Nom : Bouhlel I1 Cybersécurité

Prénom : Malek Date : 11 / 04 / 2024

Compte rendu TP Architecture Décisionnelle

TP1

- Ecrivez deux programme Python dans le dossier src/data :
  - Le premier récupère tous les dataset de 2018 à 2023 Le deuxième récupère le dernier mois
  - Les données seront téléchargées et stockées dans minio

Solution :

Vu le volume conséquent des fichiers.parquet, il a étais convenu d'en retenir que les données de 2023

def grab_data() → None :

- La fonction grab_data() télécharge les données de voyages (trip data) du service de taxi jaune à partir d'une URL spécifique.
- Pour chaque mois de l'année 2023. Les fichiers parquet téléchargés sont sauvegardés localement dans le répertoire './data/raw/'.
- La fonction gère les cas où les fichiers existent déjà localement et affiche les erreurs potentielles rencontrées lors du téléchargement.
- À la fin, elle indique si le dataset est à jour (Sinon on télécharge le dernier mois), ou s'il a été mis à jour avec succès. Enfin, un message de succès est affiché.

def write_data_minio () → None :

- La fonction write_data_minio() interagit avec un serveur MinIO pour envoyer des fichiers locaux vers un bucket spécifié.
- Le client MinIO est configuré pour se connecter à un serveur local avec l'adresse "localhost:9000", en utilisant des informations d'identification d'accès prédéfinies (access_key="minio", secret_key="minio123").
- La fonction vérifie si le bucket spécifié ("data") existe déjà, le crée s'il n'existe pas.
- Ensuite, elle parcourt le répertoire local './data/raw/' à la recherche de fichiers, puis les envoie individuellement vers le bucket MinIO avec leur chemin relatif local comme nom d'objet. Un message est affiché à chaque fichier ajouté au bucket, et la fonction s'arrête une fois tous les fichiers traités.

TP2

- Récupérez les données que vous avez téléchargé en local.
- Fusionnez les données pour avoir une unique table.
- Chargez le résultat vers votre base de données.

Solution :

Vu la taille des données j'ai dû deviser chaque fichier parquet avant de le chargé dans la BD

def write_data() → None :

- *La fonction write_data() traite les fichiers parquet situés dans le répertoire './data/raw/' en vue de les écrire dans une base de données PostgreSQL.*
- *Elle utilise la fonction clean_column_name() pour normaliser les noms de colonnes.*
- *La fonction lit chaque fichier parquet, affiche des informations telles que le nombre de lignes et la taille du fichier, puis divise le DataFrame en chunks (morceaux) pour les écrire par blocs dans la base de données.*
- *Si une erreur se produit lors de l'écriture, un message d'erreur est affiché. À la fin du processus, un message indique le succès de l'opération et affiche le nombre total de lignes traitées, ainsi que la durée totale du processus.*
- *Remarque : Une partie du code liée à la lecture des données depuis Minio vers PostgreSQL est actuellement commentée, laissant la fonction écriture locale comme méthode principale.*

def write_data_postgres(dataframe: pd.DataFrame) → Bool :

- La fonction write_data_postgres(dataframe: pd.DataFrame) enregistre un DataFrame dans une table PostgreSQL spécifiée.
- Les paramètres de configuration de la base de données sont définis dans le dictionnaire db_config.
- La fonction utilise SQLAlchemy pour créer une connexion au moteur PostgreSQL en utilisant les informations de configuration.
- Ensuite, elle envoie le DataFrame vers la table spécifiée dans la base de données.
- Si une erreur se produit lors de la connexion ou de l'opération d'écriture, un message d'erreur est affiché, et la fonction retourne False. Sinon, elle retourne True pour indiquer le succès de l'opération.

VU BD (nyc_warehouse.nyc_raw) sur pgadmin4 :

Data Output

	vendorid integer	tpep_pickup_datetime timestamp without time zone	tpep_dropoff_datetime timestamp without time zone	passenger_count double precision	trip_distance double precision	ratecodeid double precision	store_and_fwd_flag text	pulocationid integer	dolocationid integer	payment_type bigint	fare_amount double precision	extra double precision	mta_tax double pre
1	2	2023-03-07 08:15:45	2023-03-07 08:24:09	1	1.31	1	N	170	90	1	10	0	
2	2	2023-03-07 08:35:18	2023-03-07 08:52:30	2	1.37	1	N	186	233	1	15.6	0	
3	2	2023-03-07 08:56:03	2023-03-07 09:10:00	1	0.98	1	N	233	161	1	12.8	0	
4	2	2023-03-07 08:31:56	2023-03-07 08:43:06	1	1.11	1	N	229	163	1	11.4	0	
5	2	2023-03-07 08:45:25	2023-03-07 08:57:48	1	2.05	1	N	163	238	1	14.2	0	
6	2	2023-03-07 08:41:02	2023-03-07 08:54:09	1	1.29	1	N	234	246	1	12.8	0	
7	2	2023-03-07 08:17:21	2023-03-07 08:41:10	1	2.87	1	N	68	161	1	21.2	0	
8	1	2023-03-07 08:09:36	2023-03-07 08:21:25	1	1.9	1	N	142	236	2	13.5	2.5	
9	1	2023-03-07 08:25:40	2023-03-07 08:47:29	1	5.3	1	N	236	79	1	26.1	2.5	
10	1	2023-03-07 08:50:36	2023-03-07 09:14:46	1	3.2	1	N	79	161	1	22.6	2.5	
11	2	2023-03-07 08:30:38	2023-03-07 08:36:27	1	1.74	1	N	140	229	1	9.3	0	
12	2	2023-03-07 08:38:33	2023-03-07 08:50:32	1	1.59	1	N	229	100	2	12.1	0	
13	2	2023-03-07 08:53:00	2023-03-07 09:06:32	1	0.93	1	N	186	170	1	12.1	0	
14	1	2023-03-07 08:06:56	2023-03-07 08:21:13	1	1.8	1	N	142	233	2	12.8	2.5	
15	1	2023-03-07 08:45:02	2023-03-07 09:01:09	1	4.4	1	N	107	261	2	20.5	2.5	
16	2	2023-03-07 08:28:33	2023-03-07 08:34:14	4	0.64	1	N	162	161	2	7.2	0	
17	2	2023-03-07 08:39:17	2023-03-07 08:52:41	5	1.49	1	N	230	186	1	13.5	0	

TP3

- Concevez les KPI que vous souhaitez restituer lors de la phase de visualisation.
- Concevez un modèle en flocon pour les données finaux qui servira au Dashboard. Stockez les résultats de votre travail dans une DBMS OLAP

Solution :

Les KPI : dédié au Datamart. Nombre total de courses : Mesure la quantité totale de courses enregistrées dans la table de faits (FactTable).

- Revenu total : Calcule le revenu total généré par l'ensemble des courses en utilisant la colonne total_amount de la table de faits.
- Distance moyenne par course : Utilise la colonne trip_distance dans la table de faits pour calculer la distance moyenne parcourue par course.
- Durée moyenne de la course : Si la durée de la course est disponible dans les données, vous pouvez calculer la durée moyenne des courses.
- Nombre de courses par méthode de paiement : Analyse du nombre de courses par méthode de paiement à partir de la table de faits.
- Revenu par méthode de tarification : Utilise la colonne ratecodeid pour analyser le revenu généré par chaque méthode de tarification.
- Nombre de courses par fournisseur (vendor) : Analyse du nombre de courses par fournisseur en utilisant la colonne vendorid de la table de faits.
- Nombre de courses par emplacement (location) : Analyse du nombre de courses par emplacement en utilisant les colonnes pulocationid et dolocationid de la table de faits.

Le Modèle en Flocon :

1.DimensionTemporelle :

- pickup_datetime : La date et l'heure du début de la course.
- dropoff_datetime : La date et l'heure de la fin de la course.
- Cette table stocke les informations temporelles liées aux courses de taxi.

2.DimensionLieu :

- pulocationid : L'identifiant du lieu de prise en charge.
- dolocationid : L'identifiant du lieu de dépose.
- Cette table contient des informations sur les emplacements géographiques liés aux courses de taxi. 3.DimensionFournisseur :
- vendorid : L'identifiant du fournisseur de taxi.
- Cette table stocke des informations sur les fournisseurs de services de taxi.

4.DimensionTarification :

- ratecodeid : L'identifiant de la tarification utilisée pour la course.
- Cette table stocke des informations sur les différentes tarifications utilisées.

5.DimensionPaiement :

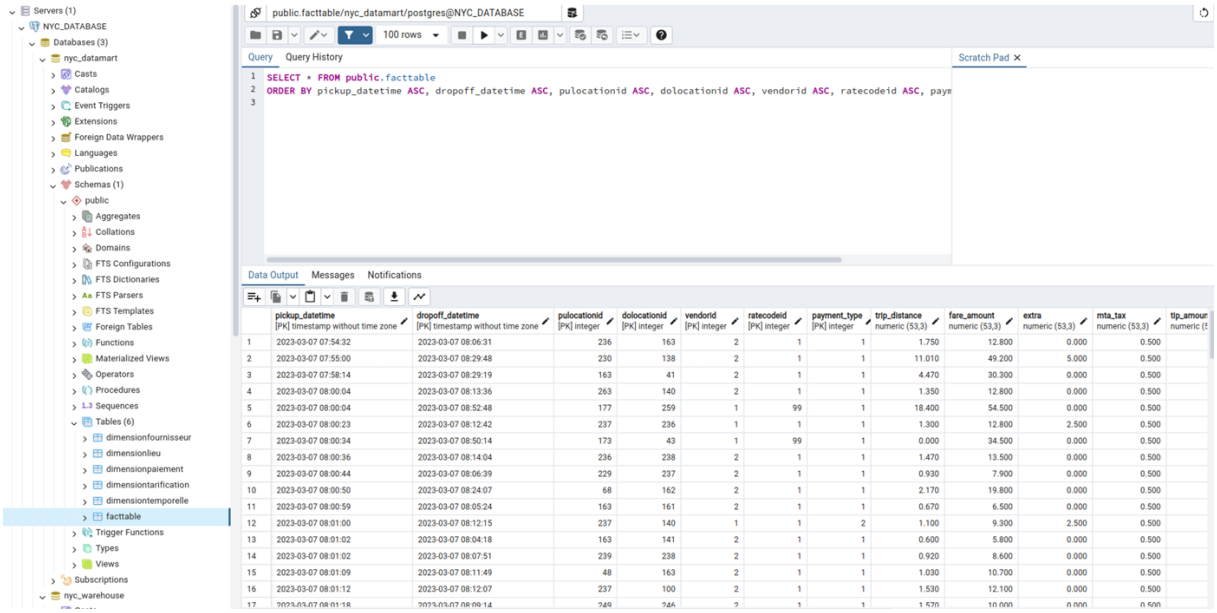
- payment_type : Le type de paiement utilisé pour la course.
- Cette table stocke des informations sur les types de paiement.

6.FactTable :

- pickup_datetime : Clé étrangère vers la table DimensionTemporelle.
- dropoff_datetime : Clé étrangère vers la table DimensionTemporelle.
- pulocationid : Clé étrangère vers la table DimensionLieu.
- dolocationid : Clé étrangère vers la table DimensionLieu.
- vendorid : Clé étrangère vers la table DimensionFournisseur.
- ratecodeid : Clé étrangère vers la table DimensionTarification.
- payment_type : Clé étrangère vers la table DimensionPaiement.
- trip_distance, fare_amount, extra, mta_tax, tip_amount, tolls_amount, improvement_surcharge, total_amount, congestion_surcharge, airport_fee : Mesures associées à la course.
- La table de faits stocke des mesures (quantités numériques) associées à chaque course, ainsi que des clés étrangères vers les tables de dimensions correspondantes.

Vu sur la DB nyc_datamart :

Vu les performances réduites de mon pc j'ai dû me restreindre une limite de 500 entrées pour simulation et débogage.



def create_table() → None :

- La fonction create_tables() établit une connexion à la base de données PostgreSQL ("nyc_datamart") en utilisant les informations d'identification spécifiées. Elle lit ensuite le script SQL de création de tables à partir du fichier "./src/data/create.sql" et exécute ce script pour créer les tables dans la base de données.
- En cas d'erreur lors de la connexion ou de l'exécution du script, un message d'erreur est affiché. Sinon, un message de succès est affiché indiquant que les tables ont été créées avec succès dans la base de données "nyc_datamart".

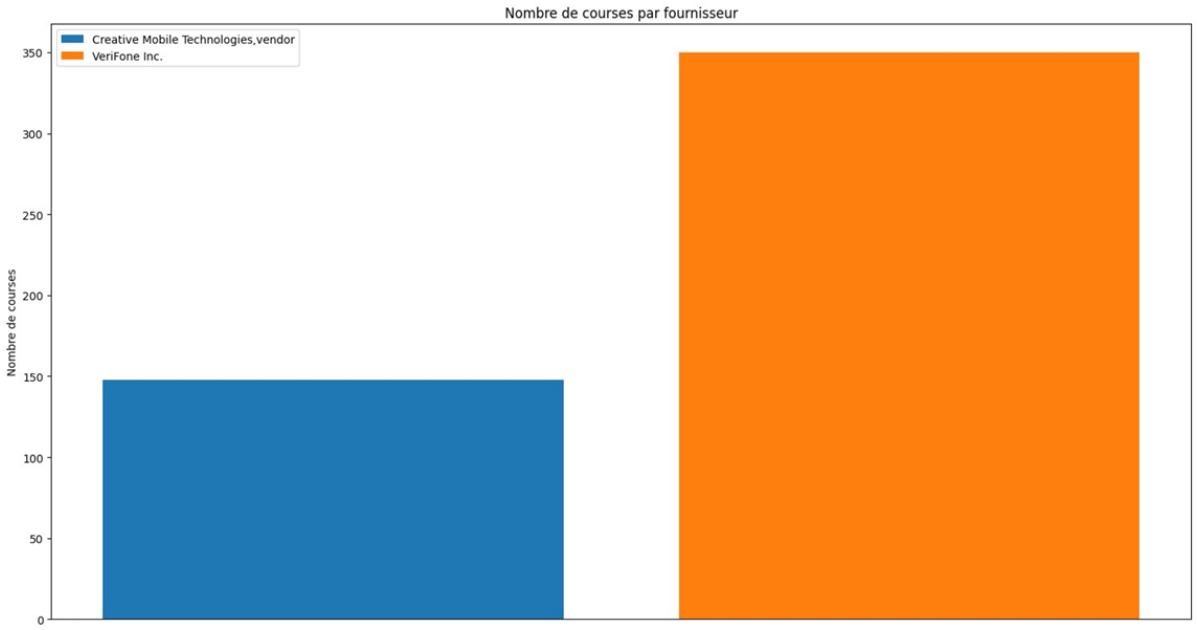
Def insert_data() → None :

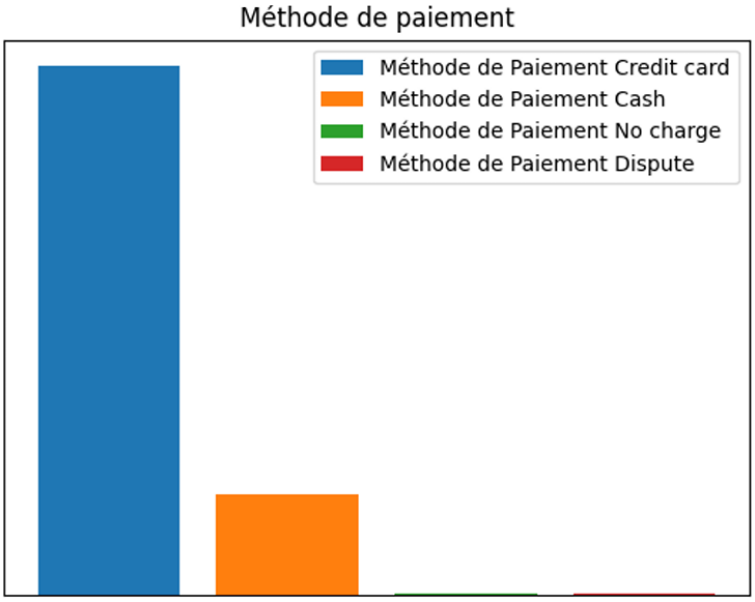
- La fonction insert_data() établit une connexion à la base de données PostgreSQL ("nyc_datamart") en utilisant les informations d'identification spécifiées. Elle lit ensuite le script SQL d'insertion de données à partir du fichier "./src/data/insert.sql" et exécute ce script pour insérer les données dans les tables de la base de données cible.
- En cas d'erreur lors de la connexion, de la lecture du script ou de l'exécution du script, un message d'erreur est affiché. Sinon, un message de succès est affiché indiquant que les données ont été insérées avec succès dans la base de données "nyc_datamart".

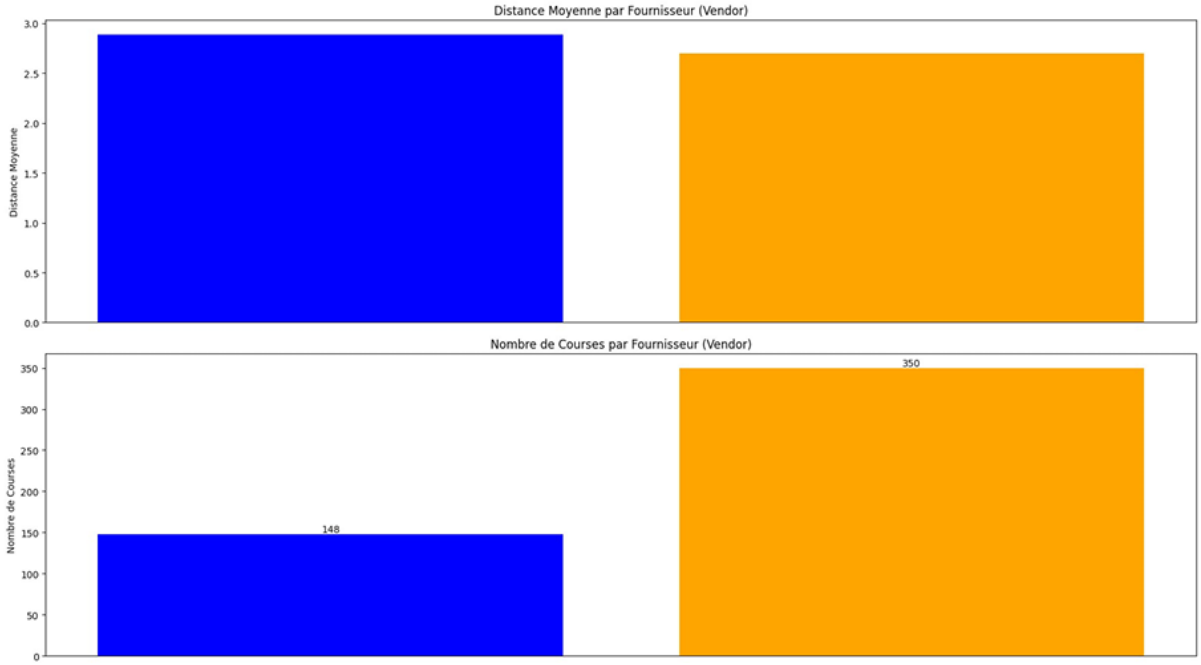
TP4

- Connectez-vous à votre DBMS Datamart.
- Faites une EDA (première partie de la visualisation)
- A la suite de votre EDA, vous devriez avoir plus de connaissance sur votre BDD, vous êtes normalement capable identifier les différentes KPI que vous souhaitez restituer.
- Avec votre outil de BI, concevez vos visualisations de données.
- Fusionner l'ensemble de vos visualisations de données vers un Dashboard. Pour cela, vous allez concevoir votre Dashboard.
- Assurez-vous que votre Dashboard se mette à jour automatiquement.

Malheureusement, sur Ubuntu, je ne peux pas utiliser POWER-Bi ni Tableau j'utilise par conséquent, la bib python matplotlib pour faire les tests de « viz » sur les tables de la BD nyc_datamart. J'ai créé un script viz_data.py mais les résultats sont loin d'être satisfaisant.




Méthode de paiement
Méthode de Paiement Credit card
Méthode de Paiement Cash
Méthode de Paiement No charge
Méthode de Paiement Dispute


Distance Moyenne par Fournisseur (Vendor)
Distance Moyenne
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Nombre de Courses par Fournisseur (Vendor)
Nombre de Courses
350
300
250
200
150
100
50
0
148
350