Module 2 — Spark SQL & DataFrames: Relational Database on AWS EC2

Soham Deshkhaire

Table of contents

{python} from pyspark.sql import SparkSession import pandas as pd import seaborn as sns import matplotlib.pyplot as plt import os

# 1. Start a Spark session

spark = SparkSession.builder.appName(“JobPostingsAnalysis”).getOrCreate()

# 2. Load the CSV file into a Spark DataFrame

df = spark.read.option(“header”, “true”).option(“inferSchema”, “true”).option(“multiLine”,“true”).option(“escape”, “"”).csv(“lightcast\_job\_postings.csv”)

# 3. Register the DataFrame as a temporary SQL view

df.createOrReplaceTempView(“job\_postings”)

# 4. Show Schema and Sample Data

print(“—This is Diagnostic check, No need to print it in the final doc—”)

# 5. comment the lines below when rendering the submission

df.printSchema() df.show(5)

# 6. Create output directory if it doesn’t exist

if not os.path.exists(‘output’): os.makedirs(‘output’)

from pyspark.sql.functions import col, monotonically\_increasing\_id

industries\_df = df.select( col(“naics\_2022\_6”), col(“naics\_2022\_6\_name”), col(“soc\_5”).alias(“soc\_code”), col(“soc\_5\_name”).alias(“soc\_name”), col(“lot\_specialized\_occupation\_name”).alias(“specialized\_occupation”), col(“lot\_occupation\_group”).alias(“occupation\_group”) ).distinct().withColumn(“industry\_id”, monotonically\_increasing\_id())

industries\_df = industries\_df.select( “industry\_id”, “naics\_2022\_6”, “naics\_2022\_6\_name”, “soc\_code”, “soc\_name”, “specialized\_occupation”, “occupation\_group” )

industries\_df.show(5, truncate=False)

locations\_df = df.select( col(“location”), col(“city\_name”), col(“state\_name”), col(“county\_name”), col(“msa”), col(“msa\_name”), ).distinct().withColumn(“location\_id”, monotonically\_increasing\_id())

from pyspark.sql import SparkSession, functions as F from pyspark.sql.window import Window companies\_df = df.select( col(“company”), col(“company\_name”), col(“company\_raw”), col(“company\_is\_staffing”), ).distinct().where(F.col(“company”).isNotNull()).where(F.col(“company\_name”) != ‘Unclassified’).withColumn(“company\_id”, monotonically\_increasing\_id()) company\_df = companies\_df.dropna()

from pyspark.sql.functions import lower job\_postings\_df = df.select( col(“ID”).alias(“job\_postings\_id”), “title\_clean”,“employment\_type\_name”,“remote\_type\_name”,“remote\_type”,“body”, “min\_years\_experience”,“max\_years\_experience”,“salary”,“salary\_from”,“salary\_to”, “posted”,“expired”,“duration”, “company”,“location”,“naics\_2022\_6” ).dropDuplicates([“job\_postings\_id”])  
.join(companies\_df.select(“company”,“company\_id”), on=“company”, how=“left”)  
.join(locations\_df.select(“location”,“location\_id”), on=“location”, how=“left”)  
.join(industries\_df.select(“naics\_2022\_6”,“industry\_id”), on=“naics\_2022\_6”, how=“left”)

companies\_df.createOrReplaceTempView(“companies”) industries\_df.createOrReplaceTempView(“industries”) locations\_df.createOrReplaceTempView(“locations”) job\_postings\_df.createOrReplaceTempView(“job\_postings”)

## 6.1 QUERY 1 Industry-Specific Salary Trends Grouped by Job Titl

tech\_salary\_trends = spark.sql(““” SELECT i.naics\_2022\_6\_name AS industry\_name, i.specialized\_occupation, PERCENTILE\_APPROX(j.salary, 0.5) AS median\_salary FROM job\_postings j JOIN industries i ON j.naics\_2022\_6 = i.naics\_2022\_6 WHERE i.naics\_2022\_6 = 518210 AND j.salary IS NOT NULL AND j.salary > 0 GROUP BY i.naics\_2022\_6\_name, i.specialized\_occupation ORDER BY median\_salary DESC “““)

tech\_salary\_pd = tech\_salary\_trends.toPandas()

plt.figure(figsize=(12, 6)) ax = sns.barplot(data=tech\_salary\_pd, x=“median\_salary”, y=“specialized\_occupation”, palette=“viridis”, hue=“specialized\_occupation”, legend=False) ax.set\_xlabel(“Median Salary”) ax.set\_ylabel(“Specialized Occupation”) ax.set\_title(“Median Salary by Specialized Occupation (NAICS 518210)”) plt.tight\_layout() plt.savefig(‘output/tech\_salary\_trends.png’) plt.show()

# 7. QUERY 2 : Top 5 Companies with the Most Remote Jobs in California

remote\_jobs\_ca = spark.sql(““” SELECT COALESCE(c.company\_name) AS company\_name, COUNT(\*) AS remote\_jobs FROM job\_postings j JOIN companies c ON j.company\_id = c.company\_id JOIN locations l ON j.location\_id = l.location\_id WHERE (j.remote\_type = 1 OR LOWER(j.remote\_type\_name) LIKE ‘remote%’) AND l.state\_name = ‘California’ GROUP BY COALESCE(c.company\_name) ORDER BY remote\_jobs DESC LIMIT 10 “““)

remote\_jobs\_ca\_pd = remote\_jobs\_ca.toPandas()

remote\_jobs\_ca\_pd.head()

plt.figure(figsize=(10, 6)) sns.barplot(x=“remote\_jobs”, y=“company\_name”, data=remote\_jobs\_ca\_pd, palette=“viridis”, hue=“company\_name”, legend=False) plt.title(“Top 10 Companies Hiring Remote Jobs in California”) plt.xlabel(“Number of Remote Jobs”) plt.ylabel(“Company”) plt.tight\_layout() plt.savefig(‘output/remote\_jobs\_ca.png’) plt.show()

# 8. Query 3 :Monthly Job Posting Trends in California

from pyspark.sql.functions import to\_date, year, month

# 9. Add parsed date, year, month

job\_postings\_with\_date = job\_postings\_df.withColumn( “posted\_date”, to\_date(“posted”, “yyyy-MM-dd”) ).withColumn( “year”, year(“posted\_date”) ).withColumn( “month”, month(“posted\_date”) )

job\_postings\_with\_date.createOrReplaceTempView(“job\_postings\_with\_date”) locations\_df.createOrReplaceTempView(“locations”)

# 10. SQL Query

monthly\_trends\_ca = spark.sql(““” SELECT j.year, j.month, COUNT(\*) AS job\_count FROM job\_postings\_with\_date j JOIN locations l ON j.location\_id = l.location\_id WHERE l.state\_name = ‘California’ AND j.posted\_date IS NOT NULL GROUP BY j.year, j.month ORDER BY j.year, j.month “““)

monthly\_trends\_ca.show(10)

monthly\_trends\_ca\_pd = monthly\_trends\_ca.toPandas()

# 11. Convert month numbers to proper labels if needed

monthly\_trends\_ca\_pd[“month”] = monthly\_trends\_ca\_pd[“month”].astype(int)

plt.figure(figsize=(12, 6)) sns.lineplot(data=monthly\_trends\_ca\_pd, x=“month”, y=“job\_count”, hue=“year”, marker=‘o’) plt.title(“Monthly Job Posting Trends in California”) plt.xlabel(“Month”) plt.ylabel(“Number of Job Postings”) plt.legend(title=“Year”) plt.grid(True) plt.savefig(‘output/monthly\_trends\_ca.png’) plt.show()

# 12. Query 4 – Salary Comparisons Across Major US Cities

msa\_list = [14460,47900,35620,41860,42660,31080,19100,26420,12420,34980,28140,19740] salary\_comparison\_cities = spark.sql(f”“” WITH base AS ( SELECT l.MSA, l.MSA\_NAME, j.SALARY FROM job\_postings j JOIN locations l ON j.LOCATION\_ID = l.LOCATION\_ID WHERE j.SALARY IS NOT NULL AND j.SALARY > 0 AND l.MSA IN ({“,”.join(map(str, msa\_list))}) ), named AS ( SELECT CASE CAST(MSA AS INT) WHEN 14460 THEN ‘Boston’ WHEN 47900 THEN ‘Washington DC’ WHEN 35620 THEN ‘New York’ WHEN 41860 THEN ‘San Francisco’ WHEN 42660 THEN ‘Seattle’ WHEN 31080 THEN ‘Los Angeles’ WHEN 19100 THEN ‘Dallas’ WHEN 26420 THEN ‘Houston’ WHEN 12420 THEN ‘Austin’ WHEN 34980 THEN ‘Nashville’ WHEN 28140 THEN ‘Kansas City’ WHEN 19740 THEN ‘Denver’ ELSE COALESCE(MSA\_NAME,‘Unknown’) END AS metro, SALARY FROM base ) SELECT metro, ROUND(AVG(SALARY), 2) AS average\_salary, COUNT(\*) AS job\_count FROM named GROUP BY metro ORDER BY average\_salary DESC “““)

salary\_comparison\_cities.show()

salary\_comparison\_cities\_pd = salary\_comparison\_cities.toPandas()

plt.figure(figsize=(12, 8)) sns.barplot( data=salary\_comparison\_cities\_pd, x=“average\_salary”, y=“metro”, palette=“viridis”, hue=“metro”, # Set hue to the y-axis variable legend=False # Hide the legend ) plt.title(“Average Salary Comparison Across Major US Metro Areas”) plt.xlabel(“Average Salary”) plt.ylabel(“Metro Area”) plt.tight\_layout() plt.savefig(‘output/salary\_comparison\_cities.png’) plt.show()