03 Working on DataFrames -KirkYagami👨‍💻🕵️

```
## PySpark Session Initialization
from pyspark.sql import SparkSession

# Initialize Spark Session
ss = SparkSession \
    .builder \
    .appName("DF_Analysis") \
    .getOrCreate()
```

Data Loading

```
# Load the CSV data into a DataFrame
disney_raw = ss.read \
    .format("csv") \
    .option("header", "true") \
    .option("inferSchema", "true") \
    .option("multiline", "true") \
    .option("quote", "\"") \
    .option("escape", "\"") \
    .load("disney_plus_shows.csv")
```

Data Inspection

1. Print Schema

```
# 1. Print the schema of the DataFrame
disney_raw.printSchema()
```

2. Display First Few Rows

```
# 2. Display the first few rows of the DataFrame
disney_raw.show(1, truncate=False, vertical=True)
```

3. Basic Statistical Information

```
# 3. Generate basic statistical summary
df_summary = disney_raw.describe().toPandas()
```

Data Cleansing

4. Convert and Handle Data Types

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

# Convert 'year' to integer and handle incorrect formats
disney_raw = disney_raw.withColumn(
    "year",
    when(col("year").rlike("^\d{4}$"), col("year").cast("integer"))
    .otherwise(None)
)

# Convert 'released_at' to a date type, handling 'N/A' values
disney_raw = disney_raw.withColumn(
    "released_at",
    when(col("released_at").isNotNull() & ~col("released_at").rlike("N/A"),
         col("released_at").cast("date"))
    .otherwise(None)
)
```

Data Selection

5. Select Specific Columns

```
# 4. Selecting specific columns
disney_raw.select("title", "year", "imdb_rating").show(5, truncate=False)
```

6. Filtering Data

```
# 5. Filtering data based on a condition
disney_raw.select("title", "year", "imdb_rating") \
          .filter(disney_raw.year > 2017).show(5)
```

7. Filter with Multiple Conditions

```
# 6. Filter data with multiple conditions
disney_raw.select("title", "year", "imdb_rating") \
          .filter((disney_raw.imdb_rating > 8.0) & (disney_raw.year > 2019)) \
          .show(5)
```

Data Manipulation

8. Renaming Columns

```
# 7. Renaming a column
df_renamed = disney_raw.withColumnRenamed("imdb_rating", "rating")
```

9. Dropping Columns

```
# 8. Dropping a single column
df_dropped_plot = df_renamed.drop("plot")
```

```
# 9. Dropping multiple columns
df_dropped_2_cols = df_renamed.drop("plot", "rated")
```

10. Dropping Rows

```
# 10. Dropping rows with any null values
df_no_nulls = disney_raw.dropna()
```

```
# 11. Dropping rows with null values in specific columns
df_cleaned = disney_raw.dropna(subset=["title", "imdb_rating"])
```

11. Drop Duplicates

```
# 13. Dropping duplicate rows
df_cleaned = disney_raw.dropDuplicates()
```

```
# 14. Dropping duplicates in specific columns
df_cleaned = disney_raw.dropDuplicates(["title"])
```

Aggregation and Grouping

12. Group By and Count

```
# 15. Grouping data by genre and counting occurrences
from pyspark.sql import functions as F

disney_raw.groupBy("genre") \
    .count() \
    .orderBy(F.desc("count")) \
    .show(10, truncate=False)
```

```
# Grouping data by type and counting occurrences
df2.groupBy("type").count().show()
```

13. Approximate Count Distinct

```
# Approximate distinct count of 'genre'
df2.select(F.approx_count_distinct("genre")).show()
```

```
# Count distinct values in 'genre'
df2.select(F.countDistinct("genre")).show()
```

```
# Approximate distinct count with specified relative standard deviation
df2.select(F.approx_count_distinct("genre", 0.05)).show()
```

14. Average

```
# 3. Average rating
df2.select(F.avg("imdb_rating")).show()
```

15. Collect List

```
# Collect list of all titles
title_list = df2.agg(F.collect_list("title"))
```

```
title_list.count()
```

16. Correlation and Covariance

```
# 5. Correlation between imdb_rating and imdb_votes
df2.select(F.corr("imdb_rating", "imdb_votes")).show()
```

```
# Covariance population between imdb_rating and imdb_votes
df2.select(F.covar_pop("imdb_rating", "imdb_votes")).show()
```

```
# Covariance sample between imdb_rating and imdb_votes
df2.select(F.covar_samp("imdb_rating", "imdb_votes")).show()
```

17. First and Last

```
# First non-null value of imdb_rating
df2.select(F.first("imdb_rating")).show()
```

```
# Last non-null value of imdb_rating
df2.select(F.last("imdb_rating")).show()
```

18. Sum and Sum Distinct

```
# Sum of imdb_votes
df2.select(F.sum("imdb_votes")).show()
```

```
# Sum of distinct imdb_votes
df2.select(F.sum_distinct("imdb_votes")).show()
```

```
# Collect set of imdb_votes grouped by genre
df2.groupBy("genre").agg(F.collect_set("imdb_votes").alias('votes_list')).show(100,
truncate=False)
```

Expressions

19. Column Operations

```
# Increase imdb_rating by 1
df2.withColumn("adjusted_rating", (F.col("imdb_rating") + 1)) \
            .select("title", "imdb_rating","adjusted_rating").show(2)
```

20. Substring and Split

```
# Extract release year using substring
df2.withColumn("release_year",
               F.substring("released_at", -4, 4)).select("title",
                                    "imdb_rating",
                                    "release_year",
                                    "released_at").show(2)
```

```
# Extract release year using split
df2.withColumn("release_year",
               F.split(("released_at"), " ").getItem(2)).select("title",
                                    "imdb_rating",
                                    "release_year",
                                    "released_at").show(2)
```

21. Conditional Column Creation

```
# Create a new column 'rating_category' based on imdb_rating
df2.withColumn('rating_category',
```

```
            F.when(F.col("imdb_rating") >= 8, "Excellent")
             .when(F.col("imdb_rating") >= 6, "Good")
             .otherwise("Average") \
            ).select("title",
                     "imdb_rating",
                     "rating_category",
                     "released_at").show(2)
```

User-Defined Functions (UDFs)

22. UDF Example

```
# Define a UDF to add a prefix to the title
from pyspark.sql.types import StringType

def add_prefix(title):
    return "Disney's " + title

add_prefix_udf = F.udf(add_prefix, StringType())

df3 = df2.dropna()
df3.withColumn("modified_title",
               add_prefix_udf(F.col("title"))).select("title",
                        "modified_title",
                       "imdb_rating",
                       "released_at").show(2)
```