improvement_surcharge) as revenue_monthly_improvement_surcharge,
sum(total_amount) as revenue_monthly_total_amount,
sum(congestion_surcharge) as revenue_monthly_congestion_surcharge,

-- Additional calculations
count(tripid) as total_monthly_trips,
avg(passenger_count) as avg_montly_passenger_count,
avg(trip_distance) as avg_montly_trip_distance

from trips_data
group by 1,2,3
```

https://github.com/DataTalksClub/data-engineering-zoomcamp/blob/main/week_5_analytics_engineering/taxi_rides_ny/models/data-marts/dm_monthly_zone_revenue.sql

```
select * from green_data
union all
select * from yellow_data
```

```
51      trips_unioned.payment_type_description,
52      trips_unioned.congestion_surcharge
53  from trips_unioned
54  inner join dim_zones as pickup_zone
55  on trips_unioned.pickup_locationid = pickup_zone.locationid
56  inner join dim_zones as dropoff_zone
57  on trips_unioned.dropoff_locationid = dropoff_zone.locationid
```

```
36  where vendorid is not null
37    -- qualify row_number() over(partition by tripid) = 1
```