Royaume du Maroc
Université Sultan Moulay Slimane
Ecole Supérieure de Technologie – Béni Mellal
Département Informatique et Techniques de Gestion

Spécialité : DUT en Sécurité Informatique et Réseaux (SIR)

COMPTE RENDU DE

Atelier N 03 :

Exploration et Analyse de Données de Transport avec PySpark

Réalisé par :

Hajar Rachid

lien :
https://colab.research.google.com/drive/1YVa_8f8tIw-cBa7O-T8Ofcfu3bhlW3hz?usp=sharing

Introduction

Cet atelier, intitulé "Exploration et Analyse de Données de Transport avec PySpark", avait pour objectif principal d'initier les étudiants aux techniques d'analyse de données massives en utilisant le framework Apache Spark via son API Python, PySpark. L'étude de cas portait sur l'analyse des données de trajets de l'entreprise TransData Maroc afin d'évaluer la performance de ses services de transport (bus, train, taxi, tram).

Les objectifs pédagogiques de cet atelier étaient clairement définis :

•Créer et manipuler un environnement Spark, notamment via Google Colab ou Docker.

•Charger, nettoyer et enrichir un jeu de données volumineux.

•Réaliser des analyses approfondies en utilisant les différentes abstractions de Spark : RDD, DataFrame et Spark SQL.

•Visualiser les résultats obtenus à l'aide de bibliothèques comme matplotlib et pandas.

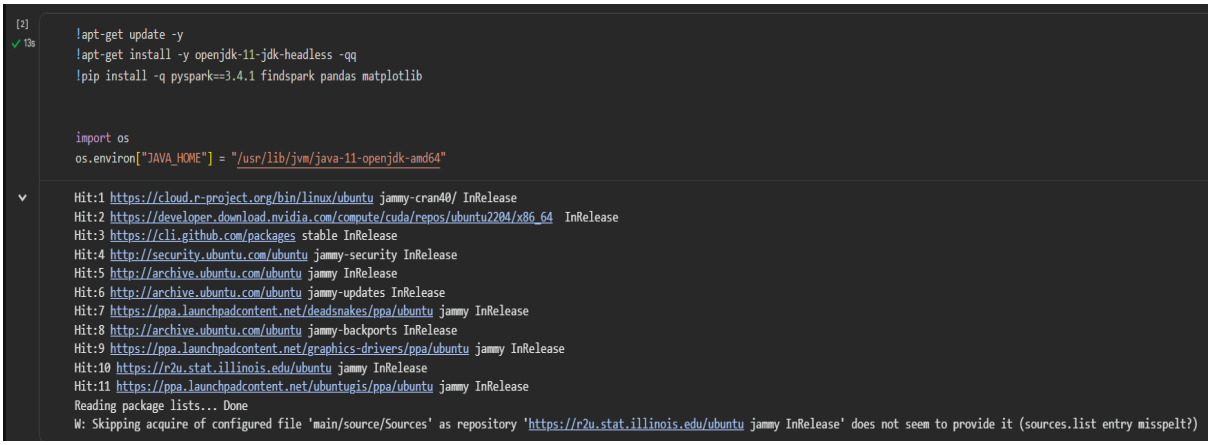
Déroulement de l'Atelier

L'atelier a été structuré en plusieurs parties progressives, allant de la mise en place de l'environnement à l'interprétation des résultats d'analyse.

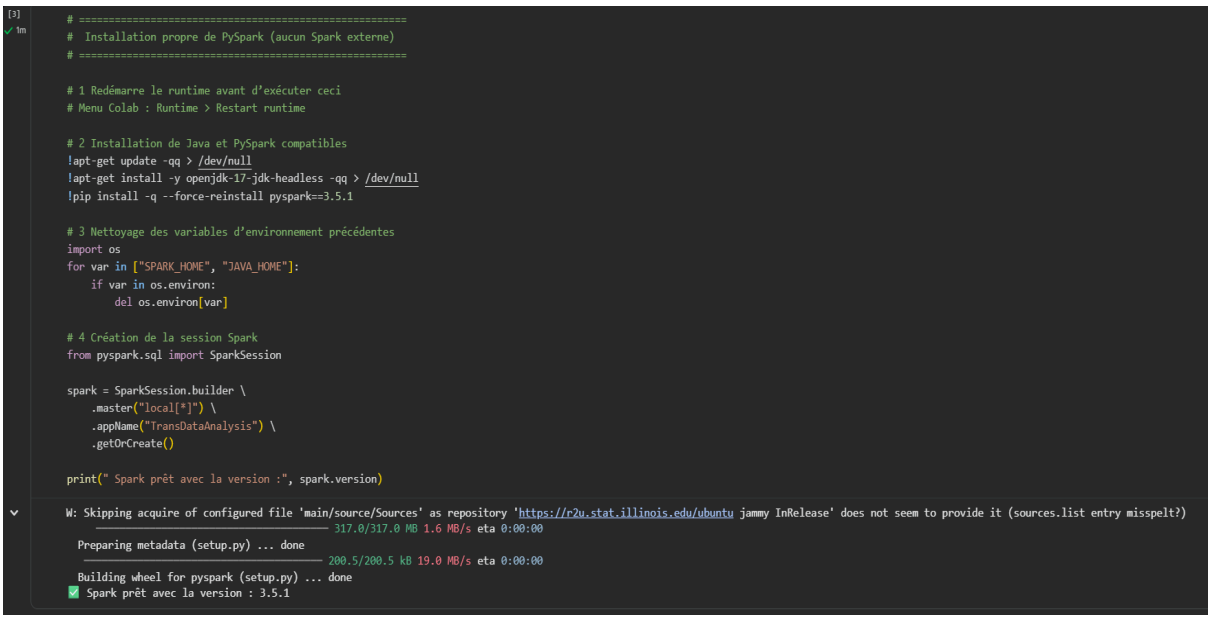
Partie 1 : Préparation de l'environnement :

Cette phase a couvert la mise en place de l'infrastructure nécessaire à l'exécution des programmes PySpark. Deux méthodes ont été présentées :

1. **Google Colab** : Une solution rapide et sans installation locale, nécessitant l'exécution de commandes pour installer et configurer PySpark directement dans le notebook.
2. **Exécution locale avec Docker** : Utilisation de l'image officielle jupyter/all-spark-notebook pour créer un environnement isolé et reproductible intégrant JupyterLab et Apache Spark.



```
[2]
✓ 13s
!apt-get update -y
!apt-get install -y openjdk-11-jdk-headless -qq
!pip install -q pyspark==3.4.1 findspark pandas matplotlib


import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-11-openjdk-amd64"

Hit:1 https://cloud.r-project.org/bin/linux/ubuntu jammy-cran40/ InRelease
Hit:2 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64  InRelease
Hit:3 https://cli.github.com/packages stable InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:6 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:7 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease
Hit:8 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:9 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy InRelease
Hit:10 https://r2u.stat.illinois.edu/ubuntu jammy InRelease
Hit:11 https://ppa.launchpadcontent.net/ubuntugis/ppa/ubuntu jammy InRelease
Reading package lists... Done
W: Skipping acquire of configured file 'main/source/Sources' as repository 'https://r2u.stat.illinois.edu/ubuntu jammy InRelease' does not seem to provide it (sources.list entry misspelt?)
```



```
[3]
✓ 1m
# ======================================================
#  Installation propre de PySpark (aucun Spark externe)
# ======================================================

# 1 Redémarre le runtime avant d'exécuter ceci
# Menu Colab : Runtime > Restart runtime

# 2 Installation de Java et PySpark compatibles
!apt-get update -qq > /dev/null
!apt-get install -y openjdk-17-jdk-headless -qq > /dev/null
!pip install -q --force-reinstall pyspark==3.5.1

# 3 Nettoyage des variables d'environnement précédentes
import os
for var in ["SPARK_HOME", "JAVA_HOME"]:
    if var in os.environ:
        del os.environ[var]

# 4 Création de la session Spark
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .master("local[*]") \
    .appName("TransDataAnalysis") \
    .getOrCreate()

print(" Spark prêt avec la version :", spark.version)

W: Skipping acquire of configured file 'main/source/Sources' as repository 'https://r2u.stat.illinois.edu/ubuntu jammy InRelease' does not seem to provide it (sources.list entry misspelt?)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 317.0/317.0 MB 1.6 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 200.5/200.5 kB 19.0 MB/s eta 0:00:00
  Building wheel for pyspark (setup.py) ... done
✅ Spark prêt avec la version : 3.5.1
```

Partie 2 : Chargement et Exploration :

Un jeu de données synthétique, trajets.csv, représentant 1000 trajets aléatoires, a été généré pour simuler les données réelles de TransData Maroc. Chaque trajet contenait des informations clés telles que la région de départ/arrivée, la distance, la durée, le nombre de passagers, le revenu généré et le moyen de transport.

L'étape a consisté à charger ce fichier CSV dans un DataFrame PySpark pour une première exploration, permettant d'afficher la structure des données (schema) et d'en visualiser quelques exemples.

```
[4]
✓ 11s
import pandas as pd
from random import randint, choice, uniform

regions = ["Casablanca", "Rabat", "Marrakech", "Fès", "Agadir"]
moyens = ["Bus", "Train", "Taxi", "Tram"]

data = [{
"trajet_id": i,
"region_depart": choice(regions),
"region_arrivee": choice(regions),
"distance_km": round(uniform(5, 600),1),
"duree_min": randint(10, 600),
"nb_passagers": randint(1, 50),
"revenu": round(uniform(50, 5000), 2),
"moyen_transport": choice(moyens)
} for i in range(1000)]

df = pd.DataFrame(data)
df.to_csv("trajets.csv", index=False)
df_spark = spark.read.csv("trajets.csv", header=True, inferSchema=True)
df_spark.printSchema()
df_spark.show(5)
```

```
root
 |-- trajet_id: integer (nullable = true)
 |-- region_depart: string (nullable = true)
 |-- region_arrivee: string (nullable = true)
 |-- distance_km: double (nullable = true)
 |-- duree_min: integer (nullable = true)
 |-- nb_passagers: integer (nullable = true)
 |-- revenu: double (nullable = true)
 |-- moyen_transport: string (nullable = true)

+---------+-------------+--------------+-----------+---------+------------+-------+--------------+
|trajet_id|region_depart|region_arrivee|distance_km|duree_min|nb_passagers| revenu|moyen_transport|
+---------+-------------+--------------+-----------+---------+------------+-------+--------------+
|        0|   Casablanca|     Marrakech|      562.0|      365|          48|2889.84|          Taxi|
|        1|    Marrakech|        Agadir|       10.4|      298|          40| 271.52|          Tram|
|        2|    Marrakech|         Rabat|      270.9|       65|          24| 832.28|           Bus|
|        3|   Casablanca|    Casablanca|      412.9|      535|           1| 4726.6|           Bus|
|        4|          Fès|           Fès|       95.5|      549|          34|2096.03|          Taxi|
+---------+-------------+--------------+-----------+---------+------------+-------+--------------+
only showing top 5 rows
```

Partie 3 : Nettoyage et Enrichissement :

Cette partie a été cruciale pour améliorer la qualité et la pertinence du jeu de données. Les opérations suivantes ont été effectuées :

• Filtrage : Élimination des trajets dont la distance était non positive.

• Ajout de colonnes calculées :

• revenu_par_km : Calcul du revenu par kilomètre.

• revenu_par_passager : Calcul du revenu par passager.

• Catégorisation : Création de la colonne catégorielle categorie_distance (Courte, Moyenne, Longue) basée sur la distance du trajet.

Ces manipulations ont permis de préparer le dataset pour des analyses plus fines.

```
[5]
✓ 0s
from pyspark.sql.functions import col, round as sround, when

# Filtrer les distances > 0
df_clean = df_spark.filter(col("distance_km") > 0)

# Calcul revenu par km
df_clean = df_clean.withColumn("revenu_par_km", sround(col("revenu") / col("distance_km"), 2))

# Calcul revenu par passager
df_clean = df_clean.withColumn("revenu_par_passager", sround(col("revenu") / col("nb_passagers"), 2))

# Catégorie de distance
df_clean = df_clean.withColumn(
    "categorie_distance",
    when(col("distance_km") <= 100, "Courte")
    .when(col("distance_km") <= 400, "Moyenne")
    .otherwise("Longue")
)

# Afficher les 5 premières lignes
df_clean.show(5)
```

```
+---------+-------------+-------------+-----------+--------+------------+-------+--------------+------------+------------------+-----------------+
|trajet_id|region_depart|region_arrivee|distance_km|duree_min|nb_passagers| revenu|moyen_transport|revenu_par_km|revenu_par_passager|categorie_distance|
+---------+-------------+-------------+-----------+--------+------------+-------+--------------+------------+------------------+-----------------+
|        0|   Casablanca|     Marrakech|      562.0|     365|          48|2889.84|          Taxi|        5.14|             60.21|           Longue|
|        1|    Marrakech|       Agadir|       10.4|     298|          40| 271.52|          Tram|       26.11|              6.79|           Courte|
|        2|    Marrakech|        Rabat|      270.9|      65|          24| 832.28|           Bus|        3.07|             34.68|          Moyenne|
|        3|   Casablanca|   Casablanca|      412.9|     535|           1| 4726.6|           Bus|       11.45|            4726.6|           Longue|
|        4|          Fès|          Fès|       95.5|     549|          34|2096.03|          Taxi|       21.95|             61.65|           Courte|
+---------+-------------+-------------+-----------+--------+------------+-------+--------------+------------+------------------+-----------------+
only showing top 5 rows
```

Partie 4 : Analyse via RDD :

L'atelier a introduit l'utilisation des Resilient Distributed Datasets (RDD), l'abstraction fondamentale de Spark. L'analyse principale réalisée avec les RDD était le calcul du revenu total par région de départ, suivi de l'affichage des 5 régions les plus rentables. Cette partie a souligné l'importance des RDD pour les opérations de bas niveau et les transformations complexes.

```
[6]
✓ 0s
    rdd = df_clean.rdd
    revenu_par_region = rdd.map(lambda x: (x["region_depart"], x["revenu"])) \
    .reduceByKey(lambda a,b: a+b) \
    .sortBy(lambda x: x[1], ascending=False)
    revenu_par_region.take(5)

    [('Marrakech', 545313.66),
     ('Agadir', 531939.9000000003),
     ('Rabat', 521583.6000000001),
     ('Casablanca', 459296.97000000003),
     ('Fès', 446370.99999999994)]
```

Partie 5 : Analyse avancée (SQL & DataFrame) :

Cette section a exploité la puissance des DataFrames et de Spark SQL pour des analyses agrégées et complexes :

• Calcul du revenu total par région de départ (réalisé également avec l'API DataFrame pour comparaison).

• Calcul des indicateurs moyens par moyen de transport : revenu moyen, distance moyenne et nombre de trajets.

• Utilisation de Spark SQL pour obtenir le revenu moyen et le nombre de trajets par categorie_distance.

```
[7]
✓ 3s
    from pyspark.sql.functions import avg, sum as ssum, count

    # Revenu total par région
    revenu_region = df_clean.groupBy("region_depart").agg(
        ssum("revenu").alias("revenu_total")
    )
    revenu_region.show()

    # Revenu moyen et distance moyenne par moyen de transport
    stats_transport = df_clean.groupBy("moyen_transport").agg(
        avg("revenu").alias("revenu_moyen"),
        avg("distance_km").alias("distance_moyenne"),
        count("*").alias("nb_trajets")
    )
    stats_transport.show()

    # Création d'une vue temporaire pour SQL
    df_clean.createOrReplaceTempView("trajets")

    # Revenu moyen et nombre de trajets par catégorie de distance
    spark.sql("""
    SELECT categorie_distance,
           AVG(revenu) AS revenu_moyen,
           COUNT(*) AS nb_trajets
    FROM trajets
    GROUP BY categorie_distance
    """).show()
```

```
+-------------+------------------+
|region_depart|      revenu_total|
+-------------+------------------+
|   Casablanca|459296.97000000003|
|          Fès|446370.99999999994|
|       Agadir| 531939.9000000003|
|        Rabat| 521583.6000000001|
|    Marrakech|         545313.66|
+-------------+------------------+

+---------------+------------------+------------------+----------+
|moyen_transport|      revenu_moyen|  distance_moyenne|nb_trajets|
+---------------+------------------+------------------+----------+
|           Taxi|2367.0500386100384| 305.7073359073356|       259|
|           Tram|2582.0808786610874|305.15020920502076|       239|
|            Bus|2583.6126612903226| 308.0661290322583|       248|
|          Train|2494.4326771653555|286.36181102362224|       254|
+---------------+------------------+------------------+----------+

+------------------+-----------------+----------+
|categorie_distance|     revenu_moyen|nb_trajets|
+------------------+-----------------+----------+
|           Longue|2447.705196374621|       331|
|           Courte|  2564.7480794702|       151|
|          Moyenne|  2523.2388996139|       518|
+------------------+-----------------+----------+
```

Partie 6 : Visualisation et Interprétation :

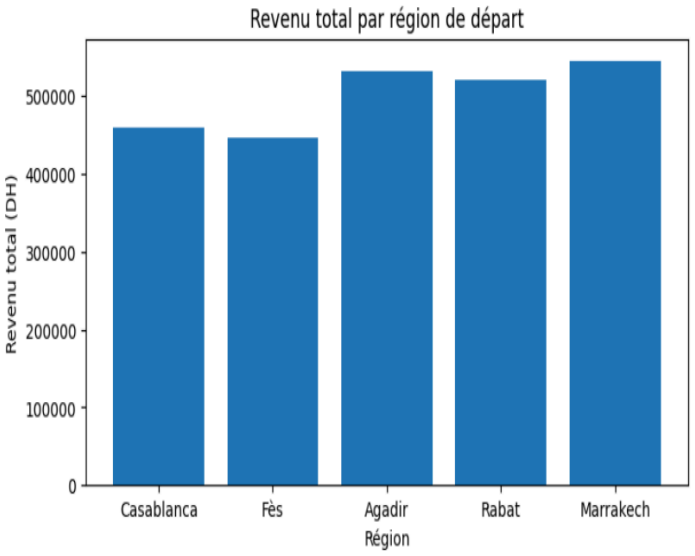
La dernière étape d'analyse a porté sur la visualisation des résultats pour en faciliter l'interprétation :

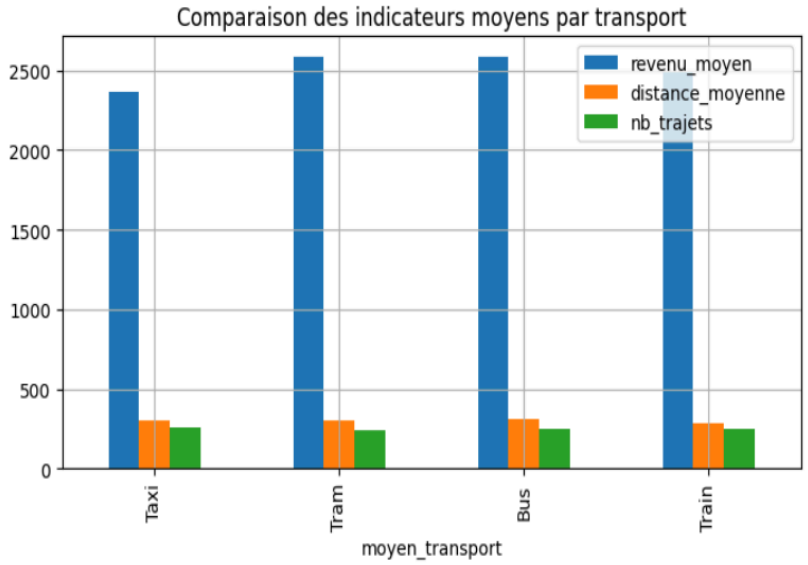
• Création d'un graphique en barres pour le revenu total par région de départ.

• Comparaison des indicateurs moyens (revenu, distance, nombre de trajets) par moyen de transport via un second graphique en barres.

L'objectif final était d'interpréter ces visualisations pour dégager des tendances, des insights et des recommandations pour TransData Maroc.

```
[8]
2s
#1. Conversion vers Pandas
pdf = revenu_region. toPandas ()
#2. Graphique en barres
import matplotlib.pyplot as plt

plt.figure(figsize=(8,4))
plt.bar(pdf["region_depart"], pdf["revenu_total"])
plt.title("Revenu total par région de départ")
plt.xlabel("Région")
plt.ylabel("Revenu total (DH)")
plt.show()
#3. Comparaison multi-indicateurs
pdf_stats = stats_transport. toPandas ()
pdf_stats.plot(x="moyen_transport", kind="bar", figsize=(8,4))
plt.title("Comparaison des indicateurs moyens par transport")
plt.grid(True)
plt.show()
```





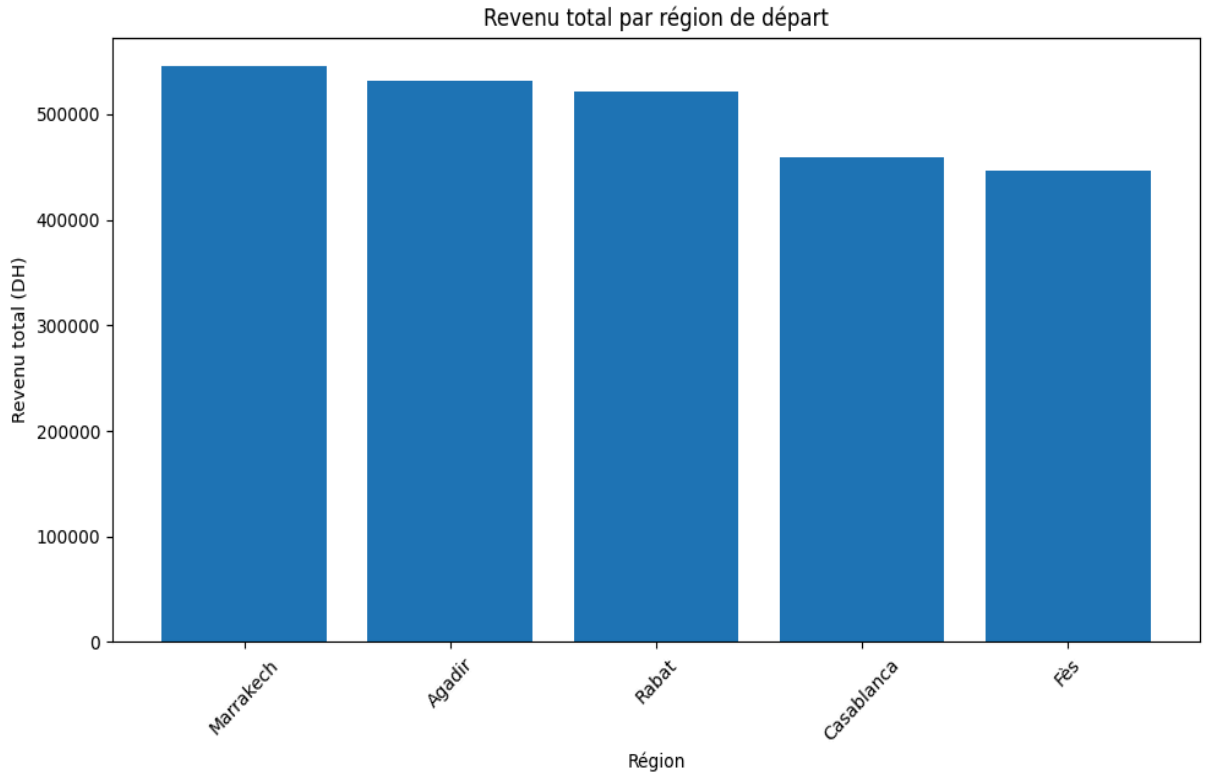
Exercice Pratique : Questions d'Analyse et de Visualisation :

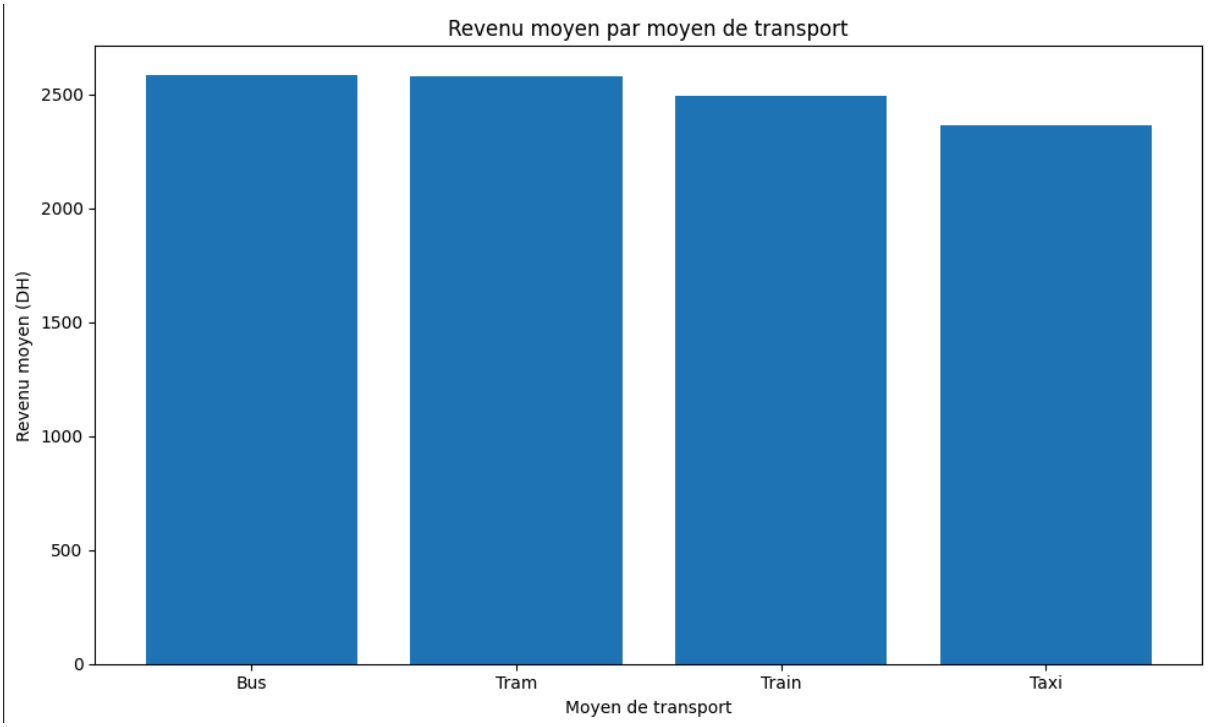
La partie pratique a consolidé les connaissances acquises à travers une série de questions d'analyse et de visualisation, exigeant la combinaison de PySpark SQL et de l'API DataFrame.

```
from pyspark.sql.functions import avg, sum as ssum, count, desc, var_pop, col, when, round as sround
import matplotlib.pyplot as plt

# 1. Revenu moyen par région de départ
revenu_moyen_depart = df_clean.groupBy("region_depart").agg(avg("revenu").alias("revenu_moyen"))
revenu_moyen_depart.show()

# 2. Nombre de trajets par région d'arrivée
trajets_par_arrivee = df_clean.groupBy("region_arrivee").agg(count("*").alias("nb_trajets"))
trajets_par_arrivee.show()

# 3. Statistiques par moyen de transport
stats_par_transport = df_clean.groupBy("moyen_transport").agg(
    ssum("revenu").alias("revenu_total"),
    avg("distance_km").alias("distance_moyenne")
)
stats_par_transport.show()

# 4. Région avec le maximum de trajets
region_max_trajets = df_clean.groupBy("region_depart").agg(count("*").alias("nb_trajets")).orderBy(desc("nb_trajets"))
region_max_trajets.show(1)

# 5. Top 3 des trajets les plus longs
top_3_distance = df_clean.orderBy(desc("distance_km"))
top_3_distance.show(3)

# 6. Revenu moyen par catégorie de distance
revenu_par_categorie = df_clean.groupBy("categorie_distance").agg(avg("revenu").alias("revenu_moyen"))
revenu_par_categorie.show()
```

```python
# 7. Moyenne et variance du revenu par région de départ
stats_revenu_region = df_clean.groupBy("region_depart").agg(
    avg("revenu").alias("moyenne_revenu"),
    var_pop("revenu").alias("variance_revenu")
)
stats_revenu_region.show()

# 8. Revenu moyen par moyen de transport (tri décroissant)
revenu_moyen_transport = df_clean.groupBy("moyen_transport").agg(avg("revenu").alias("revenu_moyen")).orderBy(desc("revenu_moyen"))
revenu_moyen_transport.show()

# 9. Pourcentage des trajets par moyen de transport
total_trajets = df_clean.count()
trajets_transport = df_clean.groupBy("moyen_transport").agg(count("*").alias("nb_trajets"))
trajets_transport = trajets_transport.withColumn("pourcentage", (col("nb_trajets") / total_trajets * 100))
trajets_transport.show()

# 10. Revenu total par région de départ (top 1)
revenu_total_region = df_clean.groupBy("region_depart").agg(ssum("revenu").alias("revenu_total")).orderBy(desc("revenu_total"))
revenu_total_region.show(1)

# 11. Visualisation du revenu total par région de départ
pdf_revenu_region = revenu_total_region.toPandas()
plt.figure(figsize=(10,6))
plt.bar(pdf_revenu_region["region_depart"], pdf_revenu_region["revenu_total"])
plt.title("Revenu total par région de départ")
plt.xlabel("Région")
plt.ylabel("Revenu total (DH)")
```

```python
# 12. Visualisation du revenu moyen par moyen de transport
pdf_revenu_transport = revenu_moyen_transport.toPandas()
plt.figure(figsize=(10,6))
plt.bar(pdf_revenu_transport["moyen_transport"], pdf_revenu_transport["revenu_moyen"])
plt.title("Revenu moyen par moyen de transport")
plt.xlabel("Moyen de transport")
plt.ylabel("Revenu moyen (DH)")
plt.tight_layout()
plt.show()
```





Conclusion

L'Atelier N° 03 a permis une immersion complète dans l'écosystème PySpark pour l'analyse de données de transport. En couvrant l'installation, le pré-traitement, l'analyse multiparadigmatique (RDD, DataFrame, SQL) et la visualisation, il a fourni une base solide pour la manipulation de Big Data. La progression de l'atelier, de la préparation des données à l'interprétation des résultats, a illustré un flux de travail typique en science des données, essentiel pour l'évaluation de la performance d'une entreprise comme TransData Maroc.

Ce travail pratique est fondamental pour maîtriser les outils et les concepts nécessaires à l'exploitation des données massives dans un contexte professionnel.