

TP DATAMART

```
git clone https://github.com/Noobzik/ATL-Datamart.git
cd ATL-datamart
```

TP 1 : Compléter les fonctions vides dans `src/data/grab_parquet.py` pour récupérer les datasets de janvier 2023 à août 2023, ainsi que le dataset du dernier mois disponible. Les données doivent être téléchargées et stockées dans Minio.

```
from minio import Minio
import urllib.request
import pandas as pd
import os
import sys

def main():
    grab_data()

def grab_data() -> None:
    """
    Télécharge les données de New York Yellow Taxi pour les mois spécifiés.
    Les fichiers sont enregistrés dans le dossier '../data/raw'.
    Cette fonction ne prend pas d'arguments et ne retourne rien.
    """
    base_url = "https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2024-"
    save_path = "../data/raw"

    # Liste des mois pour lesquels télécharger les données (de janvier à août 2023)
    months = ["2024-01", "2024-02", "2024-03", "2024-04", "2024-05", "2024-06", "2024-07", "2024-08"]

    # Crée le dossier de sauvegarde s'il n'existe pas
    if not os.path.exists(save_path):
        os.makedirs(save_path)

    for month in months:
        file_url = f"{base_url}{month}.csv"
        file_name = f"{save_path}/yellow_tripdata_{month}.csv"

        # Téléchargement du fichier
        print(f"Téléchargement de {file_url}...")
        urllib.request.urlretrieve(file_url, file_name)
        print(f"Fichier téléchargé et enregistré sous {file_name}")

def write_data_minio():
    """
    Cette fonction transfère tous les fichiers Parquet dans Minio.
    Ne pas implémenter cette méthode pour le moment.
    """
    client = Minio(
        "localhost:9000",
        secure=False,
        access_key="minio",
        secret_key="minio123"
    )
    bucket: str = "NOM_DU_BUCKET_ICI"
    found = client.bucket_exists(bucket)
    if not found:
        client.make_bucket(bucket)
    else:
        print("Bucket " + bucket + " existe déjà")

if __name__ == '__main__':
    sys.exit(main())
```

```
from minio import Minio
import urllib.request
```

```

import pandas as pd
import os
import sys
import requests

def main():
    grab_data()
    write_data_minio()

def grab_data() -> None:
    """Grab the data from New York Yellow Taxi"""
    # Dossier de destination
    output_dir = os.path.join(os.path.expanduser('~'), 'Desktop', 'tp new york')
    os.makedirs(output_dir, exist_ok=True)

    # Base URL pour le téléchargement des fichiers
    base_url = "https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2024-"

    # Liste des mois à télécharger (jusqu'à août 2024)
    months = ['01', '02', '03', '04', '05', '06', '07', '08']

    for month in months:
        file_url = f'{base_url}{month}.parquet'
        file_path = os.path.join(output_dir, f'yellow_tripdata_2024-{month}.parquet')

        try:
            print(f"Téléchargement de {file_url}...")
            response = requests.get(file_url)
            response.raise_for_status() # Vérifie si la requête a réussi
            with open(file_path, 'wb') as file:
                file.write(response.content)
            print(f"Téléchargement réussi : {file_path}")
        except requests.exceptions.HTTPError as e:
            print(f"Erreur lors du téléchargement de {file_url}: {e}")
        except Exception as e:
            print(f"Erreur inattendue lors du téléchargement de {file_url}: {e}")

def write_data_minio():
    """Put all Parquet files into Minio"""
    client = Minio(
        "localhost:9000",
        secure=False,
        access_key="minio",
        secret_key="minio123"
    )

    bucket: str = "yellow-taxi-data" # Nom valide pour le bucket
    found = client.bucket_exists(bucket)
    if not found:
        client.make_bucket(bucket)
    else:
        print("Bucket " + bucket + " existe déjà")

    # Dossier de destination
    output_dir = os.path.join(os.path.expanduser('~'), 'Desktop', 'tp new york')

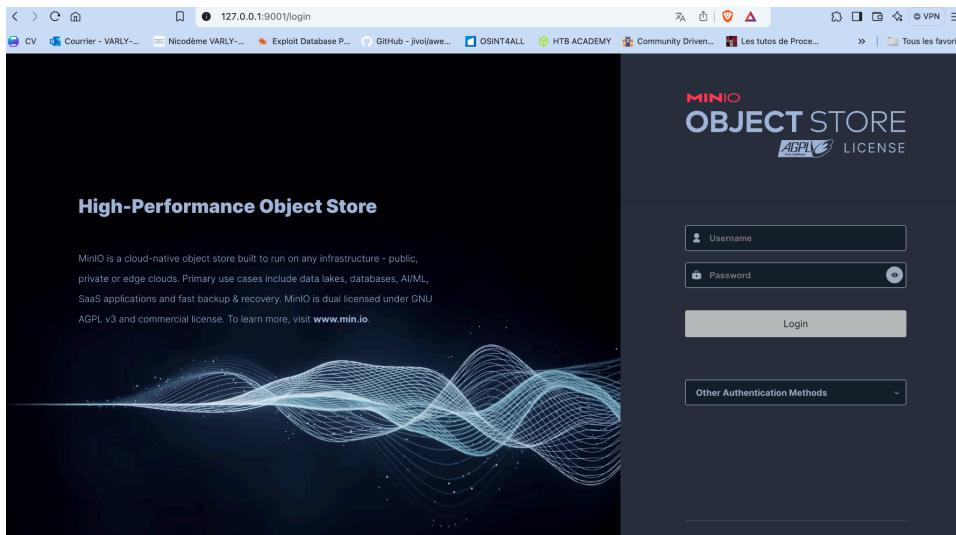
    # Téléchargez tous les fichiers Parquet dans le bucket
    for month in range(1, 9): # De 1 à 8
        file_name = f"yellow_tripdata_2024-{month:02d}.parquet"
        file_path = os.path.join(output_dir, file_name)
        if os.path.exists(file_path):
            print(f"Téléchargement de {file_name} vers Minio...")
            try:
                client.fput_object(bucket, file_name, file_path)
                print(f"{file_name} téléchargé avec succès dans Minio.")
            except Exception as e:
                print(f"Erreur lors du téléchargement de {file_name} dans Minio: {e}")
        else:
            print(f"Le fichier {file_name} n'existe pas et ne peut pas être téléchargé.")

if __name__ == '__main__':
    sys.exit(main())

```

Script python pour collecter les datasets et les stocker sur minio.

Lancement de minio



connexion à minio

Exécution du script python avec réussite car on peut apercevoir les données yellow taxi.

Name	Last Modified	Size
yellow_tripdata_2024-01.parquet	Today, 13:04	47.6 MiB
yellow_tripdata_2024-02.parquet	Today, 13:04	48.0 MiB
yellow_tripdata_2024-03.parquet	Today, 13:04	57.3 MiB
yellow_tripdata_2024-04.parquet	Today, 13:04	58.4 MiB
yellow_tripdata_2024-05.parquet	Today, 13:04	59.7 MiB
yellow_tripdata_2024-06.parquet	Today, 13:04	57.1 MiB
yellow_tripdata_2024-07.parquet	Today, 13:04	49.9 MiB
yellow_tripdata_2024-08.parquet	Today, 13:04	48.7 MiB

TP 2

Modifier `src/data/dump_to_sql.py` pour récupérer les fichiers Parquet stockés dans Minio et les charger dans une base de données relationnelle, telle que PostgreSQL.

connexion a postgresql via docker

```
nicodeme@mbp-de-nicodeme ATL-Datamart % docker exec -it atl-datamart-db-1 psql -U postgres -d postgres
psql (17.0 (Debian 17.0-1.pgdg120+1))
Type "help" for help.

postgres=# 
```

Installation des bibliothèques nécessaire :

```
nicodeme@mbp-de-nicodeme Desktop % pip install pandas pyarrow sqlalchemy psycopg2-binary boto3
```

Création du script via la base `dump_to_sql.py`

```
import gc
import os
import sys
import pandas as pd
from sqlalchemy import create_engine
from minio import Minio

# Configuration Minio
MINIO_ENDPOINT = "http://localhost:9000"
MINIO_ACCESS_KEY = "minio"
MINIO_SECRET_KEY = "minio123"
MINIO_BUCKET_NAME = "yellow-taxi-data"

# Configuration PostgreSQL
DB_HOST = "localhost"
DB_PORT = "15432"
DB_USER = "postgres"
DB_PASSWORD = "admin"
DB_NAME = "nyc_warehouse"
DB_TABLE = "nyc_raw"
```

```

def download_parquet_from_minio(parquet_file: str, download_path: str) -> None:
    """Télécharge un fichier Parquet depuis Minio."""
    client = Minio(
        "localhost:9000",
        secure=False,
        access_key=MINIO_ACCESS_KEY,
        secret_key=MINIO_SECRET_KEY
    )

    try:
        print(f"Téléchargement de {parquet_file} depuis Minio...")
        client.fget_object(MINIO_BUCKET_NAME, parquet_file, download_path)
        print(f"Téléchargement terminé : {parquet_file}")
    except Exception as e:
        print(f"Erreur lors du téléchargement de {parquet_file} : {e}")

def write_data_postgres(dataframe: pd.DataFrame) -> bool:
    """Charge le DataFrame dans PostgreSQL."""
    db_url = f"postgresql://{{DB_USER}}:{{DB_PASSWORD}}@{{DB_HOST}}:{{DB_PORT}}/{{DB_NAME}}"
    try:
        engine = create_engine(db_url)
        with engine.connect():
            print("Connexion à PostgreSQL réussie.")
            dataframe.to_sql(DB_TABLE, engine, index=False, if_exists='append')
            print("Données chargées dans PostgreSQL.")
        return True
    except Exception as e:
        print(f"Erreur lors de l'insertion dans PostgreSQL : {e}")
    return False

def clean_column_name(dataframe: pd.DataFrame) -> pd.DataFrame:
    """Nettoie les noms de colonnes en les mettant en minuscules."""
    dataframe.columns = map(str.lower, dataframe.columns)
    return dataframe

def main() -> None:
    # Dossier local où les fichiers seront téléchargés
    local_dir = os.path.join(os.path.expanduser('~'), 'Desktop', 'tp new york')

    # Liste des fichiers Parquet à traiter
    parquet_files = [f"yellow_tripdata_2024-{month:02d}.parquet" for month in range(1, 9)]

    for parquet_file in parquet_files:
        local_path = os.path.join(local_dir, parquet_file)
        download_parquet_from_minio(parquet_file, local_path)

        if os.path.isfile(local_path):
            print(f"Lecture du fichier Parquet : {local_path}")
            parquet_df = pd.read_parquet(local_path, engine='pyarrow')
            parquet_df = clean_column_name(parquet_df)

            if not write_data_postgres(parquet_df):
                del parquet_df
                gc.collect()
                return

            del parquet_df
            gc.collect()

    if __name__ == '__main__':
        sys.exit(main())

```

Tout cela est réalisé en local dans un fichier qui se nomme dump_to_sql_local.py.

Préparation de PostgreSQL :

Création de la database nyc_warehouse

```
postgres=# CREATE DATABASE nyc_warehouse;
CREATE DATABASE
postgres=# \c
postgres-# \c
You are now connected to database "postgres" as user "postgres".
postgres-# \c nyc_warehouse
You are now connected to database "nyc_warehouse" as user "postgres".
```

Création de la table nyc_raw

```
nyc_warehouse=# CREATE TABLE nyc_raw (
    vendor_id INTEGER,
    pickup_datetime TIMESTAMP,
    dropoff_datetime TIMESTAMP,
    passenger_count INTEGER,
    trip_distance FLOAT,
    rate_code INTEGER,
    store_and_fwd_flag TEXT,
    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
);
CREATE TABLE
nyc_warehouse=# \dt
      List of relations
 Schema |   Name   | Type  | Owner
-----+----------+-----+-----
 public | nyc_raw | table | postgres
(1 row)
```

Problème concernant les colonnes, j'ai dû les changer en voyant les erreurs que le script me rentrait car les noms ne correspondaient pas avec les colonnes des fichiers parquets.

Exemple de ce que j'avais comme erreur

```
[nicodeme@mp-de-nicodeme Desktop % python3 dump_to_sql_local.py
Téléchargement de yellow_tripdata_2024-01.parquet depuis Minio...
Téléchargement de yellow_tripdata_2024-01.parquet depuis Minio...
Lecture du fichier Parquet : /Users/nicodeme/Desktop/tp new/yellow_tripdata_2024-01.parquet
Connexion à PostgreSQL réussie.
Erreur lors de l'insertion dans PostgreSQL : (psycopg2.errors.UndefinedColumn) column "pulocationid" of relation "nyc_raw" does not exist
LINE 1: ...t, trip_distance, ratecodeid, store_and_fwd_flag, pulocation...
                                             ^
[SQL: INSERT INTO nyc_raw (vendorid, trip_pickup_datetime, trip_dropoff_datetime, passenger_count, trip_distance, ratecodeid, store_and_fwd_flag, pulocationid, payment_type, fare_amount, extra, mta_tax, tip_amount, tolls_amount, improvement_surcharge, _9999, passenger_count, _9999, X(airport_fee'_9999)) VALUES (%parameters: ('fare_amount',_9999: 17.7, 'extra',_9999: 1.0, 'congestion_surcharge',_9999: 2.5, 'ratecodeid',_9999: 1.0, 'payment_type',_9999: 2, 'tpep_pickup_datetime',_9999: datetime.datetime(2024, 1, 1, 0, 57, 55), 'total_amount',_9999: 22.7, 'vendorid',_9999: 186, 'tip_amount',_9999: 0.0, 'pulocationid',_9999: 79, 'passenger_count',_9999: 1.0, 'airport_fee',_9999: 0.0, 'store_and_fwd_flag',_9999: 'N', 'tolls_amount',_9999: 0.0, 'tpep_dropoff_datetime',_9999: datetime.datetime(2024, 1, 1, 1, 17, 43), 'trip_distance',_9999: 1.72, 'mta_tax',_9999: 0.0, 'fare_amount',_9999: 18.0, 'improvement_surcharge',_9999: 2.5, 'ratecodeid',_9999: 1.0, 'payment_type',_9999: 1, 'tpep_pickup_datetime',_9999: datetime.datetime(2024, 1, 1, 0, 3), 'total_amount',_9999: 18.75, 'VendorID',_9999: 1, 'extra',_9999: 3.5, 'congestion_surcharge',_9999: 1.0, 'ratecodeid',_9999: 1.0, 'payment_type',_9999: 1, 'tpep_dropoff_datetime',_9999: datetime.datetime(2024, 1, 1, 0, 36), 'trip_distance',_9999: 1.8, 'mta_tax',_9999: 0.5, 'fare_amount',_9999: 23.3, 'improvement_surcharge',_9999: 1.0, 'extra',_9999: 3.5, 'congestion_surcharge',_9999: 2.0, 'ratecodeid',_9999: 2, 'payment_type',_9999: 1, 'tpep_pickup_datetime',_9999: datetime.datetime(2024, 1, 1, 0, 17, 43), 'total_amount',_9999: 31.3, 'VendorID',_9999: 1, 'pulocationid',_9999: 236, 'tip_amount',_9999: 3.0, 'store_and_fwd_flag',_9999: 'N', 'tolls_amount',_9999: 0.0, 'tpep_dropoff_datetime',_9999: datetime.datetime(2024, 1, 1, 0, 22, 59), 'trip_distance',_9999: 1.97, 'mta_tax',_9999: 0.5, 'fare_amount',_9999: 59.0, 'improvement_surcharge',_9999: 1.0, 'extra',_9999: 3.5, 'congestion_surcharge',_9999: 2.5, 'ratecodeid',_9999: 1.0, 'payment_type',_9999: 1, 'tpep_pickup_datetime',_9999: datetime.datetime(2024, 1, 1, 0, 28, 44), 'total_amount',_9999: 79.94, 'VendorID',_9999: 1, 'pulocationid',_9999: 140, 'tip_amount',_9999: 0.0, 'dolocationid',_9999: 16, 'passenger_count',_9999: 2.0, 'airport_fee',_9999: 0.0, 'store_and_fwd_flag',_9999: 'N', 'tolls_amount',_9999: 6.94, 'tpep_dropoff_datetime',_9999: datetime.datetime(2024, 1, 1, 0, 55, 6), 'trip_distance',_9999: 15.8, 'mta_tax',_9999: 0.5, 'fare_amount',_9999: -9.3, 'improvement_surcharge',_9999: -1.0, 'extra',_9999: -1.0, 'congestion_surcharge',_9999: -2.5, 'ratecodeid',_9999: 1.0, 'payment_type',_9999: 4, 'tpep_pickup_datetime',_9999: datetime.datetime(2024, 1, 1, 0, 21, 53), 'trip_distance',_9999: 1.17, 'mta_tax',_9999: -0.5)
(Background on this error at: https://sqlalche.me/e/28/f466)
nicodeme@mp-de-nicodeme Desktop % ]
```

Remédier à mes erreurs

```
nyc_warehouse=# ALTER TABLE nyc_raw RENAME COLUMN vendor_id TO vendorid;
ALTER TABLE nyc_raw RENAME COLUMN pickup_datetime TO tpep_pickup_datetime;
ALTER TABLE nyc_raw RENAME COLUMN dropoff_datetime TO tpep_dropoff_datetime;
ALTER TABLE nyc_raw RENAME COLUMN rate_code TO ratecodeid;
ALTER TABLE nyc_raw RENAME COLUMN store_and_fwd_flag TO store_and_fwd_flag;
ALTER TABLE nyc_raw RENAME COLUMN trip_distance TO trip_distance;
ALTER TABLE nyc_raw RENAME COLUMN payment_type TO payment_type;
ALTER TABLE nyc_raw RENAME COLUMN fare_amount TO fare_amount;
ALTER TABLE nyc_raw RENAME COLUMN extra TO extra;
ALTER TABLE nyc_raw RENAME COLUMN mta_tax TO mta_tax;
ALTER TABLE nyc_raw RENAME COLUMN tip_amount TO tip_amount;
ALTER TABLE nyc_raw RENAME COLUMN tolls_amount TO tolls_amount;
ALTER TABLE nyc_raw RENAME COLUMN improvement_surcharge TO improvement_surcharge;
ALTER TABLE nyc_raw RENAME COLUMN total_amount TO total_amount;
ALTER TABLE nyc_raw RENAME COLUMN congestion_surcharge TO congestion_surcharge;
ALTER TABLE
ALTER TABLE
ALTER TABLE
ERROR: column "store_and_fwd_flag" of relation "nyc_raw" already exists
ERROR: column "trip_distance" of relation "nyc_raw" already exists
ERROR: column "payment_type" of relation "nyc_raw" already exists
ERROR: column "fare_amount" of relation "nyc_raw" already exists
ERROR: column "extra" of relation "nyc_raw" already exists
ERROR: column "mta_tax" of relation "nyc_raw" already exists
ERROR: column "tip_amount" of relation "nyc_raw" already exists
ERROR: column "tolls_amount" of relation "nyc_raw" already exists
ERROR: column "improvement_surcharge" of relation "nyc_raw" already exists
ERROR: column "total_amount" of relation "nyc_raw" already exists
ERROR: column "congestion_surcharge" of relation "nyc_raw" already exists
nyc_warehouse=# ALTER TABLE nyc_raw ADD COLUMN pulocationid INTEGER;
ALTER TABLE nyc_raw ADD COLUMN dolocationid INTEGER;
ALTER TABLE
ALTER TABLE
nyc_warehouse=# ALTER TABLE nyc_raw ADD COLUMN airport_fee FLOAT;
ALTER TABLE
```

```
|nyc_warehouse# \d nyc_raw
Table "public.nyc_raw"
 Column |          Type          | Collation | Nullable | Default
-----+---------------------+-----+-----+-----+
vendorid | integer | | |
tpep_pickup_datetime | timestamp without time zone | | |
tpep_dropoff_datetime | timestamp without time zone | | |
passenger_count | integer | | |
trip_distance | double precision | | |
ratecodeid | integer | | |
store_and_fwd_flag | text | | |
payment_type | integer | | |
fare_amount | double precision | | |
extra | double precision | | |
mta_tax | double precision | | |
tip_amount | double precision | | |
tolls_amount | double precision | | |
improvement_surcharge | double precision | | |
total_amount | double precision | | |
congestion_surcharge | double precision | | |
pulocationid | integer | | |
dolocationid | integer | | |
airport_fee | double precision | | |
```

Résultats :

Le script me dit que les données ont été chargés dans PostgreSQL

```

[nicode@mbp-de-nicode Desktop % python3 dump_to_sql_local.py
Téléchargement de yellow_tripdata_2024-01.parquet depuis Minio...
Téléchargement terminé : yellow_tripdata_2024-01.parquet
Lecture du fichier Parquet : /Users/nicode/Desktop/tp new york/yellow_tripdata_2024-01.parquet
Connexion à PostgreSQL réussie.
Données chargées dans PostgreSQL.
Téléchargement de yellow_tripdata_2024-02.parquet depuis Minio...
Téléchargement terminé : yellow_tripdata_2024-02.parquet
Lecture du fichier Parquet : /Users/nicode/Desktop/tp new york/yellow_tripdata_2024-02.parquet
Connexion à PostgreSQL réussie.
Données chargées dans PostgreSQL.
Téléchargement de yellow_tripdata_2024-03.parquet depuis Minio...
Téléchargement terminé : yellow_tripdata_2024-03.parquet
Lecture du fichier Parquet : /Users/nicode/Desktop/tp new york/yellow_tripdata_2024-03.parquet
Connexion à PostgreSQL réussie.
Données chargées dans PostgreSQL.
Téléchargement de yellow_tripdata_2024-04.parquet depuis Minio...
Téléchargement terminé : yellow_tripdata_2024-04.parquet
Lecture du fichier Parquet : /Users/nicode/Desktop/tp new york/yellow_tripdata_2024-04.parquet
Connexion à PostgreSQL réussie.
Données chargées dans PostgreSQL.
Téléchargement de yellow_tripdata_2024-05.parquet depuis Minio...
Téléchargement terminé : yellow_tripdata_2024-05.parquet
Lecture du fichier Parquet : /Users/nicode/Desktop/tp new york/yellow_tripdata_2024-05.parquet
Connexion à PostgreSQL réussie.
Données chargées dans PostgreSQL.
Téléchargement de yellow_tripdata_2024-06.parquet depuis Minio...
Téléchargement terminé : yellow_tripdata_2024-06.parquet
Lecture du fichier Parquet : /Users/nicode/Desktop/tp new york/yellow_tripdata_2024-06.parquet
Connexion à PostgreSQL réussie.
Données chargées dans PostgreSQL.
Téléchargement de yellow_tripdata_2024-07.parquet depuis Minio...
Téléchargement terminé : yellow_tripdata_2024-07.parquet
Lecture du fichier Parquet : /Users/nicode/Desktop/tp new york/yellow_tripdata_2024-07.parquet
Connexion à PostgreSQL réussie.
Données chargées dans PostgreSQL.
Téléchargement de yellow_tripdata_2024-08.parquet depuis Minio...
Téléchargement terminé : yellow_tripdata_2024-08.parquet
Lecture du fichier Parquet : /Users/nicode/Desktop/tp new york/yellow_tripdata_2024-08.parquet
Connexion à PostgreSQL réussie.
Données chargées dans PostgreSQL.
nicode@mbp-de-nicode Desktop %

```

Vérifications des données sur PostgreSQL :

`SELECT * FROM nyc_raw LIMIT 10;` cette commande permet de sélectionner toutes les colonnes de la table `nyc_raw` et restreint le nombre de lignes retournées à 10.

vendorid	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	ratecodeid	store_and_fwd_flag	payment_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount	improvement_surcharge	total_amount	congestion_surcharge	pulocationid	dolocationid	airport_fee
2	2024-01-01 00:57:55	2024-01-01 01:17:43	1	1.72	1	N		2	17.7	1	0.5	0	0	0	0			
1	2024-01-01 00:03:00	2024-01-01 00:09:36	2.5	186	79	0												
1	2024-01-01 00:17:06	2024-01-01 00:35:01	2.5	140	1.8	1	N		1	10	3.5	0.5	3.75	0	0			
1	2024-01-01 00:36:38	2024-01-01 00:44:56	31.3	236	236	0												
1	2024-01-01 00:46:51	2024-01-01 00:52:57	17	236	4.7	1	N		1	23.3	3.5	0.5	3	0	0			
1	2024-01-01 00:54:08	2024-01-01 01:26:31	16.1	236	7	1	N		1	10	3.5	0.5	2	0	0			
1	2024-01-01 00:54:08	2024-01-01 01:26:31	2.5	211	1.4	1	N		1	0.8	1	0.5	3.2	0	0			
1	2024-01-01 00:54:08	2024-01-01 01:26:31	41.5	211	148	0			1	7.9	3.5	0.5	29.6	0	0			
2	2024-01-01 00:49:44	2024-01-01 01:15:47	64.95	148	141	1	N		1	45.7	6	0.5	10	0	0			
1	2024-01-01 00:30:48	2024-01-01 00:58:40	38.4	138	10.82	1	N		1	25.4	3.5	0.5	0	0	0			
2	2024-01-01 00:26:01	2024-01-01 00:54:12	36	246	181	1	N		2	31	1	0.5	0	0	0			
2	2024-01-01 00:28:08	2024-01-01 00:29:16	2.5	161	231	1	N		2	0.64	1	0.5	0	0	0			
			8	113	261	0			2	1	0.5	0						
(10 rows)																		
{END}																		

Cette commande permet de voir le nombre total de lignes insérées

```

nyc_warehouse=# SELECT COUNT(*) FROM nyc_raw;
      count
-----
26388179
(1 row)

```

**TP 3 : Créer un modèle en flocon pour structurer les données dans un Data Mart.
Déployer le modèle sur un serveur distinct par rapport au Data warehouse.**

Création d'un deuxième serveur postgres pour la création du datamart.

Modification du fichier docker-compose.yml

```
db-datamart:
  image: postgres:latest
  restart: always
  ports:
    - "15435:5432" # Port exposé pour accéder au Data Mart
  environment:
    POSTGRES_USERNAME: admin
    POSTGRES_PASSWORD: admin
  networks:
    - spark_network
  volumes:
    - pgdata-datamart:/var/lib/postgresql/data
```

```
volumes:
  pgdata:
  postgres-db-volume:
    pgdata-datamart:
```

Commande “docker-compose up -d” pour voir si le service se lance correctement :

```
[+] Running 12/12
✓ Network spark_network          Created
✓ Volume "atl-datamart_pgdata-datamart"      Created
✓ Container minio                  Started
✓ Container atl-datamart-db-1       Started
✓ Container atl-datamart-db-datamart-1     Started
✓ Container atl-datamart-redis-1      Healthy
✓ Container atl-datamart-postgres-airflow-1 Healthy
✓ Container atl-datamart-airflow-init-1   Exited
✓ Container atl-datamart-airflow-webserver-1 Started
✓ Container atl-datamart-airflow-worker-1   Started
✓ Container atl-datamart-airflow-scheduler-1 Started
✓ Container atl-datamart-airflow-triggerer-1 Started
```

Commande “Docker ps” pour voir si db-datamart est bien actif

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
a96cb673d3af	apache/airflow:latest-python3.11	"/usr/bin/dumb-init ..."	4 minutes ago	Up 4 minutes (healthy)	8080/tcp	atl-datamart-airflow-triggerer-1
0541534ce5dc	apache/airflow:latest-python3.11	"/usr/bin/dumb-init ..."	4 minutes ago	Up 4 minutes (healthy)	0.0.0.0:8080->8080/tcp	atl-datamart-airflow-webserver-1
1082d7753988	apache/airflow:latest-python3.11	"/usr/bin/dumb-init ..."	4 minutes ago	Up 4 minutes (healthy)	8080/tcp	atl-datamart-airflow-scheduler-1
e4a2f84be9c6	apache/airflow:latest-python3.11	"/usr/bin/dumb-init ..."	4 minutes ago	Up 4 minutes (healthy)	8080/tcp	atl-datamart-airflow-worker-1
2deba4295089	minio/minio	"/usr/bin/docker-entrypoint.s..."	4 minutes ago	Up 4 minutes	0.0.0.0:9000-9001->9000-9001/tcp	minio
e6f0448db46b	redis:latest	"docker-entrypoint.s..."	4 minutes ago	Up 4 minutes (healthy)	6379/tcp	atl-datamart-redis-1
77ebaedb989	postgres:13	"docker-entrypoint.s..."	4 minutes ago	Up 4 minutes (healthy)	0.0.0.0:15433->5432/tcp	atl-datamart-postgres-airflow-1
3eeb94214763	postgres:latest	"docker-entrypoint.s..."	4 minutes ago	Up 4 minutes	0.0.0.0:15432->5432/tcp	atl-datamart-db-1
574b8e1bed58	postgres:latest	"docker-entrypoint.s..."	4 minutes ago	Up 4 minutes	0.0.0.0:15435->5432/tcp	atl-datamart-db-datamart-1

Connexion à db-datamart et création de la database datamart :

```
[nicodeme@mbp-de-nicodeme ATL-DataMart % docker exec -it atl-datamart-db-datamart-1 psql -U postgres
psql (17.0 (Debian 17.0-1.pgdg120+1))
Type "help" for help.

postgres=# CREATE DATABASE datamart;
CREATE DATABASE
```

- Création d'un script sql : creation.sql pour la création des tables en flocons avec les contraintes associés

```
--Dimension : Vendor
CREATE TABLE dim_vendor (
    vendor_id INT PRIMARY KEY,
    vendor_name TEXT
);

-- Dimension : Rate Code
CREATE TABLE dim_rate_code (
    rate_code_id INT PRIMARY KEY,
    rate_code_description TEXT
);

-- Dimension : Location
CREATE TABLE dim_location (
    location_id INT PRIMARY KEY,
    borough TEXT,
    zone TEXT,
    latitude FLOAT,
    longitude FLOAT
);
charger fichier csv

-- Dimension : Payment Type
CREATE TABLE dim_payment (
    payment_type_id INT PRIMARY KEY,
    payment_description TEXT
);
faire à la main

-- Dimension : Time
CREATE TABLE dim_time (
    time_id SERIAL PRIMARY KEY,
    date DATE,
    year INT,
    month INT,
    day INT,
    hour INT,
    minute INT
);

-- Table factuelle
CREATE TABLE rides_fact (
    ride_id SERIAL PRIMARY KEY,
    pickup_datetime TIMESTAMP NOT NULL,
    dropoff_datetime TIMESTAMP NOT NULL,
    passenger_count INT,
    trip_distance FLOAT,
    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,
    vendor_id INT REFERENCES dim_vendor(vendor_id),
    rate_code_id INT REFERENCES dim_rate_code(rate_code_id),
    pickup_location_id INT REFERENCES dim_location(location_id),
    dropoff_location_id INT REFERENCES dim_location(location_id),
    payment_type_id INT REFERENCES dim_payment(payment_type_id)
);
faire à la main
```

Commande “`docker cp ~/Desktop/creation.sql atl-datamart-db-datamart-1:/tmp/`” pour déplacer dans le db-datamart.

Ensuite utilisation du script creation.sql

```
[datamart=# \i /tmp/creation.sql
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
```

Nous pouvons voir que le script a bien fonctionné

```
[datamart=# \dt
              List of relations
 Schema |      Name      | Type  | Owner
-----+--------------+-----+-----
 public | dim_location | table | postgres
 public | dim_payment | table | postgres
 public | dim_rate_code | table | postgres
 public | dim_time | table | postgres
 public | dim_vendor | table | postgres
 public | rides_fact | table | postgres
(6 rows)
```

- Création d'un script insertion.sql pour insérer les données depuis notre base de données data warehouse vers notre base de données data mart

Création d'un lien entre nyc_warehouse et datamart via dblink pour insérer les données.

```
[datamart=# CREATE EXTENSION IF NOT EXISTS dblink;
CREATE EXTENSION
```

```
[datamart=# SELECT dblink_connect(
  'nyc_warehouse_conn',
  'host=db port=5432 dbname=nyc_warehouse user=postgres password=admin'
);
dblink_connect
-----
OK
(1 row)
```

```
-- Remplir la dimension Vendor
INSERT INTO dim_vendor (vendor_id, vendor_name)
SELECT DISTINCT COALESCE(vendorid, -1), 'Unknown Vendor'
FROM dblink(
  'nyc_warehouse_conn',
  'SELECT DISTINCT vendorid FROM nyc_raw'
) AS t(vendorid INT);

-- Remplir la dimension Rate Code
INSERT INTO dim_rate_code (rate_code_id, rate_code_description)
SELECT DISTINCT COALESCE(ratecodeid, -1), 'Unknown Rate Code'
FROM dblink(
  'nyc_warehouse_conn',
  'SELECT DISTINCT ratecodeid FROM nyc_raw'
) AS t(ratecodeid INT);

-- Remplir la dimension Location
INSERT INTO dim_location (location_id, borough, zone, latitude, longitude)
SELECT DISTINCT COALESCE(pulocationid, -1), 'Unknown Borough', 'Unknown Zone', NULL::DOUBLE PRECISION, NULL::DOUBLE
PRECISION
FROM dblink(
  'nyc_warehouse_conn',
  'SELECT DISTINCT pulocationid FROM nyc_raw'
) AS t(pulocationid INT)
```

```

UNION
SELECT DISTINCT COALESCE(dolocationid, -1), 'Unknown Borough', 'Unknown Zone', NULL::DOUBLE PRECISION, NULL::DOUBLE
PRECISION
FROM dblink(
  'nyc_warehouse_conn',
  'SELECT DISTINCT dolocationid FROM nyc_raw'
) AS t(dolocationid INT);

-- Remplir la dimension Payment Type
INSERT INTO dim_payment (payment_type_id, payment_description)
SELECT DISTINCT COALESCE(payment_type, -1), 'Unknown Payment'
FROM dblink(
  'nyc_warehouse_conn',
  'SELECT DISTINCT payment_type FROM nyc_raw'
) AS t(payment_type INT);

-- Remplir la dimension Time
INSERT INTO dim_time (date, year, month, day, hour, minute)
SELECT DISTINCT
  DATE(tpep_pickup_datetime),
  EXTRACT(YEAR FROM tpep_pickup_datetime),
  EXTRACT(MONTH FROM tpep_pickup_datetime),
  EXTRACT(DAY FROM tpep_pickup_datetime),
  EXTRACT(HOUR FROM tpep_pickup_datetime),
  EXTRACT(MINUTE FROM tpep_pickup_datetime)
FROM dblink(
  'nyc_warehouse_conn',
  'SELECT DISTINCT tpep_pickup_datetime FROM nyc_raw'
) AS t(tpep_pickup_datetime TIMESTAMP);

-- Remplir la table factuelle
INSERT INTO rides_fact (
  pickup_datetime, dropoff_datetime, passenger_count, trip_distance,
  fare_amount, extra, mta_tax, tip_amount, tolls_amount,
  improvement_surcharge, total_amount, congestion_surcharge, airport_fee,
  vendor_id, rate_code_id, pickup_location_id, dropoff_location_id, payment_type_id
)
SELECT
  tpep_pickup_datetime, tpep_dropoff_datetime, passenger_count, trip_distance,
  fare_amount, extra, mta_tax, tip_amount, tolls_amount,
  improvement_surcharge, total_amount, congestion_surcharge, airport_fee,
  COALESCE(vendorid, -1), -- Utilise -1 pour les valeurs NULL
  COALESCE(ratecodeid, -1), -- Utilise -1 pour les valeurs NULL
  COALESCE(pulocationid, -1), -- Utilise -1 pour les valeurs NULL
  COALESCE(dolocationid, -1), -- Utilise -1 pour les valeurs NULL
  COALESCE(payment_type, -1) -- Utilise -1 pour les valeurs NULL
FROM dblink(
  'nyc_warehouse_conn',
  'SELECT tpep_pickup_datetime, tpep_dropoff_datetime, passenger_count, trip_distance,
    fare_amount, extra, mta_tax, tip_amount, tolls_amount,
    improvement_surcharge, total_amount, congestion_surcharge, airport_fee,
    vendorid, ratecodeid, pulocationid, dolocationid, payment_type
  FROM nyc_raw'
) AS t(
  tpep_pickup_datetime TIMESTAMP, tpep_dropoff_datetime TIMESTAMP, passenger_count INT, trip_distance FLOAT,
  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,
  vendorid INT, ratecodeid INT, pulocationid INT, dolocationid INT, payment_type INT
);

```

Utilisation du script insertion.sql

```
[datamart=# \i /tmp/insertion.sql
INSERT 0 3
INSERT 0 8
INSERT 0 263
INSERT 0 6
INSERT 0 351020
INSERT 0 26388179
datamart=# ]
```

Remplissage des dimensions :

```
datamart=# UPDATE dim_rate_code
SET rate_code_description = CASE rate_code_id
    WHEN 1 THEN 'Standard rate'
    WHEN 2 THEN 'JFK'
    WHEN 3 THEN 'Newark'
    WHEN 4 THEN 'Nassau or Westchester'
    WHEN 5 THEN 'Negotiated fare'
    WHEN 6 THEN 'Group ride'
END
[WHERE rate_code_description = 'Unknown Rate Code';
UPDATE 8
datamart=# ]
```

```
datamart=# INSERT INTO dim_time (date, year, month, day, hour, minute)
VALUES
    ('2024-12-10', 2024, 12, 10, 14, 30),
    ('2024-12-11', 2024, 12, 11, 9, 15),
    ('2024-12-12', 2024, 12, 12, 16, 45)
ON CONFLICT (time_id) DO NOTHING;
INSERT 0 3
datamart=# ]
```

```
datamart=# INSERT INTO dim_vendor (vendor_id, vendor_name)
VALUES
    (1, 'Creative Mobile Technologies, LLC'),
    (2, 'VeriFone Inc.')
ON CONFLICT (vendor_id)
DO UPDATE SET vendor_name = EXCLUDED.vendor_name;
INSERT 0 2
datamart=# ]
```

```
datamart=# UPDATE dim_payment
SET payment_description = CASE payment_type_id
    WHEN 1 THEN 'Credit card'
    WHEN 2 THEN 'Cash'
    WHEN 3 THEN 'No charge'
    WHEN 4 THEN 'Dispute'
    WHEN 5 THEN 'Unknown' -- Si vous voulez garder 'Unknown' pour certaines valeurs
    WHEN 6 THEN 'Voided trip'
    ELSE 'Unknown Payment' -- Par défaut, pour toute autre valeur inattendue
END
WHERE payment_description = 'Unknown Payment';
UPDATE 6
datamart=# ]
```

Pour la table dim_location il fallait télécharger le fichier taxi_zone_lookup.csv et le mettre dans dim_location.

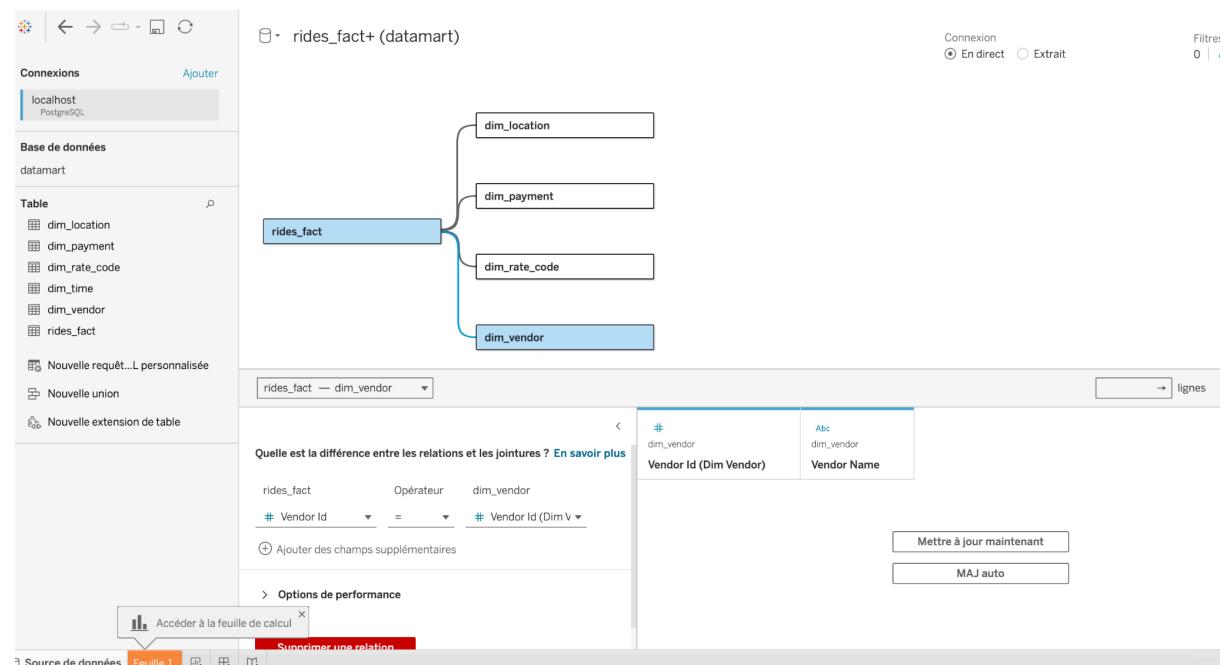
J'ai eu quelques problèmes (concernant les clés primaires) et pour les régler j'ai créé une table

```
datamart=# CREATE TEMP TABLE temp_location (
    location_id INT,
    borough TEXT,
    zone TEXT,
    service_zone TEXT
);
CREATE TABLE
datamart=# \COPY temp_location(location_id, borough, zone, service_zone)
FROM '/tmp/taxi_zone_lookup.csv'
DELIMITER ',' CSV HEADER;
COPY 265
datamart=# INSERT INTO dim_location(location_id, borough, zone, service_zone)
SELECT location_id, borough, zone, service_zone
FROM temp_location
ON CONFLICT (location_id)
DO UPDATE SET
    borough = EXCLUDED.borough,
    zone = EXCLUDED.zone,
    service_zone = EXCLUDED.service_zone;
INSERT 0 265
```

TP 4 : visualisation de données. Se connecter à Datamart avec Tableau ou Power BI et faire une EDA. Concevoir nos visualisations de données vers un Dashboard (concevoir le dashboard) et s'assurer qu'il se mette à jour automatiquement

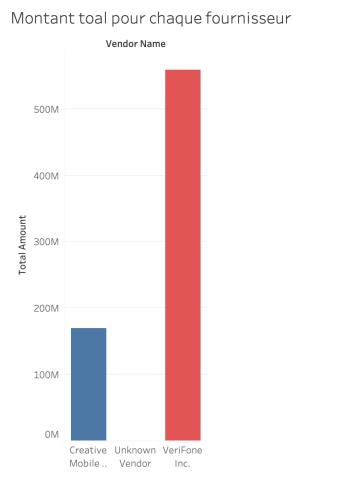
En amont il faut installer les pilotes nécessaires pour qu'on puisse se connecter à la bdd postgres.

Création des sources de données

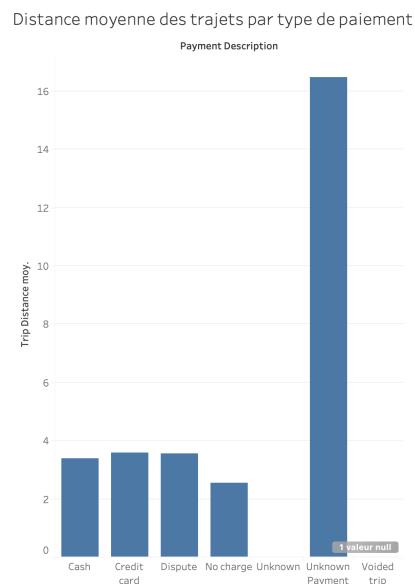


Voici mon EDA ainsi que les tableaux effectués :

- Montant total pour chaque fournisseur

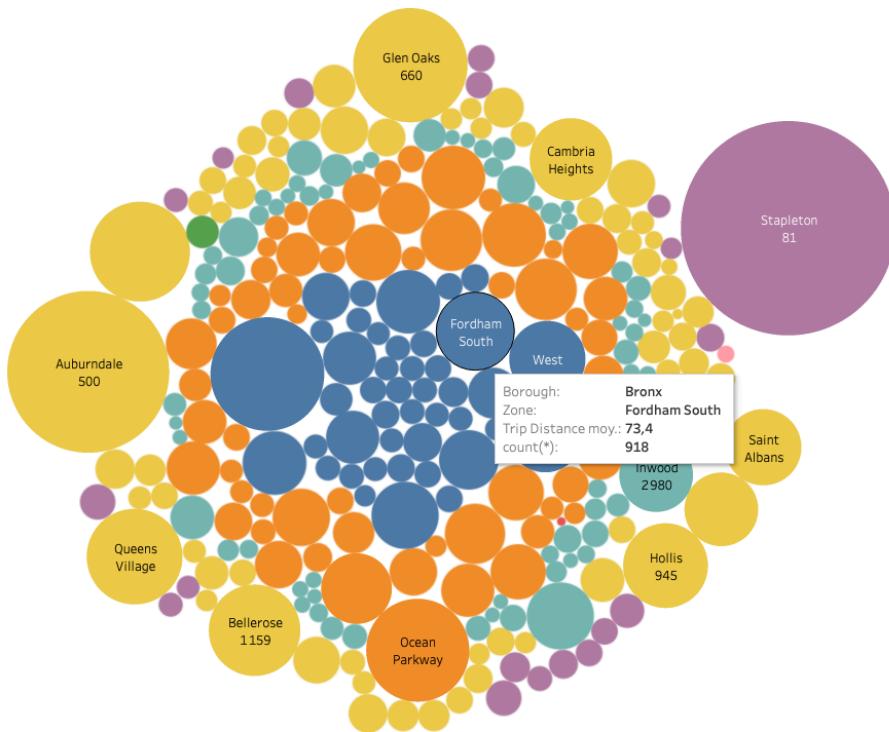


- Distance moyenne des trajets par type de paiement



- Analyse des trajets par distance moyenne

Analyse des trajets par distance moyenne



- Pourboires moyens par type de paiement

pourboires moyens par type de paiement

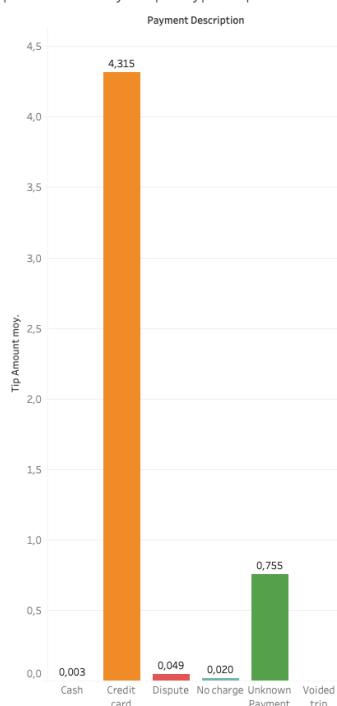
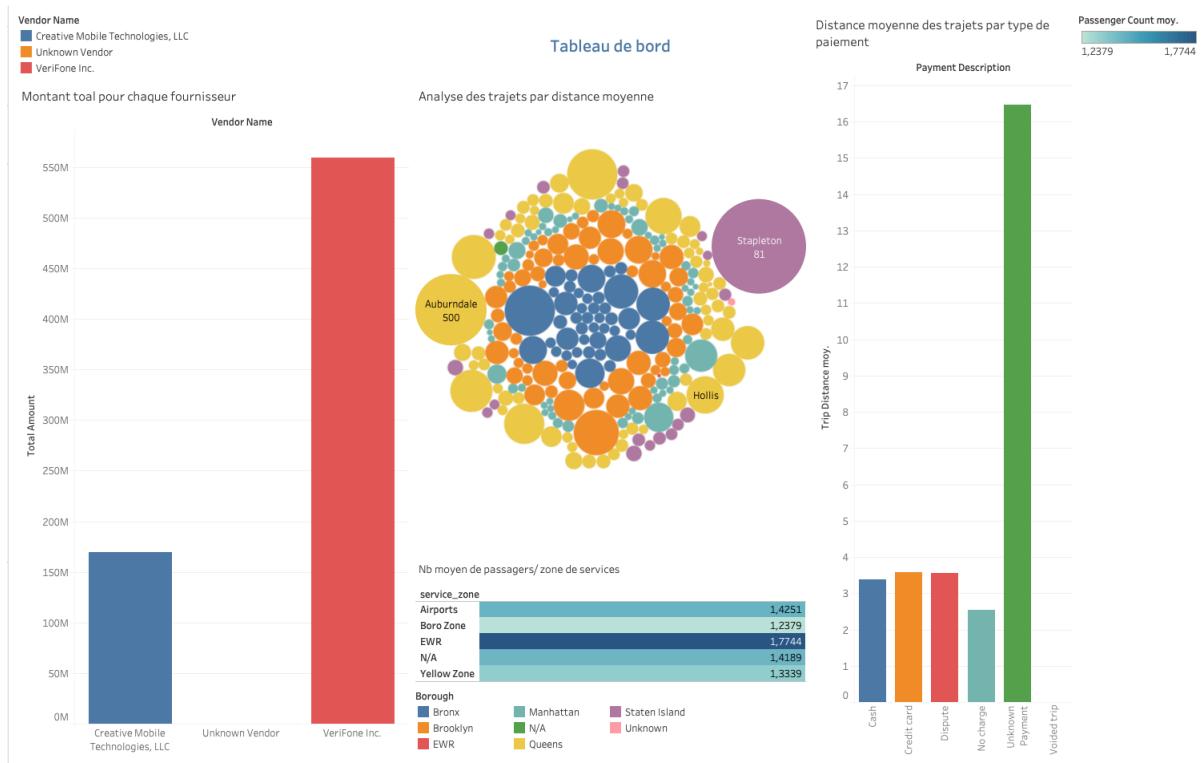


Tableau de bord qui contient des visualisations



Tp 5: Rédaction d'une DAG dans Apache Airflow en reprenant le code du tp1 sur la fonction "récupérer le mois dernier" qui va s'exécuter tous les 1er du mois

```

from airflow import DAG
from airflow.operators.python import PythonOperator
from datetime import datetime
import urllib.request
import os
from minio import Minio
from minio.error import S3Error

# Configuration de MinIO
MINIO_ENDPOINT = "minio:9000"
MINIO_BUCKET_NAME = "yellow-taxi-data"
MINIO_ACCESS_KEY = "minio"
MINIO_SECRET_KEY = "minio123"

# URL de téléchargement des données NYC
NYC_DATA_URL = "https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page"

# Fonction pour télécharger les données
def download_parquet_file():
    local_file_path = "/tmp/yellow_tripdata_2024-01.parquet"
    try:
        print(f"Téléchargement du fichier depuis {NYC_DATA_URL}...")
        urllib.request.urlretrieve(NYC_DATA_URL, local_file_path)
        print(f"Fichier téléchargé avec succès : {local_file_path}")
    except Exception as e:
        raise RuntimeError(f"Erreur lors du téléchargement du fichier : {str(e)}")

```

```

return local_file_path

# Fonction pour envoyer le fichier vers MinIO
def upload_to_minio(local_file_path):
    try:
        print("Connexion à MinIO...")
        client = Minio(
            MINIO_ENDPOINT,
            access_key=MINIO_ACCESS_KEY,
            secret_key=MINIO_SECRET_KEY,
            secure=False,
        )

        # Vérifier si le bucket existe, sinon le créer
        if not client.bucket_exists(MINIO_BUCKET_NAME):
            client.make_bucket(MINIO_BUCKET_NAME)
            print(f"Bucket {MINIO_BUCKET_NAME} créé.")
        else:
            print(f"Bucket {MINIO_BUCKET_NAME} déjà existant.")

        # Téléverser le fichier
        object_name = os.path.basename(local_file_path)
        client.fput_object(MINIO_BUCKET_NAME, object_name, local_file_path)
        print(f"Fichier téléversé avec succès : {object_name} dans le bucket {MINIO_BUCKET_NAME}")
    except ConnectionRefusedError:
        raise RuntimeError("Impossible de se connecter à MinIO. Vérifiez que MinIO est en cours d'exécution.")
    except S3Error as e:
        raise RuntimeError(f"Erreur MinIO : {str(e)}")
    except Exception as e:
        raise RuntimeError(f"Erreur inconnue : {str(e)}")
    finally:
        # Supprimer le fichier local après l'envoi
        if os.path.exists(local_file_path):
            os.remove(local_file_path)
            print(f"Fichier local supprimé : {local_file_path}")

# Définir le DAG et les tâches
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
}

with DAG(
    dag_id="grab_nyc_data_to_minio",
    default_args=default_args,
    description="Télécharge des données NYC et les stocke dans MinIO",
    schedule="0 12 * * *", # Exécuter tous les jours à midi
    start_date=datetime(2024, 12, 15),
    catchup=False,
) as dag:

    # Tâche 1 : Télécharger le fichier Parquet
    download_task = PythonOperator(
        task_id="download_parquet_file",
        python_callable=download_parquet_file,

```

```

)
# Tâche 2 : Envoyer le fichier vers MinIO
upload_task = PythonOperator(
    task_id="upload_to_minio",
    python_callable=lambda: upload_to_minio(download_parquet_file()),
)

```

```

# Définir l'ordre des tâches
download_task >> upload_task

```

The screenshot shows the Airflow UI with the 'Logs' tab selected for the 'upload_to_minio' task. The log output is as follows:

```

[2024-12-18, 14:38:49 CET] [local_task_job_runner.py:123] ▶ Pre task execution logs
[2024-12-18, 14:38:49 CET] [logging_mixin.py:190] INFO - Téléchargement du fichier depuis https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page...
[2024-12-18, 14:38:50 CET] [logging_mixin.py:190] INFO - Fichier téléchargé avec succès : /tmp/yellow_tripdata_2024-01.parquet
[2024-12-18, 14:38:50 CET] [logging_mixin.py:190] INFO - Connexion à MinIO...
[2024-12-18, 14:38:56 CET] [logging_mixin.py:190] INFO - Fichier local supprimé : /tmp/yellow_tripdata_2024-01.parquet
[2024-12-18, 14:38:56 CET] [logging_mixin.py:190] INFO - Task instance in failure state
[2024-12-18, 14:38:56 CET] [logging_mixin.py:190] INFO - Task start:2024-12-18 13:38:49.212800+00:00 end:2024-12-18 13:38:56.493915+00:00 duration:7.281115
[2024-12-18, 14:38:56 CET] [logging_mixin.py:190] INFO - Task:<Task(PythonOperator): upload_to_minio> dag:grab_nyc_data_to_minio dagrun:< DagRun grab_nyc_data_to_minio at 2024-12-18 14:38:56 CET > host:localhost port:9000 > Max retries exceeded with url: /taskinstance.py:1225
[2024-12-18, 14:38:56 CET] [taskinstance.py:340] ▶ Post task execution logs
[2024-12-18, 14:38:56 CET] [local_task_job_runner.py:123] ▶ Pre task execution logs
[2024-12-18, 15:48:54 CET] [logging_mixin.py:190] INFO - Téléchargement du fichier depuis https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page...
[2024-12-18, 15:48:54 CET] [logging_mixin.py:190] INFO - Fichier téléchargé avec succès : /tmp/yellow_tripdata_2024-01.parquet
[2024-12-18, 15:48:55 CET] [logging_mixin.py:190] INFO - Connexion à MinIO...
[2024-12-18, 15:48:55 CET] [logging_mixin.py:190] INFO - Bucket yellow-taxi-data déjà existant.
[2024-12-18, 15:48:55 CET] [logging_mixin.py:190] INFO - Fichier téléversé avec succès : yellow_tripdata_2024-01.parquet dans le bucket yellow-taxi-data
[2024-12-18, 15:48:55 CET] [logging_mixin.py:190] INFO - Fichier local supprimé : /tmp/yellow_tripdata_2024-01.parquet
[2024-12-18, 15:48:55 CET] [python.py:240] INFO - Done. Returned value was: None
[2024-12-18, 15:48:56 CET] [taskinstance.py:340] ▶ Post task execution logs

```

The screenshot shows the Airflow UI with the DAG graph selected for the 'grab_nyc_data_to_minio' DAG. The graph displays two tasks: 'download_parquet_file' and 'upload_to_minio'. An arrow points from 'download_parquet_file' to 'upload_to_minio', indicating a dependency. Both tasks are shown as green boxes labeled 'success'.

yellow-taxi-data		
Created on: Mon, Oct 28 2024 14:16:41 (GMT+1) Access: PRIVATE 435.5 MiB - 9 Objects		
		Rewind ⏪ Refresh ⌛ Upload ⚡
◀	yellow-taxi-data	Create new path ⌘F
☐	▲ Name	Last Modified
☐	yellow_tripdata_2024-01.parquet	Today, 15:48
☐	yellow_tripdata_2024-02.parquet	Thu, Nov 28 2024 06:25 (GMT+1)
☐	yellow_tripdata_2024-03.parquet	Thu, Nov 28 2024 06:25 (GMT+1)
☐	yellow_tripdata_2024-04.parquet	Thu, Nov 28 2024 06:25 (GMT+1)
☐	yellow_tripdata_2024-05.parquet	Thu, Nov 28 2024 06:25 (GMT+1)
☐	yellow_tripdata_2024-06.parquet	Thu, Nov 28 2024 06:26 (GMT+1)