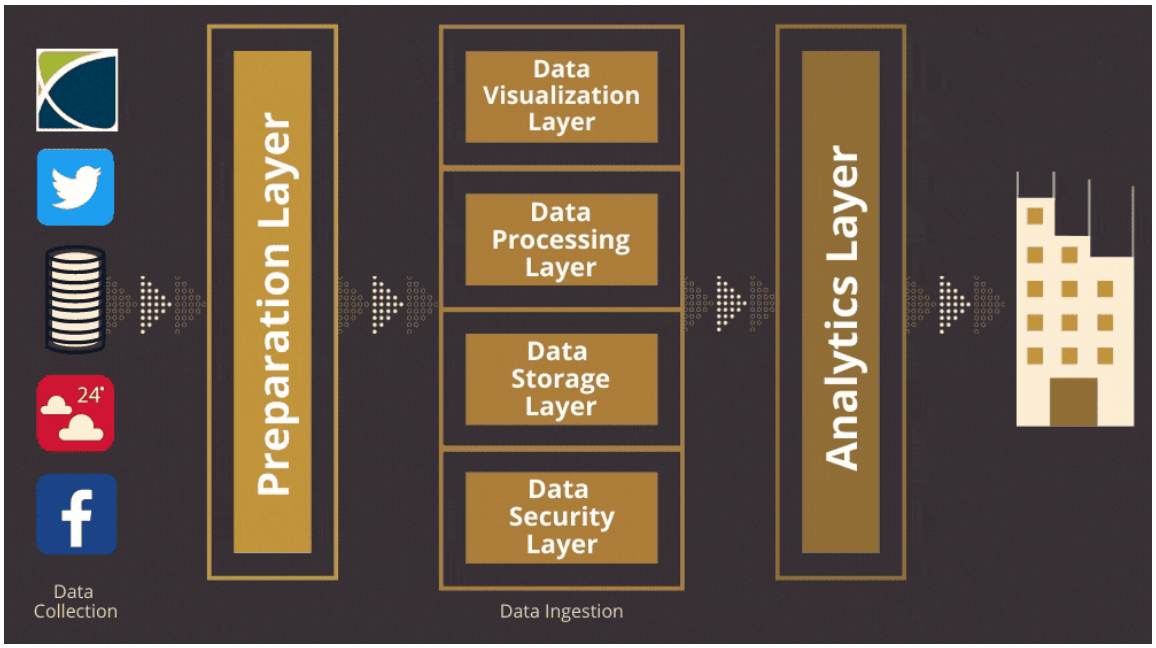
Computer Science and Mathematics

2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

Data Management and Organization (STAT6158016)
Group Assignment

1. Explain the architecture of data streaming



Source: https://estuary.dev/data-streaming-architecture/

Data streaming is a process that allows analysts to integrate new data in real time rather than waiting for a whole file to transfer from one platform to another. Streaming refers to a method in which data is processed as soon as it enters rather than being saved and processed batch-wise. This streamlined process enables instant insights and actions, which is critical in rapid decision-making. Usually, this technology is used for live internet websites and/or applications for real-time data processing in analytics which includes, but not limited to financial transactions and social media analytics.

Referring to the image above, the process goes as follows:

1. Data collection
   The initial process involves gathering data from numerous sources. Data is sourced from a variety of sites. The data itself can be in different formats such as JSON or CSV, and can have different characteristics.
2. Preparation Layer
   Preparation layer is a stage or component that is responsible for preparing or transforming raw incoming data before it enters the main processing phase. This preparation could

2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

involve tasks like data cleansing, normalization, schema validation, or any other operations that make the data suitable for further processing.

3. Data Ingestion
   Data ingestion involves bringing the previously collected data into the system for processing. This process involves services capable of handling real time data streams, for example the Apache Kafka, Google Cloud Pub/Sub, and AWS Kinesis.
4. Data Visualization
   Data visualization is the presentation of data in a graphical or visual format to help analysts and decision makers understand complex information. Visualization provides a clear and intuitive way to understand patterns, trends, and anomalies in streaming data. Graphical representations help decision makers quickly understand the current situation and make informed decisions.
5. Data Processing
   Data processing in data streaming involves applying transformations, filters, and calculations to incoming data in real time. Unlike batch processing, data is processed as it arrives, allowing for immediate analysis and response. Processing may involve complex algorithms, filters, aggregations, and enrichments to derive meaningful insights. Apache Flink, Apache Storm, and Apache Kafka Streams are examples of stream processing engines that enable real-time data processing. CEP systems process patterns and relationships within streaming data.
6. Data Storage
   The mechanisms and systems employed to store and manage data that is being continuously generated and processed in real-time or near-real-time. Data streaming involves the constant flow of data, and efficient storage solutions are required to manage and make this data available for analysis, reporting, or other purposes.
7. Data Security
   Use policies and procedures to maintain the availability, confidentiality, and integrity of data during transmission, processing, and archiving in streaming settings that operate in real-time from near real-time. policies and procedures for maintaining the availability, confidentiality, and integrity of data during transmission, processing, and archiving in streaming settings operating in real-time or near real-time. Preventing unauthorized access, data breaches, and other security issues requires ensuring data security security issues requires ensuring data security in data streaming.in data streaming.
8. Analytics Layer
   Analytics layer is a component or stage in the streaming architecture that is focused on performing various types of analytics on the streaming data. This layer is responsible for

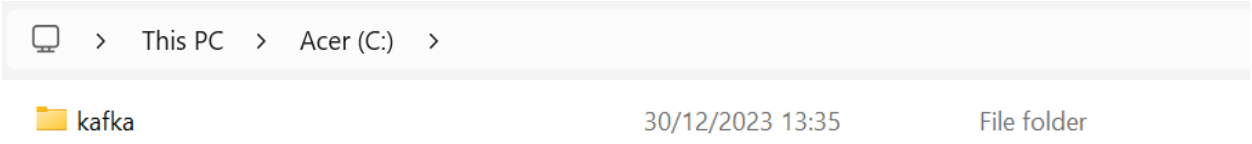
2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

processing and analyzing the incoming data streams in real-time or near-real-time, extracting meaningful insights, and generating valuable information for decision-making.

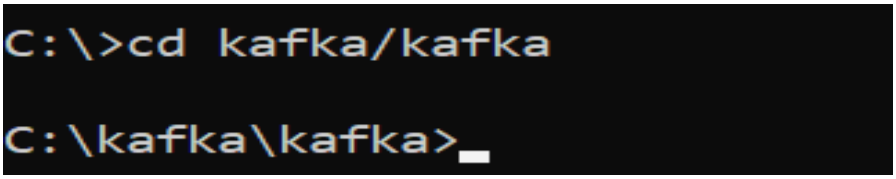
2. How to simulate data streaming with Kafka, Spark Streaming, and PySpark (with example)

- Good if you can try it and share the result from your testing

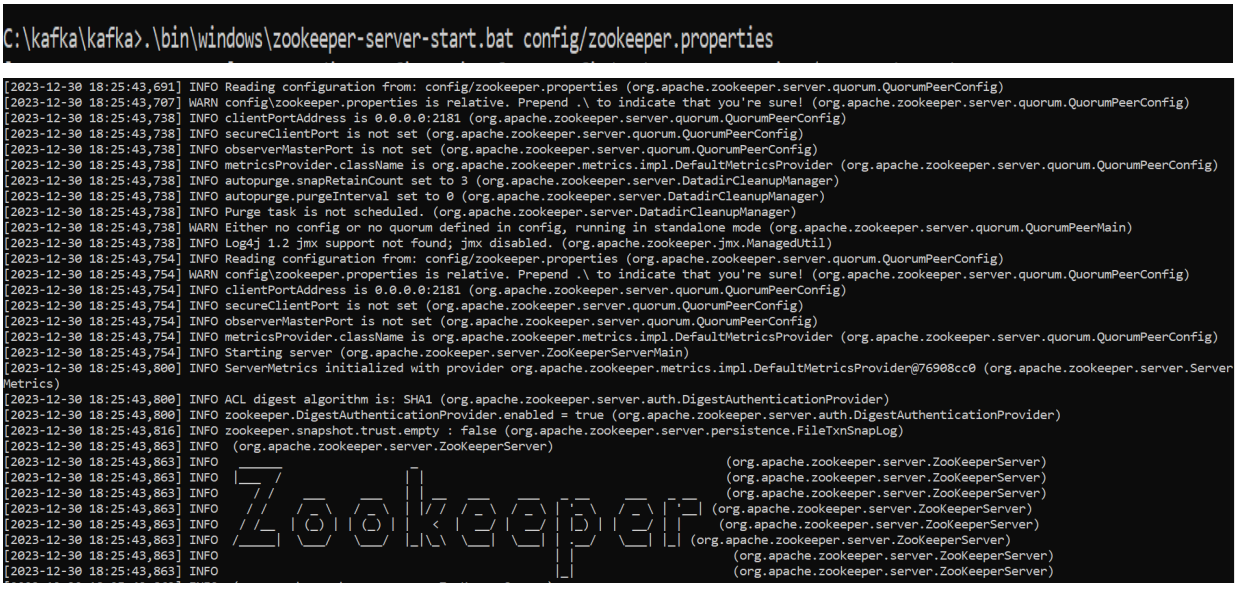
1. Set up Kafka
   Download and install Kafka from https://kafka.apache.org/downloads.
2. Unzip the File. Do it manually or with command prompt.



3. Navigate to the Kafka Directory



4. Start ZooKeeper



2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

```
C:\kafka\kafka>cd bin

C:\kafka\kafka\bin>zookeeper-server-start.sh config/zookeeper.properties

C:\kafka\kafka\bin>
[main 2024-01-07T03:00:24.579Z] update#setState idle
[main 2024-01-07T03:00:29.004Z] WSL is not installed, so could not detect WSL profiles
[main 2024-01-07T03:00:54.601Z] update#setState checking for updates
[main 2024-01-07T03:00:54.666Z] update#setState idle
```

5. Start the Kafka Server
   Open a separate terminal window and start the Kafka server:

```
C:\kafka\kafka>.\bin\windows\kafka-server-start.bat .\config\server.properties
```

```
[2023-12-30 18:27:39,705] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
[2023-12-30 18:27:40,455] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS renegotiation (org.apache.zookeeper.common.X509Uti
l)
[2023-12-30 18:27:40,689] INFO starting (kafka.server.KafkaServer)
[2023-12-30 18:27:40,689] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2023-12-30 18:27:40,736] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)
[2023-12-30 18:27:40,736] INFO Client environment:zookeeper.version=3.8.3-6ad6d364c7c0bcf0de452d54ebefa3058098ab56, built on 2023-10-05 10:34 UTC (org.apache.zookeeper.ZooK
eeper)
```

```
C:\kafka\kafka>cd bin

C:\kafka\kafka\bin>kafka-server-start.sh config/server.properties
```

zookeeper-server-start.sh

```
#!/bin/bash
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

if [ $# -lt 1 ];
then
    echo "USAGE: $0 [-daemon] zookeeper.properties"
    exit 1
fi
base_dir=$(dirname $0)
```

zookeeper-server-start.sh

```
if [ "x$KAFKA_LOG4J_OPTS" = "x" ]; then
    export KAFKA_LOG4J_OPTS="-Dlog4j.configuration=file:$base_dir/../config/log4j.properties"
fi

if [ "x$KAFKA_HEAP_OPTS" = "x" ]; then
    export KAFKA_HEAP_OPTS="-Xmx512M -Xms512M"~
fi

EXTRA_ARGS=${EXTRA_ARGS-'-name zookeeper -loggc'}

COMMAND=$1
case $COMMAND in
  -daemon)
    EXTRA_ARGS="-daemon "$EXTRA_ARGS
    shift
    ;;
 *)
    ;;
esac

exec $base_dir/kafka-run-class.sh $EXTRA_ARGS org.apache.zookeeper.server.quorum.QuorumPeerMain "$@"
```

6. Verify Installation:
   Create a test topic to confirm Kafka is running correctly:

2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

```
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\kafka\kafka\bin\windows>kafka-topics.bat --create --topic topic-cobaKafka --bootstrap-server localhost:9092
Created topic topic-cobaKafka.
```

7. Input any kind of datas into the test topic:

```
C:\kafka\kafka\bin\windows>kafka-console-producer.bat --topic topic-cobaKafka --bootstrap-server localhost:9092
>Hello
>Ujicoba
>untuk data streaming
>demi tugas projek DMO
>group assignment
>^CTerminate batch job (Y/N)? y
```

8. To make sure that the datas have been stored inside the topic, check the datas by using kafka-console-consumer.bat then display the data from beginning:

```
C:\kafka\kafka\bin\windows>kafka-console-consumer.bat --topic topic-cobaKafka --from-beginning --bootstrap-server localhost:9092
Hello
Ujicoba
untuk data streaming
demi tugas projek DMO
group assignment
```

zookeeper-server-start.sh | DMO.py | DMO2.py | kafka-server-start.sh | kafka-topics.sh

D: > Download > kafka-3.6.1-src > bin > kafka-topics.sh

```
#!/bin/bash
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

exec $(dirname $0)/kafka-run-class.sh kafka.admin.TopicCommand "$@"
```

9. Create the spark session for the kafka file in pyspark

```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("KafkaStreaming") \
    .getOrCreate()
```

Computer Science and Mathematics

2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

10. Make a code to read file from kafka, the kafka.bootstrap.servers fills with a host name and port name in kafka.

```
streaming_df = spark.readStream\
    .format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:2181") \
    .option("subscribe", "test-topic") \
    .option("startingOffsets", "earliest") \
    .load()
```

11. Make a schema for the dataframe. To make the schema, use structype and structfield and types of the attributes of the dataframe using libraries from pyspark.sql.types.

```
from pyspark.sql.types import StringType, StructField, StructType, ArrayType, LongType

json_schema = StructType([StructField('customerId', StringType(), True), \
StructField('data', StructType([StructField('devices', ArrayType(StructType([ \
StructField('deviceId', StringType(), True), \
StructField('measure', StringType(), True), \
StructField('status', StringType(), True), \
StructField('temperature', LongType(), True)]), True), True)]), True), \
StructField('eventId', StringType(), True), \
StructField('eventOffset', LongType(), True), \
StructField('eventPublisher', StringType(), True), \
StructField('eventTime', StringType(), True)])
```

12. Read the data from value column, cast to string and expand the JSON

```
json_df = streaming_df.selectExpr("cast(value as string) as value")

from pyspark.sql.functions import from_json

json_expanded_df = json_df.withColumn("value", from_json(json_df["value"],
json_schema)).select("value.*")
```

13. Then, explode the dataframe to get the devices array

2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

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

exploded_df = json_expanded_df \
   .select("customerId", "eventId", "eventOffset", "eventPublisher", "eventTime",
"data") \
   .withColumn("devices", explode("data.devices")) \
   .drop("data")
```

14. Flatten the dataframe that has been exploded

```
flattened_df = exploded_df \
   .selectExpr("customerId", "eventId", "eventOffset", "eventPublisher","cast(eventTime as
timestamp) as eventTime",
            "devices.deviceId as deviceId", "devices.measure as measure",
            "devices.status as status", "devices.temperature as temperature")
```

15. Finally, write the data to console in outputMode as complete

```
writing_df = agg_df.writeStream \
   .format("console") \
   .option("checkpointLocation","checkpoint_dir") \
   .outputMode("complete") \
   .start()

# Start the streaming application to run until the following happens
# 1. Exception in the running program
# 2. Manual Interruption
writing_df.awaitTermination()
```

Source:

https://subhamkharwal.medium.com/pyspark-structured-streaming-read-from-kafka-64c40767155f

2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

3. PySpark:

Data used: https://www.kaggle.com/datasets/nelgiriyewithana/top-spotify-songs-2023?resource=download

- Find data csv from Data Exploration, Data Visualization & Regression
- Perform Data Exploration, Data Visualisation & Regression

1. Importing necessary library and data and printing its Schema

Code:

```
%pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("SpotifyDataAnalysis").getOrCreate()

df = spark.read.csv("file:///home/verulam-blue/working/spotify-2023.csv", header=True,
inferSchema=True)
df.printSchema()
```

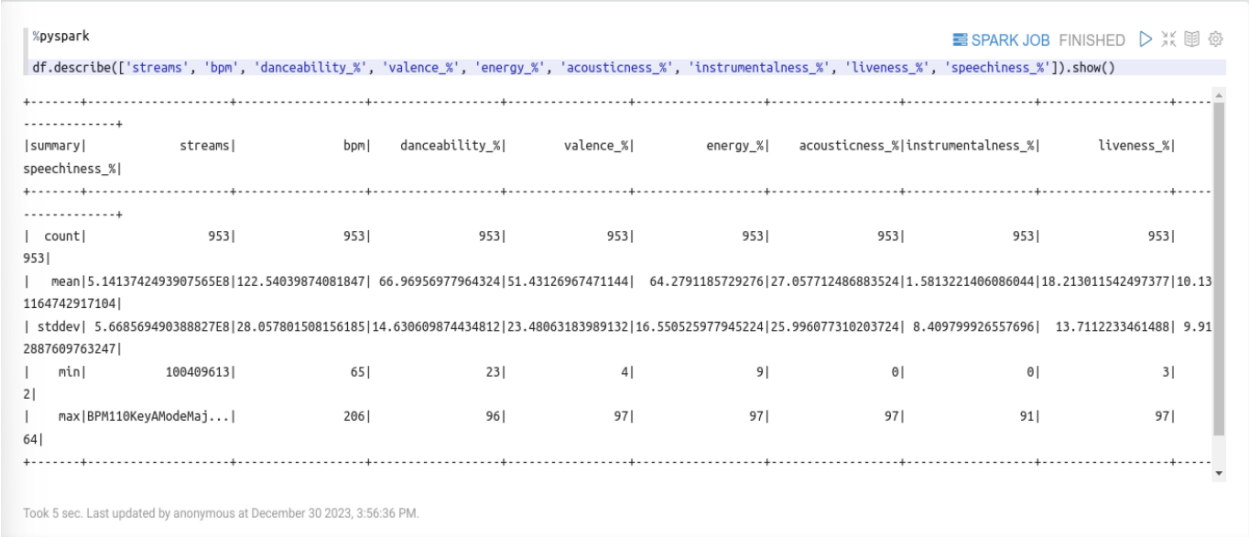
Results:

```
|-- track_name: string (nullable = true)
 |-- artist(s)_name: string (nullable = true)
 |-- artist_count: integer (nullable = true)
 |-- released_year: integer (nullable = true)
 |-- released_month: integer (nullable = true)
 |-- released_day: integer (nullable = true)
 |-- in_spotify_playlists: integer (nullable = true)
 |-- in_spotify_charts: integer (nullable = true)
 |-- streams: string (nullable = true)
 |-- in_apple_playlists: integer (nullable = true)
 |-- in_apple_charts: integer (nullable = true)
 |-- in_deezer_playlists: string (nullable = true)
 |-- in_deezer_charts: integer (nullable = true)
 |-- in_shazam_charts: string (nullable = true)
 |-- bpm: integer (nullable = true)
 |-- key: string (nullable = true)
 |-- mode: string (nullable = true)
 |-- danceability_%: integer (nullable = true)
 |-- valence_%: integer (nullable = true)
 |-- energy_%: integer (nullable = true)
 |-- acousticness_%: integer (nullable = true)
 |-- instrumentalness_%: integer (nullable = true)
 |-- liveness_%: integer (nullable = true)
```

2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

```
|-- speechiness_%: integer (nullable = true)
```

2. Summary statistics of numerical columns



```
%pyspark
df.describe(['streams', 'bpm', 'danceability_%', 'valence_%', 'energy_%', 'acousticness_%', 'instrumentalness_%', 'liveness_%', 'speechiness_%']).show()
```

SPARK JOB FINISHED

```
+-------+------------------+----------------+-----------------+----------------+----------------+----------------+-----------------+----------------+-----------------+
|summary|           streams|             bpm|   danceability_%|      valence_%|        energy_%|   acousticness_%|instrumentalness_%|      liveness_%|     speechiness_%|
+-------+------------------+----------------+-----------------+----------------+----------------+----------------+-----------------+----------------+-----------------+
|  count|               953|             953|              953|             953|             953|             953|              953|             953|              953|
|   mean|5.141374249390756E8|122.54039874081847| 66.96956977964324|51.43126967471144| 64.2791185729276|27.057712486883524|1.5813221406086044|18.213011542497377|10.131164742917104|
| stddev| 5.668569490388827E8|28.057801508156185|14.630609874434812|23.48063183989132|16.550525977945224|25.996077310203724| 8.409799926557696| 13.7112233461488| 9.912887609763247|
|    min|         100409613|              65|               23|               4|               9|               0|                0|               3|                2|
|    max|BPM110KeyAModeMaj...|             206|               96|              97|              97|              97|               91|              97|               64|
+-------+------------------+----------------+-----------------+----------------+----------------+----------------+-----------------+----------------+-----------------+
```

Took 5 sec. Last updated by anonymous at December 30 2023, 3:56:36 PM.

3. Calculating the sum of unique tracks and artists

```
%pyspark

df.select(F.countDistinct("track_name")).alias('unique_tracks').show()
df.select(F.countDistinct("artist(s)_name")).alias('artist(s)_name').show()
```

```
+--------------------------+
|count(DISTINCT track_name)|
+--------------------------+
|                       943|
+--------------------------+


+-----------------------------+
|count(DISTINCT artist(s)_name)|
+-----------------------------+
|                          645|
+-----------------------------+
```

Took 22 sec. Last updated by anonymous at December 30 2023, 3:59:31 PM.

4. Calculate the average number of streams per artists

2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

```
%pyspark

df.groupBy("artist(s)_name").agg(F.avg("streams").alias('avg_streams')).orderBy('avg_streams', ascending=False).show()
```

```
+--------------------+-------------+
|      artist(s)_name|  avg_streams|
+--------------------+-------------+
|         Tones and I|2.864791672E9|
|Post Malone, Swae...| 2.80809655E9|
| Drake, WizKid, Kyla| 2.71392235E9|
|Justin Bieber, Th...|2.665343922E9|
|The Chainsmokers,...|2.591224264E9|
|The Weeknd, Daft ...|2.565529693E9|
|       Glass Animals|2.557975762E9|
|Shawn Mendes, Cam...|2.484812918E9|
|Billie Eilish, Kh...|2.355719893E9|
|The Chainsmokers,...|2.204080728E9|
```

5. Finding the most common key and mode used by popular artists

```
%pyspark

df.groupBy("key").count().orderBy('count', ascending=False).show()
df.groupBy("mode").count().orderBy('count', ascending=False).show()
```

```
+----+-----+
| key|count|
+----+-----+
|  C#|  120|
|   G|   96|
|null|   95|
|  G#|   91|
|   F|   89|
|   D|   81|
|   B|   81|
|   A|   75|
|  F#|   73|
|   E|   62|
|  A#|   57|
|  D#|   33|
+----+-----+
```

```
+-----+-----+
| mode|count|
+-----+-----+
|Major|  550|
|Minor|  403|
+-----+-----+
```

6. Finding the average danceability, valence, and energy by month of release

2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

```
%pyspark

df.groupBy("released_month").agg(
    F.avg("danceability_%").alias('danceability'),
    F.avg("valence_%").alias('valence'),
    F.avg("energy_%").alias('energy')
).orderBy("released_month").show()
```

```
+--------------+-----------------+-----------------+-----------------+
|released_month|     danceability|          valence|           energy|
+--------------+-----------------+-----------------+-----------------+
|             1| 65.5223880597015|57.208955223880594| 65.80597014925372|
|             2|67.49180327868852| 57.90163934426229|  66.0327868852459|
|             3|67.18604651162791| 54.08139534883721| 67.84883720930233|
|             4|67.71212121212122| 46.53030303030303| 62.22727272727273|
|             5|        68.609375|         51.140625|        63.0390625|
|             6|71.96511627906976|51.325581395348834| 65.46511627906976|
|             7|67.14516129032258| 52.70967741935484| 65.25806451612904|
|             8|66.78260869565217|49.130434782608695| 67.19565217391305|
|             9|68.42857142857143|48.642857142857146| 65.41071428571429|
|            10|62.67123287671233| 41.36986301369863|59.794520547945204|
|            11|          65.4125|             49.0|            62.125|
|            12|64.42666666666666|            52.56| 61.81333333333333|
+--------------+-----------------+-----------------+-----------------+
```

7. Data Visualization

Code:

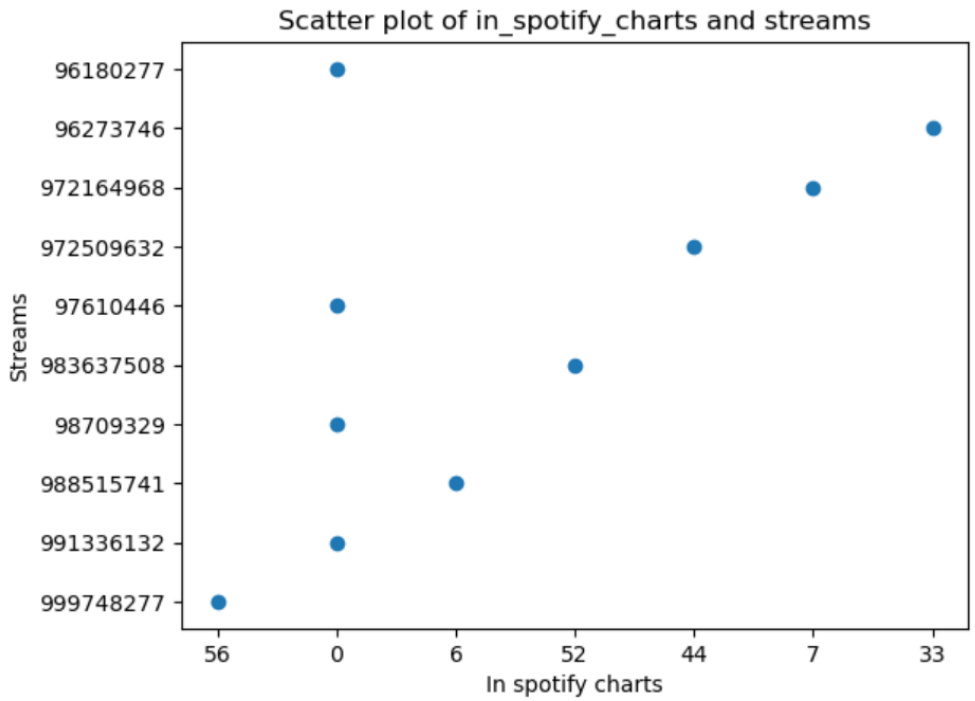
```
import matplotlib.pyplot as plt
from pyspark.sql import SparkSession

spark =
SparkSession.builder.appName("SpotifyDataAnalysis").config("spark.some.config.option",
"some-value").getOrCreate()

df = spark.read.format("csv").option("inferSchema", "false").option("header",
"true").option("sep", ",").load("spotify-2023.csv")

# Drop row with track_name "Love Grows (Where My Rosemary Goes)"
df = df.filter(df["track_name"] != "Love Grows (Where My Rosemary Goes)")

df.createOrReplaceTempView('spotifyData')

data = spark.sql(
```

2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

```
'''
SELECT * FROM spotifyData
ORDER BY streams DESC Limit 10
'''
).toPandas()
```

```
plt.scatter(data["in_spotify_charts"], data["streams"])
plt.title("Scatter plot of in_spotify_charts and streams")
plt.xlabel("In spotify charts")
plt.ylabel("Streams")
plt.show()
```

Result:

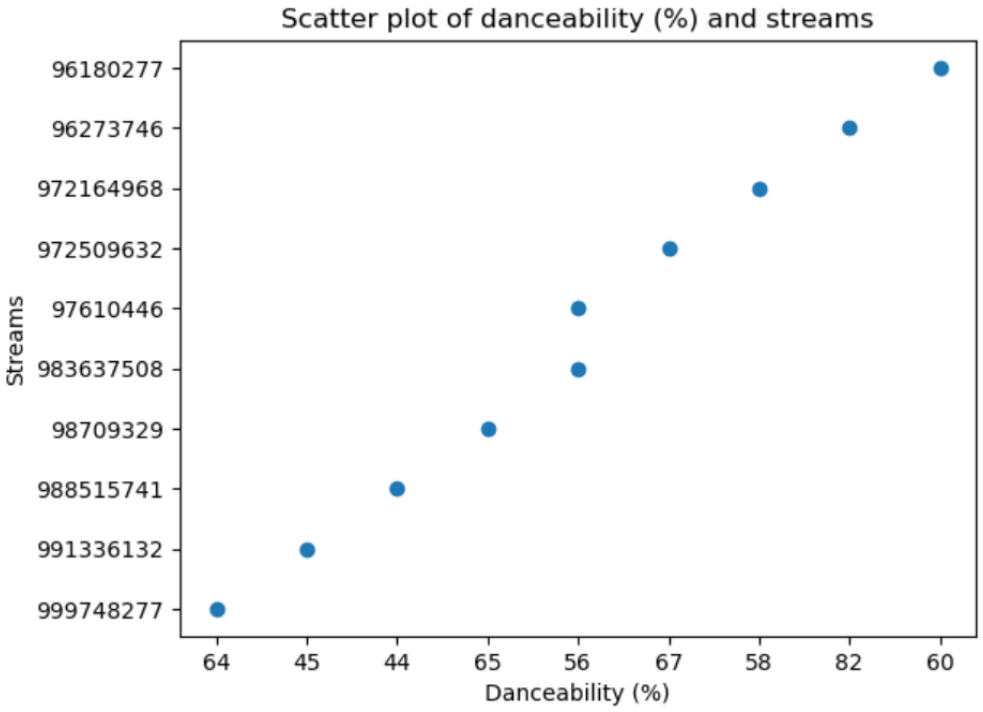



```
plt.scatter(data["danceability_%"], data["streams"])
plt.title("Scatter plot of danceability (%) and streams")
plt.xlabel("Danceability (%)")
```

2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

```
plt.ylabel("Streams")
plt.show()
```

Result:

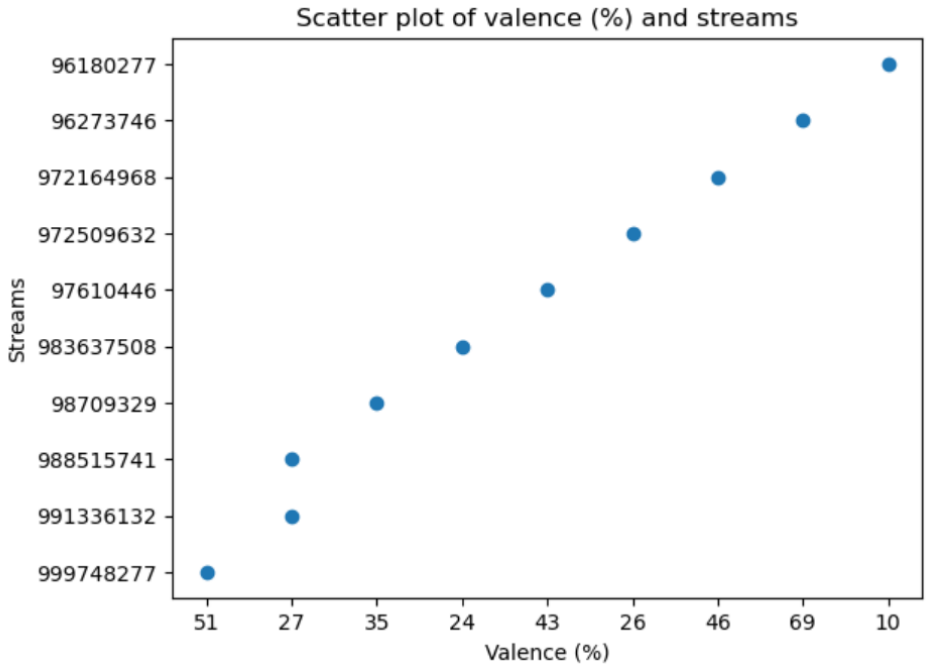



```
plt.scatter(data["valence_%"], data["streams"])
plt.title("Scatter plot of valence (%) and streams")
plt.xlabel("Valence (%)")
plt.ylabel("Streams")
plt.show()
```

Result:

Computer Science and Mathematics

2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

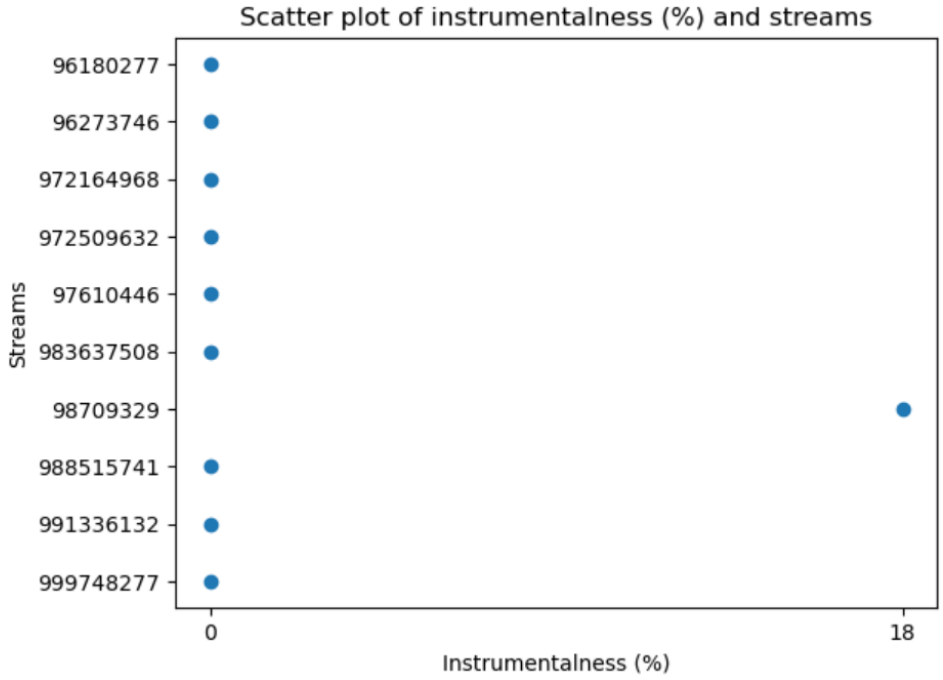



```
plt.scatter(data["instrumentalness_%"], data["streams"])
plt.title("Scatter plot of instrumentalness (%) and streams")
plt.xlabel("Instrumentalness (%)")
plt.ylabel("Streams")
plt.show()
```

Result:

Computer Science and Mathematics

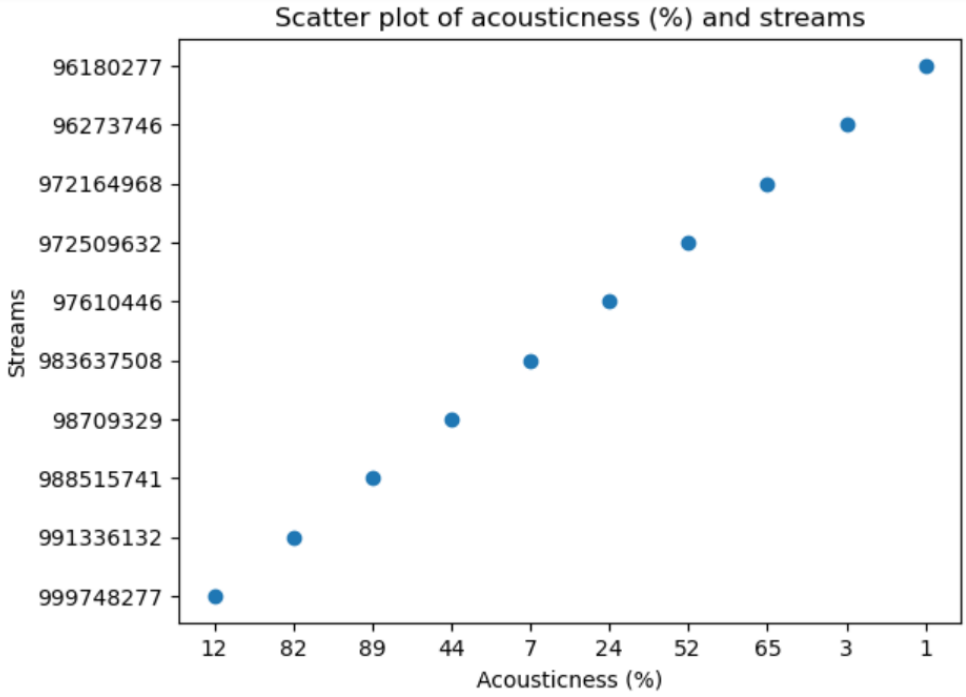
2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama




```
plt.scatter(data["acousticness_%"], data["streams"])
plt.title("Scatter plot of acousticness (%) and streams")
plt.xlabel("Acousticness (%)")
plt.ylabel("Streams")
plt.show()
```

Computer Science and Mathematics

2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

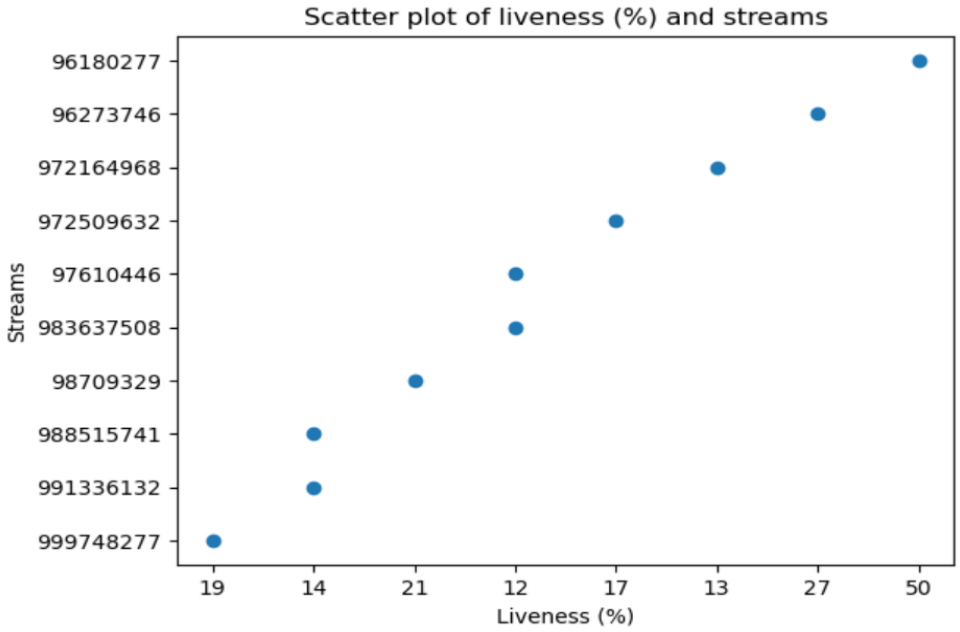



```
plt.scatter(data["liveness_%"], data["streams"])
plt.title("Scatter plot of liveness (%) and streams")
plt.xlabel("Liveness (%)")
plt.ylabel("Streams")
plt.show()
```

Result:

Computer Science and Mathematics

2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

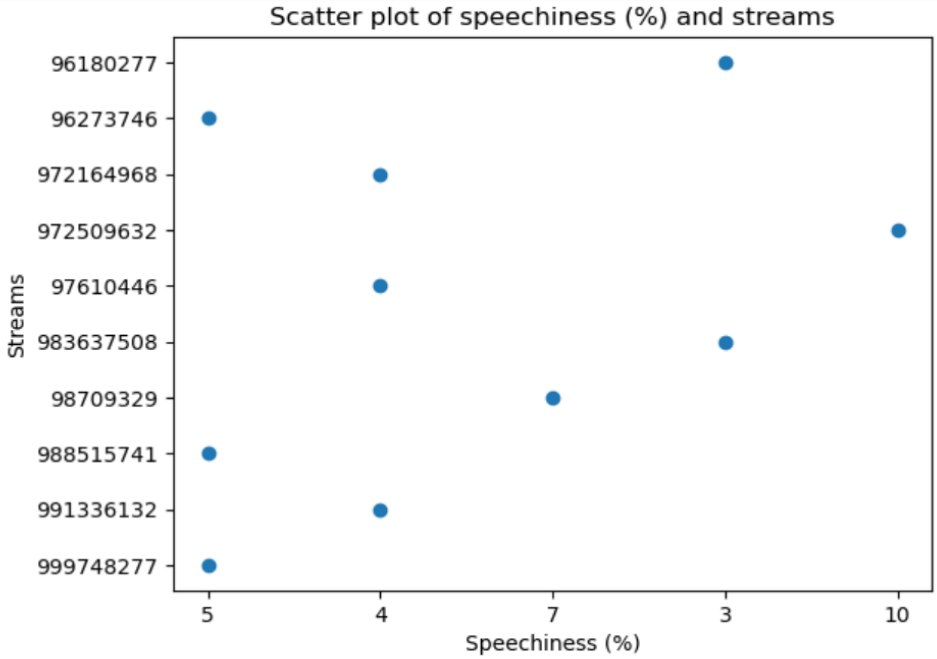



```
plt.scatter(data["speechiness_%"], data["streams"])
plt.title("Scatter plot of speechiness (%) and streams")
plt.xlabel("Speechiness (%)")
plt.ylabel("Streams")
plt.show()
```

Result:

Computer Science and Mathematics

2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

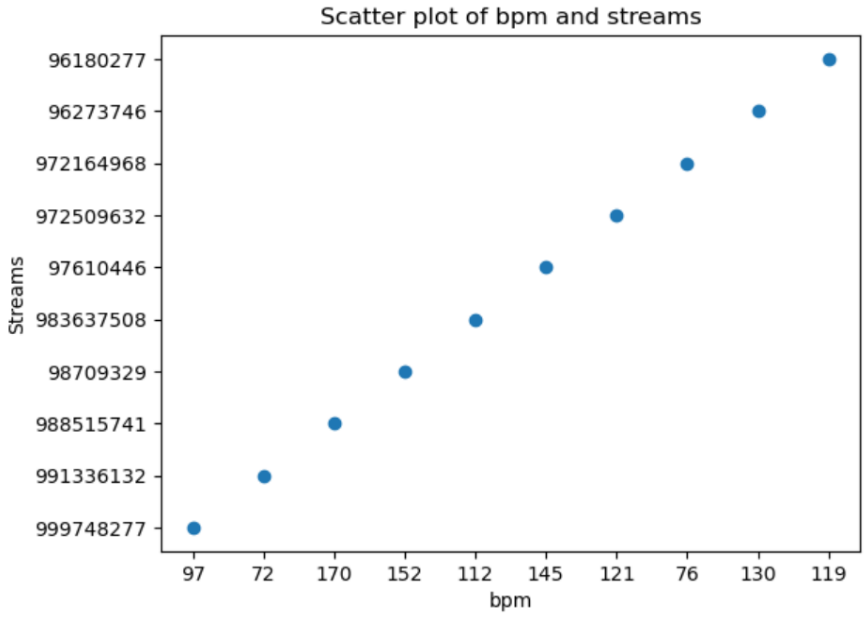



```
plt.scatter(data["bpm"], data["streams"])
plt.title("Scatter plot of bpm and streams")
plt.xlabel("bpm")
plt.ylabel("Streams")
plt.show()
```

Result:

Computer Science and Mathematics

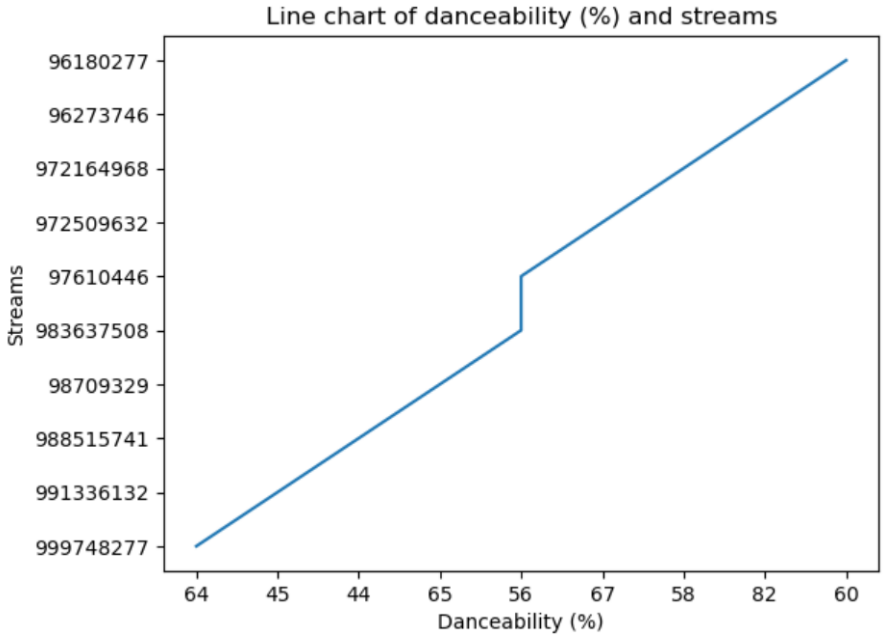
2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama




```
# Line chart

plt.plot(data["danceability_%"], data["streams"])
plt.title("Line chart of danceability (%) and streams")
plt.xlabel("Danceability (%)")
plt.ylabel("Streams")
plt.show()
```

Result:

2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

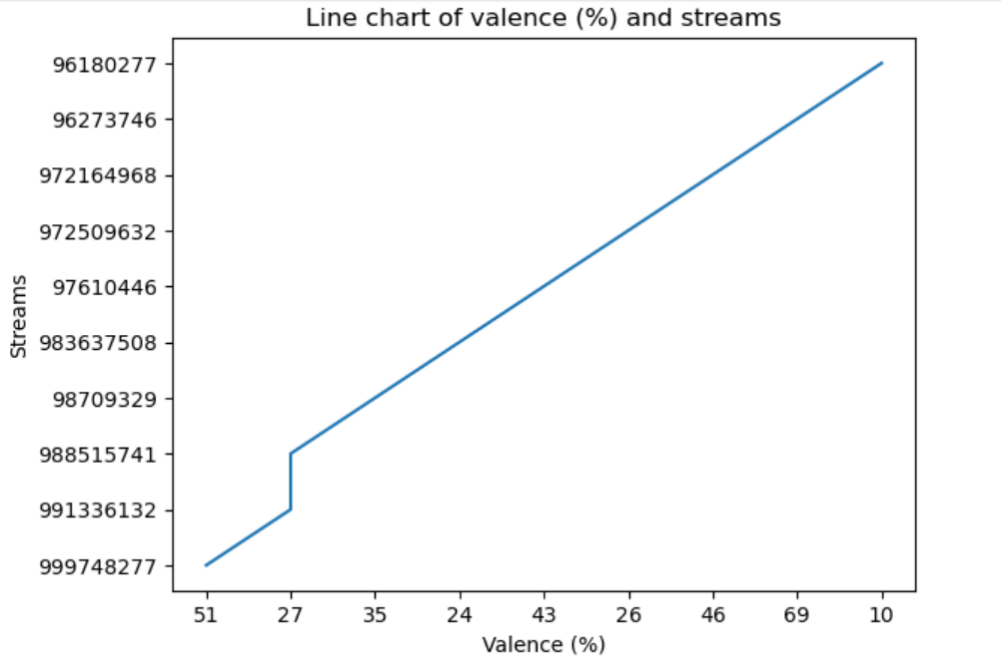



```
plt.plot(data["valence_%"], data["streams"])
plt.title("Line chart of valence (%) and streams")
plt.xlabel("Valence (%)")
plt.ylabel("Streams")
plt.show()
```

Result:

Computer Science and Mathematics

2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

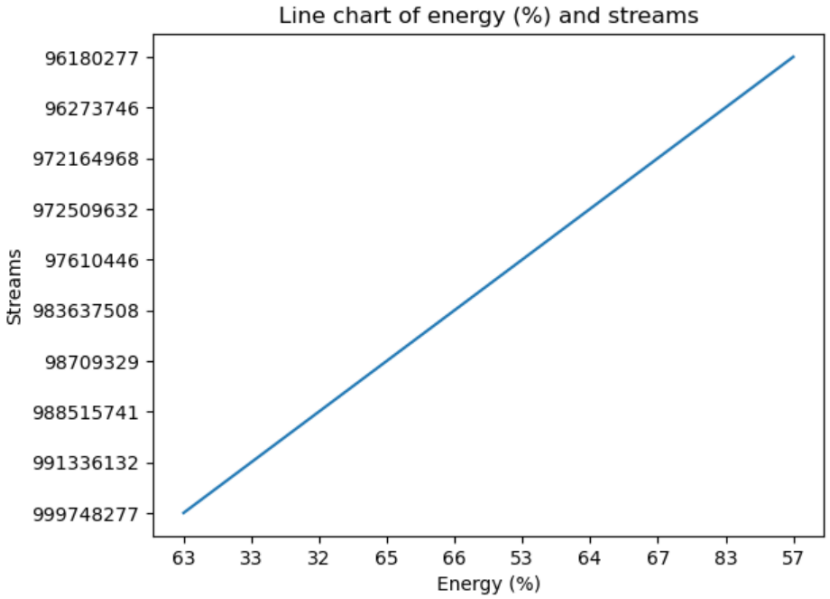



```
plt.plot(data["energy_%"], data["streams"])
plt.title("Line chart of energy (%) and streams")
plt.xlabel("Energy (%)")
plt.ylabel("Streams")
plt.show()
```

Result:

Computer Science and Mathematics

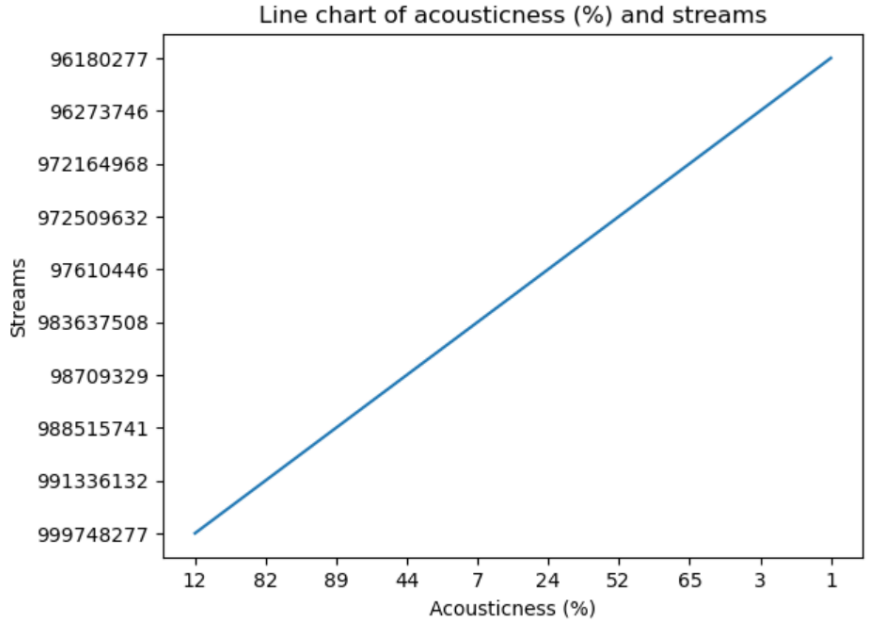
2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama




```
plt.plot(data["acousticness_%"], data["streams"])
plt.title("Line chart of acousticness (%) and streams")
plt.xlabel("Acousticness (%)")
plt.ylabel("Streams")
plt.show()
```

Result:

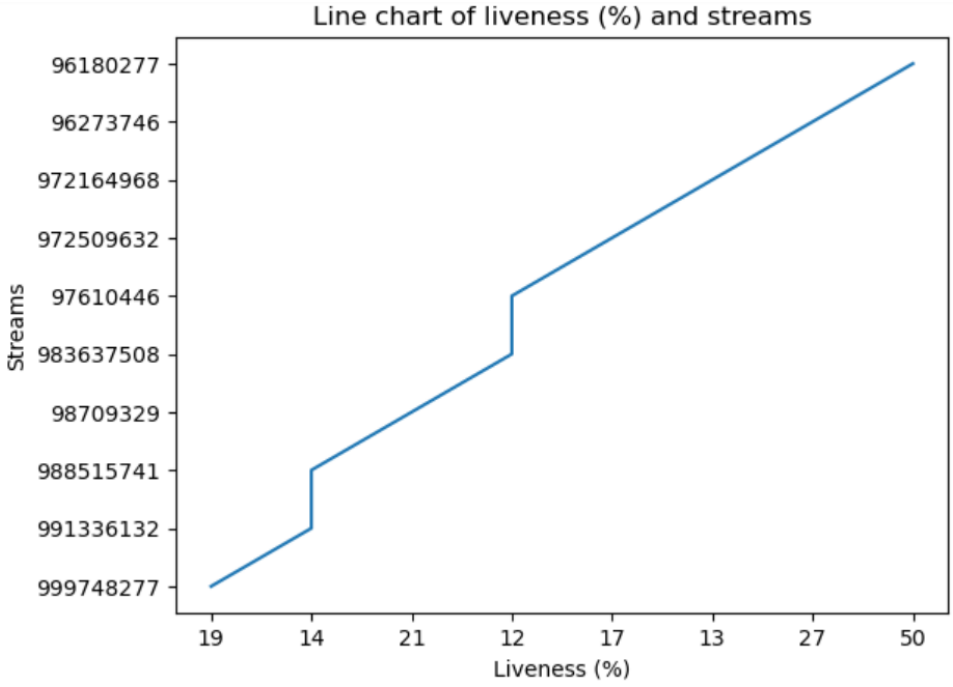
2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama




```
plt.plot(data["liveness_%"], data["streams"])
plt.title("Line chart of liveness (%) and streams")
plt.xlabel("Liveness (%)")
plt.ylabel("Streams")
plt.show()
```

Result:

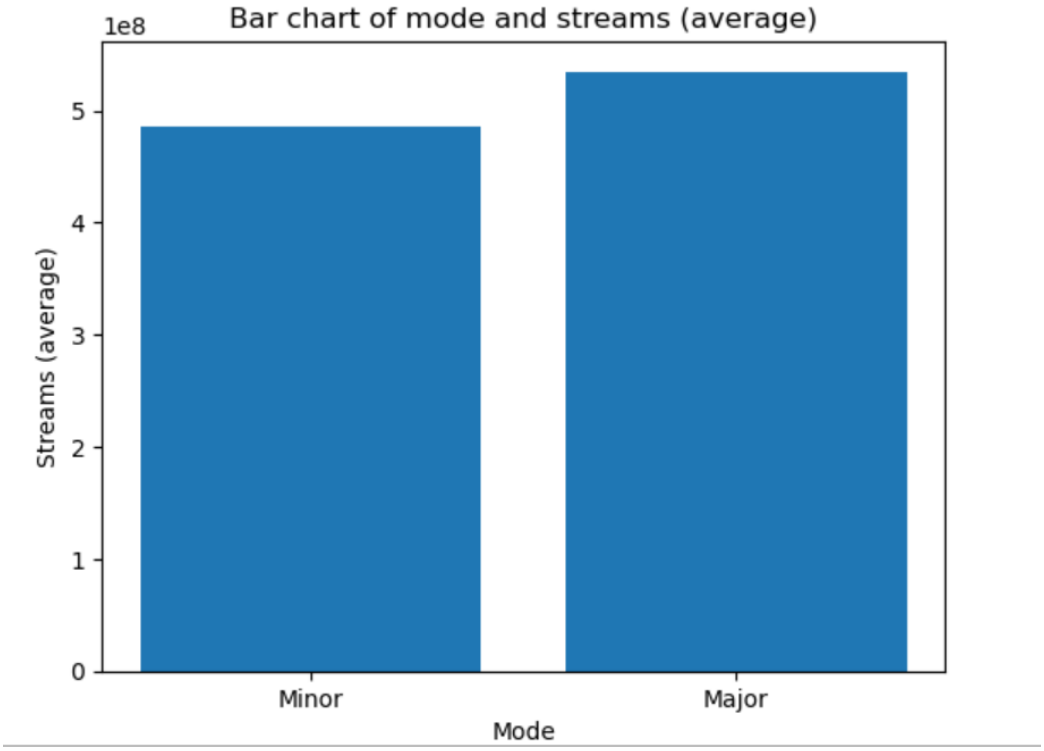
2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama




```
data2 = spark.sql(
''' SELECT mode, AVG(streams) AS average_streams FROM spotifyData
    GROUP BY mode '''
).toPandas()

# Bar chart
plt.bar(data2["mode"], data2["average_streams"])
plt.title("Bar chart of mode and streams (average)")
plt.xlabel("Mode")
plt.ylabel("Streams (average)")
plt.show()
```

Result:

2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama



8. Regression

Code:

```
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import LinearRegression
from pyspark.ml.evaluation import RegressionEvaluator

df = df.withColumn("streams", df["streams"].cast("double"))

assembler = VectorAssembler(
    inputCols = ["danceability_%", "valence_%", "energy_%"],
    outputCol="features"
)

%pyspark
data = assembler.transform(df)
final_data = data.select("features", "streams")
```

2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

```
train_data, test_data = final_data.randomSplit([0.7, 0.3])

lr = LinearRegression(featuresCol = 'features', labelCol = 'streams')
lr_model = lr.fit(train_data)

test_results = lr_model.transform(test_data)

%pyspark
evaluator = RegressionEvaluator(labelCol = "streams", predictionCol = "prediction",
metricName = "rmse")
rmse = evaluator.evaluate(test_results)
print("RMSE: ", rmse)

print("Coefficients: ", lr_model.coefficients)
print("Intecepts: ", lr_model.intercept)

%pyspark
test_results.select("prediction", "streams").show()
```

Results:

```
RMSE:  504085918.6618116
Coefficients:  [-3281541.4089495777,-663268.66696044,-280364.74571030756]
Intecepts:  800405432.9413372
```

Computer Science and Mathematics

2501972493 Gaizkia Adeline Atmaka
2501972625 Shelly Alfianda
2502003895 Glory Daniella
2540124450 Bernard Hugo
2502034912 Luthfi Izza Pratama

prediction	streams
6.544181356949354E8	6.63832097E8
6.98680388007961E8	2.9732896E8
6.856699932355715E8	1.21913181E8
6.5410749163258E8	4.21135627E8
6.713875666910793E8	1.056760045E9
6.655498072469023E8	2.0245286E8
6.579527063318149E8	7.26434358E8
6.516276371888299E8	6.00976848E8
6.443109010194528E8	2.42767149E8
6.584797110942295E8	1.8610431E8
6.32003182185955E8	8.24420218E8
6.558482066643715E8	2.84908316E8
6.60142985337697E8	6.24101957E8
6.545356904234909E8	8.41749534E8