Ejercicio 1 :

from pyspark.sql import SparkSession  
# Create a SparkSession  
spark = SparkSession.builder.appName("PySparkExample").getOrCreate()  
# Create a Data Frame  
data = [("Alice", 25), ("Bob", 30), ("Charlie", 35)]  
df = spark.createDataFrame(data, ["Name", "Age"])  
# Print the DataFrame  
df.show()  
# Perform a simple transformation  
df\_filtered = df.filter(df["Age"] > 30)  
# Print the filtered DataFrame  
df\_filtered.show()  
# Stop the SparkSession  
spark.stop()

Ejercicio 2 :

# Selecting specific columns  
df.select("Name", "Age").show()

# Adding a new column  
df\_with\_gender = df.withColumn("Gender", "Female")  
df\_with\_gender.show()

# Filtering rows based on a condition  
df\_filtered = df.filter(df["Age"] > 30)  
df\_filtered.show()

# Sorting the DataFrame  
df\_sorted = df.orderBy(df["Age"])  
df\_sorted.show()

# Grouping and aggregating data  
df\_grouped = df.groupBy("Gender").agg({"Age": "avg"})  
df\_grouped.show()

Ejercicio 3 : (Este ejercicio debe realizarse con archivos csv,json,parquet)

# Reading a CSV file  
df\_csv = spark.read.csv("data.csv", header=True, inferSchema=True)  
# Writing a DataFrame to a CSV file  
df\_csv.write.csv("output.csv", header=True)

# Reading a Parquet file  
df\_parquet = spark.read.parquet("data.parquet")  
# Writing a DataFrame to a Parquet file  
df\_parquet.write.parquet("output.parquet")

# Reading a JSON file  
df\_json = spark.read.json("data.json")  
# Writing a DataFrame to a JSON file  
df\_json.write.json("output.json")

Ejercicio 4 :

# Reading data from a database table  
df\_jdbc = spark.read.format("jdbc").option("url", "jdbc:postgresql://localhost:5432/mydatabase") \  
.option("dbtable", "mytable").option("user", "myuser").option("password", "mypassword").load()  
# Writing data to a database table  
df\_jdbc.write.format("jdbc").option("url", "jdbc:postgresql://localhost:5432/mydatabase") \  
.option("dbtable", "mytable").option("user", "myuser").option("password", "mypassword").mode("append").save()

Ejercicio 5 :

from pyspark.sql.types import StructType, StructField, StringType, IntegerType  
# Define the schema  
schema = StructType([  
StructField("Name", StringType(), nullable=False),  
StructField("Age", IntegerType(), nullable=True),  
])  
# Create a DataFrame with the defined schema  
df\_with\_schema = spark.createDataFrame(data, schema)

# Renaming a column  
df\_renamed = df.withColumnRenamed("Name", "Full Name")  
df\_renamed.show()

# Dropping a column  
df\_dropped = df.drop("Age")  
df\_dropped.show()

# Checking for null values  
df\_null = df.filter(df["Age"].isNull())  
# Filling null values with a default value  
df\_filled = df.fillna({"Age": 0})  
# Dropping rows with null values  
df\_no\_null = df.dropna()

Ejercicio 6 :

# Dropping rows with missing data  
df\_no\_missing = df.dropna()

# Filling missing data with a default value  
df\_filled = df.fillna(0)  
# Filling missing data with the mean value of a column  
mean\_age = df.select("Age").agg({"Age": "mean"}).first()[0]  
df\_filled = df.fillna

Ejercicio 7 :

# Grouping and aggregating data  
df\_grouped = df.groupBy("Gender").agg({"Age": "avg", "Salary": "sum"})  
df\_grouped.show()

# Calculating descriptive statistics  
df\_stats = df.describe(["Age", "Salary"])  
df\_stats.show()

# Creating a pivot table  
df\_pivot = df.groupBy("Gender").pivot("City").agg({"Salary": "sum"})  
df\_pivot.show()

# Performing an inner join  
df\_joined = df1.join(df2, df1["ID"] == df2["ID"], "inner")  
df\_joined.show()

# Performing an outer join  
df\_joined = df1.join(df2, df1["ID"] == df2["ID"], "outer")  
df\_joined.show()

# Combining DataFrames using union  
df\_combined = df1.union(df2)  
df\_combined.show()

Ejercicio 8 :

from pyspark.sql.functions import from\_json  
from pyspark.sql.types import StructType, StringType, IntegerType  
# Define the schema for the streaming data  
schema = StructType().add("name", StringType()).add("age", IntegerType())  
# Create a Streaming DataFrame from a Kafka topic  
streaming\_df = spark.readStream.format("kafka") \  
.option("kafka.bootstrap.servers", "localhost:9092") \  
.option("subscribe", "topic-name") \  
.load() \  
.select(from\_json(col("value").cast("string"), schema).alias("data"))

# Write the transformed streaming data to the console  
query = streaming\_df.writeStream.outputMode("append").format("console").start()  
query.awaitTermination()