

# TP D'ARCHITECTURE DECISIONNELLE

## ENCADRANT:

- SHEIKH Rakib

## GROUPE :

- SARR Awa
- SEME Mamadou
- MBOUP Birame
- TAWATIEU Meghane
- THIAM Oumou

# SOMMAIE

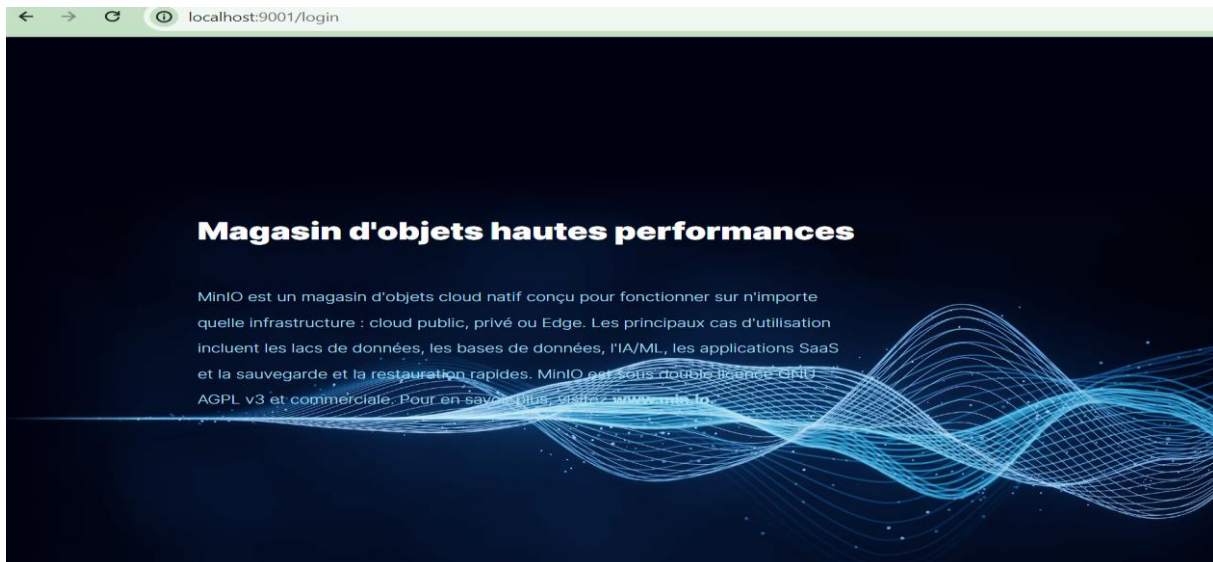
<b>SOMMAIE .....</b>	<b>1</b>
<b>OBJECTIF DU TP :.....</b>	<b>2</b>
<b>Prérequis : .....</b>	<b>2</b>
<b>I. TP1 : Télécharger des fichiers et upload to Minio.....</b>	<b>3</b>
1. Tâche 1 : Téléchargeons les fichiers de données d'enregistrement des voyages TLC pour les mois de novembre et de décembre 2023. ....	3
2. Tâche 2: Upload des fichiers téléchargés to Minio .....	4
<b>II. TP 2: Transfert des fichiers depuis Minio vers la base de données .....</b>	<b>4</b>
<b>III. TP 3 : Scripts SQL pour la création et l'insertion de données dans un modèle en flocon .....</b>	<b>5</b>
1. Grille de lecture .....	5
2. Présentation de notre modèle en flacon.....	6
3. Creation.sql.....	6
4. Insertion.sql .....	8
5. Verification des données insérées .....	9
<b>IV. TP4: Visualisation de nos données à partir d'un outil de visualisation. 10</b>	
1. Power BI.....	10
2. Réalisation des Dashboard .....	11
a. Le type de paiements utilisés par les passagers .....	11
b. La répartition des voyages par type de tarification.....	12
c. Le montant total facturé aux passagers .....	12
<b>V. TP 5 (optionnel) .....</b>	<b>13</b>

## OBJECTIF DU TP :

Le but de ce TP est de déployer une architecture data dès la récupération de la donnée vers la restitution sous la forme de dataviz en passant par un Datalake, Data Warehouse et Data Mart. Ce TP est divisé en plusieurs parties, allant du TP1 jusqu'au TP5. Nous aborderons chaque TP séquentiellement pour progresser de manière organisée dans notre projet.

### Prérequis :

1. Installer **GIT**
2. Installer **Docker**
3. Installer miniconda (**PYTHON**)
4. Création de compte **Github** si on n'a pas de compte
5. <https://github.com/Noobzik/ATL-datamart> + Fork
6. Cd ATL-Datamart + Git clone <https://github.com/oumishou/ATL-Datamart.git> (dans mo cmd)
7. Docker compose UP
8. Vérifie si le docker est bon : à travers localhost : 9001



## I. TP1 : Télécharger des fichiers et upload to Minio

Dans ce TP1, l'objectif est de récupérer les fichiers de données d'enregistrement des voyages TLC pour les mois de novembre et de décembre 2023 depuis le site officiel <https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page>. Ces fichiers seront téléchargés sur le disque dur local avant d'être transférés vers un service de stockage en ligne, tel que Minio.

### 1. Tâche 1 : Téléchargeons les fichiers de données d'enregistrement des voyages TLC pour les mois de novembre et de décembre 2023.

Pour ce faire, nous allons se positionner sur le fichier qui se situe à `src/data/grab_parquet.py` et compléter les fonctions qui sont vides.

Le script ajouté :

```
#Novembre 2023
url1 = "https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2023-11.parquet"
destination1 = "C:\\Users\\OUMOU THIAM\\ATL-Datamart\\data\\raw\\novembre.parquet"

#Decembre 2023
url = "https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2023-12.parquet"
destination = "C:\\Users\\OUMOU THIAM\\ATL-Datamart\\data\\raw\\decembre.parquet"

# Téléchargement du fichier depuis l'URL
urllib.request.urlretrieve(url, destination)
urllib.request.urlretrieve(url1, destination1)
print(f"Le fichier a été téléchargé avec succès à l'emplacement : {destination}")
print(f"Le fichier a été téléchargé avec succès à l'emplacement : {destination1}")
```

Après l'exécution du programme :

```
"C:\\Users\\OUMOU THIAM\\ATL-Datamart\\.venv\\Scripts\\python.exe" "C:\\Users\\OUMOU THIAM\\ATL-Datamart\\src\\data\\grab_parquet.py"
Le fichier a été téléchargé avec succès à l'emplacement : C:\\Users\\OUMOU THIAM\\ATL-Datamart\\data\\raw\\decembre.parquet
Le fichier a été téléchargé avec succès à l'emplacement : C:\\Users\\OUMOU THIAM\\ATL-Datamart\\data\\raw\\novembre.parquet

Process finished with exit code 0
```

Vérification sur le disque local :

> Ce PC > Disque local (C:) > Utilisateurs > OUMOU THIAM > ATL-Datamart > data > raw				
	Nom	Modifié le	Type	Taille
	decembre.parquet	07/03/2024 15:20	Fichier PARQUET	55 473 Ko
	novembre.parquet	06/03/2024 22:08	Fichier PARQUET	54 780 Ko

## 2. Tâche 2: Upload des fichiers téléchargés to Minio

Pour commencer, nous avons créé un Bucket dans Minio que nous avons nommé "oumishou", ensuite, nous avons définis la fonction `write_data_minio()`. Cette fonction commence par créer une instance de client Minio avec les détails de notre serveur Minio (hôte, clé d'accès, clé secrète) ect...

### Après l'exécution du programme :

```
"C:\Users\OUMOU THIAM\ATL-Datamart\.venv\Scripts\python.exe" "C:\Users\OUMOU THIAM\ATL-Datamart\src\data\grab_parquet.py"
Les fichiers C:\Users\OUMOU THIAM\ATL-Datamart\data\raw\novembre.parquet ont été téléversés avec succès dans MinIO.
Les fichiers C:\Users\OUMOU THIAM\ATL-Datamart\data\raw\decembre.parquet ont été téléversés avec succès dans MinIO.

Process finished with exit code 0
```

### Vérification dans Minio :

oumishou		
Created on: Wed, Mar 06 2024 22:20:39 (GMT) Access: PRIVATE 107.7 MiB - 2 Objects		
<div>oumishou</div>		
<input type="checkbox"/>	<b>Name</b>	<b>Last Modified</b>
<input type="checkbox"/>	decembre.parquet	Thu, Mar 07 2024 15:20 (GMT)
<input type="checkbox"/>	novembre.parquet	Thu, Mar 07 2024 15:20 (GMT)

## II. TP 2: Transfert des fichiers depuis Minio vers la base de données

Dans le cadre du TP2, l'objectif est de transférer les fichiers stockés dans Minio vers notre base de données. En modifiant le programme du fichier `src/data/dump_to_sql.py` pour qu'il puisse récupérer les fichiers depuis Minio et les charge dans notre base de données.

**NB :** Notre base de données s'appelle `oumou_warehouse` et la table `oumou_raw`.

### Explication du programme :

Ce script Python a pour but de transférer des fichiers stockés dans Minio vers notre base de données `oumou_warehouse`. Il se connecte à Minio, récupère les fichiers Parquet du seau spécifié (`oumishou`), les lit en tant que DataFrame Pandas, modifie les noms de colonnes en **minuscules**, puis les charge dans la base de données. En cas d'erreur, il affiche un message approprié.

### Après l'exécution du programme :

Nous avons vérifié les données chargées en utilisant Beekeeper. Nous avons exécuté la commande SQL `"SELECT * FROM oumou_raw LIMIT 10;"` pour afficher les dix premières lignes de notre table.

	vendorid	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	rateschid	store_and_ford_flag	pulocationid	dislocationid	payment_type	fare_amount	extra	mta_tax
1	1	2023-12-01 00:06:06	2023-12-01 00:15:47	0	1.1	1	N	230	48	1	10	3.5	0.5
2	1	2023-12-01 00:22:26	2023-12-01 00:26:53	0	1.5	1	N	142	238	1	9.3	3.5	0.5
3	1	2023-12-01 00:59:44	2023-12-01 01:13:22	2	2.2	1	N	114	186	1	13.5	3.5	0.5
4	2	2023-12-01 00:22:17	2023-12-01 00:30:59	1	0.66	1	N	79	79	2	7.2	1	0.5
5	2	2023-12-01 00:18:16	2023-12-01 00:25:32	2	2.2	1	N	229	263	1	11.4	1	0.5
6	1	2023-12-01 00:13:17	2023-12-01 00:23:53	0	5.7	1	N	88	141	1	23.3	3.5	0.5
7	2	2023-12-01 00:17:09	2023-12-01 00:33:31	1	5.33	1	N	45	162	1	24.7	1	0.5
8	2	2023-12-01 00:40:49	2023-12-01 00:44:10	1	0.76	1	N	170	107	1	5.8	1	0.5
9	2	2023-12-01 00:19:04	2023-12-01 00:34:36	1	3.33	1	N	186	209	1	17.7	1	0.5
10	1	2023-12-01 00:08:39	2023-12-01 00:16:18	1	2.1	1	N	163	262	1	12.1	3.5	0.5

### III. TP 3 : Scripts SQL pour la création et l'insertion de données dans un modèle en flocon

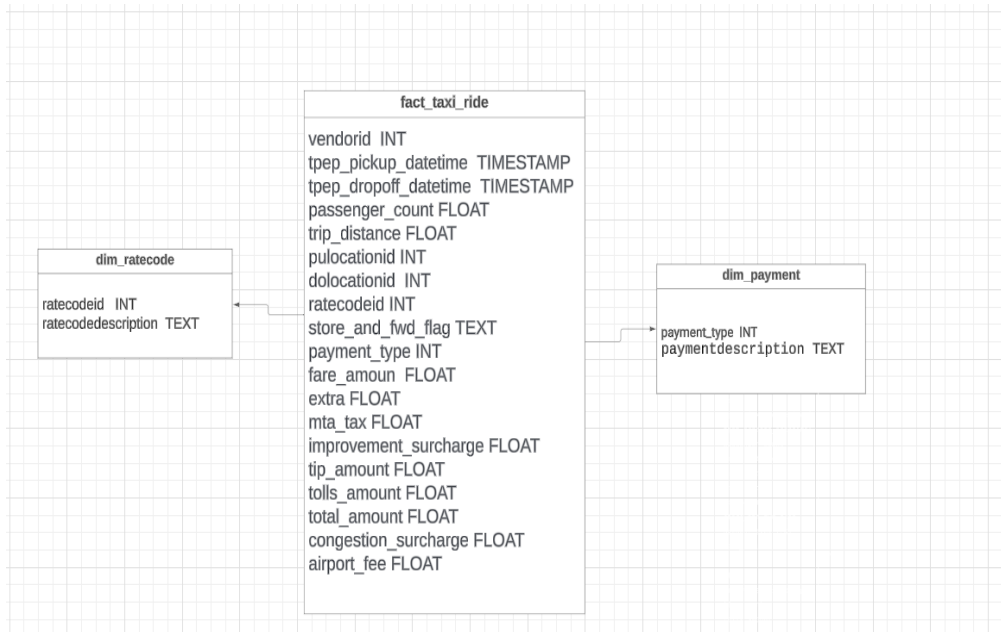
Dans le cadre du TP2, l'objectif est de créer des scripts SQL pour mettre en œuvre un modèle en flocon dans une base de données. Le script [creation.sql](#) sera utilisé pour créer les tables en flocon avec les contraintes associées, tandis que le script [insertion.sql](#) sera utilisé pour insérer les données de la base de données [oumou\\_warehouse](#) vers la base de données [oumou\\_datamart](#).

#### 1. Grille de lecture

Nous avons créé une grille de lecture pour notre ensemble de données de taxi afin de mieux comprendre chaque champ et sa signification. Cela nous aide à interpréter et analyser les données, ainsi qu'à formuler des requêtes SQL adaptées à nos besoins spécifiques. En résumé, cette grille de lecture nous permet de comprendre, identifier, structurer et utiliser les informations de nos données pour des analyses, des requêtes et des visualisations.

**NB : Le fichier sur la grille de lecture pour notre ensemble de données sera en jointure sur le fichier Github.**

## 2. Présentation de notre modèle en flocon



Notre modèle en flocon est organisé autour d'une table de faits centrale (**fact\_taxi\_ride**) qui contient les mesures principales de notre système.

Les tables de dimensions (**dim\_ratecode** et **dim\_Payment**) fournissent des informations détaillées et des descriptions pour les codes tarifaires et les modes de paiement utilisés dans les trajets en taxi.

En utilisant des clés étrangères, nous établissons des relations entre la table de faits et les tables de dimensions, ce qui permet de naviguer facilement entre les différentes entités et d'effectuer des analyses multidimensionnelles.

En résumé, notre modèle en flocon est conçu pour faciliter l'analyse et la compréhension des données liées aux trajets en taxi, en organisant les informations de manière structurée et hiérarchique.

## 3. Creation.sql

Nous allons utiliser un script SQL pour la création des tables en flocons avec les contraintes associés.

## Voici le script utilisé :

```
-- Création de la table de dimension pour ratecode dans DataMart
CREATE TABLE IF NOT EXISTS dim_ratecode (
  ratecodeid INT PRIMARY KEY,
  ratecodedescription TEXT
);

-- Création de la table de dimension pour payment dans DataMart
CREATE TABLE IF NOT EXISTS dim_payment (
  payment_type INT PRIMARY KEY,
  paymentdescription TEXT
);

-- Création de la table de faits taxi jaune dans DataMart
CREATE TABLE IF NOT EXISTS fact_taxi_ride (
  vendorid INT,
  tpep_pickup_datetime TIMESTAMP (6),
  tpep_dropoff_datetime TIMESTAMP (6),
  passenger_count FLOAT,
  trip_distance FLOAT,
  pulocationid INT,
  dolocationid INT,
  ratecodeid INT,
  store_and_fwd_flag TEXT,
  payment_type INT,
  fare_amount FLOAT,
  extra FLOAT,
  mta_tax FLOAT,
  improvement_surcharge FLOAT,
  tip_amount FLOAT,
  tolls_amount FLOAT,
  total_amount FLOAT,
  congestion_surcharge FLOAT,
  airport_fee FLOAT,
  PRIMARY KEY (vendorid, tpep_pickup_datetime, tpep_dropoff_datetime),
  FOREIGN KEY (ratecodeid) REFERENCES dim_ratecode(ratecodeid),
  FOREIGN KEY (payment_type) REFERENCES dim_payment(payment_type)
);
```

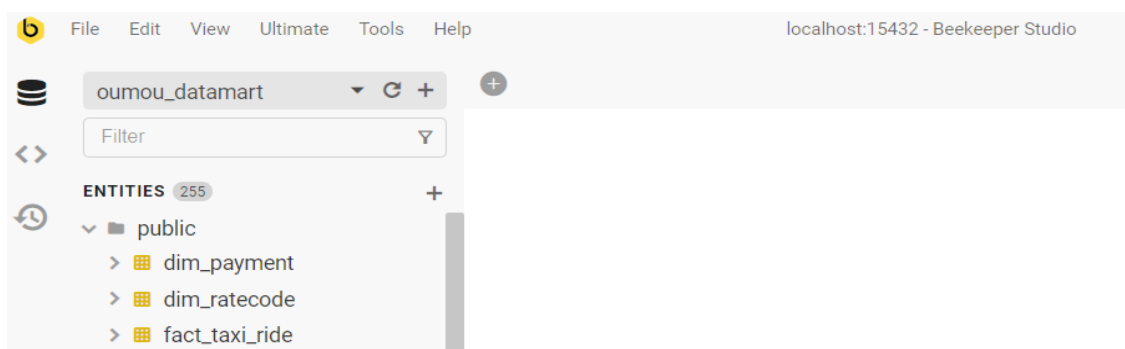
Il faut noter que nous avons d'abord créé une nouvelle base de données nommée **oumou\_datamart** avant de créer nos tables (dimensions et fait).

Nous avons jugé nécessaire que la clé primaire de la table de faits "**fact\_taxi\_ride**" sera composée de trois colonnes :

- **vendorid** : Identifiant du fournisseur de taxi.
- **tpep\_pickup\_datetime** : Date et heure de prise en charge du taxi.
- **tpep\_dropoff\_datetime** : Date et heure de dépose du taxi.

Ces trois colonnes combinées forment une clé primaire composite qui garantit l'unicité de chaque enregistrement dans la table de faits. Cela signifie qu'aucune combinaison de valeurs dans ces colonnes ne se répétera dans la table, ce qui permet d'identifier de manière unique chaque trajet en taxi enregistré dans la base de données.

## Les tables créées sur Beekeeper :



**NB** : Le fichier **creation.sql** sera en jointure sur le fichier Github.



## 4. Insertion.sql

```
*****INSERTIONS DES TABLES DIMENSIONS ET LA TABLE DE FAIT*****

-- Connexion à oumou_warehouse
CREATE EXTENSION IF NOT EXISTS dblink;
SELECT dblink_connect('oumou_warehouse_conn', 'dbname=oumou_warehouse');

-- Récupération des données de la table
CREATE TEMP TABLE temp_data AS
SELECT *
FROM dblink('oumou_warehouse_conn', 'SELECT * FROM oumou_raw')
AS t(vendorid INT, tpep_pickup_datetime TIMESTAMP (6), tpep_dropoff_datetime TIMESTAMP(6), passenger_count FLOAT, trip_distance FLOAT, ratecodeid INT,
    store_and_fwd_flag TEXT, pulocationid INT, dolocationid INT, payment_type INT, 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);

-- Affichage des données récupérées
SELECT * FROM temp_data limit 10;

-- Fonction de mappage pour la colonne paymentdescription
CREATE OR REPLACE FUNCTION map_payment_description(payment_type INT)
RETURNS TEXT AS $$
BEGIN
    CASE payment_type
        WHEN 1 THEN RETURN 'Carte de crédit';
        WHEN 2 THEN RETURN 'Espèces';
        WHEN 3 THEN RETURN 'Gratuit';
        WHEN 4 THEN RETURN 'Contestation';
        WHEN 5 THEN RETURN 'Inconnu';
        WHEN 6 THEN RETURN 'Trajet annulé';
        ELSE RETURN 'Non spécifié';
    END CASE;
END;
$$ LANGUAGE plpgsql;

-- Fonction de mappage pour la colonne ratecodedescription
CREATE OR REPLACE FUNCTION map_ratecode_description(ratecodeid INT)
RETURNS TEXT AS $$
BEGIN
    CASE ratecodeid
        WHEN 1 THEN RETURN 'Standard rate';
        WHEN 2 THEN RETURN 'JFK';
        WHEN 3 THEN RETURN 'Newark';
        WHEN 4 THEN RETURN 'Nassau or Westchester';
        WHEN 5 THEN RETURN 'Negotiated fare';
        WHEN 6 THEN RETURN 'Group ride';
        ELSE RETURN 'Unknown';
    END CASE;
END;
$$ LANGUAGE plpgsql;

-- Insertion des données dans la table dim_payment depuis temp_data
INSERT INTO dim_payment (payment_type, paymentdescription)
SELECT DISTINCT payment_type, map_payment_description(payment_type)
FROM temp_data
WHERE payment_type IS NOT NULL
ON CONFLICT (payment_type) DO UPDATE SET paymentdescription = EXCLUDED.paymentdescription;

-- Insertion des données dans la table dim_ratecode depuis temp_data
INSERT INTO dim_ratecode (ratecodeid, ratecodedescription)
SELECT DISTINCT ratecodeid, map_ratecode_description(ratecodeid)
FROM temp_data
WHERE ratecodeid IS NOT NULL
ON CONFLICT (ratecodeid) DO UPDATE SET ratecodedescription = EXCLUDED.ratecodedescription;

-- Insertion des données dans la table fact_taxi_ride depuis temp_data
INSERT INTO fact_taxi_ride (vendorid, tpep_pickup_datetime, tpep_dropoff_datetime, passenger_count,
    trip_distance, pulocationid, dolocationid, ratecodeid, store_and_fwd_flag, payment_type,
    fare_amount, extra, mta_tax, improvement_surcharge, tip_amount, tolls_amount, total_amount,
    congestion_surcharge, airport_fee)
SELECT DISTINCT vendorid, tpep_pickup_datetime, tpep_dropoff_datetime, passenger_count,
    trip_distance, pulocationid, dolocationid, ratecodeid, store_and_fwd_flag, payment_type,
    fare_amount, extra, mta_tax, improvement_surcharge, tip_amount, tolls_amount, total_amount,
    congestion_surcharge, airport_fee
FROM temp_data
ON CONFLICT DO NOTHING; -- Ignore les doublons lors de l'insertion

-- Fermeture de la connexion dblink
SELECT dblink_disconnect('oumou_warehouse_conn');
```

### Explication du script de insertion.sql:

Ce script SQL est conçu pour transférer les données de la table de faits et des tables de dimensions depuis la table source "oumou\_raw" vers notre entrepôt de données "oumou\_warehouse" en utilisant l'extension **dblink**. Tout d'abord, il se connecte à la base de données "oumou\_warehouse" via **dblink**, puis récupère les données de "oumou\_raw" et les stocke temporairement dans une table temporaire nommée "temp\_data".


Pour vérification, il affiche quelques lignes de données. Ensuite, il utilise des fonctions de mappage pour traduire les valeurs des colonnes "**ratecodedescription**" et "**paymentdescription**" à partir de leurs identifiants respectifs. Les données uniques sont ensuite insérées dans les tables de dimensions, en évitant les doublons. Enfin, les données distinctes de "temp\_data" sont insérées dans la table de faits, en évitant également les doublons. Une fois les opérations terminées, la connexion **dblink** est fermée pour libérer les ressources.

**NB :** Le fichier `insertion.sql` sera en jointure sur le fichier Github.

## 5. Verification des données insérées

Il est essentiel de vérifier si les données ont été insérées correctement. Pour ce faire, nous allons sur notre base de données `oumou_datamart` et visualisé nos données.

### Table dim\_payment



	payment_type	paymentdescription
1	0	Non spécifié
2	1	Carte de crédit
3	2	Espèces
4	3	Gratuit
5	4	Contestation

La table de `dim_payment` fournira une liste des différents modes de paiement disponibles, avec une description pour chaque mode.

### Table dim\_ratecode



	ratecodeid	ratecodedescription
1	1	Standard rate
2	2	JFK
3	3	Newark
4	4	Nassau or Westche...
5	5	Negotiated fare
6	6	Group ride
7	99	Unknown

La table `dim_ratecode` fournit une liste des différents codes tarifaires utilisés, avec une description pour chaque code.

## Table fact\_taxi\_ride

The screenshot shows a data tool interface. On the left, a sidebar lists entities under 'public', including 'fact\_taxi\_ride'. The main area displays a SQL query: 'select \* from fact\_taxi\_ride limit 10;'. Below the query, a table of results is shown with columns: vendorid, tpep\_pickup\_datetime, tpep\_dropoff\_datetime, passenger\_count, trip\_distance, pulocationid, dolocationid, ratecodeid, store\_and\_fwd..., and pa. The first 10 rows of data are visible.

	vendorid	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	pulocationid	dolocationid	ratecodeid	store_and_fwd...	pa
1	1	2023-11-01 07:08:53	2023-11-01 07:17:20	1	2.7	262	161	1	N	
2	2	2023-11-01 05:35:25	2023-11-01 05:47:03	1	2.96	236	100	1	N	
3	2	2023-11-01 02:28:55	2023-11-01 02:42:38	1	5.12	79	262	1	N	
4	2	2023-11-01 07:03:10	2023-11-01 08:22:07	1	17.23	132	161	2	N	
5	1	2023-11-01 10:35:41	2023-11-01 11:12:46	1	6	140	223	1	N	
6	1	2023-11-01 07:09:40	2023-11-01 07:19:26	1	1.2	74	152	1	N	
7	2	2023-11-01 09:48:34	2023-11-01 10:00:08	1	1.47	162	186	1	N	
8	1	2023-11-01 09:58:33	2023-11-01 10:08:58	2	1.7	249	186	1	N	
9	2	2023-11-01 09:44:23	2023-11-01 10:03:51	1	3.91	140	79	1	N	
10	2	2023-11-01 07:08:53	2023-11-01 07:17:20	1	2.7	262	161	1	N	

## IV. TP4: Visualisation de nos données à partir d'un outil de visualisation

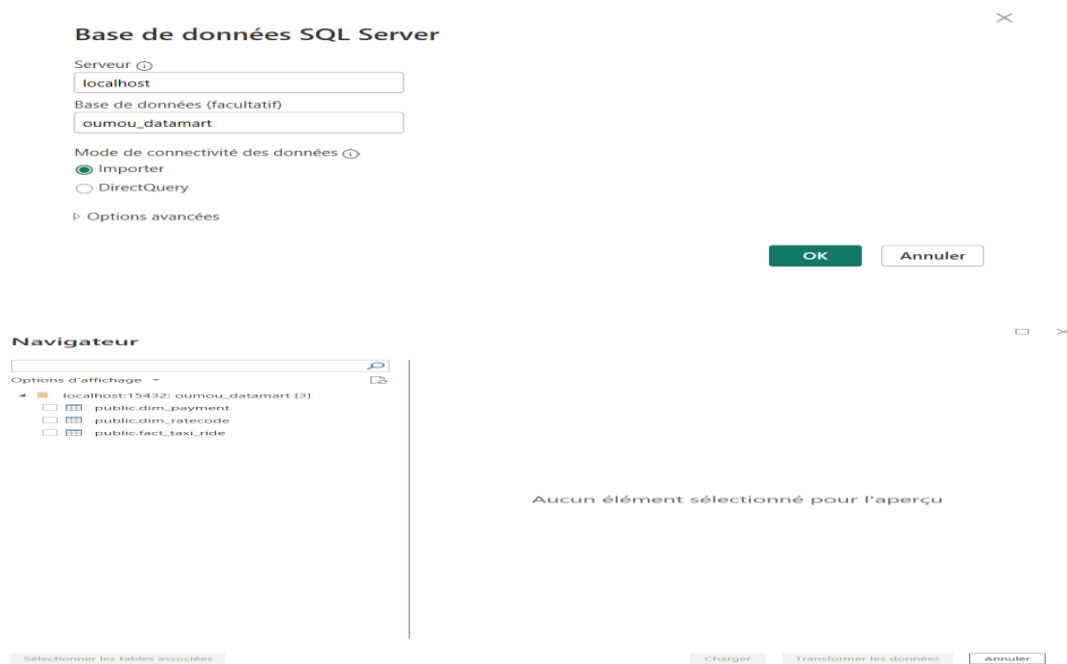
Dans le cadre du TP4, L'objectif est de nous permettre d'avoir une compréhension plus rapide, plus claire et plus profonde des informations contenues dans un ensemble de données. Pour ce faire, nous allons utiliser Power BI pour la visualisation de nos données.

### 1. Power BI

Power BI est un excellent outil pour la visualisation et l'analyse de données, nous allons importer notre base de données **oumou\_datamart** dans Power BI.

Voici les étapes à suivre :

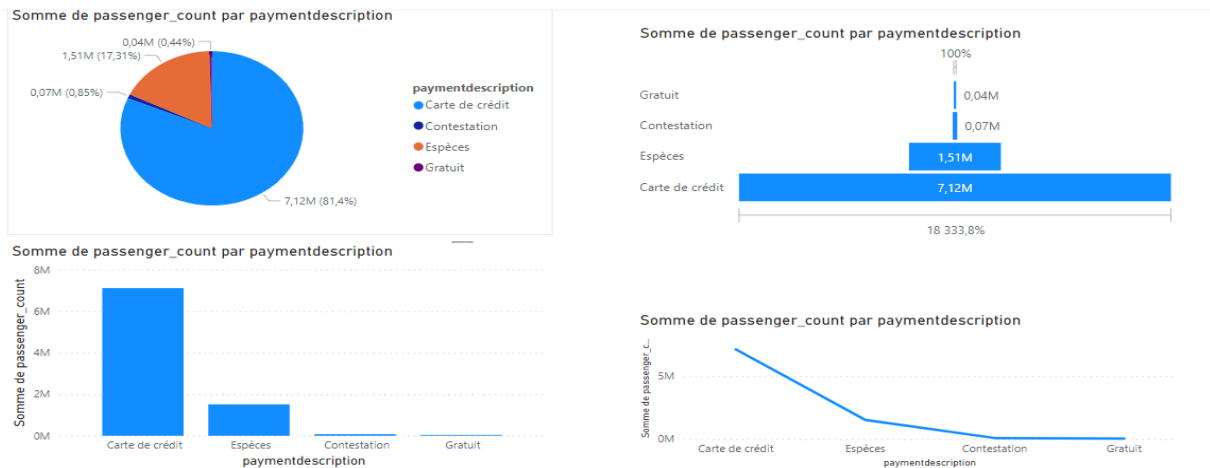




## 2. Réalisation des Dashboard

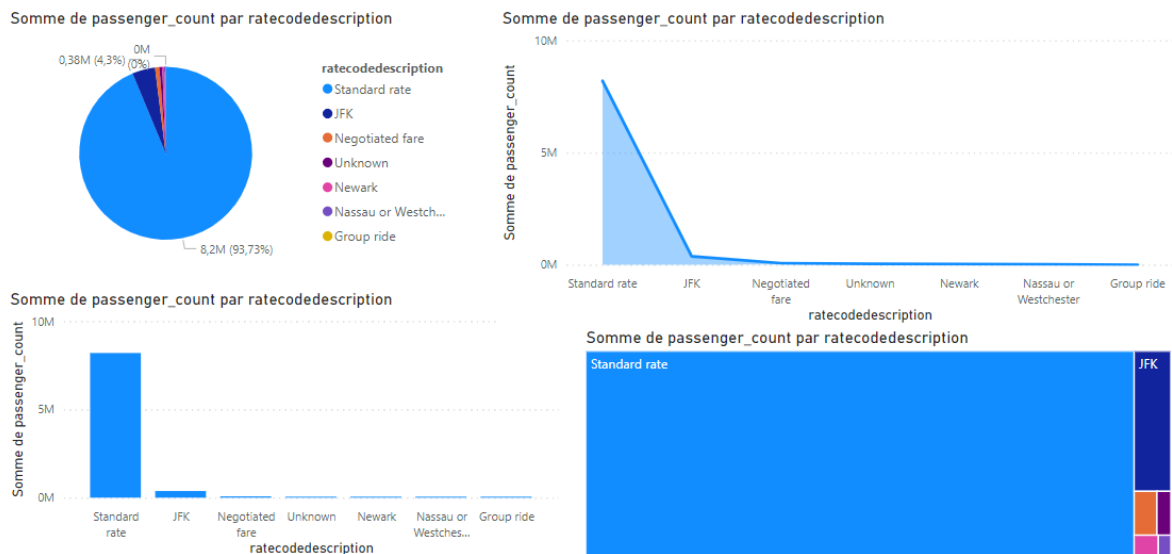
Nous allons présenter quelques Dashboard, mettant en valeur les faits issus des données sources.

### a. Le type de paiements utilisés par les passagers



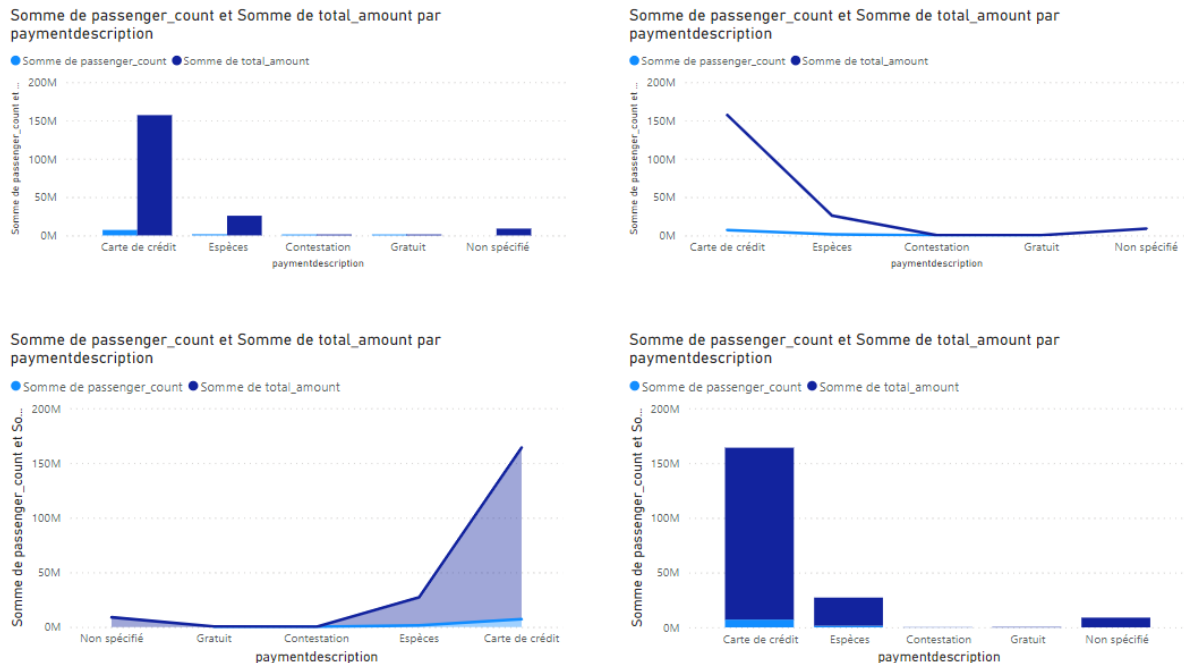
Cette visualisation montre la répartition des paiements effectués par les passagers lors de leurs voyages en taxi jaune et nous constatons une forte utilisation de **la carte de crédit** comme mode de paiement dans les voyages cela suggère un changement dans les habitudes de paiement des passagers et souligne l'importance pour les entreprises de transport de répondre à ces préférences en offrant des options de paiement électronique sécurisées et pratiques.

## b. La répartition des voyages par type de tarification



Nous observons que **le tarif standard** est le type de tarification le plus couramment utilisé pour les voyages en taxi jaune. Cela pourrait indiquer que la plupart des passagers optent pour un **tarif standard** plutôt que des tarifs spéciaux comme les trajets vers les **aéroports (JFK)** ou les **tarifs négociés**.

## c. Le montant total facturé aux passagers



Cette visualisation présente le montant total facturé aux passagers pour l'ensemble des voyages en taxi jaune.

## V. TP 5 (optionnel)

Cette partie du TP vous servira d'introduction à l'orchestration des tâches d'un projet Big Data. C'est-à-dire de lancer des scripts python de manière totalement automatisée sur un interval définie.

Pour le moment, je vous demande de réaliser une dag qui permet de télécharger un parquet du dernier mois en vigueur (TP 1) et de le stocker vers Minio.

Une fois que vous avez compris le fonctionnement des dags, vous pouvez vous amuser à automatiser le TP 2 et 3 afin de rendre le TP 4 totalement autonome.

**Pour le TP 5, il faudra créer vous-même le répertoire suivant : Sinon vous risquez d'avoir des problèmes au lancement des conteneurs.**

```
|— airflow
| |— config    <- Configuration files related to the Airflow Instance
| |— dags      <- Folder that contains all the dags
| |— logs      <- Contains the logs of the previously dags run
| |— plugins   <- Should be empty : Contains all needed plugins to make the dag work
```