COMPTE RENDU TP3

MARIEM EJIWEN 23018

9 MAI 2025

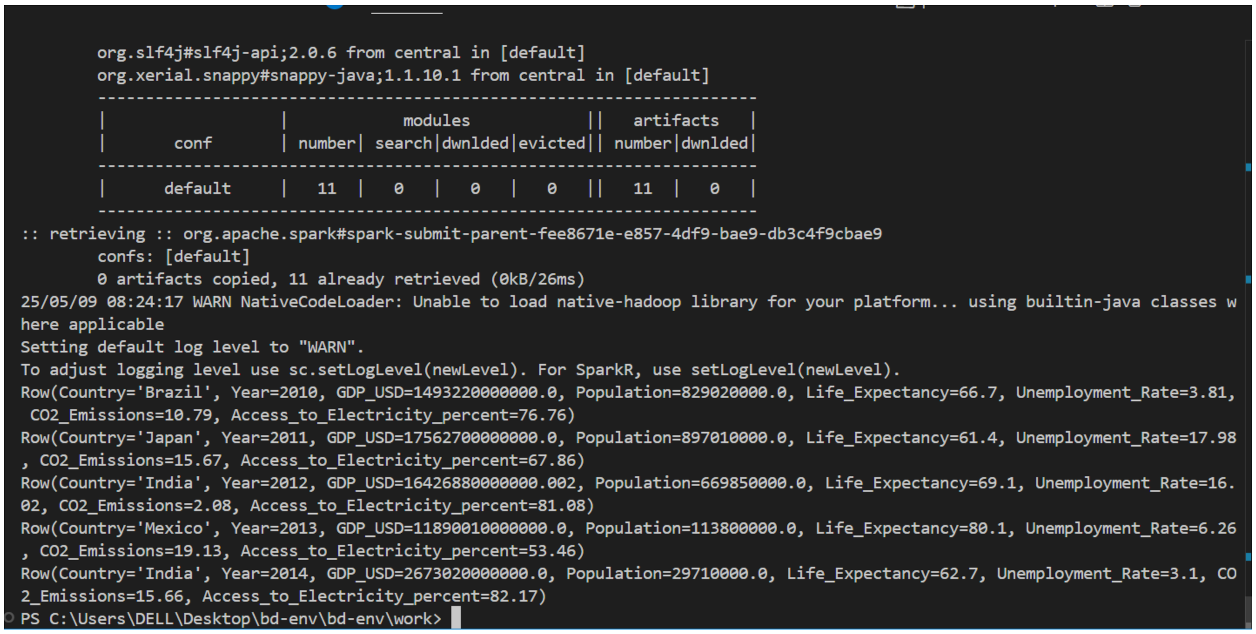
EXO1 :

question1:

```
from pyspark.sql import SparkSession


spark = SparkSession.builder.appName("WorldBank")\
    .config("spark.jars.packages",
"org.apache.spark:spark-sql-kafka-0-10_2.12:3.4.1") \
    .getOrCreate()
```

Question 2 : Charger le fichier CSV dans un DataFrame, le convertir en RDD **country_data**, puis afficher les 5 premières lignes

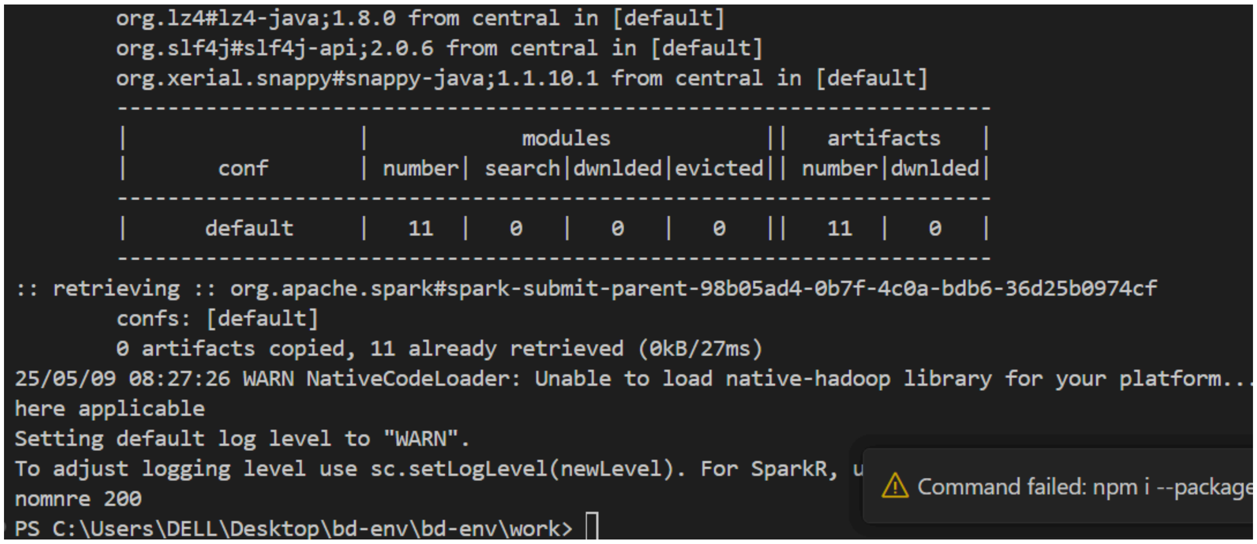
```
df = spark.read.csv("world_bank_dataset.csv", header=True,
inferSchema=True)
country_data = df.rdd
country_data.take(5)
for row in rows:
    print(row)
```



```
        org.slf4j#slf4j-api;2.0.6 from central in [default]
        org.xerial.snappy#snappy-java;1.1.10.1 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   11  |   0   |   0   |   0   ||   11  |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent-fee8671e-e857-4df9-bae9-db3c4f9cbae9
        confs: [default]
        0 artifacts copied, 11 already retrieved (0kB/26ms)
25/05/09 08:24:17 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes w
here applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Row(Country='Brazil', Year=2010, GDP_USD=1493220000000.0, Population=829020000.0, Life_Expectancy=66.7, Unemployment_Rate=3.81,
 CO2_Emissions=10.79, Access_to_Electricity_percent=76.76)
Row(Country='Japan', Year=2011, GDP_USD=17562700000000.0, Population=897010000.0, Life_Expectancy=61.4, Unemployment_Rate=17.98
, CO2_Emissions=15.67, Access_to_Electricity_percent=67.86)
Row(Country='India', Year=2012, GDP_USD=16426880000000.002, Population=669850000.0, Life_Expectancy=69.1, Unemployment_Rate=16.
02, CO2_Emissions=2.08, Access_to_Electricity_percent=81.08)
Row(Country='Mexico', Year=2013, GDP_USD=11890010000000.0, Population=113800000.0, Life_Expectancy=80.1, Unemployment_Rate=6.26
, CO2_Emissions=19.13, Access_to_Electricity_percent=53.46)
Row(Country='India', Year=2014, GDP_USD=2673020000000.0, Population=29710000.0, Life_Expectancy=62.7, Unemployment_Rate=3.1, CO
2_Emissions=15.66, Access_to_Electricity_percent=82.17)
PS C:\Users\DELL\Desktop\bd-env\bd-env\work>
```

Question 3 : Afficher le nombre total d'enregistrements dans le RDD

```
print("nomnre",country_data.count())
```

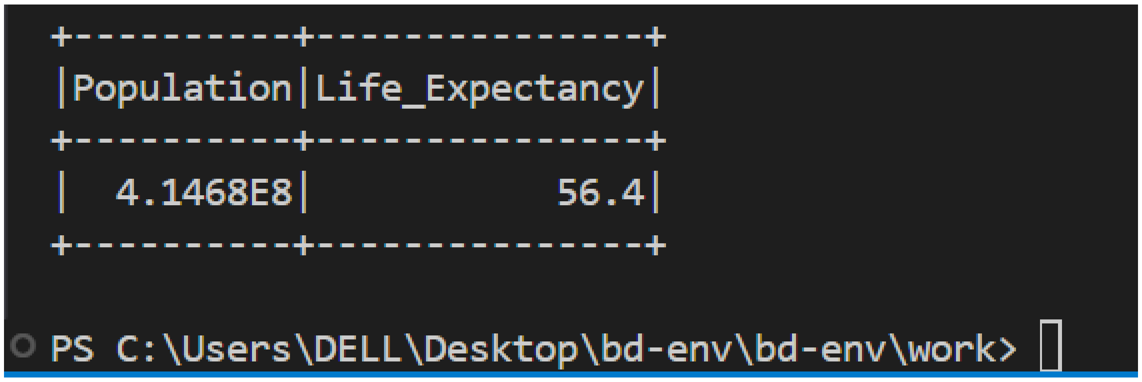


```
        org.lz4#lz4-java;1.8.0 from central in [default]
        org.slf4j#slf4j-api;2.0.6 from central in [default]
        org.xerial.snappy#snappy-java;1.1.10.1 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   11  |   0   |   0   |   0   ||   11  |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent-98b05ad4-0b7f-4c0a-bdb6-36d25b0974cf
        confs: [default]
        0 artifacts copied, 11 already retrieved (0kB/27ms)
25/05/09 08:27:26 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform..
here applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, u
nomnre 200
PS C:\Users\DELL\Desktop\bd-env\bd-env\work>
```

⚠ Command failed: npm i --package

Question 4 : Quelle est la population et l’espérance de vie du Canada en 2014 ?

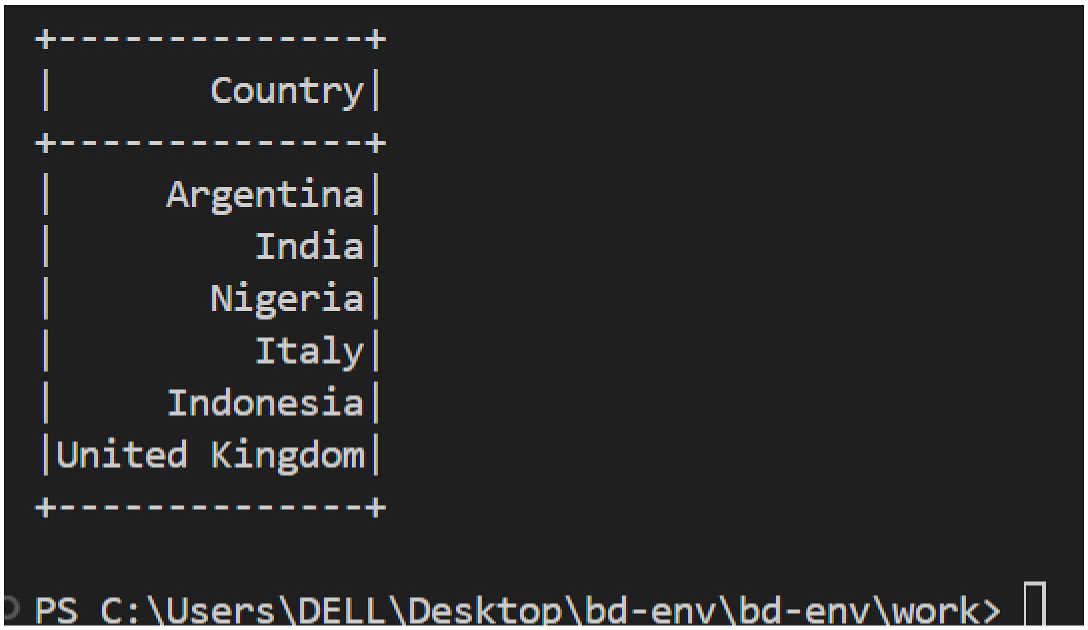
```
df.filter((df["Country"] == "Canada") & (df["Year"] == 2014)) \
  .select("Population", "Life_Expectancy") \
  .show()
```



```
+----------+---------------+
|Population|Life_Expectancy|
+----------+---------------+
|  4.1468E8|           56.4|
+----------+---------------+

PS C:\Users\DELL\Desktop\bd-env\bd-env\work>
```

Question 5 : Afficher le nom de tous les pays dont le PIB dépasse 10 000 000 000 000 USD en 2015

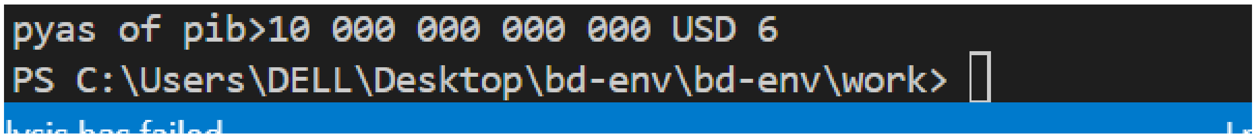
```
df.filter((df["Year"] == 2015) & (df["GDP_USD"] > 1e13)) \
  .select("Country") \
  .distinct() \
  .show()
```



```
+--------------+
|       Country|
+--------------+
|     Argentina|
|         India|
|       Nigeria|
|         Italy|
|     Indonesia|
|United Kingdom|
+--------------+

PS C:\Users\DELL\Desktop\bd-env\bd-env\work>
```

Question 6 : Trouver le nombre de pays dont le PIB dépasse 10 000 000 000 000 USD en 2015

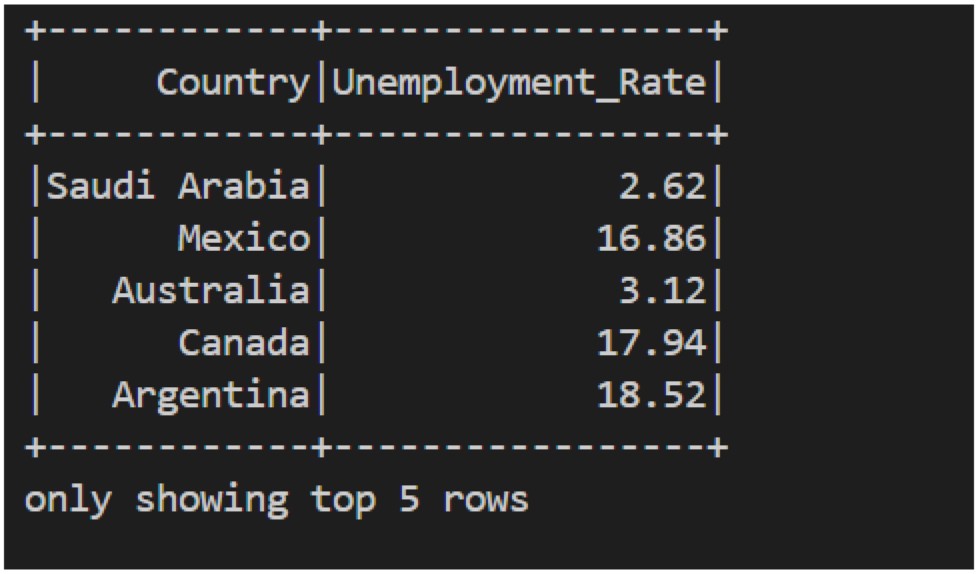
```
print("pyas of pib>10 000 000 000 000 USD",df.filter((df["Year"] == 2015)
& (df["GDP_USD"] > 1e13)) \

  .select("Country") \

  .distinct() \

  .count())
```



```
pyas of pib>10 000 000 000 000 USD 6
PS C:\Users\DELL\Desktop\bd-env\bd-env\work>
```

Question 7 : Afficher les 5 pays ayant le plus grand taux de chômage et leur taux

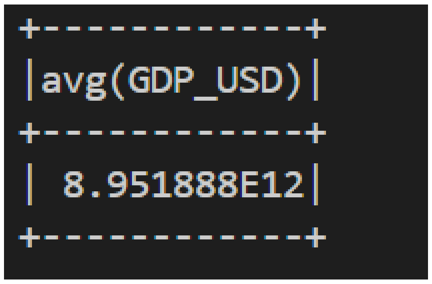
```
df.select("Country", "Unemployment_Rate") \
  .orderBy(df["Unemployment_Rate"].desc()) \
  .dropna(subset=["Unemployment_Rate"]) \
  .distinct() \
  .show(5)
```



```
+------------+-----------------+
|     Country|Unemployment_Rate|
+------------+-----------------+
|Saudi Arabia|             2.62|
|      Mexico|            16.86|
|   Australia|             3.12|
|      Canada|            17.94|
|   Argentina|            18.52|
+------------+-----------------+
only showing top 5 rows
```

Question 8 : Quel est le PIB moyen des pays en 2014 ?

```
df.filter(df["Year"] == 2014) \
  .select("GDP_USD") \
  .groupBy().avg() \
  .show()
```



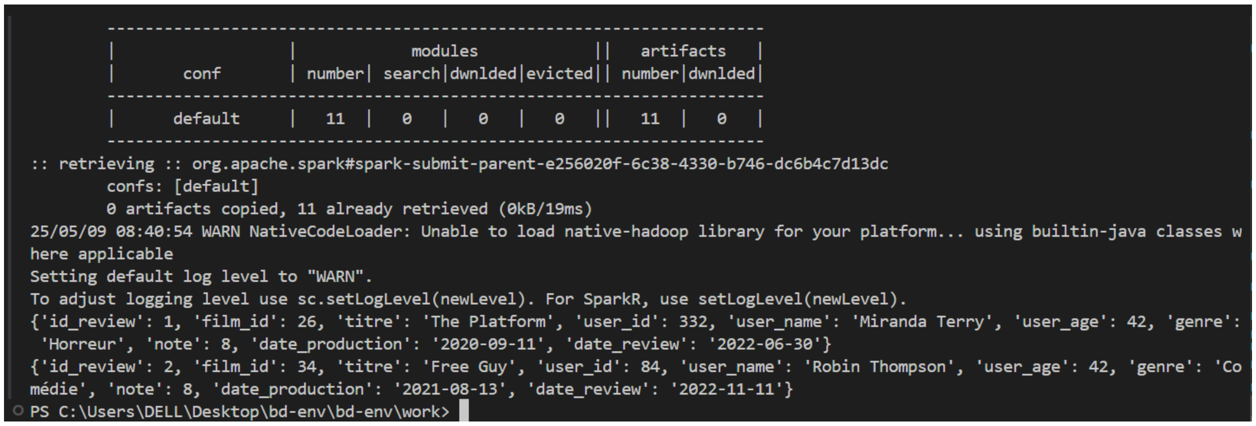
```
+------------+
|avg(GDP_USD)|
+------------+
| 8.951888E12|
+------------+
```

EXO2:

Question 1 :

Charger les données JSON du fichier reviews_nd.json dans un RDD et afficher les 2 premiers documents.

```
import json

from pyspark.sql import SparkSession


spark = SparkSession.builder.appName("ReviewsRDD")\

    .config("spark.jars.packages",
"org.apache.spark:spark-sql-kafka-0-10_2.12:3.4.1") \

    .getOrCreate()

rdd_raw = spark.sparkContext.textFile("reviews_nd.json")


reviews_rdd = rdd_raw.map(lambda line: json.loads(line))


for review in reviews_rdd.take(2):

    print(review)
```



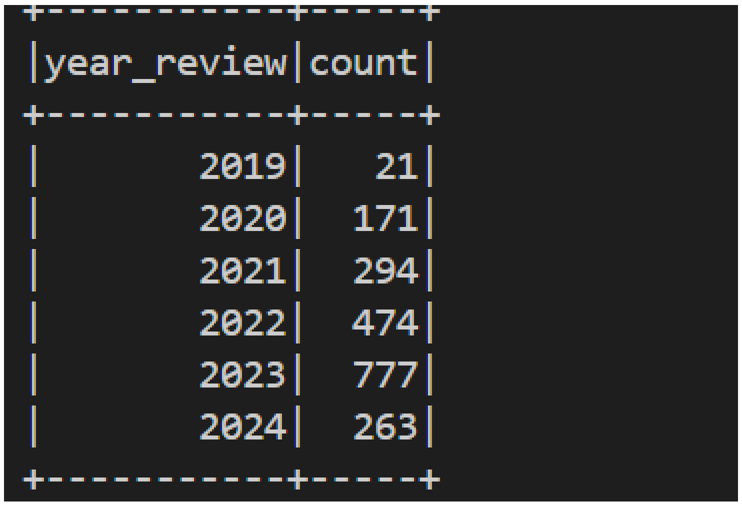
```
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   11  |   0   |   0   |   0   ||   11  |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent-e256020f-6c38-4330-b746-dc6b4c7d13dc
        confs: [default]
        0 artifacts copied, 11 already retrieved (0kB/19ms)
25/05/09 08:40:54 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes w
here applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
{'id_review': 1, 'film_id': 26, 'titre': 'The Platform', 'user_id': 332, 'user_name': 'Miranda Terry', 'user_age': 42, 'genre':
 'Horreur', 'note': 8, 'date_production': '2020-09-11', 'date_review': '2022-06-30'}
{'id_review': 2, 'film_id': 34, 'titre': 'Free Guy', 'user_id': 84, 'user_name': 'Robin Thompson', 'user_age': 42, 'genre': 'Co
médie', 'note': 8, 'date_production': '2021-08-13', 'date_review': '2022-11-11'}
PS C:\Users\DELL\Desktop\bd-env\bd-env\work>
```

Question 2 :

Calculer le nombre de films notés chaque année, ordonné par **année croissante.**

Cela se base sur le champ "date_review" . Il faut extraire l'année, grouper, puis compter.

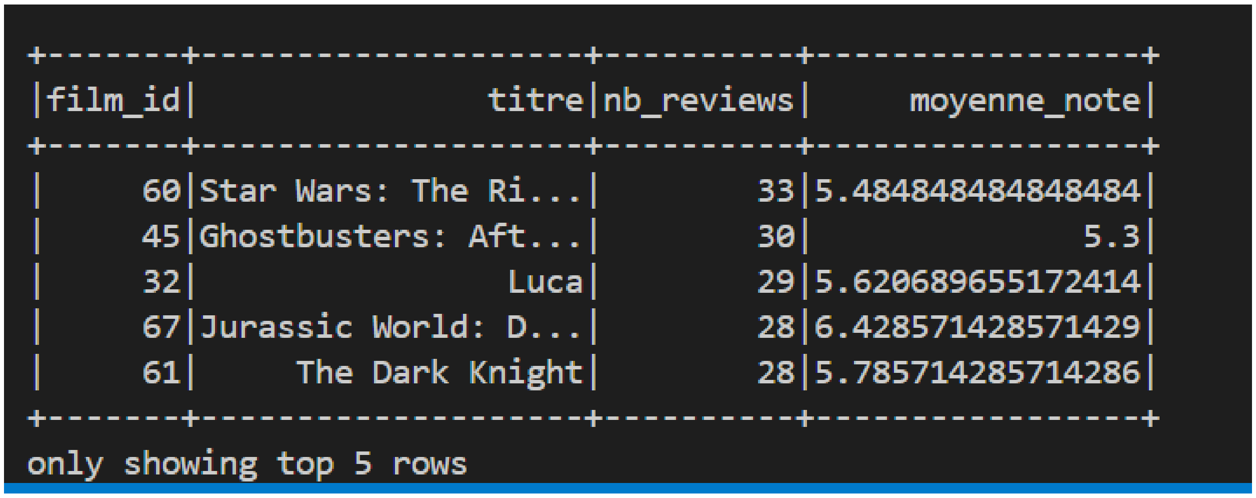
```
from pyspark.sql import functions as F

df= spark.read.json("reviews_nd.json")

df.withColumn("year_review", F.year("date_review")) \

  .groupBy("year_review") \

  .count() \

  .orderBy("year_review") \

  .show()
```



```
+-----------+-----+
|year_review|count|
+-----------+-----+
|       2019|   21|
|       2020|  171|
|       2021|  294|
|       2022|  474|
|       2023|  777|
|       2024|  263|
+-----------+-----+
```

Q3. Nombre de reviews et moyenne des notes par film

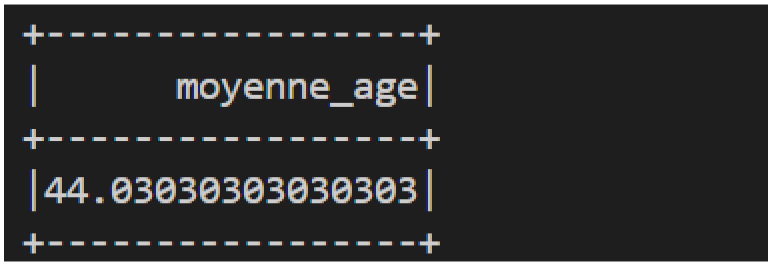
```
df.groupBy("film_id", "titre") \
  .agg(F.count("*").alias("nb_reviews"),
F.avg("note").alias("moyenne_note")) \
  .orderBy(F.col("nb_reviews").desc(), F.col("moyenne_note").desc()) \
  .show(5)
```



```
+-------+--------------------+----------+-----------------+
|film_id|               titre|nb_reviews|     moyenne_note|
+-------+--------------------+----------+-----------------+
|     60|Star Wars: The Ri...|        33|5.484848484848484|
|     45|Ghostbusters: Aft...|        30|              5.3|
|     32|                Luca|        29|5.620689655172414|
|     67|Jurassic World: D...|        28|6.428571428571429|
|     61|     The Dark Knight|        28|5.785714285714286|
+-------+--------------------+----------+-----------------+
only showing top 5 rows
```

Q4. Moyenne d'âge des utilisateurs du film le plus noté

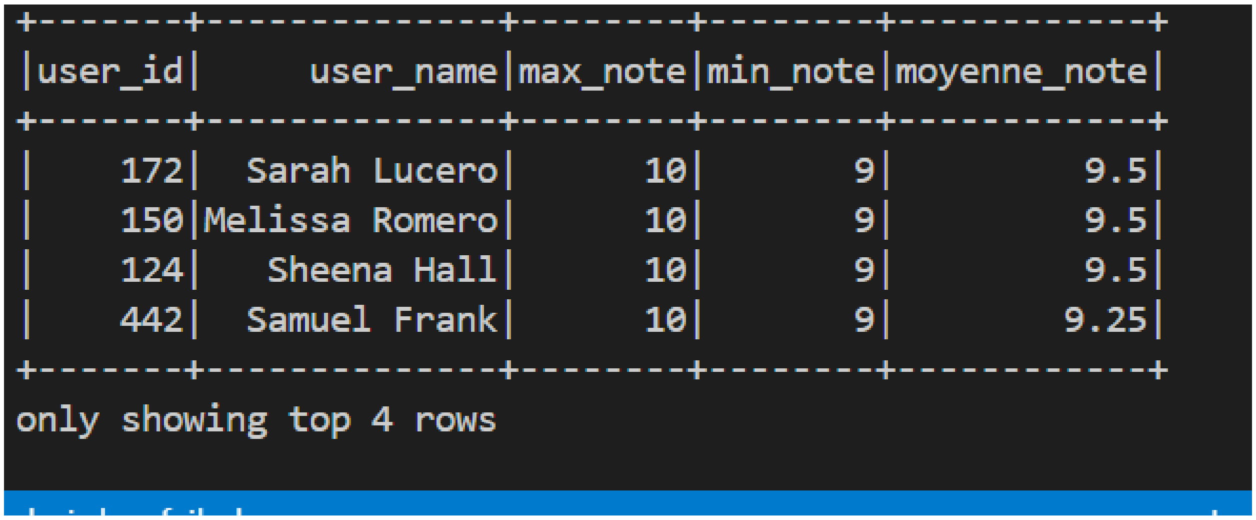
```
top_film=df.groupBy("film_id").count().orderBy(F.desc("count")).first()["f
ilm_id"]


df.filter(df["film_id"] == top_film) \
  .agg(F.avg("user_age").alias("moyenne_age")) \
  .show()
```



```
+-----------------+
|      moyenne_age|
+-----------------+
|44.03030303030303|
+-----------------+
```

Q5. Statistiques par utilisateur (top 4)

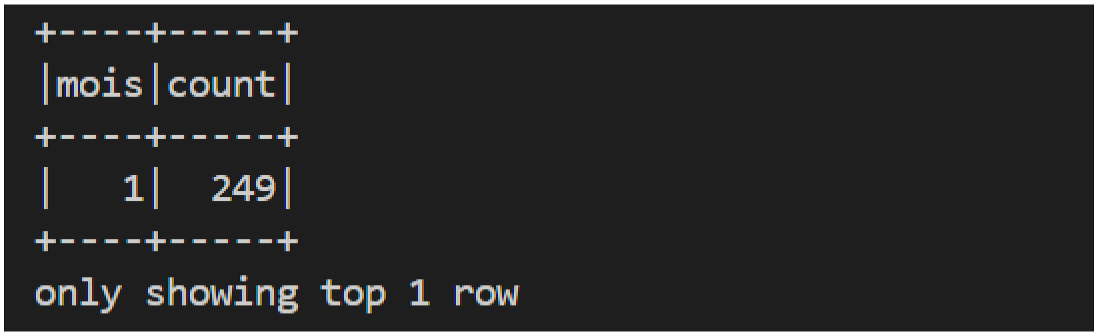
```
df.groupBy("user_id", "user_name") \
  .agg(
    F.max("note").alias("max_note"),
    F.min("note").alias("min_note"),
    F.avg("note").alias("moyenne_note")
  ) \
  .orderBy(F.col("moyenne_note").desc()) \
  .show(4)
```



```
+-------+--------------+--------+--------+------------+
|user_id|     user_name|max_note|min_note|moyenne_note|
+-------+--------------+--------+--------+------------+
|    172|  Sarah Lucero|      10|       9|         9.5|
|    150|Melissa Romero|      10|       9|         9.5|
|    124|   Sheena Hall|      10|       9|         9.5|
|    442|  Samuel Frank|      10|       9|        9.25|
+-------+--------------+--------+--------+------------+
only showing top 4 rows
```

Q6. Mois avec le plus de reviews

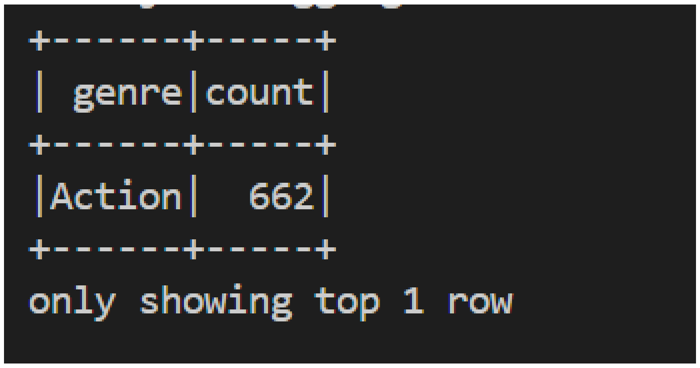
```
df.withColumn("mois", F.month("date_review")) \

  .groupBy("mois") \

  .count() \

  .orderBy(F.col("count").desc()) \

  .show(1)
```



```
+----+-----+
|mois|count|
+----+-----+
|   1|  249|
+----+-----+
only showing top 1 row
```

Q7. Genre le plus populaire (en nombre de reviews)

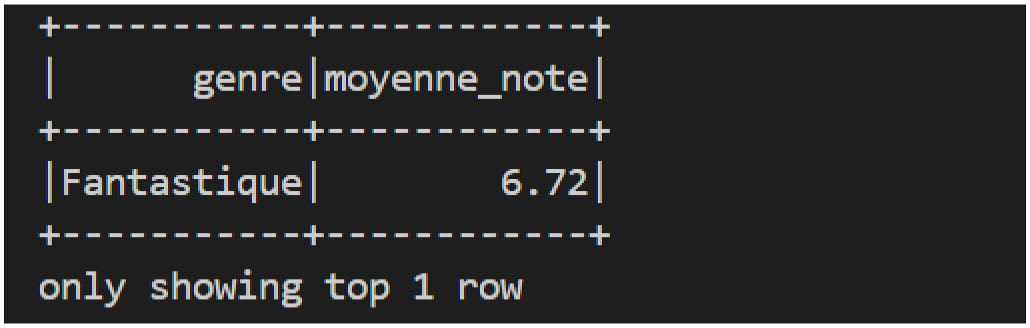
```
df.groupBy("genre") \

  .count() \

  .orderBy(F.desc("count")) \
  .show(1)
```



```
+------+-----+
| genre|count|
+------+-----+
|Action|  662|
+------+-----+
only showing top 1 row
```

Q8. Genre le mieux noté (en moyenne)

```
df.groupBy("genre") \
  .agg(F.avg("note").alias("moyenne_note")) \
  .orderBy(F.desc("moyenne_note")) \
  .show(1)
```

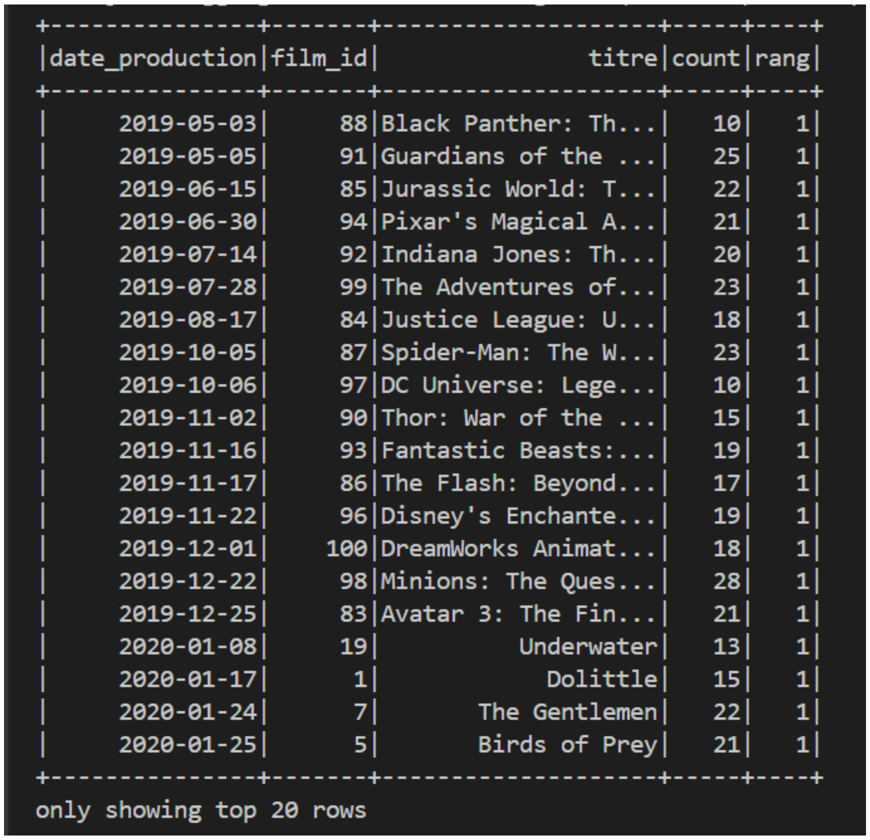


```
+----------+------------+
|     genre|moyenne_note|
+----------+------------+
|Fantastique|        6.72|
+----------+------------+
only showing top 1 row
```

Q9. Pour chaque année de production, film ayant reçu le plus de notes

```
wfrom pyspark.sql.window import Window


indow = Window.partitionBy("date_production").orderBy(F.desc("count"))
```

```
df.groupBy("date_production", "film_id", "titre") \
  .count() \
  .withColumn("rang", F.row_number().over(window)) \
  .filter(F.col("rang") == 1) \
  .orderBy("date_production") \
  .show()
```

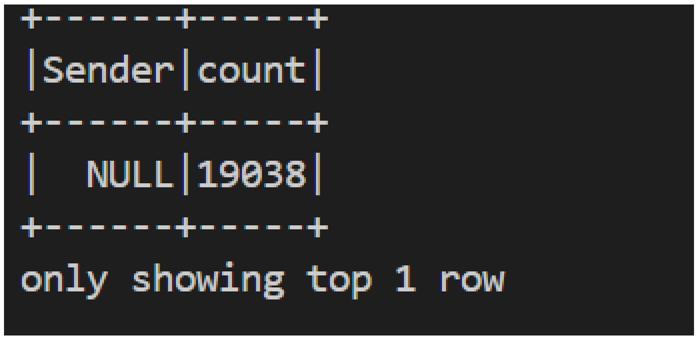


```
+---------------+-------+--------------------+-----+----+
|date_production|film_id|               titre|count|rang|
+---------------+-------+--------------------+-----+----+
|     2019-05-03|     88|Black Panther: Th...|   10|   1|
|     2019-05-05|     91|Guardians of the ...|   25|   1|
|     2019-06-15|     85|Jurassic World: T...|   22|   1|
|     2019-06-30|     94|Pixar's Magical A...|   21|   1|
|     2019-07-14|     92|Indiana Jones: Th...|   20|   1|
|     2019-07-28|     99|The Adventures of...|   23|   1|
|     2019-08-17|     84|Justice League: U...|   18|   1|
|     2019-10-05|     87|Spider-Man: The W...|   23|   1|
|     2019-10-06|     97|DC Universe: Lege...|   10|   1|
|     2019-11-02|     90|Thor: War of the ...|   15|   1|
|     2019-11-16|     93|Fantastic Beasts:...|   19|   1|
|     2019-11-17|     86|The Flash: Beyond...|   17|   1|
|     2019-11-22|     96|Disney's Enchante...|   19|   1|
|     2019-12-01|    100|DreamWorks Animat...|   18|   1|
|     2019-12-22|     98|Minions: The Ques...|   28|   1|
|     2019-12-25|     83|Avatar 3: The Fin...|   21|   1|
|     2020-01-08|     19|          Underwater|   13|   1|
|     2020-01-17|      1|            Dolittle|   15|   1|
|     2020-01-24|      7|        The Gentlemen|   22|   1|
|     2020-01-25|      5|        Birds of Prey|   21|   1|
+---------------+-------+--------------------+-----+----+
only showing top 20 rows
```

EXO3 :

Question 1 : Qui est l'employé le plus actif dans l'envoi des emails ?

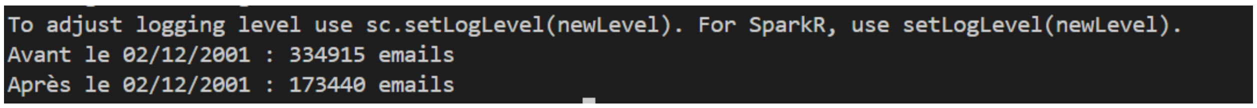
```
df_enron.groupBy("Sender") \

    .count() \

    .orderBy(F.desc("count")) \

    .show(1)
```



```
+------+-----+
|Sender|count|
+------+-----+
|  NULL|19038|
+------+-----+
only showing top 1 row
```

Q2. Nombre d'emails envoyés avant et après le 2 décembre 2001

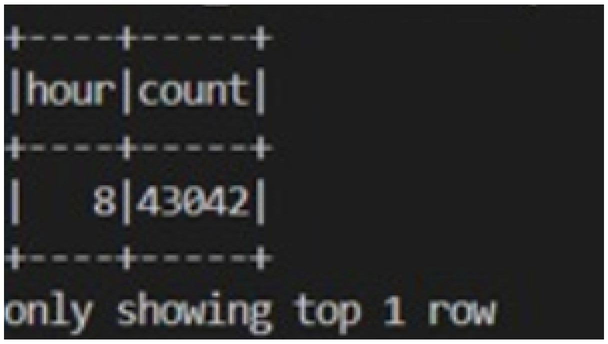
```
before = df_enron.filter(F.col("Date") < "2001-12-02").count()

after = df_enron.filter(F.col("Date") >= "2001-12-02").count()


print(f"Avant le 02/12/2001 : {before} emails")

print(f"Après le 02/12/2001 : {after} emails")
```



```
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Avant le 02/12/2001 : 334915 emails
Après le 02/12/2001 : 173440 emails
```

Q3. Heure d'activité la plus intense entre 8h et 17h

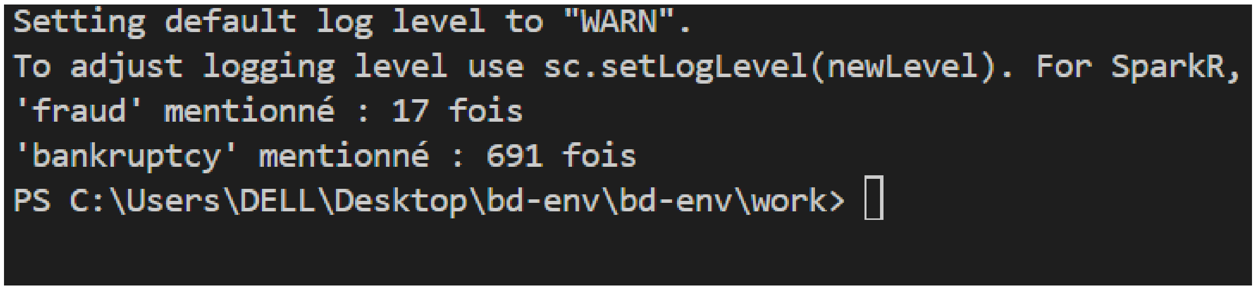
```
df = df_enron.withColumn("Date", to_timestamp("Date", "dd-MM-yyyy
HH:mm:ss"))


df.filter((F.col("Hour") >= 8) & (F.col("Hour") <= 17)) \

  .groupBy("Hour") \

  .count() \

  .orderBy(F.desc("count")) \

  .show(1)
```



```
+----+-----+
|hour|count|
+----+-----+
|   8|43042|
+----+-----+
only showing top 1 row
```

Q4. Mentions de "fraud" et "bankruptcy" dans le sujet

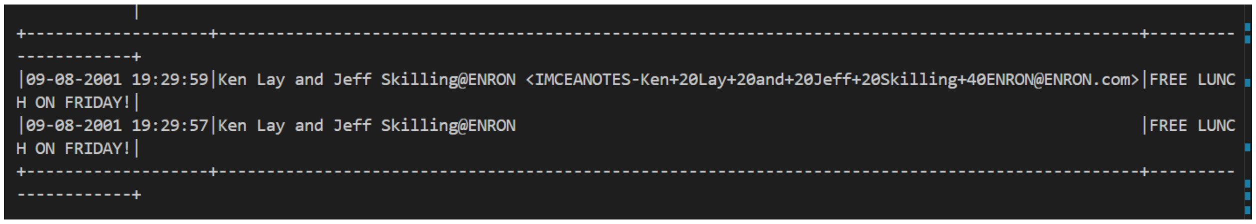
```
fraud_count =
df_enron.filter(F.lower(F.col("Subject")).contains("fraud")).count()

bankruptcy_count =
df_enron.filter(F.lower(F.col("Subject")).contains("bankruptcy")).count()


print(f"'fraud' mentionné : {fraud_count} fois")
print(f"'bankruptcy' mentionné : {bankruptcy_count} fois")
```



```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR,
'fraud' mentionné : 17 fois
'bankruptcy' mentionné : 691 fois
PS C:\Users\DELL\Desktop\bd-env\bd-env\work>
```

Q5. Emails envoyés par Jeff.Skilling@enron avec le sujet "FREE LUNCH ON FRIDAY!"

```
from pyspark.sql.functions import lower, col


df_enron.filter(

    (lower(col("Sender")).like("%skilling%")) &

    (lower(col("Subject")) == "free lunch on friday!")

).select("Date", "Sender", "Subject").show(truncate=False)
```

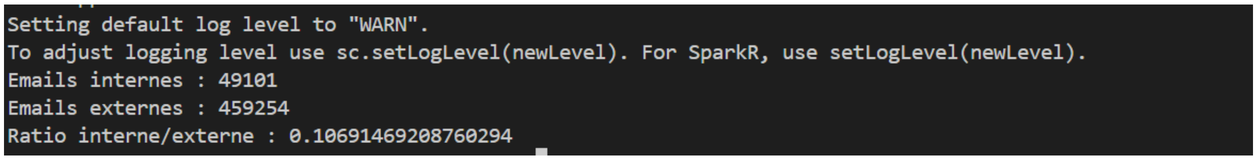


```
+-------------------+--------------------------------------------------------------------------------------------------+---------
------------+
|09-08-2001 19:29:59|Ken Lay and Jeff Skilling@ENRON <IMCEANOTES-Ken+20Lay+20and+20Jeff+20Skilling+40ENRON@ENRON.com>|FREE LUNC
H ON FRIDAY!|
|09-08-2001 19:29:57|Ken Lay and Jeff Skilling@ENRON                                                                  |FREE LUNC
H ON FRIDAY!|
+-------------------+--------------------------------------------------------------------------------------------------+---------
------------+
```

Q6. Ratio d’emails internes vs externes

```
df = df_enron.withColumn("is_internal",
F.col("Recipients").contains("@enron"))


nb_total = df.count()
```

```
nb_internal = df.filter(F.col("is_internal")).count()

nb_external = nb_total - nb_internal


ratio = nb_internal / nb_external if nb_external else "Infini"


print(f"Emails internes : {nb_internal}")

print(f"Emails externes : {nb_external}")

print(f"Ratio interne/externe : {ratio}")
```



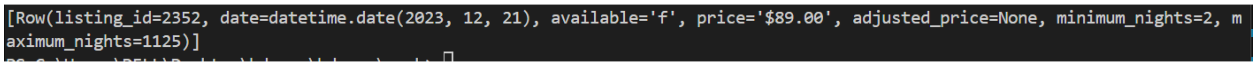
```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Emails internes : 49101
Emails externes : 459254
Ratio interne/externe : 0.10691469208760294
```

Exo4:

Q1. Transformer le dataset en RDD

```
import json

from pyspark.sql import SparkSession

from pyspark.sql import functions as F

from pyspark.sql.window import Window

from pyspark.sql.functions import to_timestamp, hour

from pyspark.sql.functions import lower, col
```

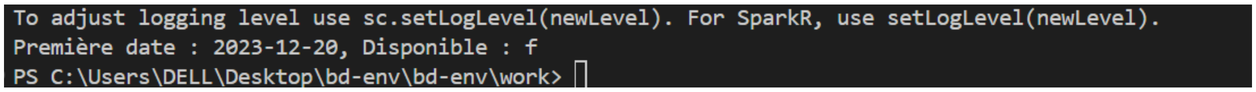
```
spark = SparkSession.builder.appName("AirbnbAnalysis")\

    .config("spark.jars.packages",
"org.apache.spark:spark-sql-kafka-0-10_2.12:3.4.1") \

    .getOrCreate()

df = spark.read.csv("airbnb.csv", header=True, inferSchema=True)



rdd = df.rdd

rdd.take(1)

print(rdd.take(1)
)
```



```
[Row(listing_id=2352, date=datetime.date(2023, 12, 21), available='f', price='$89.00', adjusted_price=None, minimum_nights=2, m
aximum_nights=1125)]
```

Q2. Quelle est la première date ? Le logement était-il disponible ?

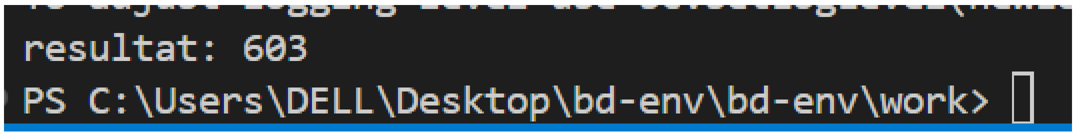
```
first_date_row = df.orderBy("date").select("date", "available").first()

print(f"Première date : {first_date_row['date']}, Disponible :
{first_date_row['available']}")
```



```
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Première date : 2023-12-20, Disponible : f
PS C:\Users\DELL\Desktop\bd-env\bd-env\work>
```

Q3. Combien d'annonces étaient disponibles le 31/12/2023 ?

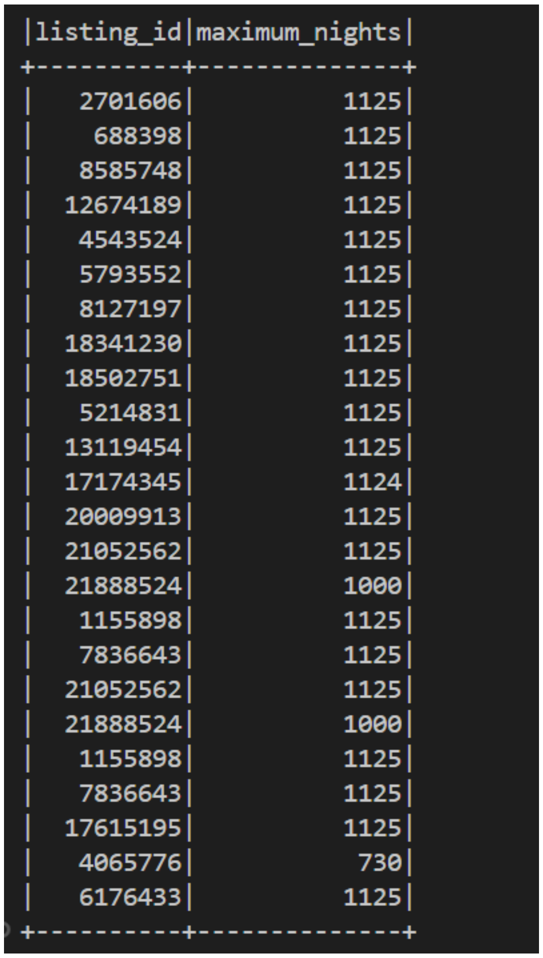
```
print("resultat:",df.filter((F.col("date") == "2023-12-31") &
(F.col("available") == "t")).count())
```



```
resultat: 603
PS C:\Users\DELL\Desktop\bd-env\bd-env\work>
```

Q4. Annonces ayant maximum_nights > 365

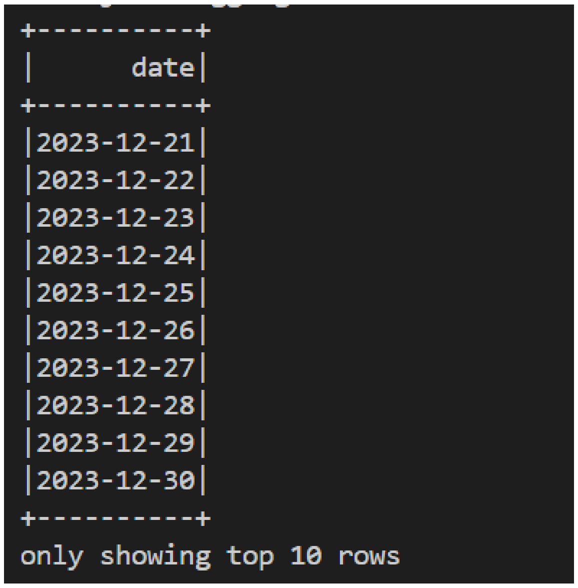
```
df.filter(F.col("maximum_nights") > 365).select("listing_id",
"maximum_nights").distinct().show(

)
```



```
|listing_id|maximum_nights|
+----------+--------------+
|   2701606|          1125|
|    688398|          1125|
|   8585748|          1125|
|  12674189|          1125|
|   4543524|          1125|
|   5793552|          1125|
|   8127197|          1125|
|  18341230|          1125|
|  18502751|          1125|
|   5214831|          1125|
|  13119454|          1125|
|  17174345|          1124|
|  20009913|          1125|
|  21052562|          1125|
|  21888524|          1000|
|   1155898|          1125|
|   7836643|          1125|
|  21052562|          1125|
|  21888524|          1000|
|   1155898|          1125|
|   7836643|          1125|
|  17615195|          1125|
|   4065776|           730|
|   6176433|          1125|
+----------+--------------+
```

Q5. Pour listing_id = 2352, afficher les 10 premières dates non disponibles

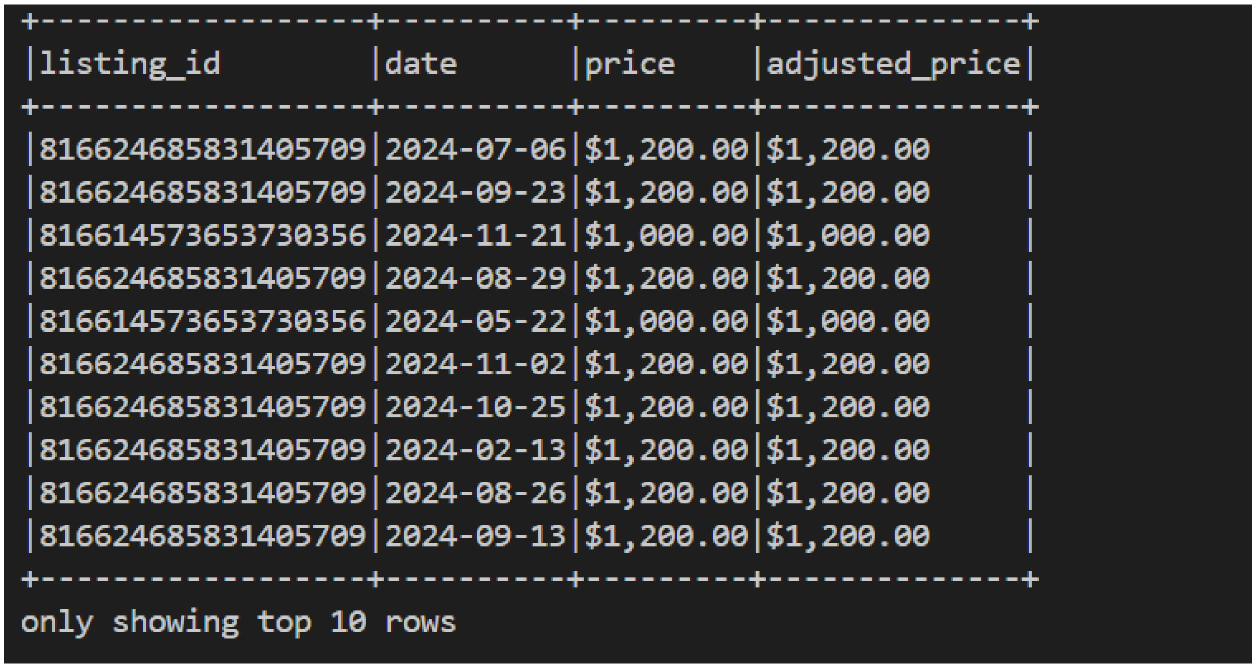
```
df.filter((F.col("listing_id") == 2352) & (F.col("available") == "f")) \
  .orderBy("date") \
  .select("date") \
  .show(10)
```



```
+----------+
|      date|
+----------+
|2023-12-21|
|2023-12-22|
|2023-12-23|
|2023-12-24|
|2023-12-25|
|2023-12-26|
|2023-12-27|
|2023-12-28|
|2023-12-29|
|2023-12-30|
+----------+
only showing top 10 rows
```

Q6. 10 annonces aléatoires où price == adjusted_price

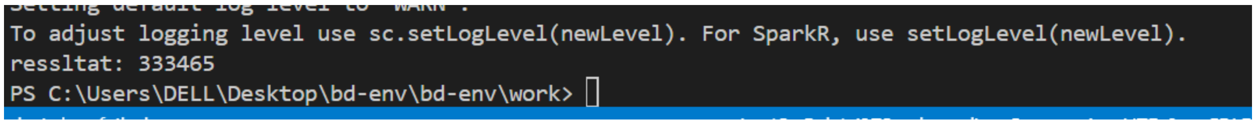
```
from pyspark.sql.functions import rand


df.filter((F.col("price") == F.col("adjusted_price"))) \
  .orderBy(rand()) \
  .select("listing_id", "date", "price", "adjusted_price") \
  .show(10, truncate=False)
```



```
+------------------+----------+---------+--------------+
|listing_id        |date      |price    |adjusted_price|
+------------------+----------+---------+--------------+
|816624685831405709|2024-07-06|$1,200.00|$1,200.00     |
|816624685831405709|2024-09-23|$1,200.00|$1,200.00     |
|816614573653730356|2024-11-21|$1,000.00|$1,000.00     |
|816624685831405709|2024-08-29|$1,200.00|$1,200.00     |
|816614573653730356|2024-05-22|$1,000.00|$1,000.00     |
|816624685831405709|2024-11-02|$1,200.00|$1,200.00     |
|816624685831405709|2024-10-25|$1,200.00|$1,200.00     |
|816624685831405709|2024-02-13|$1,200.00|$1,200.00     |
|816624685831405709|2024-08-26|$1,200.00|$1,200.00     |
|816624685831405709|2024-09-13|$1,200.00|$1,200.00     |
+------------------+----------+---------+--------------+
only showing top 10 rows
```

Question 7 :Combien d'annonces ont une minimum_nights supérieure à la moyenne globale ?

```
from pyspark.sql import functions as F


moyenne_min_nights = df.select(F.avg("minimum_nights")).first()[0]


print("ressltat:",df.filter(F.col("minimum_nights") >
moyenne_min_nights).count())
```



```
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
ressltat: 333465
PS C:\Users\DELL\Desktop\bd-env\bd-env\work> 
```