DATA ENGINEER BATCH 4
Mentee: Yovina Silvia
Mentor: Bilal Benefit

TASK 2 - Ingestion Data

1. We are going to create a DataFrame from a parquet file on our datasets.

- File Parquet yang dimaksud adalah `yellow_tripdata_2023-01.parquet` yang terdapat di direktori `./dataset/`.
- DataFrame dibuat dengan membaca file Parquet tersebut menggunakan library `fastparquet`.
- Kelas `Extraction` digunakan untuk mengekstraksi data dari file parquet.
- Metode `local_file` menerima path file parquet dan memuatnya ke dalam DataFrame menggunakan metode `__read_parquetfile`.
- `__read_parquetfile` membaca file parquet menggunakan pustaka `fastparquet` dan mengonversinya ke DataFrame.

```
import pandas as pd
from fastparquet import ParquetFile
from sqlalchemy import create_engine, text
from sqlalchemy.types import BigInteger, DateTime, Boolean, Float, Integer
from sqlalchemy.exc import SQLAlchemyError
from prettytable import PrettyTable

# 1. Buat DataFrame dari file parquet di folder datasets kita.
class Extraction:
    def __init__(self) -> None:
        self.path = ""
        self.dataframe = pd.DataFrame()

    def local_file(self, path: str):
        self.path = path
        self.__read_parquetfile()
        print("1. DataFrame created from parquet file:")
        print(self.dataframe.head())  # Print the first few rows of the DataFrame
        self.investigate_schema()
        self.cast_data()
```

```
        return self.dataframe

    def __read_parquetfile(self):
        parquetfile = ParquetFile(self.path)
        self.dataframe = parquetfile.to_pandas()
```

```
1. DataFrame created from parquet file:
   VendorID tpep_pickup_datetime  ... congestion_surcharge  airport_fee
0         2  2023-01-01 00:32:10  ...                 2.5         0.00
1         2  2023-01-01 00:55:08  ...                 2.5         0.00
2         2  2023-01-01 00:25:04  ...                 2.5         0.00
3         1  2023-01-01 00:03:48  ...                 0.0         1.25
4         2  2023-01-01 00:10:29  ...                 2.5         0.00

[5 rows x 19 columns]
```

2. Load the parquet file to a DataFrame with fastparquet library.

- Menggunakan library `fastparquet`, file Parquet dibaca dan diubah menjadi DataFrame.
- Metode `investigate_schema` digunakan untuk menampilkan skema DataFrame dengan menampilkan tipe data dari setiap kolom.

```
# 2. Memuat file parquet ke dalam DataFrame menggunakan pustaka fastparquet
    def investigate_schema(self):
        pd.set_option('display.max_columns', None)
        print("\n2. Schema of the DataFrame:")
        print(self.dataframe.dtypes)  # Print the schema of the DataFrame
```

```
2. Schema of the DataFrame:
VendorID                        int64
tpep_pickup_datetime     datetime64[ns]
tpep_dropoff_datetime    datetime64[ns]
passenger_count                float64
trip_distance                  float64
RatecodeID                     float64
store_and_fwd_flag              object
PULocationID                    int64
DOLocationID                    int64
payment_type                    int64
fare_amount                    float64
extra                          float64
mta_tax                        float64
tip_amount                     float64
tolls_amount                   float64
improvement_surcharge          float64
total_amount                   float64
congestion_surcharge           float64
airport_fee                    float64
dtype: object
```

3. Clean the Yellow Trip dataset.

- Dataset dibersihkan dengan metode `cast_data(self)` dari kelas `Extraction`, di mana beberapa kolom diubah tipe datanya.
- Metode `cast_data` digunakan untuk membersihkan dan mengonversi tipe data pada kolom DataFrame agar sesuai dengan tipe data yang benar.
- Langkah-langkah pembersihan:
  - Kolom `passenger_count` dikonversi menjadi tipe `Int8`.
  - Kolom `store_and_fwd_flag` dikonversi menjadi nilai boolean (`False` untuk "N" dan `True` untuk "Y").
  - Kolom `tpep_pickup_datetime` dan `tpep_dropoff_datetime` dikonversi menjadi tipe datetime.

```python
# 3. Bersihkan dataset Yellow Trip.
    def cast_data(self):
        self.dataframe["passenger_count"] = self.dataframe["passenger_count"].astype("Int8")
        self.dataframe["store_and_fwd_flag"] = self.dataframe["store_and_fwd_flag"].map({"N": False, "Y":
True}).astype("boolean")
        self.dataframe["tpep_pickup_datetime"] = pd.to_datetime(self.dataframe["tpep_pickup_datetime"])
        self.dataframe["tpep_dropoff_datetime"] = pd.to_datetime(self.dataframe["tpep_dropoff_datetime"])
        print("\n3. Cleaned DataFrame with correct data types:")
        print(self.dataframe.head())  # Print the cleaned DataFrame
```

```
3. Cleaned DataFrame with correct data types:
   VendorID tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0         2  2023-01-01 00:32:10   2023-01-01 00:40:36                1
1         2  2023-01-01 00:55:08   2023-01-01 01:01:27                1
2         2  2023-01-01 00:25:04   2023-01-01 00:37:49                1
3         1  2023-01-01 00:03:48   2023-01-01 00:13:25                0
4         2  2023-01-01 00:10:29   2023-01-01 00:21:19                1

   trip_distance  RatecodeID  store_and_fwd_flag  PULocationID  DOLocationID  \
0           0.97         1.0               False           161           141
1           1.10         1.0               False            43           237
2           2.51         1.0               False            48           238
3           1.90         1.0               False           138             7
4           1.43         1.0               False           107            79

   payment_type  fare_amount  extra  mta_tax  tip_amount  tolls_amount  \
0             2          9.3   1.00      0.5        0.00           0.0
1             1          7.9   1.00      0.5        4.00           0.0
2             1         14.9   1.00      0.5       15.00           0.0
3             1         12.1   7.25      0.5        0.00           0.0
4             1         11.4   1.00      0.5        3.28           0.0

   improvement_surcharge  total_amount  congestion_surcharge  airport_fee
0                    1.0         14.30                   2.5         0.00
1                    1.0         16.90                   2.5         0.00
2                    1.0         34.90                   2.5         0.00
3                    1.0         20.85                   0.0         1.25
4                    1.0         19.68                   2.5         0.00
```

4. Define the data type schema when using to_sql method.

- Kelas `Load` digunakan untuk menghubungkan ke database PostgreSQL dan memuat DataFrame ke dalam tabel PostgreSQL.
- Metode `__create_connection` membuat koneksi ke database PostgreSQL.
- Metode `to_postgres` menerima nama database dan DataFrame, lalu mendefinisikan skema tipe data untuk metode `to_sql`.

```
class Load:
    def __init__(self) -> None:
        self.engine = None

    def __create_connection(self):
        user = "postgres"
        password = "admin"
        host = "localhost"
        database = "mydb"
        port = 5437
        conn_string = f"postgresql://{user}:{password}@{host}:{port}/{database}"
        self.engine = create_engine(conn_string)

# 4. Tentukan skema tipe data saat menggunakan metode to_sql.
    def to_postgres(self, db_name: str, data: pd.DataFrame) -> None:
        self.__create_connection()
        df_schema = {
            "VendorID": BigInteger,
            "tpep_pickup_datetime": DateTime,
            "tpep_dropoff_datetime": DateTime,
            "passenger_count": BigInteger,
            "trip_distance": Float,
            "RatecodeID": Float,
            "store_and_fwd_flag": Boolean,
            "PULocationID": Integer,
```

```
            "DOLocationID": Integer,
            "payment_type": Integer,
            "fare_amount": Float,
            "extra": Float,
            "mta_tax": Float,
            "tip_amount": Float,
            "tolls_amount": Float,
            "improvement_surcharge": Float,
            "total_amount": Float,
            "congestion_surcharge": Float,
            "airport_fee": Float
        }
        print("\n4. Data type schema defined for to_sql method.")
        print(data.info())
```

```
4. Data type schema defined for to_sql method.
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 19 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   VendorID               100000 non-null  int64
 1   tpep_pickup_datetime   100000 non-null  datetime64[ns]
 2   tpep_dropoff_datetime  100000 non-null  datetime64[ns]
 3   passenger_count        100000 non-null  Int8
 4   trip_distance          100000 non-null  float64
 5   RatecodeID             100000 non-null  float64
 6   store_and_fwd_flag     100000 non-null  boolean
 7   PULocationID           100000 non-null  int64
 8   DOLocationID           100000 non-null  int64
 9   payment_type           100000 non-null  int64
 10  fare_amount            100000 non-null  float64
 11  extra                  100000 non-null  float64
 12  mta_tax                100000 non-null  float64
 13  tip_amount             100000 non-null  float64
 14  tolls_amount           100000 non-null  float64
 15  improvement_surcharge  100000 non-null  float64
 16  total_amount           100000 non-null  float64
 17  congestion_surcharge   100000 non-null  float64
 18  airport_fee            100000 non-null  float64
dtypes: Int8(1), boolean(1), datetime64[ns](2), float64(11), int64(4)
memory usage: 13.4 MB
None
```

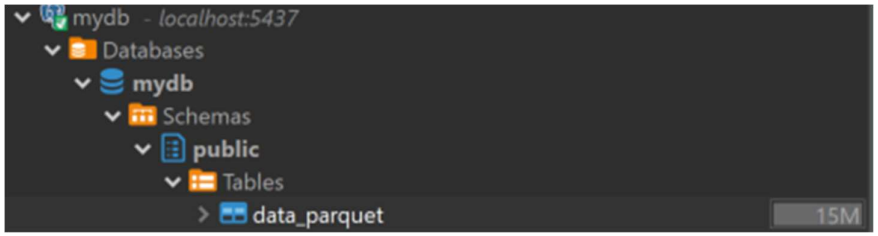
5. Ingest the Yellow Trip dataset to PostgreSQL

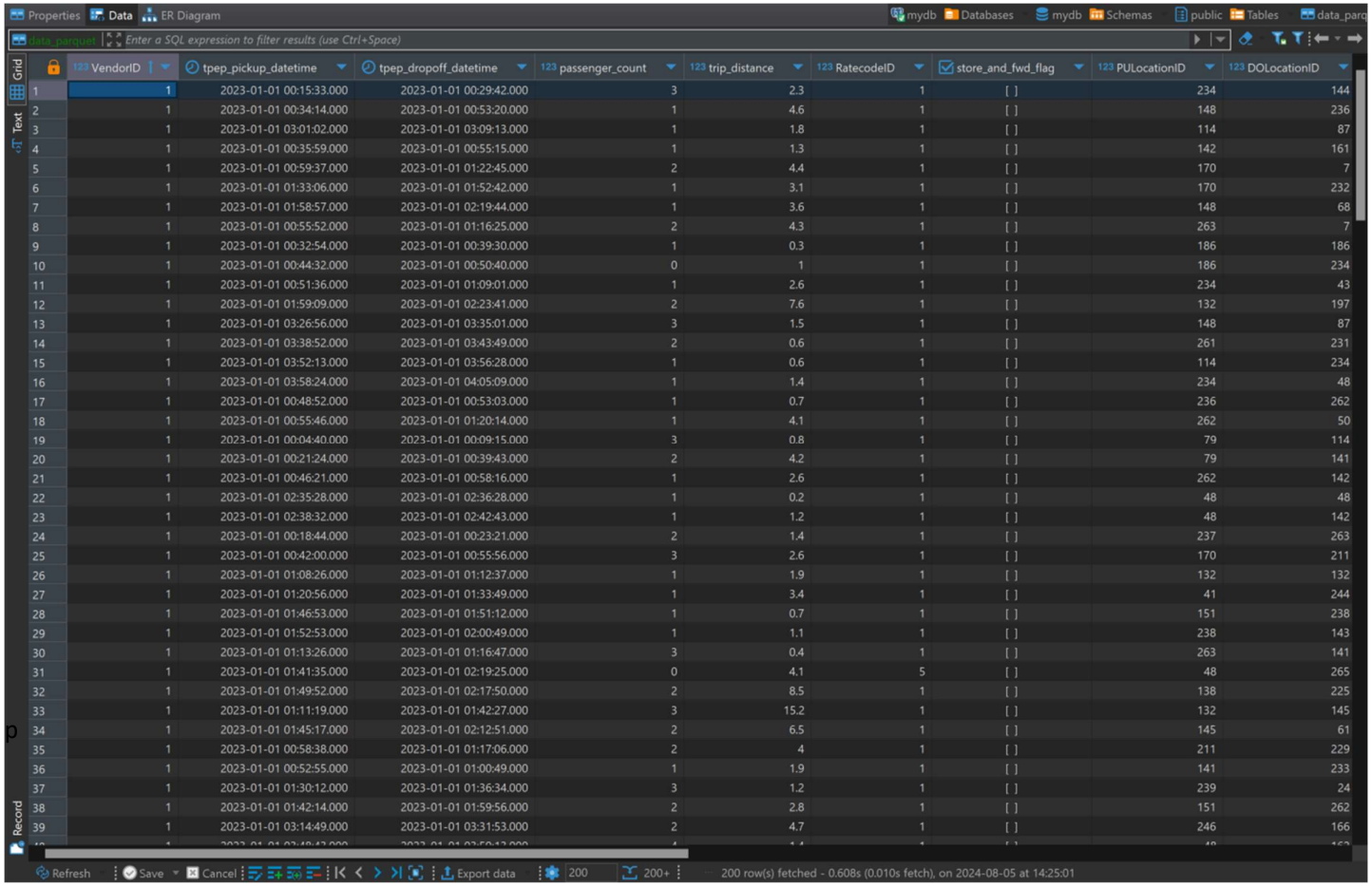
- DataFrame dimasukkan ke dalam tabel PostgreSQL menggunakan metode `to_sql` dengan skema tipe data yang telah didefinisikan.
- Jika terjadi kesalahan, pesan error akan dicetak.

```
        try:
# 5. Masukkan dataset Yellow Trip ke PostgreSQL.
            data.to_sql(name=db_name, con=self.engine, if_exists="replace", index=False, schema="public",
dtype=df_schema, method=None, chunksize=5000)
            print("\n5. Dataset ingested to PostgreSQL, check the result on PostgreSQL")
        except SQLAlchemyError as err:
            print(f"error >> {err}")
```

```
5. Dataset ingested to PostgreSQL, check the result on PostgreSQL
```

Data berhasil di-ingest ke Postgres





	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID
1	1	2023-01-01 00:15:33.000	2023-01-01 00:29:42.000	3	2.3	1	[ ]	234	144
2	1	2023-01-01 00:34:14.000	2023-01-01 00:53:20.000	1	4.6	1	[ ]	148	236
3	1	2023-01-01 03:01:02.000	2023-01-01 03:09:13.000	1	1.8	1	[ ]	114	87
4	1	2023-01-01 00:35:59.000	2023-01-01 00:55:15.000	1	1.3	1	[ ]	142	161
5	1	2023-01-01 00:59:37.000	2023-01-01 01:22:45.000	2	4.4	1	[ ]	170	7
6	1	2023-01-01 01:33:06.000	2023-01-01 01:52:42.000	1	3.1	1	[ ]	170	232
7	1	2023-01-01 01:58:57.000	2023-01-01 02:19:44.000	1	3.6	1	[ ]	148	68
8	1	2023-01-01 00:55:52.000	2023-01-01 01:16:25.000	2	4.3	1	[ ]	263	7
9	1	2023-01-01 00:32:54.000	2023-01-01 00:39:30.000	1	0.3	1	[ ]	186	186
10	1	2023-01-01 00:44:32.000	2023-01-01 00:50:40.000	0	1	1	[ ]	186	234
11	1	2023-01-01 00:51:36.000	2023-01-01 01:09:01.000	1	2.6	1	[ ]	234	43
12	1	2023-01-01 01:59:09.000	2023-01-01 02:23:41.000	2	7.6	1	[ ]	132	197
13	1	2023-01-01 03:26:56.000	2023-01-01 03:35:01.000	3	1.5	1	[ ]	148	87
14	1	2023-01-01 03:38:52.000	2023-01-01 03:43:49.000	2	0.6	1	[ ]	261	231
15	1	2023-01-01 03:52:13.000	2023-01-01 03:56:28.000	1	0.6	1	[ ]	114	234
16	1	2023-01-01 03:58:24.000	2023-01-01 04:05:09.000	1	1.4	1	[ ]	234	48
17	1	2023-01-01 00:48:52.000	2023-01-01 00:53:03.000	1	0.7	1	[ ]	236	262
18	1	2023-01-01 00:55:46.000	2023-01-01 01:20:14.000	1	4.1	1	[ ]	262	50
19	1	2023-01-01 00:04:40.000	2023-01-01 00:09:15.000	3	0.8	1	[ ]	79	114
20	1	2023-01-01 00:21:24.000	2023-01-01 00:39:43.000	2	4.2	1	[ ]	79	141
21	1	2023-01-01 00:46:21.000	2023-01-01 00:58:16.000	1	2.6	1	[ ]	262	142
22	1	2023-01-01 02:35:28.000	2023-01-01 02:36:28.000	1	0.2	1	[ ]	48	48
23	1	2023-01-01 02:38:32.000	2023-01-01 02:42:43.000	1	1.2	1	[ ]	48	142
24	1	2023-01-01 00:18:44.000	2023-01-01 00:23:21.000	2	1.4	1	[ ]	237	263
25	1	2023-01-01 00:42:00.000	2023-01-01 00:55:56.000	3	2.6	1	[ ]	170	211
26	1	2023-01-01 01:08:26.000	2023-01-01 01:12:37.000	1	1.9	1	[ ]	132	132
27	1	2023-01-01 01:20:56.000	2023-01-01 01:33:49.000	1	3.4	1	[ ]	41	244
28	1	2023-01-01 01:46:53.000	2023-01-01 01:51:12.000	1	0.7	1	[ ]	151	238
29	1	2023-01-01 01:52:53.000	2023-01-01 02:00:49.000	1	1.1	1	[ ]	238	143
30	1	2023-01-01 01:13:26.000	2023-01-01 01:16:47.000	3	0.4	1	[ ]	263	141
31	1	2023-01-01 01:41:35.000	2023-01-01 02:19:25.000	0	4.1	5	[ ]	48	265
32	1	2023-01-01 01:49:52.000	2023-01-01 02:17:50.000	2	8.5	1	[ ]	138	225
33	1	2023-01-01 01:11:19.000	2023-01-01 01:42:27.000	3	15.2	1	[ ]	132	145
34	1	2023-01-01 01:45:17.000	2023-01-01 02:12:51.000	2	6.5	1	[ ]	145	61
35	1	2023-01-01 00:58:38.000	2023-01-01 01:17:06.000	2	4	1	[ ]	211	229
36	1	2023-01-01 00:52:55.000	2023-01-01 01:00:49.000	1	1.9	1	[ ]	141	233
37	1	2023-01-01 01:30:12.000	2023-01-01 01:36:34.000	3	1.2	1	[ ]	239	24
38	1	2023-01-01 01:42:14.000	2023-01-01 01:59:56.000	2	2.8	1	[ ]	151	262
39	1	2023-01-01 03:14:49.000	2023-01-01 03:31:53.000	2	4.7	1	[ ]	246	166

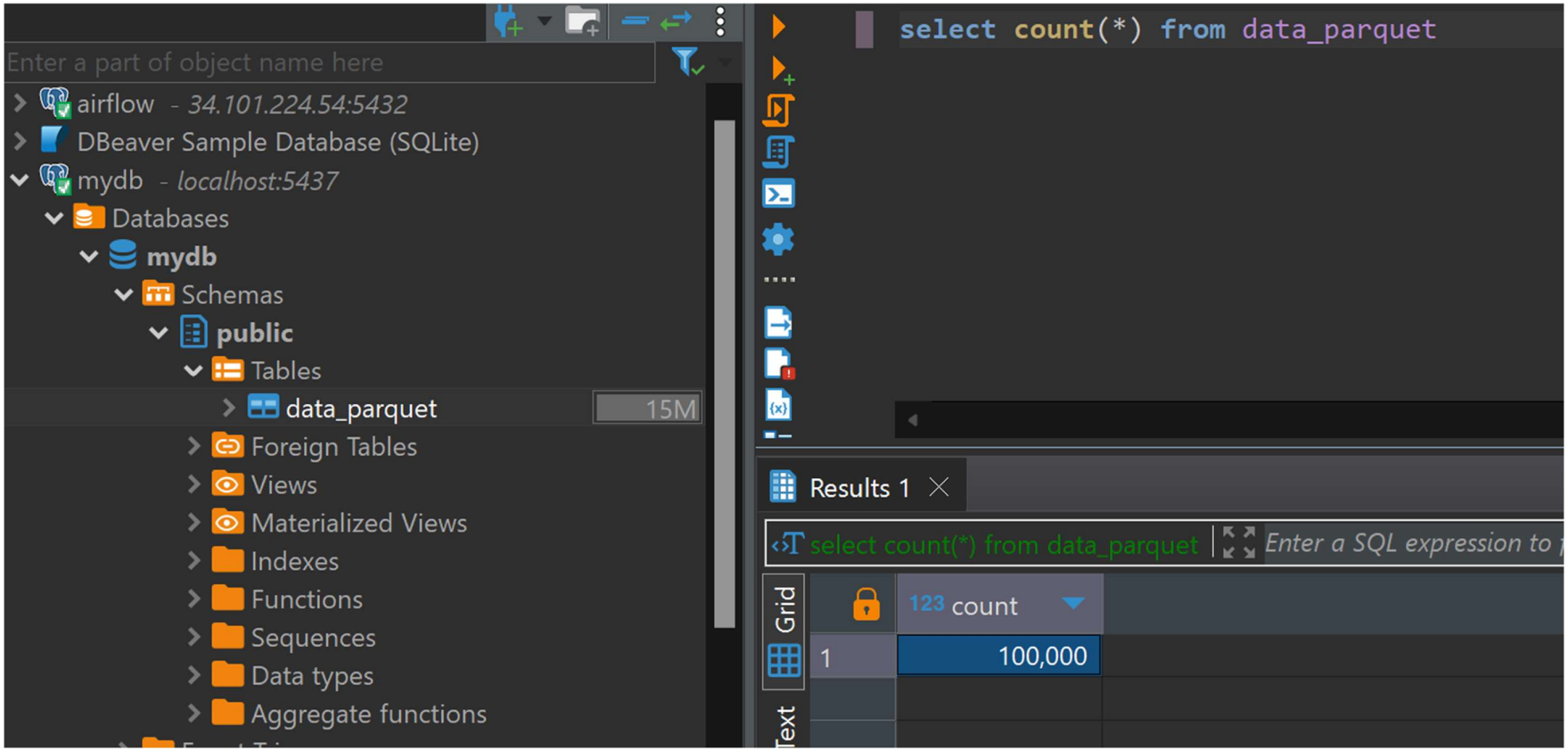
p

6. Count how many rows are ingested.
   - Fungsi `main` digunakan untuk menjalankan semua langkah-langkah di atas.
   - File parquet dimuat ke dalam DataFrame, lalu DataFrame tersebut dimasukkan ke dalam database PostgreSQL.
   - Jumlah baris yang dimasukkan ke database dihitung dan ditampilkan menggunakan pustaka `PrettyTable`.

```python
def main():
    extract = Extraction()
    file_path = "C:/Users/USER/Alta/belajar-bc/ingestion-demo/dataset/yellow_tripdata_2023-01.parquet"
    df_result = extract.local_file(file_path)

    load = Load()
    db_name = "data_parquet"
    load.to_postgres(db_name, df_result)

# 6. Hitung berapa baris yang dimasukkan.
    row_count = len(df_result)
    table = PrettyTable()
    table.field_names = ["Number of rows ingested"]
    table.add_row([row_count])
    print("\n6. Number of rows ingested:")
    print(table)

if __name__ == "__main__":
    main()
```

```
6. Number of rows ingested:
+------------------------+
| Number of rows ingested |
+------------------------+
|         100000         |
+------------------------+
```

Count on Postgres how many rows are ingested



Screeshots code:

test2.py U

ingestion-demo > ingestion_data > test2.py > main

```python
import pandas as pd
from fastparquet import ParquetFile
from sqlalchemy import create_engine, text
from sqlalchemy.types import BigInteger, DateTime, Boolean, Float, Integer
from sqlalchemy.exc import SQLAlchemyError
from prettytable import PrettyTable

# 1. Buat DataFrame dari file parquet di folder datasets kita.
class Extraction:
    def __init__(self) -> None:
        self.path = ""
        self.dataframe = pd.DataFrame()

    def local_file(self, path: str):
        self.path = path
        self.__read_parquetfile()
        print("1. DataFrame created from parquet file:")
        print(self.dataframe.head())  # Print the first few rows of the DataFrame
        self.investigate_schema()
        self.cast_data()
        return self.dataframe

    def __read_parquetfile(self):
        parquetfile = ParquetFile(self.path)
        self.dataframe = parquetfile.to_pandas()

# 2. Memuat file parquet ke dalam DataFrame menggunakan pustaka fastparquet
    def investigate_schema(self):
        pd.set_option('display.max_columns', None)
        print("\n2. Schema of the DataFrame:")
        print(self.dataframe.dtypes)  # Print the schema of the DataFrame

# 3. Bersihkan dataset Yellow Trip.
    def cast_data(self):
        self.dataframe["passenger_count"] = self.dataframe["passenger_count"].astype("Int8")
        self.dataframe["store_and_fwd_flag"] = self.dataframe["store_and_fwd_flag"].map({"N": False, "Y": True}).asty
        self.dataframe["tpep_pickup_datetime"] = pd.to_datetime(self.dataframe["tpep_pickup_datetime"])
        self.dataframe["tpep_dropoff_datetime"] = pd.to_datetime(self.dataframe["tpep_dropoff_datetime"])
        print("\n3. Cleaned DataFrame with correct data types:")
```

```python
class Extraction:
    def cast_data(self):
        print(self.dataframe.head())  # Print the cleaned DataFrame

class Load:
    def __init__(self) -> None:
        self.engine = None

    def __create_connection(self):
        user = "postgres"
        password = "admin"
        host = "localhost"
        database = "mydb"
        port = 5437
        conn_string = f"postgresql://{user}:{password}@{host}:{port}/{database}"
        self.engine = create_engine(conn_string)

# 4. Tentukan skema tipe data saat menggunakan metode to_sql.
    def to_postgres(self, db_name: str, data: pd.DataFrame) -> None:
        self.__create_connection()
        df_schema = {
            "VendorID": BigInteger,
            "tpep_pickup_datetime": DateTime,
            "tpep_dropoff_datetime": DateTime,
            "passenger_count": BigInteger,
            "trip_distance": Float,
            "RatecodeID": Float,
            "store_and_fwd_flag": Boolean,
            "PULocationID": Integer,
            "DOLocationID": Integer,
            "payment_type": Integer,
            "fare_amount": Float,
            "extra": Float,
            "mta_tax": Float,
            "tip_amount": Float,
            "tolls_amount": Float,
            "improvement_surcharge": Float,
            "total_amount": Float,
            "congestion_surcharge": Float,
```

```python
class Load:
    def to_postgres(self, db_name: str, data: pd.DataFrame) -> None:
                "tip_amount": Float,
                "tolls_amount": Float,
                "improvement_surcharge": Float,
                "total_amount": Float,
                "congestion_surcharge": Float,
                "airport_fee": Float
            }
            print("\n4. Data type schema defined for to_sql method.")
            print(data.info())
            try:
# 5. Masukkan dataset Yellow Trip ke PostgreSQL.
                data.to_sql(name=db_name, con=self.engine, if_exists="replace", index=False, schema="public", dtype=df_sch
                print("\n5. Dataset ingested to PostgreSQL, the result check on PostgreSQL")
            except SQLAlchemyError as err:
                print(f"error >> {err}")
                # Menghitung dan menampilkan jumlah baris yang di-ingest

def main():
    extract = Extraction()
    file_path = "C:/Users/USER/Alta/belajar-bc/ingestion-demo/dataset/yellow_tripdata_2023-01.parquet"
    df_result = extract.local_file(file_path)

    load = Load()
    db_name = "data_parquet"
    load.to_postgres(db_name, df_result)

# 6. Hitung berapa baris yang dimasukkan.
    row_count = len(df_result)
    table = PrettyTable()
    table.field_names = ["Number of rows ingested"]
    table.add_row([row_count])
    print("\n6. Number of rows ingested:")
    print(table)

if __name__ == "__main__":
    main()
```