Learning Data Engineering

General

- NY Taxi Data is available as Parquet format
    - Pandas has functions to convert to CSV which can be easily ingested into Pandas Dataframe
- GCP Account is required for the program. 300 USD free account is available for first time users.
    - Service Account with adequate permissions to GCP Bucket and BigQuery
- Tools required for the program
    - Anaconda Environment
    - Pandas
    - Jupyter Notebook to try commands
    - Docker Commands to manager containers
    - Postgres and PGAdmin
    - Mage AI (Python) for ELT pipelines to move data from Buckets or files to BigQuery
    - Dbt for data pipeline to transform data within Warehouse
    - GCP BigQuery
    - GCP Looker Studio for visualization

Basic Commands

Docker Commands

```
# Create a Docker Image from a base image
Docker run -it ubuntu bash

#List docker images
Docker images list

#List  Running containers
Docker ps -a

#List with full container ids
Docker ps -a --no-trunc

#Add onto existing image to create new image
Docker commit -a <User_Name> -m "Message" container_id New_Image_Name

# Create a Docker Image with an entrypoint from a base image
Docker run -it --entry_point=bash python:3.11


#Attach to a stopped container
Docker start -ai <Container_Name>

#Attach to a running container
docker exec -it <Container_ID> bash
```

#copying from host to container
Docker cp <SRC_PATH/file> <containerid>:<dest_path>

#copying from container to host
Docker cp <containerid>:<Srct_path> <Dest Path on host/file>

#Create an image from a docker file
Docker build -t <Image_Name> <Location of Dockerfile>

#DockerFile Options and best practices
https://devopscube.com/build-docker-image/

#Docker delete all images forcefully
docker rmi -f $(docker images -aq)

#Docker delete all containers forcefully
docker rm -f $(docker ps -qa)

#docker compose creation
https://www.composerize.com/

Docker for PostGreSQL and PGAdmin

If Docker is created individually

Network
Docker network create pg-network

PostGres
```
docker run -it \
-e POSTGRES_USER="root" \
-e POSTGRES_PASSWORD="root" \
-e POSTGRES_DB="ny_taxi" \
-v $(pwd)/ny_taxi_data:/var/lib/postgresql/data \
-p 5432:5432 \
--network="pg-network" \
--name="pgdatabase" \
postgres:16
```

PGAdmin4

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

With Docker Compose

```
services:
  pgdatabase:
    image: postgres:16
    environment:
      - POSTGRES_USER=root
      - POSTGRES_PASSWORD=root
      - POSTGRES_DB=ny_taxi
    volumes:
      - "./ny_taxi_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"
  taxi-ingest:
    image: taxi_ingest:v001
    command: --user=root --password=root --host=pgdatabase --port=5432 --db=ny_taxi
             --table_name=yellow_taxi_data --
url=https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2023-10.parquet
```

GCP Commands

1. Create SSH Keys
2. Added to the Settings of Compute Engine VM Instance
3. SSH-ed into the VM Instance with a config similar to following

```
Host my-website.com
    HostName my-website.com
    User my-user
    IdentityFile ~/.ssh/id_rsa
```

4. Installed Anaconda by installing the sh file through bash <Anaconda.sh>
5. Install Docker after
    a. Sudo apt-get update
    b. Sudo apt-get docker
6. To run Docker without SUDO permissions
    a. https://github.com/sindresorhus/guides/blob/main/docker-without-sudo.md
7. Google cloud remote copy
    a. gcloud compute scp LOCAL_FILE_PATHVM_NAME:REMOTE_DIR

Install GCP Cloud SDK on Docker Machine

https://stackoverflow.com/questions/23247943/trouble-installing-google-cloud-sdk-in-ubuntu

sudo apt-get install apt-transport-https ca-certificates gnupg && echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg] https://packages.cloud.google.com/apt cloud-sdk main"| sudo tee -a /etc/apt/sources.list.d/google-cloud-sdk.list&& curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key --keyring /usr/share/keyrings/cloud.google.gpg add - && sudo apt-get update && sudo apt-get install google-cloud-sdk && sudo apt-get install google-cloud-sdk-app-engine-java && sudo apt-get install google-cloud-sdk-app-engine-python && gcloud init

Anaconda Commands

#Activate environment
Conda Activate <environment_name>

#DeActivate environment
Conda DeActivate <environment_name>

#Start iterm without conda environment
conda config --set auto_activate_base false

Using Conda forge as default (Community driven packaging recipes and solutions)
https://conda-forge.org/docs/user/introduction.html

conda --version
conda update conda
```
conda config --add channels conda-forge
conda config --set channel_priority strict
```

#Using Libmamba as Solver
conda install pgcli --solver=libmamba

Linux/MAC Commands

Starting and Stopping Services on Linux

- sudo systemctl start postgresql
- sudo systemctl stop postgresql

Starting and Stopping Services on MAC

- launchctl start postgresql
- launchctl stop postgresql

Identifying processes listening to a Port across MAC/Linux

sudo lsof -i -P -n | grep LISTEN
$ sudo netstat -tulpn | grep LISTEN
$ sudo ss -tulpn | grep LISTEN
$ sudo lsof -i:22 ## see a specific port such as 22 ##
$ sudo nmap -sTU -O IP-address-Here

Installing a package on Debian

sudo apt install <packagename>

Listing all package on Debian

Dpkg -l | grep <packagename>

UnInstalling a package on Debian

Sudo apt remove <packagename>
Sudo apt autoclean && sudo apt autoremove

List all Processes on Debian/Ubuntu

Ps -aux

apt-get update && apt-get install procps
apt-get install iproute2 for ss -tulpn

#Postgres Install
sudo sh -c 'echo "deb https://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
sudo apt-get update
sudo apt-get -y install postgresql

#Changing Postgresql port to 5432

- sudo service postgresql stop - sed -e `'s/^port.*/port = 5432/'` /etc/postgresql/10/main/postgresql.conf > postgresql.conf
- sudo chown postgres postgresql.conf
- sudo mv postgresql.conf /etc/postgresql/10/main
- sudo systemctl restart postgresql

Module 1 – Docker and Docker compose to create docker images, starting and stopping

- **Setting up GCP VM Instance**
  - Activated the 300 USD Free tier
  - Created a GCP Project
  - Created Service Account with permissions to Compute Admin, Compute Engine Service Agent
  - Created E2-Small VM Instance
  - Got the Public IP Address
- **Creating the local SSH Key**
  - https://cloud.google.com/compute/docs/connect/create-ssh-keys#console
  - ssh-keygen -t rsa -f ~/.ssh/KEY_FILENAME-C USERNAME-b 2048
  - Cat keyname.pub | pbcopy
- SSHed into Google VM with public iP Address
  Created a Config for easy VM
  Host my-website.com
  HostName my-website.com
  User my-user
  IdentityFile ~/.ssh/id_rsa

- Installed Anaconda
  - Sudo apt-get update
  - Wget https://repo.anaconda.com/archive/Anaconda3-2023.09-0-Linux-x86_64.sh
  - Bash Anaconda3-2023.09-0-Linux-x86_64.sh
  - Post Installing Conda, if Conda command is not found, please try
    - source ~/anaconda3/etc/profile.d/conda.sh

- Adding Code-Forge (Community repository of recipes) as a Channel and LibMamba as Solver for faster search of Conda Repository

  conda update conda

  ```
  conda config --add channels conda-forge
  conda config --set channel_priority strict

  #Using Libmamba as Solver
  conda install pgcli  --solver=libmamba
  ```

- If PIP Install pgcli gives issue, conda install can be used to install pgcli

- Install Docker
  - Sudo apt-get install docker.io
  - Running Docker without sudo
    1. To run Docker without SUDO permissions
       1. https://github.com/sindresorhus/guides/blob/main/docker-without-sudo.md

- Install Docker Compose
  - Install Docker compose from github release location https://github.com/docker/compose/releases/tag/v2.24.2
  - Add Path to bin
- Install PGCLI
  - If PIP Install PGCLI gives error, Here pgcli is installed from conda-forge with conda install -c conda-forge pgcli
  - Followed by "pip install -U mycli"
- Leverage Docker Compose to install Postgres and PGAdmin

```yaml
services:
  pgdatabase1:
    image: postgres:16
    environment:
      - POSTGRES_USER=root
      - POSTGRES_PASSWORD=root
      - POSTGRES_DB=ny_taxi1
    volumes:
      - "./ny_taxi1_data:/var/lib/postgresql/data:rw"
    ports:
      - "5432:5432"
    restart: always
  pgadmin1:
    image: dpage/pgadmin4
    environment:
      - PGADMIN_DEFAULT_EMAIL=admin@admin.com
      - PGADMIN_DEFAULT_PASSWORD=root
    ports:
      - "8080:80"
    restart: always
```

  - Leverage Docker Compose to install Postgres and PGAdmin

Module 2 – Workflow Orchestration

- Workflow Orchestration helps setting up a Workflow pipeline for Data Load/Ingestion, transform and load
- Mage.AI supports connectors to various data sources and has support for languages such as Python, SQL and R
- With Mage, in this module, following were the exercises
    - Setup One Pipeline
        - Loading the data from CSV or Parquet
        - Transform step to clean the data to remove tables with passenger count 0
        - Data Exporter to Postgresql
        - Data Exporter to S3 bucket
        - Data Exporter to S3 bucket (Partitioned – tpep_pickup_time)
        - Data Exporter to S3 bucket (Partitioned and Clustered – Cluster by Vendor ID)
        - Partition and Cluster help increase the speed when the data is high
    - Another pipeline
        - Load data from S3 bucket
        - Make column names lower
        - Load it into BQ

Module 3 – Data Warehouse (GCP BQ)

- Workflow Orchestration helps setting up a Workflow pipeline for Data Load/Ingestion, transform and load
- Clustering vs Partitioning in BQ
- Creating Partitioned and Clustered tables for Yellow taxi data

Partitioning vs Clustering

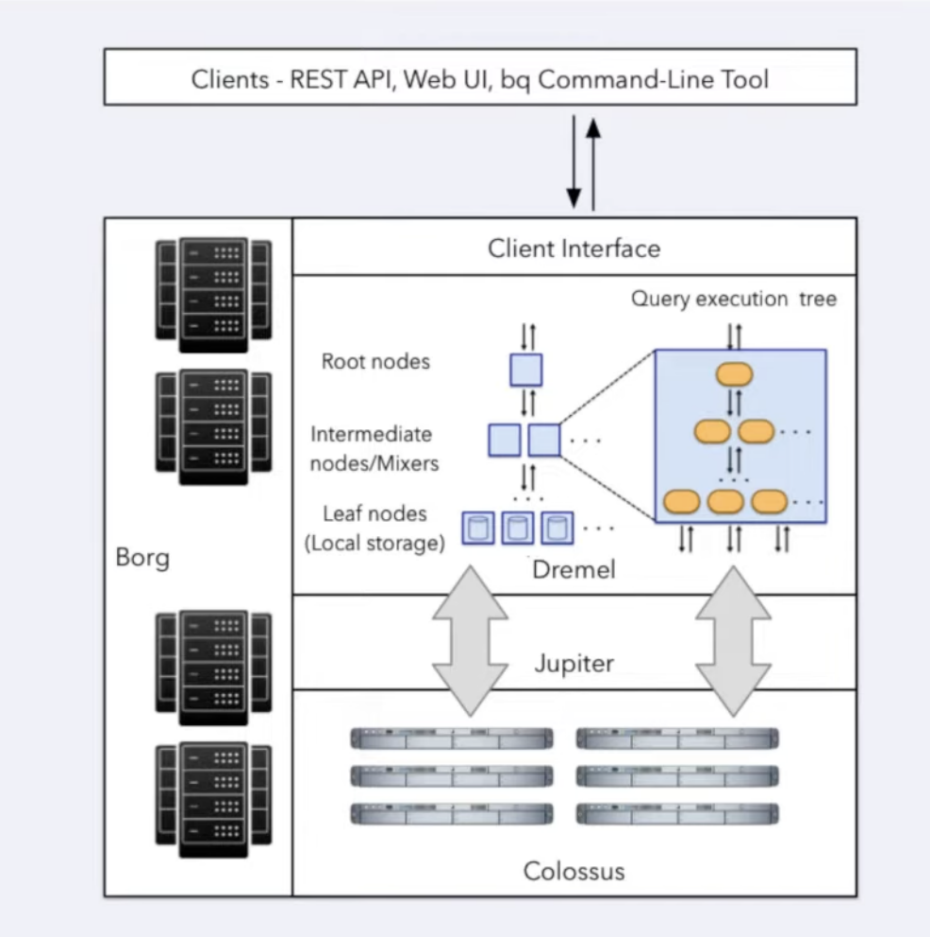
Clustering	Partitoning
Cost benefit unknown	Cost known upfront
You need more granularity than partitioning alone allows	You need partition-level management.
Your queries commonly use filters or aggregation against multiple particular columns	Filter or aggregate on single column
The cardinality of the number of values in a column or group of columns is large	

- BQ Best practices

- Query performance
    - Filter on partitioned columns
    - Denormalizing data
    - Use nested or repeated columns
    - Use external data sources appropriately
    - Don't use it, in case u want a high query performance
    - Reduce data before using a JOIN
    - Do not treat WITH clauses as prepared statements
    - Avoid oversharding tables

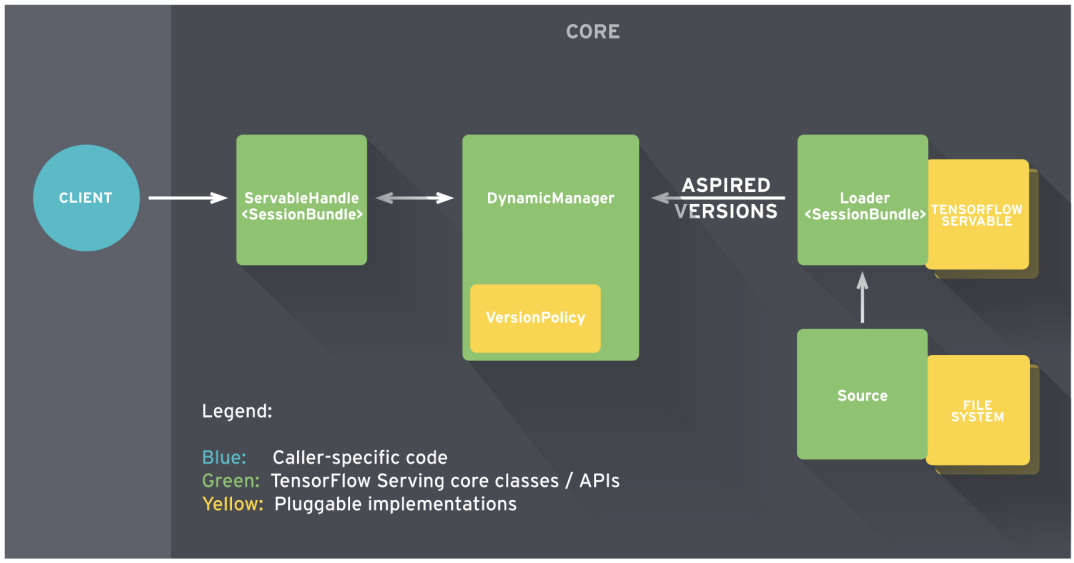
- Query performance
    - Avoid JavaScript user-defined functions
    - Use approximate aggregation functions (HyperLogLog++)
    - Order Last, for query operations to maximize performance
    - Optimize your join patterns
    - As a best practice, place the table with the largest number of rows first, followed by the table with the fewest rows, and then place the remaining tables by decreasing size.

- BQ Internals – Uses column oriented structure
    - Colossus – Cheap storage in Columnar format
    - Dremel – Compute – Query execution engine which executes each sub query
    - Jupiter – High speed network – 1 TBPS
    - Borg – Orchestrator precursor to Kubernetes



- In BQ, a model was created to predict Tips
    - A table was created to support Model with columns of interest such as Pickup Location, Drop Location, Payment type, Fare Amount, Passenger count, tip amount, tolls amount.
    - A default Model with 'Linear Regression' created with input as 'Tips Amount'
    - ML – Feature_Info
    - ML – Evaluate
    - ML – Explain_Predict
- Model Deployment
    - Extract the model into a GS Bucket
    - In the GCP VM, copy from bucket to a temp folder
    - Pull Tensorflow serving to host the model
    - Call the model with Postman to test the model

```
## Model deployment
[Tutorial](https://cloud.google.com/bigquery-ml/docs/export-model-tutorial)
### Steps
- gcloud auth login
- bq --project_id taxi-rides-ny extract -m nytaxi.tip_model gs://taxi_ml_model/tip_model
- mkdir /tmp/model
- gsutil cp -r gs://taxi_ml_model/tip_model /tmp/model
- mkdir -p serving_dir/tip_model/1
- cp -r /tmp/model/tip_model/* serving_dir/tip_model/1
- docker pull tensorflow/serving
- docker run -p 8501:8501 --mount type=bind,source=`pwd`/serving_dir/tip_model,target=
/models/tip_model -e MODEL_NAME=tip_model -t tensorflow/serving &
- curl -d '{"instances": [{"passenger_count":1, "trip_distance":12.2, "PULocationID":"193", "DOLocationI
- http://localhost:8501/v1/models/tip_model
```



Module 4 – Analytics Engineering

- Data Processing with DBT

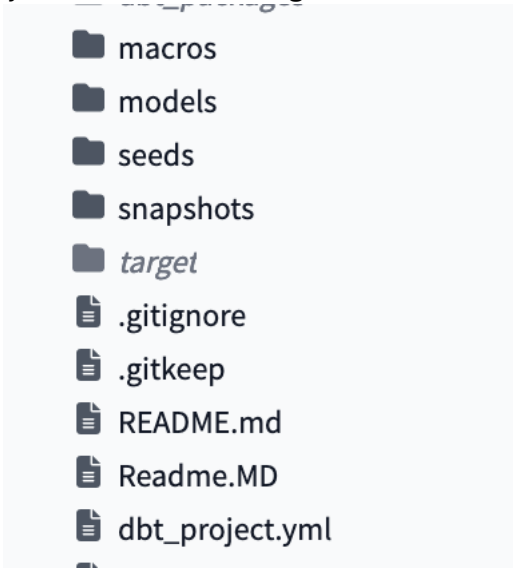
Alternative A	Alternative B
Setting up dbt for using BigQuery (cloud)	Setting up dbt for using Postgres locally
- Open a free developer dbt cloud account following this link	- Open a free developer dbt cloud account following this link
- Following these instructions to connect to your BigQuery instance	- follow the official dbt documentation or - follow the dbt core with BigQuery on Docker guide to setup dbt locally on docker or - use a docker image from oficial Install with Docker.
- More detailed instructions in dbt_cloud_setup.md	- You will need to install the latest version with the BigQuery adapter (dbt-bigquery).
	- You will need to install the latest version with the postgres adapter (dbt-postgres).
	After local installation you will have to set up the connection to PG in the `profiles.yml`, you can find the templates here

DBT helps with ‘T’ of the ELT part. Traditionally, ETL was done (Extract, Transform and Load) but with Modern Data stack, it is ELT in which data is extracted from various places, loaded/ingested into Data Lake or Warehouses. And transformation happens with DBT.

One of the confusions I had was, the data steps were like whatever was done with Mage AI. For example, could not have I achieved creating the Facts and everything from Mage itself instead of DBT (Data Build tool). Then my limited understanding was the following

- DBT Modelling is centered around Data Analysts.
- Language is SQL.
- The joins, tables or Views and creation of facts happen within the data lake.
- DBT helps with following.
    - Version Control like Coding in another language
    - Helps with Lineage, which would be a big thing if data sources are disparate, and the flow is complex.
- Reusable Macros
- DBT can be a step in orchestrators such as Mage AI or AirFlow or Perfect.

Steps done in this exercise.

- Created a Cloud IDE Project (Non-Enterprise are limited to one project in Cloud DBT)
- Connected to Github Account
- Connected to BigQuery
- The project had the following structure when created
    - 

- In dbt_project.yml, changed the Project_name in
    - Name
    - Models
- Added a Staging folder to Models. Created schema.yml
    - Under Schema, imported the two tables from BigQuery under Sources. (There is Auto-complete)

```yaml
version: 2

sources:
  - name: staging
    database: trans-array-412413
    schema: ny_taxi

    tables:
      Generate model
      - name: yellow_cab_data
      Generate model
      - name: green_cab_data
```

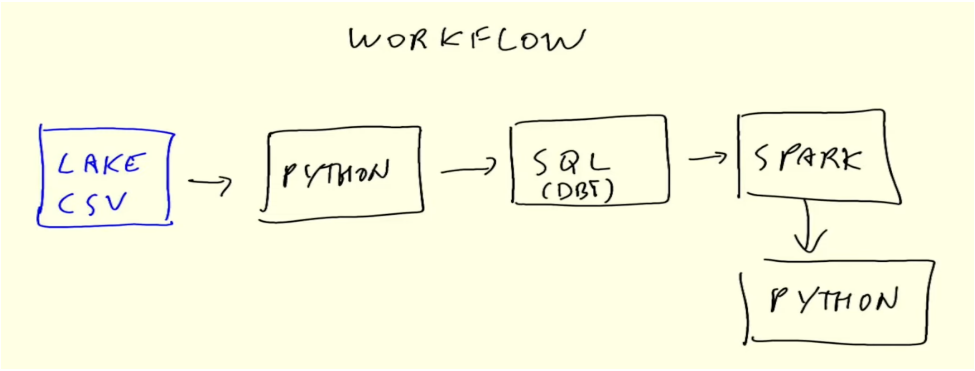
    - Generate Model helped generate the model for the two tables from above.
    - Once generated, Lineage is created with SRC (Source) and Model.
- JINJA templates
    - DBT uses Jinja templates to resolve. {{ }}
    - {{ ref() }} refers to data sources
    - {% %} for expressions, statements
    - {% macro %} for macros.
- Three folders in DBT project are important
    - Models
    - Macros
    - Seeds
- Macros are User-Defined functions which can be called from Models
    - Get_Payment_description macro is created and added to Green Cab Data model and Yellow Cab Data model.
- Seeds is a folder, in which raw files can be loaded at Project initiation. In this case, Taxi_Zone_Lookup.csv is added.
- Dbt_utils and dbt codegen are packages in dbt with lots of custom functions. Dbt deps imports all the functions after including the following packages. Code gen helps generating schema on the fly.

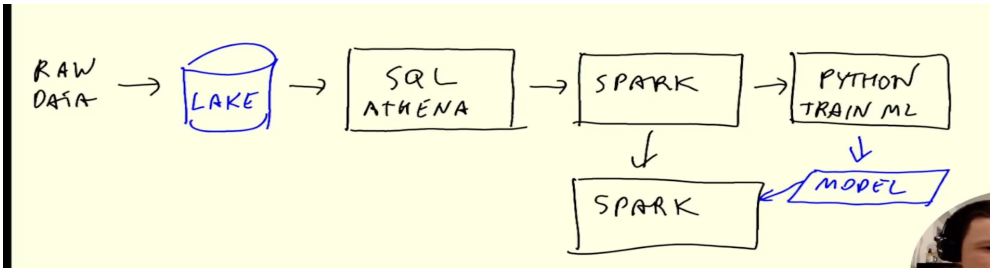
```yaml
packages:
  - package: dbt-labs/dbt_utils
    version: 1.1.1
  - package: dbt-labs/codegen
    version: 0.12.1
```

- Create a Core folder under Models. Create the following.
    - Dim_Zones
    - Fact_trips
- Adding tests and documentation for the DBT
    - Dbt docs help create documentation for the DBT package
    - Observability can be added.
- Setup Nightly builds for DBT to ensure data freshness and facts are created based on recent data

Module 5

- Data Processing
  - Batch Processing
  - Streaming Processing





To run Linux Shell Commands on Python, append with a !.

!wget https://d37ci6vzurychx.cloudfront.net/trip-data/fhvhv_tripdata_2023-12.parquet

```
import pandas as pd
df = pd.read_parquet('fhvhv_tripdata_2023-12.parquet')
df.to_csv('fhvhv.csv')
!wc -l fhvhv.csv
!head -1001 fhvhv.csv > head.csv
```

This module posed few interesting challenges

- My VM Instance type was e2-. Now I changed it to e2-standard-8.
- Version mismatch between PySpark and Pandas. Got this error

  AttributeError: 'DataFrame' object has no attribute 'iteritems'
  https://datatalks-club.slack.com/archives/C01FABYF2RG/p1708957776952089?thread_ts=1708957678.763619&cid=C01FABYF2RG

  PySpark - 3.5.1
  Pandas 2.2.1

- Memory Error for PySpark. So when creating the SparkSession, initialised with higher Executor Memory and Driver Memory of 4GB as default is only 1 GB

https://sparkbyexamples.com/spark/how-to-set-apache-spark-executor-memory/

I choose the below method

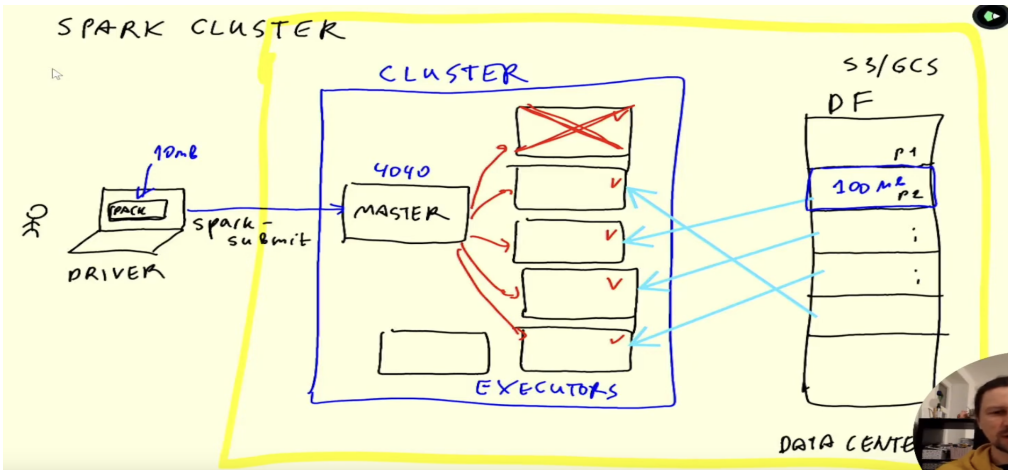
```
spark = SparkSession.builder \
        .master("local[*]") \
        .config("spark.executor.memory", "4g") \
        .config("spark.driver.memory", "4g") \
        .appName('test') \
        .getOrCreate()
```

Spark Module

- Spark Python
- Spark DataFrames
  - Create a Pandas Data Frame
  - Spark DataFrame is created with following code for the top 100 rows to derive schema
    - spark.createDataFrame(df_pandas)
  - Schema is derived
  - Create a Schema.Py which is compatible with Python SQL Types
  - Create Spark DF for the whole CSV

```
df = spark.read \
.option("header", "true") \
.schema(schema) \
.csv('fhvhv.csv')
```

  - *Leverage Spark Functions (from pyspark.sql import functions as F)*
  - User Defined Functions
    - crazy_stuff_udf = F.udf(crazy_stuff, returnType=types.StringType())
- Used a Shell Script to download all the Yellow and Green trip data to CSV
- Spark SQL
  - With Spark SQL, arrive at revenue models and write to revenue Parquet
- Anatomy of Spark
  - Executors



- Spark Group By
- Spark Joins
- Resilient Distributed Datasets
- Spark on Docker
- Spark on Cloud