Projet Big Data : Analyse et gestion pour une entreprise VTC à New-York

Yannick Mokto, Julien Mfochive, Mohammed

Lundi 22 Avril 2024

Resumé

L’entreprise de VTC basée à New York, est positionnée dans un marché très compétitif où l'efficacité opérationnelle et la satisfaction du client sont primordiales pour le succès. Dans cet environnement dynamique, l'utilisation stratégique des données devient un levier crucial pour optimiser les opérations et améliorer les services offerts. Le projet envisagé cherche à exploiter les données collectées et disponibles via les sources publiques pour transformer ces informations en connaissances actionnables, soutenant ainsi des décisions d'affaires éclairées.

TP1 Récupération des données

Nous allons écrire deux programmes qui vont récupérer les données du TLC Trip Record Data de la ville de New York, et les stocker dans Minio qui agira en tant que Data Lake. Pour ce faire, nous aurons besoin des bibliothèques Python suivantes : **requests** pour faire des requêtes HTTP afin de télécharger les données, et **minio** pour interagir avec le Data Lake Minio.

Pré-requis:

- Fork le git https://github.com/Noobzik/ATL-Datamart
- Installer docker https://docs.docker.com/engine/install/
- Installation des paquets nécessaires pip install requests minio

Grab_parquet.py : Télécharger les données de Novembre à Décembre 2023

```python
script_dir = os.path.dirname(os.path.abspath(__file__))
# Construct the relative path to the folder
folder_path = os.path.join(script_dir, '..', '..', 'data', 'raw')
# URL des données des taxis de NYC
data_url = 'https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2023'
# Répertoire pour récupérer les données
save_dir = '../../data/raw'
os.makedirs(save_dir, exist_ok=True)
# Téléchargement
for month in range(1, 9):  # Récupérer de janvier à août
    month_str = str(month).zfill(2)  # Formatage du mois sur deux chiffres avec un zéro devant si nécessaire
    file_url = f'{data_url}-{month_str}.parquet'
    file_path = os.path.join(folder_path, f'2023-{month_str}-tripdata.parquet')
    try:
        urllib.request.urlretrieve(file_url, file_path)
        print(f'Downloaded: {file_path}')
    except urllib.error.HTTPError as e:      You, il y a 2 semaines • Uncommitted changes
        print(f"Error downloading {file_url}: {e}")
```

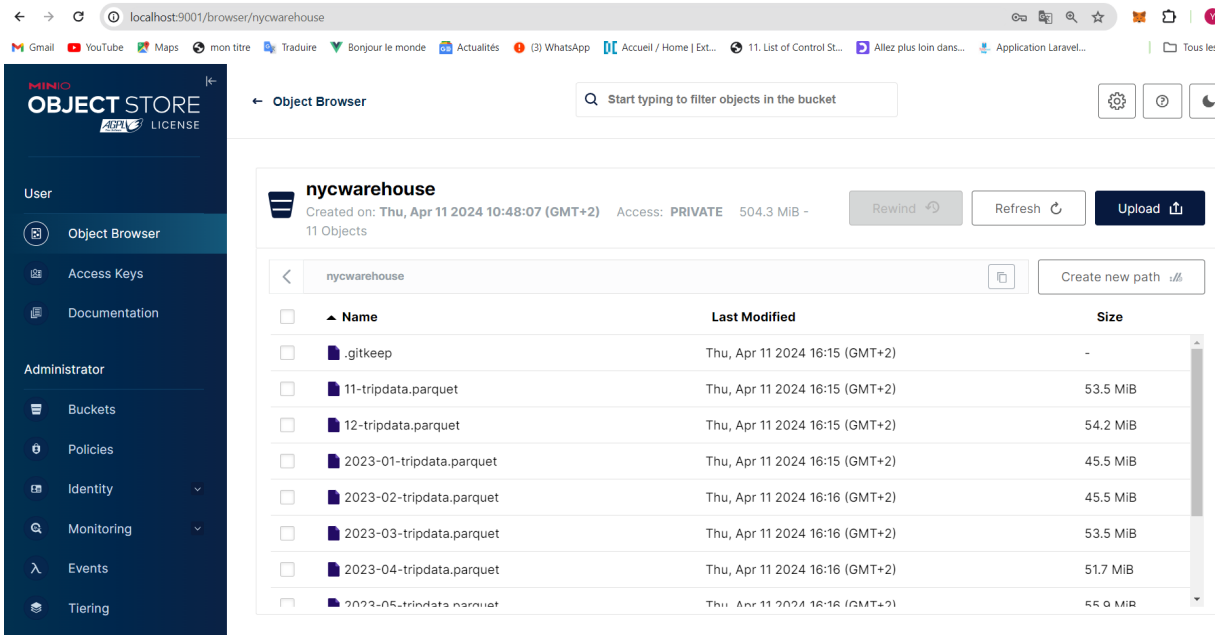
Télécharger les données du dernier mois disponible

```python
script_dir = os.path.dirname(os.path.abspath(__file__))
# Construct the relative path to the folder
folder_path = os.path.join(script_dir, '..', '..', 'data', 'raw')
# Récupération de l'année actuelle
current_year = datetime.datetime.now().year
# URL des données des taxis de NYC pour l'année actuelle
data_url = f'https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_{current_year}'
# Répertoire pour récupérer les données
save_dir = os.path.join(script_dir, '..', '..', 'data', 'raw')
os.makedirs(save_dir, exist_ok=True)
# Récupération du mois actuel
current_month = datetime.datetime.now().month
current_month_str = str(current_month).zfill(2)  # Formatage du mois sur deux chiffres avec un zéro devant si nécessaire
# Téléchargement du mois actuel
file_url = f'{data_url}-{current_month_str}.parquet'
file_path = os.path.join(folder_path, f'{current_year}-{current_month_str}-tripdata.parquet')
try:
    urllib.request.urlretrieve(file_url, file_path)
    print(f'Downloaded latest month data: {file_path}')
except urllib.error.HTTPError as e:
    print(f"Error downloading {file_url}: {e}")
```

```python
# Créez une instance Minio avec les détails de connexion
client = Minio(
    "localhost:9000",
    secure=False,
    access_key="minio",
    secret_key="minio123"
)

bucket_name = "nycwarehouse"  # Nom du bucket Minio

# Vérifiez si le bucket existe, sinon, créez-le
found = client.bucket_exists(bucket_name)
if not found:
    client.make_bucket(bucket_name)
else:
    print("Le bucket existe déjà")

# Parcourez les fichiers dans le dossier de destination
for filename in os.listdir('../../data/raw'):
    # Assurez-vous que le fichier est un fichier (pas un répertoire) avant de l'envoyer à Minio
    if os.path.isfile(os.path.join('../../data/raw', filename)):
        file_path = os.path.join('../../data/raw', filename)
        # Téléversez le fichier dans Minio
        object_name = os.path.basename(file_path)
        client.fput_object(bucket_name, object_name, file_path)
        print(f"Fichier téléchargé dans Minio : {object_name}")
```

Données récupérées par Minio

Liste d'objets dans la bucket « nycwarehouse » du MINIO Object Storage.

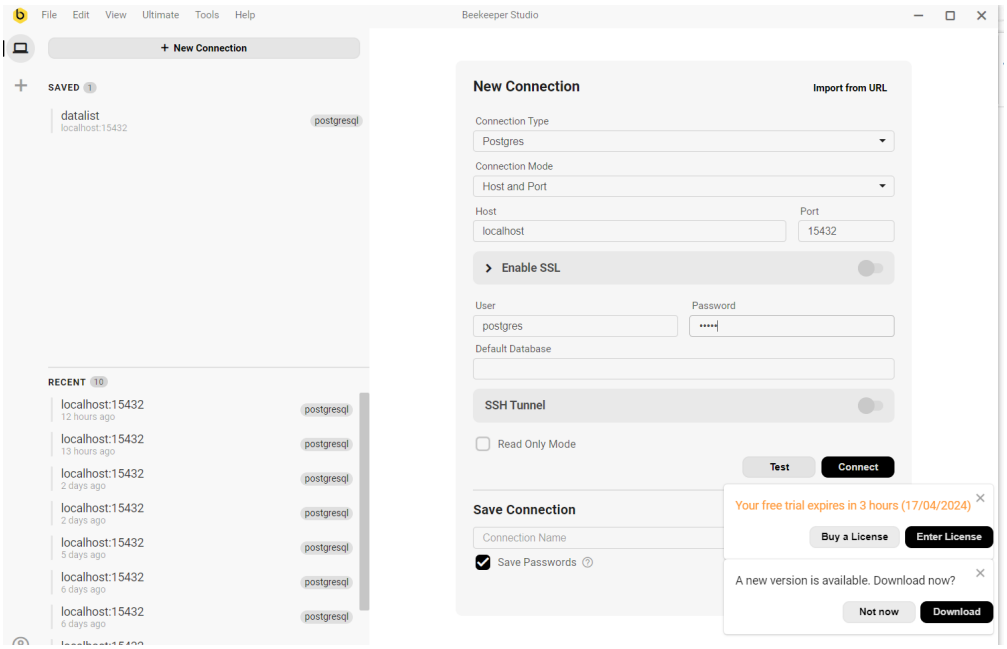


TP2 récupération de par postgres

Nous allons écrire un script qui récupère les données stockées dans Minio et les charge dans une base de données PostgreSQL sans aucune transformation. Pour cela, nous aurons besoin d'utiliser la bibliothèque **psycopg2** pour interagir avec PostgreSQL, et **minio** pour interagir avec Minio.

Prérequis : Installation du package nécessaire `pip install psycopg2-binary minio pandas`

PostgreSQL doit être en cours d'exécution et accessible (port 15432, user :postegres password :admin)



Modifier les fichiers dump_to_sql.py en adaptant les paramètres de connexion

```python
db_config = {
    "dbms_engine": "postgresql",
    "dbms_username": "postgres",
    "dbms_password": "admin",
    "dbms_ip": "localhost",
    "dbms_port": "15432",
    "dbms_database": "nyc_warehouse",
    "dbms_table": "nyc_raw"
}

db_config["database_url"] = (
    f"{db_config['dbms_engine']}://{db_config['dbms_username']}:{db_config['dbms_password']}@"
    f"{db_config['dbms_ip']}:{db_config['dbms_port']}/{db_config['dbms_database']}"
)
try:
    engine = create_engine(db_config["database_url"])
    with engine.connect():
        success: bool = True
        print("Connection successful! Processing parquet file")
        dataframe.to_sql(db_config["dbms_table"], engine, index=False, if_exists='append')
```

TP3 Mise en place du modèle

Étape 1 : Conception du modèle flocon

Considérons les éléments de données suivants :

- **Date/Heure** : informations détaillées sur le temps (année, mois, jour, heure).
- **Véhicules** : détails sur les véhicules utilisés (type, modèle, année).
- **Chauffeurs** : informations sur les chauffeurs (ID, nom, licence).
- **Trajets** : détails spécifiques aux trajets (ID trajet, point départ, point arrivée, distance, durée, coût).

Imaginons les tables suivantes pour le schéma en flocon :

- **Tab Trajets** : stocke les clés étrangères vers toutes les dimensions ainsi que les mesures (distance, durée, coût).
- **Dim Date** : année, mois, jour, heure.
- **Dim Véhicules** : type de véhicule, modèle, année.
- **Dim Chauffeurs** : ID, nom, numéro de licence.
- **Dim Loc : I**D, adresse, latitude, longitude.

Étape 2 : Implémentation de la BD

```sql
CREATE TABLE dim_date (
    date_id SERIAL PRIMARY KEY,
    year INT,
    month INT,
    day INT,
    hour INT
);
```

```sql
CREATE TABLE dim_vehicles (
    vehicle_id SERIAL PRIMARY KEY,
    type VARCHAR(50),
    model VARCHAR(50),
    year INT
);
```

```sql
CREATE TABLE fact_trips (
    trip_id SERIAL PRIMARY KEY,
    date_id INT,
    vehicle_id INT,
    driver_id INT,
    start_location_id INT,
    end_location_id INT,
    distance FLOAT,
    duration INT,
    cost FLOAT,
    FOREIGN KEY (date_id) REFERENCES dim_date(date_id),
    FOREIGN KEY (vehicle_id) REFERENCES dim_vehicles(vehicle_id),
```

Supprimer les trajets sans passagers (passenger_count = 0) :

```
DELETE FROM public.fact_taxi_trip
WHERE passenger_count <= 0;
```

Supprimer les trajets avec des montants de paiement nuls ou négatifs

```
DELETE FROM public.fact_taxi_trip
WHERE fare_amount <= 0
  OR tip_amount < 0
  OR tolls_amount < 0
  OR total_amount <= 0;
```

Supprimer les trajets qui ont une durée de trajet improbable (par exemple, négative)

```
DELETE FROM public.fact_taxi_trip
WHERE pickup_datetime > dropoff_datetime;
DELETE FROM public.fact_taxi_trip
WHERE pickup_datetime IS NULL
  OR dropoff_datetime IS NULL
  OR passenger_count IS NULL
  OR trip_distance IS NULL
  OR ratecodeid IS NULL
  OR pulocationid IS NULL
  OR dolocationid IS NULL
  OR payment_type IS NULL
  OR fare_amount IS NULL
  OR total_amount IS NULL;


DELETE FROM public.fact_taxi_trip
WHERE EXTRACT(YEAR FROM pickup_datetime) = 2022
  OR EXTRACT(YEAR FROM dropoff_datetime) = 2022;
```

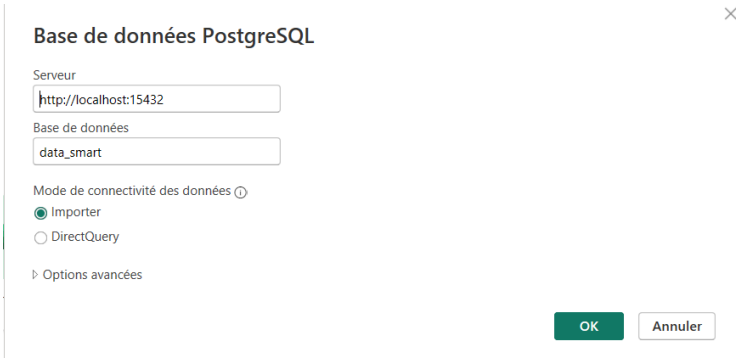
TP4 Dashboard

Dans cette partie, nous transformerons les données brutes en insights visuels et interactifs

Connexion à POWER BI

- Aller à 'Get Data'.
- Choisir 'PostgreSQL database' et entrez les détails de connexion.
- Importer les tables nécessaires pour charger des données spécifiques.

Connecte à la base de données à partie de power BI



Première Analyse des Données (EDA)

Jupytr Notebook pour réaliser une exploration de données (EDA). Points clés explorés :

- Statistiques descriptives (moyenne, médiane, écart-type, etc.).
- Distribution des variables principales (coût des trajets, durée, distances).
- Corrélations entre les différentes mesures (durée et coût, distance et coût).
- Identification des valeurs aberrantes

Sur la base de notre EDA, KPI pertinents sont :

- Total des revenus par mois.
- Nombre de trajets par jour/semaine/mois.
- Durée moyenne des trajets.

Charger des données et Réaliser les Dashboard