MLOps

1강

최민우

Introduction

목차

MLOps

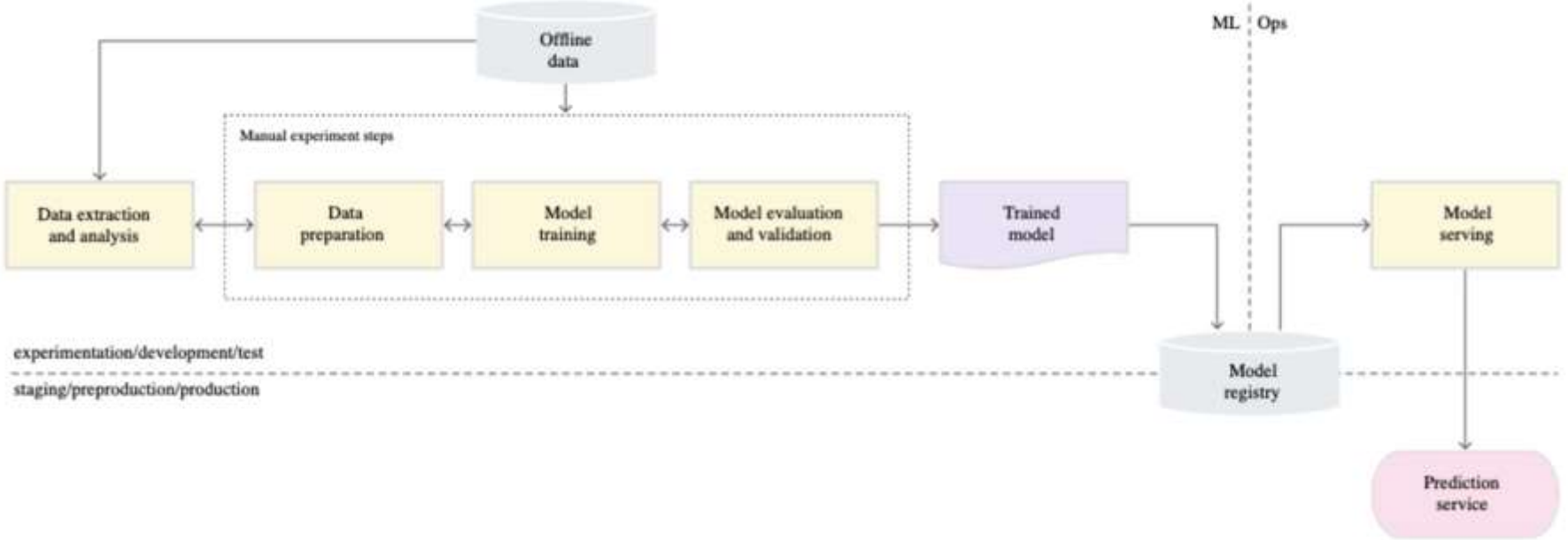
1.Machine Learning + DevOps(Development + Operation)

2.소프트웨어 개발에서 DevOps의 원칙과 방법론을 머신러닝에 적용

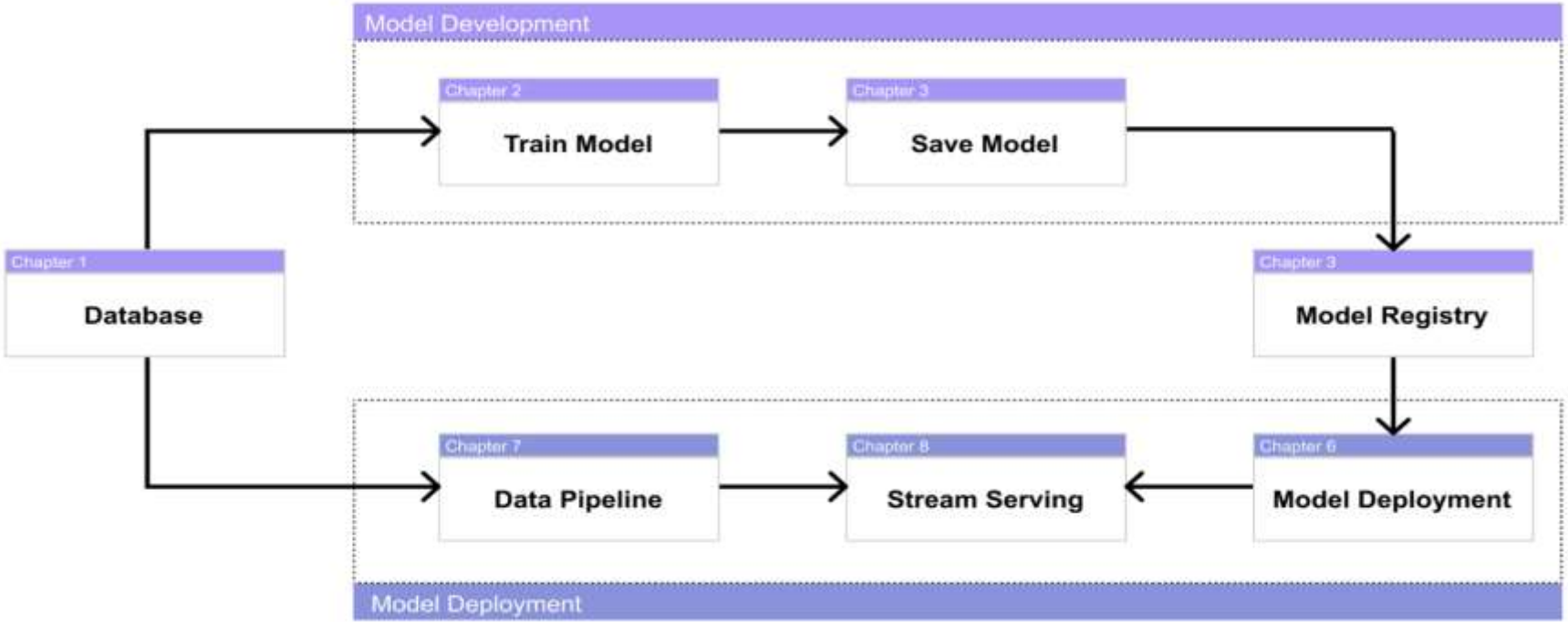
3.머신러닝 모델을 안정적으로 배포하고, 효율적으로 유지보수하기 위한 방법론

4.예) 머신러닝 = 자동차 엔진, MLOps = 엔진으로 자동차를 만드는 과정

MLOps


Offline data
ML
Ops
Manual experiment steps
Data extraction and analysis
Data preparation
Model training
Model evaluation and validation
Trained model
Model serving
experimentation/development/test
staging/preproduction/production
Model registry
Prediction service

MLOps


Model Development
Chapter 2
Train Model
Chapter 3
Save Model
Chapter 1
Database
Chapter 3
Model Registry
Chapter 7
Data Pipeline
Chapter 8
Stream Serving
Chapter 6
Model Deployment
Model Deployment

Docker

1.모든 컴포넌트를 Docker로 실행

2.머신러닝 모델을 다룰 때, 성능 재현은 가장 중요한 부분

3.OS, 파이썬 버전, 패키지 버전, 코드, 가중치 등이 모델을 학습했을 때의 환경과 동일

4.동일한 환경을 제공하기 위한 소프트웨어 = Docker

5.이러한 이유로, MLOps 엔지니어는 Docker를 다룰 수 있어야 함

https://www.youtube.com/watch?v=Ps8HDIAyPD0&list=PLuHgQVnccGMDeMJsGq2O-55Ymtx0IdKWf&index=1

작업환경 구축

1.Docker 설치: https://docs.docker.com/get-docker/

2.설치 완료하면, CMD에서 docker images 입력(Desktop에서 실행)

3.PostgreSQL 설치: https://www.postgresql.org/download/

4.application stack builder는 수행하지 않아도 됨

Database

목차

Overview

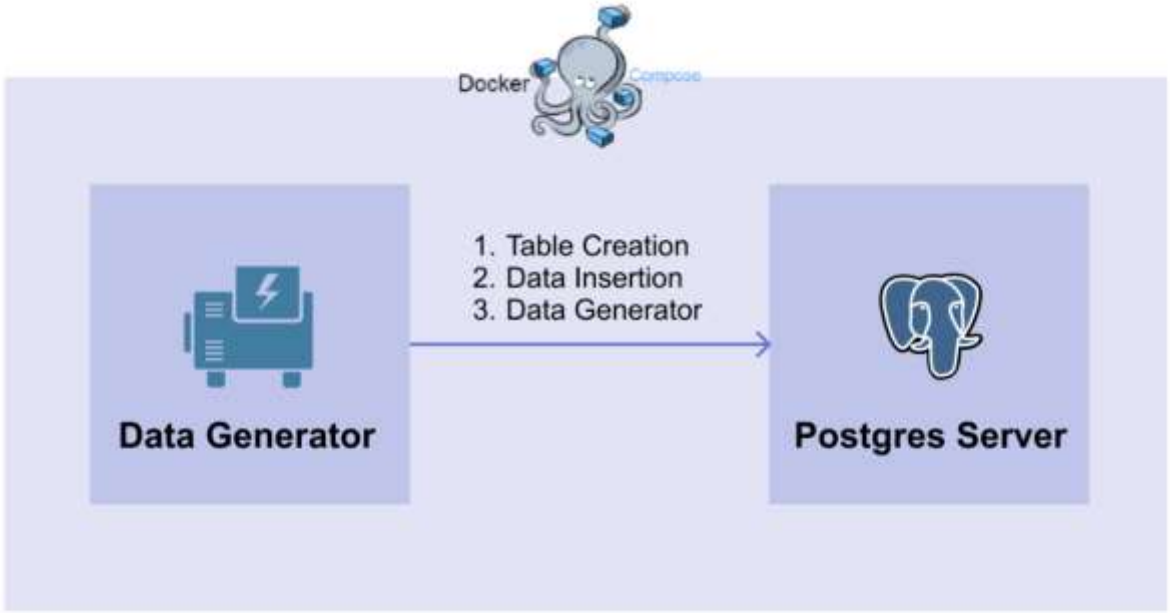
1.Docker 를 이용하여 DB server 를 생성

2.psycopg2 패키지를 이용하여 테이블 생성 및 데이터 삽입

3.Dockerfile 과 Docker Compose 파일을 생성

4.Docker 컨테이너 안에서 계속해서 데이터를 생성하는 서비스를 구축

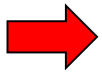
Overview


Docker
Compose
1. Table Creation
2. Data Insertion
3. Data Generator
Data Generator
Postgres Server

DB 서버 생성

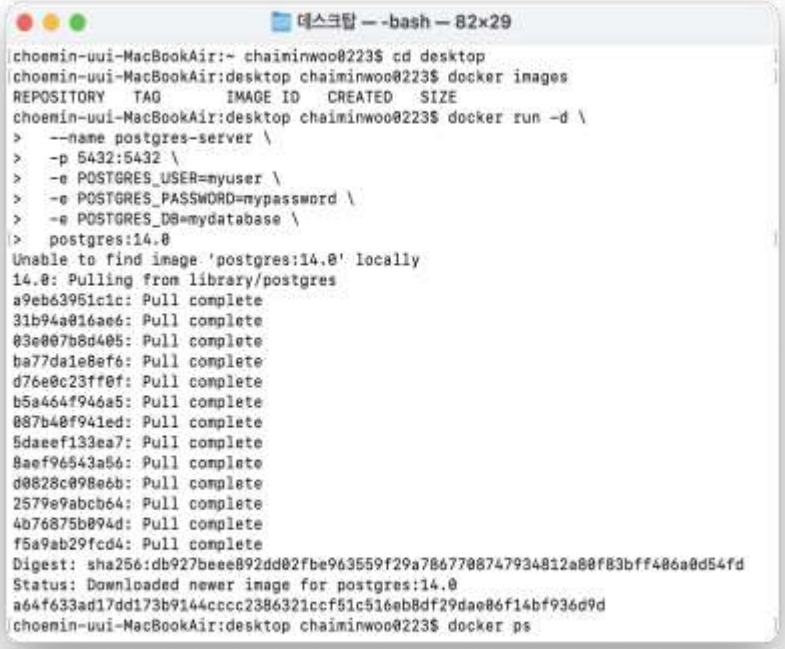
1.docker run 명령어를 사용하면, 간단한 옵션들을 통해 DB 서버를 생성
2.docker ps 명령어를 통해, DB 서버가 잘 동작하는지 확인

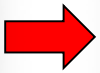
```
$ docker run -d \
  --name postgres-server \
  -p 5432:5432 \
  -e POSTGRES_USER=myuser \
  -e POSTGRES_PASSWORD=mypassword \
  -e POSTGRES_DB=mydatabase \
  postgres:14.0
```

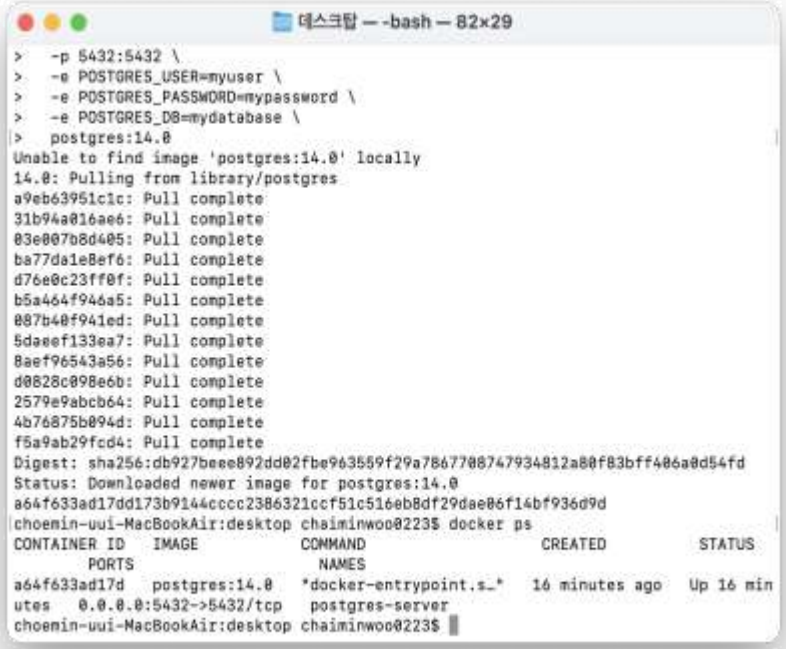


```
$ docker ps
```

DB 서버 생성


데스크탑 — -bash — 82×29
choemin-uui-MacBookAir:~ chaiminwoo0223$ cd desktop
choemin-uui-MacBookAir:desktop chaiminwoo0223$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
choemin-uui-MacBookAir:desktop chaiminwoo0223$ docker run -d \
> --name postgres-server \
> -p 5432:5432 \
> -e POSTGRES_USER=myuser \
> -e POSTGRES_PASSWORD=mypassword \
> -e POSTGRES_DB=mydatabase \
> postgres:14.0
Unable to find image 'postgres:14.0' locally
14.0: Pulling from library/postgres
a9eb63951c1c: Pull complete
31b94a016ae6: Pull complete
03e007b8d405: Pull complete
ba77da1e8ef6: Pull complete
d76e0c23ff0f: Pull complete
b5a464f946a5: Pull complete
087b40f941ed: Pull complete
5daeef133ea7: Pull complete
8aef96543a56: Pull complete
d0828c098e6b: Pull complete
2579e9abcb64: Pull complete
4b76875b094d: Pull complete
f5a9ab29fcd4: Pull complete
Digest: sha256:db927beee892dd02fbe963559f29a7867708747934812a80f83bff406a0d54fd
Status: Downloaded newer image for postgres:14.0
a64f633ad17dd173b9144cccc2386321ccf51c516eb8df29dae06f14bf936d9d
choemin-uui-MacBookAir:desktop chaiminwoo0223$ docker ps




데스크탑 — -bash — 82×29
> -p 5432:5432 \
> -e POSTGRES_USER=myuser \
> -e POSTGRES_PASSWORD=mypassword \
> -e POSTGRES_DB=mydatabase \
> postgres:14.0
Unable to find image 'postgres:14.0' locally
14.0: Pulling from library/postgres
a9eb63951c1c: Pull complete
31b94a016ae6: Pull complete
03e007b8d405: Pull complete
ba77da1e8ef6: Pull complete
d76e0c23ff0f: Pull complete
b5a464f946a5: Pull complete
087b40f941ed: Pull complete
5daeef133ea7: Pull complete
8aef96543a56: Pull complete
d0828c098e6b: Pull complete
2579e9abcb64: Pull complete
4b76875b094d: Pull complete
f5a9ab29fcd4: Pull complete
Digest: sha256:db927beee892dd02fbe963559f29a7867708747934812a80f83bff406a0d54fd
Status: Downloaded newer image for postgres:14.0
a64f633ad17dd173b9144cccc2386321ccf51c516eb8df29dae06f14bf936d9d
choemin-uui-MacBookAir:desktop chaiminwoo0223$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS
PORTS NAMES
a64f633ad17d postgres:14.0 "docker-entrypoint.s…" 16 minutes ago Up 16 min
utes 0.0.0.0:5432->5432/tcp postgres-server
choemin-uui-MacBookAir:desktop chaiminwoo0223$

DB 서버 확인

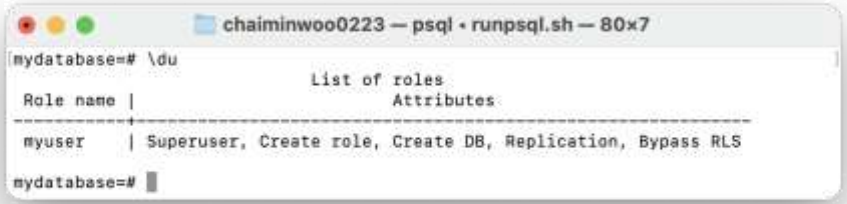
1.psql 실행
2.psql 명령어를 통해, DB 서버가 잘 동작하는지 확인

```
PGPASSWORD=mypassword psql -h localhost -p 5432 -U myuser -d mydatabase
```

- *PGPASSWORD=* :
  - 접속할 유저의 비밀번호를 입력합니다.
- *-h* :
  - 호스트를 지정합니다.
- *-p* :
  - 포트를 지정합니다.
- *-U* :
  - 접속할 유저의 이름을 입력합니다.
- *-d* :
  - DB 의 이름을 입력합니다.

DB 서버 확인

1.접속 성공
2.\du 를 통해 DB 의 role name 과 attributes 을 확인



테이블 생성

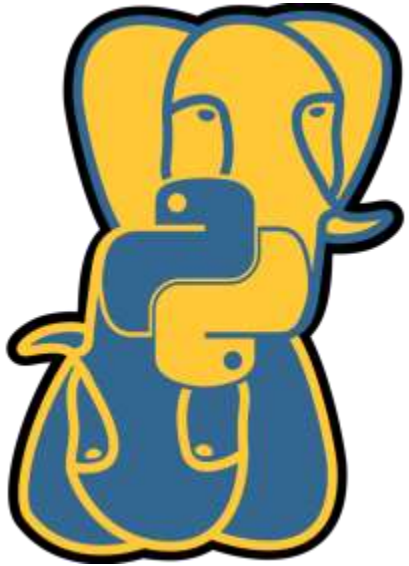
1.패키지 설치: pip3 install pandas psycopg2-binary scikit-learn

2.Python을 이용하여, PostgreSQL DB 서버에 접근하는 코드를 구현(psycopg2 패키지)

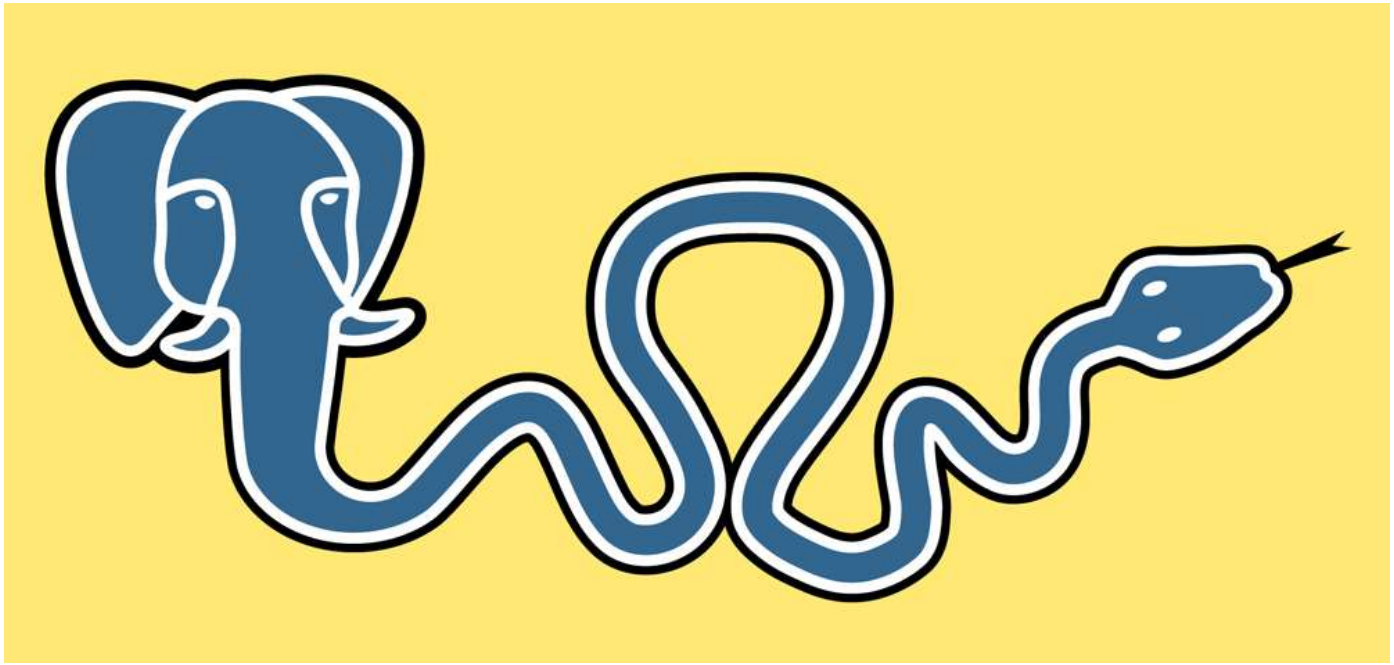
3.DB Connection → Table Creation Query → Query 실행

테이블 생성

테이블 생성



테이블 생성



테이블 생성

1) DB Connection

- psycopg2 를 이용하여 DB 에 접근하기 위해서는 connect 함수를 사용
- DB 에 연결할 때 *user*, *password*, *host*, *port*, *database* 의 총 5가지 정보가 필요

```
import psycopg2

db_connect = psycopg2.connect(
    user="myuser",
    password="mypassword",
    host="localhost",
    port=5432,
    database="mydatabase",
)
```

테이블 생성

2) Table Creation Query

- PostgreSQL에서는 float64, int 64를 지원하지 않음
- PostgreSQL에서는 float8, int를 사용

```
create_table_query = """
CREATE TABLE IF NOT EXISTS iris_data (
    id SERIAL PRIMARY KEY,
    timestamp timestamp,
    sepal_length float8,
    sepal_width float8,
    petal_length float8,
    petal_width float8,
    target int
);"""
```

테이블 생성

3) Query 실행

- Visual Studio Code 터미널에서 python table_creator.py를 입력
- 또는, 오른쪽 상단의 실행 버튼을 클릭

```
$ python table_creator.py

    CREATE TABLE IF NOT EXISTS iris_data (
        id SERIAL PRIMARY KEY,
        timestamp timestamp,
        sepal_length float8,
        sepal_width float8,
        petal_length float8,
        petal_width float8,
        target int
    );
```

테이블 확인

1.psql 실행

2.\d를 통해 생성된 테이블들의 목록을 확인

3.iris_data 테이블의 데이터 전체를 확인: select * from iris_data;

```
mydatabase=# \d
                List of relations
 Schema |       Name        |   Type   | Owner
--------+-------------------+----------+--------
 public | iris_data         | table    | myuser
 public | iris_data_id_seq  | sequence | myuser
(2 rows)
```

```
mydatabase=# select * from iris_data;
 id | timestamp | sepal_length | sepal_width | petal_length | petal_width | target
----+-----------+--------------+-------------+--------------+-------------+--------
(0 rows)
```

데이터 삽입

1.생성한 테이블에 iris 데이터 한 줄 삽입

2.iris 데이터 불러오기 → Data Insertion Query → Query 실행

데이터 삽입

1) iris 데이터 불러오기

- 생성된 테이블의 Column 이름과 일치하도록 수정

```
import pandas as pd
from sklearn.datasets import load_iris

X, y = load_iris(return_X_y=True, as_frame=True)
df = pd.concat([X, y], axis="columns")
rename_rule = {
    "sepal length (cm)": "sepal_length",
    "sepal width (cm)": "sepal_width",
    "petal length (cm)": "petal_length",
    "petal width (cm)": "petal_width",
}
df = df.rename(columns=rename_rule)
```

데이터 삽입

2) Data Insertion Query

- 삽입 순서 중요

```
insert_row_query = f"""
INSERT INTO iris_data
    (timestamp, sepal_length, sepal_width, petal_length, petal_width, target)
    VALUES (
        NOW(),
        {data.sepal_length},
        {data.sepal_width},
        {data.petal_length},
        {data.petal_width},
        {data.target}
    );"""
```

데이터 삽입

3) Query 실행

- Visual Studio Code 터미널에서 python data_insertion.py를 입력
- 또는, 오른쪽 상단의 실행 버튼을 클릭

```
$ python data_insertion.py

    INSERT INTO iris_data
        (timestamp, sepal_length, sepal_width, petal_length, petal_width, target)
        VALUES (
            NOW(),
            6.2,
            2.9,
            4.3,
            1.3,
            1.0
        );
```

데이터 확인

1.psql 실행

2.iris_data 테이블의 데이터 전체를 확인: select * from iris_data;

```
mydatabase=# select * from iris_data;
 id | timestamp | sepal_length | sepal_width | petal_length | petal_width | target
----+-----------+--------------+-------------+--------------+-------------+--------
(0 rows)

mydatabase=# select * from iris_data;
 id |         timestamp          | sepal_length | sepal_width | petal_length | petal_width | t
arget
----+----------------------------+--------------+-------------+--------------+-------------+--
------
  1 | 2023-12-30 11:03:42.166353 |            6 |         2.7 |          5.1 |         1.6 |
    1
(1 row)
```

데이터 생성 Loop

1.생성된 테이블 안에 데이터를 계속해서 추가

2.Loop 추가 → Query 실행 → 데이터 확인

데이터 생성 Loop

1) Loop 추가

- while True
- 너무 빠른 시간에 데이터가 추가되면, DB에 과부하가 생길 수 있음
- 데이터를 삽입 후 잠시 대기하는 시간을 추가: time.sleep(1)

```
import time

def generate_data(db_connect, df):
    while True:
        insert_data(db_connect, df.sample(1).squeeze())
        time.sleep(1)
```

데이터 생성 Loop

2) Query 실행

- Visual Studio Code 터미널에서 python data_insertion.py를 입력
- 또는, 오른쪽 상단의 실행 버튼을 클릭

데이터 생성 Loop

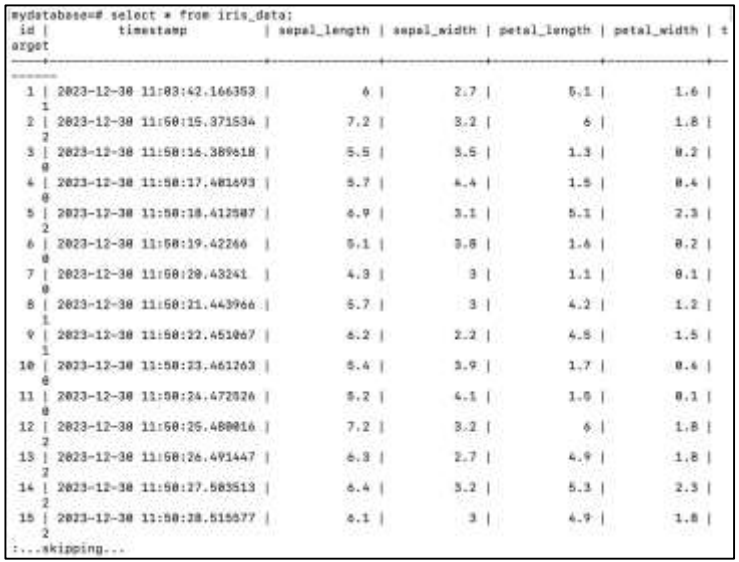
2) Query 실행

```
$ python data_insertion_loop.py

    INSERT INTO iris_data
        (timestamp, sepal_length, sepal_width, petal_length, petal_width, target)
        VALUES (
            NOW(),
            6.5,
            2.8,
            4.6,
            1.5,
            1.0
        );


    INSERT INTO iris_data
        (timestamp, sepal_length, sepal_width, petal_length, petal_width, target)
        VALUES (
            NOW(),
            5.5,
            4.2,
            1.4,
            0.2,
            0.0
        );
```

데이터 생성 Loop

3) 데이터 확인

- psql 실행
- iris_data 테이블의 데이터 전체를 확인: select * from iris_data;



```
mydatabase=# select * from iris_data;
 id |         timestamp          | sepal_length | sepal_width | petal_length | petal_width | t
arget
----+----------------------------+--------------+-------------+--------------+-------------+--
------
  1 | 2023-12-30 11:03:42.166353 |            6 |         2.7 |          5.1 |         1.6 |
  1
  2 | 2023-12-30 11:50:15.371534 |          7.2 |         3.2 |            6 |         1.8 |
  2
  3 | 2023-12-30 11:50:16.389618 |          5.5 |         3.5 |          1.3 |         0.2 |
  0
  4 | 2023-12-30 11:50:17.401693 |          5.7 |         4.4 |          1.5 |         0.4 |
  0
  5 | 2023-12-30 11:50:18.412587 |          6.9 |         3.1 |          5.1 |         2.3 |
  2
  6 | 2023-12-30 11:50:19.42266  |          5.1 |         3.8 |          1.6 |         0.2 |
  0
  7 | 2023-12-30 11:50:20.43241  |          4.3 |           3 |          1.1 |         0.1 |
  0
  8 | 2023-12-30 11:50:21.443966 |          5.7 |           3 |          4.2 |         1.2 |
  1
  9 | 2023-12-30 11:50:22.451067 |          6.2 |         2.2 |          4.5 |         1.5 |
  1
 10 | 2023-12-30 11:50:23.461263 |          5.4 |         3.9 |          1.7 |         0.4 |
  0
 11 | 2023-12-30 11:50:24.472026 |          5.2 |         4.1 |          1.5 |         0.1 |
  0
 12 | 2023-12-30 11:50:25.480016 |          7.2 |         3.2 |            6 |         1.8 |
  2
 13 | 2023-12-30 11:50:26.491447 |          6.3 |         2.7 |          4.9 |         1.8 |
  2
 14 | 2023-12-30 11:50:27.503513 |          6.4 |         3.2 |          5.3 |         2.3 |
  2
 15 | 2023-12-30 11:50:28.515577 |          6.1 |           3 |          4.9 |         1.8 |
  2
:...skipping...
```

참고자료

- https://mlops-for-mle.github.io/tutorial/
- https://www.youtube.com/playlist?list=PLuHgQVnccGMDeMJsGq2O-55Ymtx0IdKWf
- ChatGPT 4

Thank You