Université Sultan Moulay Slimane
EST – Beni Mellal
Département : ITG
Pr. Siham BAKKOURI

Filières : DUT – IDIA & SIR – S3
Module : Big Data & BD NoSQL (M236)
Année universitaire : 2025/2026

Analyse des ventes avec Spark SQL

I. Introduction à Spark SQL

→ Qu'est-ce que Spark SQL ?

Spark SQL est un module de PySpark qui permet d'utiliser le langage SQL pour manipuler des DataFrames. → Il combine la simplicité du SQL et la puissance de Spark.

Exemple

```
spark.sql("SELECT region, SUM(prix) FROM ventes GROUP BY region")
```

➔ C'est exactement comme du SQL classique, sauf que les données sont distribuées sur plusieurs machines.

→ Les vues (views) dans Spark

Une **vue** (ou *View*) est comme une table virtuelle. Elle ne contient pas de données en elle-même, mais elle fait référence à un **DataFrame**.

- **Deux types de vues**

Type de vue	Commande	Portée
Temporaire	***createOrReplaceTempView("nom")***	Accessible seulement dans la session Spark actuelle
Globale	***createGlobalTempView("nom")***	Accessible depuis toutes les sessions Spark (dans le même cluster)

Exemple

```
df.createOrReplaceTempView("ventes")
spark.sql("SELECT * FROM ventes").show()
```

II. Mise en place de l'environnement

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import *

# Création du fichier CSV
data = """date,region,produit,quantite,prix
2025-10-19,Guelmim-Oued Noun,Laptop,10,10401
2025-10-26,Tanger-Tétouan-Al Hoceïma,Imprimante,2,2230
2025-10-15,Fès-Meknès,Casque,5,775
2025-10-02,Fès-Meknès,Imprimante,9,3129
2025-10-02,Marrakech-Safi,Laptop,1,8514
2025-10-30,Tanger-Tétouan-Al Hoceïma,USB,9,85
2025-10-19,Fès-Meknès,Laptop,4,9466
2025-10-25,Marrakech-Safi,Imprimante,9,2216
"""
with open("ventes.csv", "w") as f:
    f.write(data)
# Chargement du fichier CSV
df = spark.read.option("header", True).option("inferSchema", True).csv("ventes.csv")

df.show()
df.printSchema()
```

III. Rappel sur les requêtes SQL de base

```
# Création d'une vue temporaire à partir du DataFrame ventes
df.createOrReplaceTempView("ventes")

# 1. Afficher toutes les données
spark.sql("SELECT * FROM ventes").show()

# 2. Sélection de colonnes spécifiques
spark.sql("SELECT date, region, produit, prix FROM ventes").show()

# 3. Filtrage (WHERE)
spark.sql("SELECT * FROM ventes WHERE prix > 5000").show()

# 4. Tri (ORDER BY)
spark.sql("SELECT * FROM ventes ORDER BY prix DESC").show()

# 5. Supprimer les doublons
spark.sql("SELECT DISTINCT region FROM ventes").show()
```

IV. Calculs et agrégations

1. Somme totale par région

```
spark.sql("""
SELECT region, SUM(prix) AS total_prix
FROM ventes
GROUP BY region
""").show()
```

2. Moyenne des prix par produit

```
spark.sql("""
SELECT produit, AVG(prix) AS prix_moyen
FROM ventes
GROUP BY produit
""").show()
```

3. Total par région et produit

```
spark.sql("""
SELECT region, produit, SUM(prix) AS total
FROM ventes
GROUP BY region, produit
""").show()
```

4. Condition sur l'agrégation (HAVING)

```
spark.sql("""
SELECT region, SUM(prix) AS total
FROM ventes
GROUP BY region
HAVING total > 10000
""").show()
```

V. Création de nouvelles vues

On peut créer une nouvelle vue à partir d'une requête SQL :

```
# Création d'une nouvelle vue
spark.sql("""
CREATE OR REPLACE TEMP VIEW ventes_resumees AS
SELECT region, produit, SUM(quantite) AS qte_totale, SUM(prix) AS total_prix
FROM ventes
GROUP BY region, produit
""")

spark.sql("SELECT * FROM ventes_resumees").show()
```

➔ Cette vue ***ventes_resumees*** contient des données agrégées utiles pour d'autres analyses.

VI. Jointures entre vues

Créons un mini DataFrame des catégories :

```
data_cat = [
    ("Laptop", "Informatique"),
    ("Imprimante", "Bureau"),
    ("Casque", "Audio"),
    ("USB", "Accessoires")
]

df_cat = spark.createDataFrame(data_cat, ["produit", "categorie"])
df_cat.createOrReplaceTempView("categories")
```

➔ **Jointure entre ventes et categories :**

```
spark.sql("""
SELECT v.region, v.produit, c.categorie, v.quantite, v.prix
FROM ventes v
JOIN categories c ON v.produit = c.produit
""").show()
```

VII. Fonctions SQL avancées

1. Extraire l'année et le mois

```
spark.sql("""
SELECT
  date,
  SUBSTRING(date, 1, 4) AS annee,
  SUBSTRING(date, 6, 2) AS mois,
  produit,
  prix
FROM ventes
""").show()
```

- ***SUBSTRING(date, 1, 4)*** : Extrait les 4 premiers caractères de la colonne `date`, soit l'année (exemple : "2025").
- ***SUBSTRING(date, 6, 2) :*** Extrait le 6e et 7e caractère, soit le mois (exemple : "10").

➔ **analyser les ventes par année et par mois** pour détecter des tendances temporelles.

2. Ajouter des colonnes calculées

```
spark.sql("""
SELECT *,
  quantite * prix AS total_vente
FROM ventes
""").show()
```

3. Utiliser des fonctions d'agrégation multiples

```
spark.sql("""
SELECT
  region,
  MAX(prix) AS prix_max,
  MIN(prix) AS prix_min,
  ROUND(AVG(prix),2) AS prix_moyen
FROM ventes
GROUP BY region
""").show()
```

- ***MAX(prix)*** : Renvoie le **prix le plus élevé** pour chaque région.
- ***MIN(prix)*** : Renvoie le **prix le plus bas** pour chaque région.
- ***AVG(prix)*** : Calcule la **moyenne des prix** pour chaque région.
- ***ROUND(...,2)*** : Arrondit la moyenne à 2 décimales pour plus de lisibilité.
- ***GROUP BY region*** : Regroupe les lignes par région pour effectuer les agrégations.

→ **Synthèse**

- Spark SQL permet de manipuler les données comme en SQL classique.
- Les vues facilitent la réutilisation des requêtes complexes.
- Les fonctions d'agrégation et de jointure sont essentielles pour l'analyse.
- On peut combiner SQL + DataFrame API selon le besoin (flexibilité).

Exercice pratique : Analyse des ventes

- Dataset : ***ventes.csv***
- Colonnes : ***date, region, produit, quantite, prix***

1. Créez une vue temporaire ***ventes_view*** pour pouvoir exécuter toutes les requêtes en SQL.
2. Affichez le nombre total d'unités vendues par région (`SUM(quantite)`) et triez les résultats par ordre décroissant.
3. Pour chaque produit, affichez le prix maximum et identifiez le produit le plus cher du dataset.
4. Créez une colonne calculée `total_vente = quantite * prix` et affichez le chiffre d'affaires total par produit (`SUM(total_vente)`).
5. Analyse mensuelle des ventes
   - Extraire l'année et le mois à partir de la colonne `date`.
   - Affichez le chiffre d'affaires total par mois pour l'année 2025.
6. Affichez toutes les ventes où le chiffre d'affaires total dépasse 10 000 (`total_vente > 10000`).
7. Pour chaque région, identifiez le produit le plus vendu en quantité.
8. Pour chaque produit, calculez :
   - Le prix minimum (`MIN(prix)`),
   - Le prix maximum (`MAX(prix)`),
   - Le prix moyen (`AVG(prix)`).

9. Créez une nouvelle vue `ventes_analyse` contenant :
   - **`region`, `produit`, `quantite`, `prix`, `total_vente`, `annee`, `mois`**
   - Cette vue servira pour toutes les analyses suivantes.
10. Affichez les 3 produits générant le plus de chiffre d'affaires (`SUM(total_vente)`), triés par ordre décroissant.
11. Calculez le chiffre d'affaires total par région et identifiez la région avec le plus haut chiffre d'affaires.