Hakim RAHHOU
Rapport du projet Data Engineer

Encadré par M. Kévin letup

2025- 2026

Test sur les deux requettes :

```sql
-- Nb d'emplacements disponibles de vélos dans une ville
SELECT dm.NAME, tmp.SUM_BICYCLE_DOCKS_AVAILABLE
FROM DIM_CITY dm INNER JOIN (
    SELECT CITY_ID, SUM(BICYCLE_DOCKS_AVAILABLE) AS
SUM_BICYCLE_DOCKS_AVAILABLE
    FROM FACT_STATION_STATEMENT
    WHERE CREATED_DATE = (SELECT MAX(CREATED_DATE) FROM CONSOLIDATE_STATION)
    GROUP BY CITY_ID
) tmp ON dm.ID = tmp.CITY_ID
WHERE lower(dm.NAME) in ('paris', 'nantes', 'vincennes', 'toulouse');
```

Results:

```
PS C:\Users\hakim\OneDrive\Desktop\IS5\Data Enigineering\de-projet\Data_Eng_Project> .\duckdb.exe data/duckdb/mobility_analysis.du
ckdb
>>
D SELECT dm.NAME, tmp.SUM_BICYCLE_DOCKS_AVAILABLE
┌  FROM DIM_CITY dm INNER JOIN (
┌      SELECT CITY_ID, SUM(BICYCLE_DOCKS_AVAILABLE) AS SUM_BICYCLE_DOCKS_AVAILABLE
┌      FROM FACT_STATION_STATEMENT
┌      WHERE CREATED_DATE = (SELECT MAX(CREATED_DATE) FROM CONSOLIDATE_STATION)
┌      GROUP BY CITY_ID
┌  ) tmp ON dm.ID = tmp.CITY_ID
┌  WHERE lower(dm.NAME) in ('paris', 'nantes', 'vincennes', 'toulouse');
```

NAME varchar	SUM_BICYCLE_DOCKS_AVAILABLE int128
Paris	18890
Vincennes	170
Nantes	1072

Requête 1 : Somme des bornes disponibles par ville

NAME	SUM_BICYCLE_DOCKS_AVAILABLE
Paris	18890
Vincennes	170
Nantes	1072

```sql
-- Nb de vélos disponibles en moyenne dans chaque station
SELECT ds.name, ds.code, ds.address, tmp.avg_dock_available
FROM DIM_STATION ds JOIN (
    SELECT station_id, AVG(BICYCLE_AVAILABLE) AS avg_dock_available
    FROM FACT_STATION_STATEMENT
    GROUP BY station_id
) AS tmp ON ds.id = tmp.station_id;
```

Results:

```
PS C:\Users\hakim\OneDrive\Desktop\IS5\Data Enigineering\de-projet\Data_Eng_Project> .\duckdb.exe data/duckdb/mobility_analysis.duckdb
>>
D SELECT ds.name, ds.code, ds.address, tmp.avg_dock_available
FROM DIM_STATION ds JOIN (
    SELECT station_id, AVG(BICYCLE_AVAILABLE) AS avg_dock_available
    FROM FACT_STATION_STATEMENT
    GROUP BY station_id
) AS tmp ON ds.id = tmp.station_id;
```

NAME varchar	CODE varchar	ADDRESS varchar	avg_dock_available double
Saint-Sulpice	6003	NULL	9.5
Place Nelson Mandela	25006	NULL	4.0
Jules Guesde - Pont du Port à l'Ang.	44017	NULL	15.0
Gare de Nogent-le-Perreux	41303	NULL	19.0
Boulangers - Cardinal Lemoine	5022	NULL	8.5
Commandant Schloesing - Pétrarque	16202	NULL	0.0
Quai de la Seine	19003	NULL	12.5
Sibelle - Alésia	14012	NULL	6.0
Place de Barcelone - Mirabeau	16030	NULL	43.0
Général De Gaulle - Alouette	41601	NULL	4.0
Marseille - Beaurepaire	10014	NULL	31.5
Auguste Cain - Jean Moulin	14136	NULL	3.0
Félix Faure - Sadi Carnot	33009	NULL	12.0
Place Violet	15033	NULL	12.5
18 juin 1940 - Buzenval	25005	NULL	10.0
Anatole France - Jean Lolive	35019	NULL	7.0
Gravelle - Route du Bac	12126	NULL	13.0
Pascal - Claude Bernard	5026	NULL	30.5
Labouret - Saint-Denis	27008	NULL	13.5
Capitaine Glarner - Gabriel Péri	34002	NULL	11.0
.	.	.	.
.	.	.	.
.	.	.	.
089-SAINT-AIGNAN	89	65, boulevard Saint-Aignan	2.0
064-SAINT CLÉMENT	64	24, rue du Maréchal Joffre	3.0
034-MÉDIATHEQUE	34	24, quai de la Fosse	6.0

🚲 Pipeline ETL – Analyse des stations de vélos (Paris + Nantes)

📌 Description du projet

Ce projet implémente un **pipeline ETL complet** permettant :

- L'ingestion de données open-data en temps réel
- La consolidation dans une base **DuckDB**
- La modélisation **dimensionnelle (Data Warehouse)**
- L'analyse des stations de vélos en libre-service

Le pipeline était initialement fourni pour **Paris**, et j'ai enrichi le projet avec :
✓ Les données temps réel de la ville de Nantes
✓ L'intégration multi-ville dans les tables de consolidation
✓ Les dimensions et faits pour Paris + Nantes
✓ Les requêtes analytiques demandées dans le sujet

🏗️ Architecture du projet

```
Data_Eng_Project/
│
├── src/
│   ├── data_ingestion.py
│   ├── data_consolidation.py
│   ├── data_agregation.py
│   ├── main.py
│
├── data/
│   ├── raw_data/
│   │   └── YYYY-MM-DD/
│   │       ├── paris_realtime_bicycle_data.json
│   │       ├── nantes_realtime_bicycle_data.json
│   ├── duckdb/
│       └── mobility_analysis.duckdb
│
├── data/sql_statements/
│   ├── create_consolidate_tables.sql
│   ├── create_agregate_tables.sql
│
└── README.md
```

⚙ Fonctionnement du pipeline

1 Ingestion

📄 Fichier : `data_ingestion.py`

- Récupère les données Velib Paris via API open-data
- Récupère les données Nantes via l'endpoint Naolib
- Stocke les données dans `/data/raw_data/YYYY-MM-DD/`

```
get_paris_realtime_bicycle_data()
get_nantes_realtime_bicycle_data()
```

---

2 Consolidation

📄 Fichier : `data_consolidation.py`

Création des tables :

- `CONSOLIDATE_CITY`
- `CONSOLIDATE_STATION`
- `CONSOLIDATE_STATION_STATEMENT`

Intégration multi-villes :
✓ Paris
✓ Nantes

Chaque station est enrichie :

- coordonnées GPS
- capacité
- statut
- code INSEE de la ville
- date d'ingestion

---

3 Agrégation (modèle dimensionnel)

📄 Fichier : `data_agregation.py`

Tables créées :

📌 Dimensions

- `DIM_CITY`
- `DIM_STATION`

📌 Fait

- `FACT_STATION_STATEMENT`
  (une ligne = disponibilité d'une station un jour donné)

▶ Exécution du pipeline

```
git clone <repo>
cd Data_Eng_Project

python -m venv .venv
source .venv/bin/activate  # Linux/Mac
# OU
.\.venv\Scripts\activate   # Windows

pip install -r requirements.txt

python src/main.py
```

📊 Requêtes finales (validations)

📌 1. Nombre total de places disponibles par ville

```
SELECT dm.NAME, tmp.SUM_BICYCLE_DOCKS_AVAILABLE
FROM DIM_CITY dm INNER JOIN (
    SELECT CITY_ID, SUM(BICYCLE_DOCKS_AVAILABLE) AS
SUM_BICYCLE_DOCKS_AVAILABLE
    FROM FACT_STATION_STATEMENT
    WHERE CREATED_DATE = (SELECT MAX(CREATED_DATE) FROM CONSOLIDATE_STATION)
    GROUP BY CITY_ID
) tmp ON dm.ID = tmp.CITY_ID
WHERE lower(dm.NAME) in ('paris', 'nantes', 'vincennes');
```

✓ Résultats obtenus

Ville	Places disponibles
Paris	18890
Nantes	1072
Vincennes	170

📌 2. Moyenne de vélos disponibles par station

```
SELECT ds.name, ds.code, ds.address, tmp.avg_dock_available
FROM DIM_STATION ds JOIN (
    SELECT station_id, AVG(BICYCLE_AVAILABLE) AS avg_dock_available
    FROM FACT_STATION_STATEMENT
    GROUP BY station_id
) AS tmp ON ds.id = tmp.station_id;
```

📡 Schéma du pipeline