# Azure file 1

storageAccountKey = ''

spark.conf.set("fs.azure.account.key.demo1storageaccounts.dfs.core.windows.net",storageAccountKey)

import pandas

sourceFileURL = 'https://retailpricing.blob.core.windows.net/labs/lab1/PW\_MW\_DR\_01012023.csv'

bronzelayerCSVFilePath = 'abfss://bronze-labs@demo1storageaccounts.dfs.core.windows.net/bronze/daily-pricing/csv'

sourceFilePandasDF = pandas.read\_csv(sourceFileURL)

spark.createDataFrame(sourceFilePandasDF)

sourceFileSparkDF = spark.createDataFrame(sourceFilePandasDF)

print(sourceFilePandasDF )

display(sourceFileSparkDF)

sourceFileSparkDF.write.mode("overwrite").csv(bronzelayerCSVFilePath)

# Azure File 2

storageAccountKey = ''

spark.conf.set("fs.azure.account.key.demo1storageaccounts.dfs.core.windows.net",storageAccountKey)

sourceCSVFilePath = "abfss://bronze-labs@demo1storageaccounts.dfs.core.windows.net/bronze/daily-pricing/csv"

targetJsonFilePath = "abfss://bronze-labs@demo1storageaccounts.dfs.core.windows.net/bronze/daily-pricing/json"

sourceCSVFileDF=(

spark.

read.

option("header", "true").

csv(

sourceCSVFilePath

)

)

from pyspark.sql.types import \*

sourceCSVFileSchema=StructType([

StructField("DATE\_OF\_PRICING", StringType(), True),

StructField("ROW\_ID", IntegerType(), True),

StructField("STATE\_NAME", StringType(), True),

StructField("MARKET\_NAME", StringType(), True) ,

StructField("PRODUCTGROUP\_NAME", StringType(), True),

StructField("PRODUCT\_NAME", StringType(), True),

StructField("VARIETY", StringType(), True),

StructField("ORIGIN", StringType(), True),

StructField("ARRIVAL\_IN\_TONNES", DecimalType(10,2),True),

StructField("MINIMUM\_PRICE", StringType(),True),

StructField("MAXIMUM\_PRICE",StringType(),True),

StructField("MODAL\_PRICE", DecimalType(10,2),True)

])

sourceCSVFileDF=spark.read.schema(sourceCSVFileSchema).csv(sourceCSVFilePath)

sourceCSVFileDF.write.mode("overwrite")

sourceCSVFileDF=(

spark.

read.

schema(sourceCSVFileSchema).

csv(sourceCSVFilePath)

)

display(sourceCSVFileDF)

sourceCSVFileDF.printSchema()

(

sourceCSVFileDF.

write.

mode("overwrite").

json(targetJsonFilePath)

)

# Azure File 3

storageAccountKey = ''

spark.conf.set("fs.azure.account.key.demo1storageaccounts.dfs.core.windows.net",storageAccountKey)

sourceCSVFilePath = "abfss://bronze-labs@demo1storageaccounts.dfs.core.windows.net/bronze/daily-pricing/csv"

targetPARQUETFilePath = 'abfss://bronze-labs@demo1storageaccounts.dfs.core.windows.net/bronze/daily-pricing/parquet'

sourceCSVFileDF=(spark.

read.

load(sourceCSVFilePath,format='csv')

)

from pyspark.sql.types import \*

sourceCSVFileSchema = StructType([

StructField("DATE\_OF\_PRICING",StringType(),True),

StructField("ROW\_ID",IntegerType(),True),

StructField("STATE\_NAME",StringType(),True),

StructField("MARKET\_NAME",StringType(),True),

StructField("PRODUCTGROUP\_NAME",StringType(),True),

StructField("PRODUCT\_NAME",StringType(),True),

StructField("VARIETY",StringType(),True),

StructField("ORIGIN",StringType(),True),

StructField("ARRIVAL\_IN\_TONNES",DecimalType(10,2),True),

StructField("MINIMUM\_PRICE",StringType(),True),

StructField("MAXIMUM\_PRICE",StringType(),True),

StructField("MODAL\_PRICE",StringType(),True),

])

sourceCSVFileDF = ( spark.

read.

schema(sourceCSVFileSchema).

csv(sourceCSVFilePath)

)

display(sourceCSVFileDF)

from pyspark.sql.functions import col

sourceCSVFileDF.withColumn("ARRIVAL\_IN\_KILOGRAMS",col("ARRIVAL\_IN\_TONNES")\*1000)

sourceCSVFileTransDF=(sourceCSVFileDF.

withColumn("ARRIVAL\_IN\_KILOGRAMS",col("ARRIVAL\_IN\_TONNES")\*1000)

)

display(sourceCSVFileTransDF)

from pyspark.sql.functions import sum

(sourceCSVFileTransDF.

groupBy("STATE\_NAME","PRODUCT\_NAME").

agg(sum("ARRIVAL\_IN\_KILOGRAMS")).show())

from pyspark.sql.functions import sum

(sourceCSVFileTransDF.

groupBy("STATE\_NAME","PRODUCT\_NAME").

agg(sum("ARRIVAL\_IN\_KILOGRAMS").alias("TOTAL\_ARRIVALS\_IN\_KILOGRAMS")).show())

from pyspark.sql.functions import sum,desc

(sourceCSVFileTransDF.

groupBy("STATE\_NAME","PRODUCT\_NAME").

agg(sum("ARRIVAL\_IN\_KILOGRAMS").alias("TOTAL\_ARRIVALS\_IN\_KILOGRAMS")).

orderBy(desc("TOTAL\_ARRIVALS\_IN\_KILOGRAMS")).

show())

(sourceCSVFileTransDF.write

.option("mergeSchema","true")

.mode("overwrite")

.format("delta")

.save(targetPARQUETFilePath))

(spark.

read.load(targetPARQUETFilePath).

display()

)

# Azure File 4

storageAccountKey = ''

spark.conf.set("fs.azure.account.key.demo1storageaccounts.dfs.core.windows.net",storageAccountKey)

sourceCSVFilePath = 'abfss://bronze-labs@demo1storageaccounts.dfs.core.windows.net/bronze/daily-pricing/csv'

sourcePARQUETFilePath='abfss://bronze-labs@demo1storageaccounts.dfs.core.windows.net/bronze/daily-pricing/parquet'

sourcePARQUETFileDF=(spark

.read

.load(sourcePARQUETFilePath)

)

from pyspark.sql.functions import current\_timestamp

sourcePARQUETTransFileDF=(sourcePARQUETFileDF

.withColumn("datalake\_file\_load\_date",current\_timestamp())

)

display(sourcePARQUETTransFileDF)

from pyspark.sql.functions import year,quarter

(sourcePARQUETTransFileDF

.select("datalake\_file\_load\_date")

.withColumn("datalake\_file\_load\_year",year("datalake\_file\_load\_date"))

.withColumn("datalake\_file\_load\_quarter", quarter("datalake\_file\_load\_date"))

.show()

)

from pyspark.sql.functions import month,dayofmonth

(sourcePARQUETTransFileDF

.select("datalake\_file\_load\_date")

.withColumn("datalake\_file\_load\_month",month("datalake\_file\_load\_date"))

.withColumn("datalake\_file\_load\_day",dayofmonth("datalake\_file\_load\_date"))

.show()

)

from pyspark.sql.functions import concat

from sklearn.model\_selection import train\_test\_split

(sourcePARQUETTransFileDF

.select("datalake\_file\_load\_date")

.withColumn("datalake\_file\_load\_date\_formatted",concat(year("datalake\_file\_load\_date"),- month("datalake\_file\_load\_date"),- dayofmonth("datalake\_file\_load\_date")))

.show()

)

from pyspark.sql.functions import date\_format

(sourcePARQUETTransFileDF

.select("datalake\_file\_load\_date")

.withColumn("datalake\_file\_load\_date",date\_format("datalake\_file\_load\_date","yyyy-MM-dd"))

.show()

)

sourcePARQUETTransFileDF.printSchema()

from pyspark.sql.functions import to\_date

sourcePARQUETTransFileDF1=(sourcePARQUETTransFileDF

.select("DATE\_OF\_PRICING")

.withColumn("PRICING\_DATE",to\_date("DATE\_OF\_PRICING","dd/MM/yyyy"))

)

(sourcePARQUETTransFileDF1

.select("PRICING\_DATE")

.withColumn("PRICING\_DATE\_FROMATTED",date\_format("PRICING\_DATE","dd/MM/yyyy"))

.show()

)

# Azure Assignment File

storageAccountKey = ''

spark.conf.set("fs.azure.account.key.demo1storageaccounts.dfs.core.windows.net",storageAccountKey)

sourceFileURL = 'https://date.nager.at/api/v3/PublicHolidays/2025/US'

azureStoragePath = 'abfss://assignmentdata@demo1storageaccounts.dfs.core.windows.net/bronze/daily-pricing/csv'

import pandas as pd

import requests

response = requests.get(sourceFileURL)

data=response.json()

df = pd.DataFrame(data)

display(df)

spark\_df=spark.createDataFrame(df)

spark\_df.printSchema()

spark\_df.show(5)

from pyspark.sql.functions import col, to\_date, date\_format, month

spark\_df = spark\_df.withColumn("date", to\_date(col("date"), "yyyy-MM-dd"))

spark\_df = spark\_df.withColumn("weekday", date\_format(col("date"), "EEEE"))

spark\_df = spark\_df.withColumn("holiday\_month", month("date"))

display(spark\_df)

from pyspark.sql.functions import count

weekday\_count = spark\_df.groupBy("weekday").agg(count("\*").alias("holiday\_count"))

weekday\_count.orderBy(col("holiday\_count").desc()).show()

month\_count = spark\_df.groupBy("holiday\_month").agg(count("\*").alias("month\_holiday\_count"))

month\_count.filter(col("month\_holiday\_count") > 1).orderBy("holiday\_month").show()

from pyspark.sql.functions import concat\_ws

# Convert array<string> columns to single string

flattened\_df = (

spark\_df.withColumn('launchYear', col('launchYear').cast('string'))

.withColumn("counties", concat\_ws(",", "counties"))

.withColumn("types", concat\_ws(",", "types"))

)

# Now write to CSV

flattened\_df.write.mode("overwrite").csv(azureStoragePath, header=True)

output\_path\_json = "abfss://assignmentdata@demo1storageaccounts.dfs.core.windows.net/bronze/daily-pricing/json"

output\_path\_parquet = "abfss://assignmentdata@demo1storageaccounts.dfs.core.windows.net/bronze/daily-pricing/parquet"

flattened\_df.write.mode("overwrite").json(output\_path\_json)

flattened\_df.write.mode("overwrite").parquet(output\_path\_parquet)s