PySpark Essentials

Spark Session

Code:

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("Week2").getOrCreate()
```

Use Case:
Sets up the environment to load and analyze the spotify.csv dataset with music track details.

Spark Config

Code:

```
spark.conf.set("spark.sql.shuffle.partitions", "200")
spark.conf.set("spark.driver.memory", "2g")
```

Use Case:
Optimizes partition size and memory usage when processing large datasets like spotify.csv for ETL pipelines.

Read Data (CSV, JSON, Parquet)

CSV Code:

```
df = spark.read.option("header", "true").option("inferSchema",
"true").csv("spotify.csv")
```

Use Case:
Imports the `spotify.csv` file to analyze music features such as popularity and danceability.

JSON Code:

```
df_json = spark.read.option("header", "true").json("spotify.json")
```

Use Case:
Loads JSON data from an external API providing artist biographies or additional track metadata.

Parquet Code:

```
df_parquet = spark.read.parquet("spotify.parquet")
```

Use Case:
Reads optimized Parquet files for faster querying in an ETL workflow.

Rename Columns

Code:

```
df_renamed = df.withColumnRenamed("track_name", "song_title")
```

Use Case:
Renames "track_name" to "song_title" to align with a standardized naming convention in the dataset.

Create or Update Column

Code:

```
from pyspark.sql.functions import col
df_updated = df.withColumn("duration_min", col("duration_ms") /
60000)
```

Use Case:
Creates a "duration_min" column by converting "duration_ms" to minutes for user-friendly reporting.

Apply PySpark Functions

Split Code:

```
from pyspark.sql.functions import split
df_split = df.withColumn("artist_array", split(col("artists"),
";"))
```

Use Case:
Splits multi-artist entries (e.g., "Ingrid Michaelson;Rock") into an array for individual artist analysis.

Filter Code:

```
df_filtered = df.filter(col("popularity") > 70)
```

Use Case:
Filters the dataset to focus on highly popular tracks (popularity > 70) for a top-hits report.

Select Columns Code:

```
df_selected = df.select(col("track_name"), col("popularity"))
```

Use Case:
Selects specific columns to create a subset DataFrame for targeted analysis.

Cast Column Code:

```
df_casted = df.withColumn("popularity",
col("popularity").cast("integer"))
```

Use Case:
Casts the "popularity" column to integer type for integer-specific operations or storage.

Group By and Aggregate Functions

Count Code:

```
df_grouped = df.groupBy("artists").count()
```

Use Case:
Counts the number of tracks per artist, e.g., to identify how many tracks Jason Mraz has.

Average Code:

```
from pyspark.sql.functions import avg
df_avg =
df.groupBy("key").agg(avg("popularity").alias("avg_popularity"))
```

Use Case:
Computes the average popularity for each musical key to analyze genre trends.

Collect List Code:

```
from pyspark.sql.functions import collect_list
df_list =
df.groupBy("artists").agg(collect_list("track_name").alias("tracks
"))
```

Use Case:
Collects all track names into a list per artist for a comprehensive artist profile.

Join Types

Inner Join Code:

```
df_joined = df.join(df_spotify, "track_id", "inner")
```

Use Case:
Joins `one data frame` with another data frame to retain only matching track IDs for enriched analysis.

Left Join Code:

```
df_left = df.join(df_spotify, "track_id", "left")
```

Use Case:
Keeps all tracks from `df`, adding Spotify data where matches exist, for a complete dataset view.

Anti Join Code:

```
df_anti = df.join(df_spotify, "track_id", "left_anti")
```

Use Case:
Identifies tracks in `df` that are not in df_spotify for data gap analysis.

Broadcast Join Code:

```
from pyspark.sql.functions import broadcast
df_broadcast = df.join(broadcast(df_spotify), "track_id", "inner")
```

Use Case:
Optimizes joining a small DataFrame with a larger one by broadcasting the smaller table.