import pandas as pd  
import numpy as np  
from pyspark.sql import SparkSession  
import plotly.express as px  
from pyspark.sql import functions as F

spark = SparkSession.builder.appName('assignment03').getOrCreate()  
df = spark.read.csv('data/lightcast\_job\_postings.csv', header=True, inferSchema=True)  
df.show(5)

# Clean salary columns and compute medians  
from pyspark.sql import functions as F  
  
df2 = df.withColumn("SALARY\_FROM", F.col("SALARY\_FROM").cast("double")) \  
 .withColumn("SALARY\_TO", F.col("SALARY\_TO").cast("double"))  
  
df2 = df2.withColumn("Average\_Salary", (F.col("SALARY\_FROM") + F.col("SALARY\_TO"))/2)  
  
median\_from = df2.approxQuantile("SALARY\_FROM", [0.5], 0.01)[0]  
median\_to = df2.approxQuantile("SALARY\_TO", [0.5], 0.01)[0]  
median\_avg = df2.approxQuantile("Average\_Salary", [0.5], 0.01)[0]  
  
print("Median From:", median\_from)  
print("Median To:", median\_to)  
print("Median Avg:", median\_avg)  
print("Row count:", df2.count())

# Clean education levels (remove newlines etc.)  
df2 = df2.withColumn("EDUCATION\_LEVELS\_NAME", F.regexp\_replace("EDUCATION\_LEVELS\_NAME", "[\n\r]", ""))  
df2.select("EDUCATION\_LEVELS\_NAME").distinct().show(20, truncate=False)

# Average salary by industry  
industry\_avg = df2.groupBy("INDUSTRY\_NAME").agg(F.avg("Average\_Salary").alias("avg\_salary"))  
industry\_pd = industry\_avg.toPandas().sort\_values("avg\_salary", ascending=False).head(15)  
  
import plotly.express as px  
fig = px.bar(industry\_pd, x="INDUSTRY\_NAME", y="avg\_salary",  
 title="Top 15 Industries by Average Salary")  
fig.show()

# Average salary by education level  
edu\_avg = df2.groupBy("EDUCATION\_LEVELS\_NAME").agg(F.avg("Average\_Salary").alias("avg\_salary"))  
edu\_pd = edu\_avg.toPandas().sort\_values("avg\_salary", ascending=False)  
  
fig = px.bar(edu\_pd, x="EDUCATION\_LEVELS\_NAME", y="avg\_salary",  
 title="Average Salary by Education Level")  
fig.show()

# Average salary by occupation  
occupation\_avg = df2.groupBy("OCCUPATION\_NAME").agg(F.avg("Average\_Salary").alias("avg\_salary"))  
occupation\_pd = occupation\_avg.toPandas().sort\_values("avg\_salary", ascending=False).head(15)  
  
fig = px.bar(occupation\_pd, x="OCCUPATION\_NAME", y="avg\_salary",  
 title="Top 15 Occupations by Average Salary")  
fig.show()

# Convert POSTED\_DATE to proper date  
df3 = df2.withColumn("POSTED\_DATE", F.to\_date("POSTED\_DATE", "yyyy-MM-dd"))  
  
# Count postings per month  
trend = df3.groupBy(F.date\_format("POSTED\_DATE", "yyyy-MM").alias("month")) \  
 .count() \  
 .orderBy("month")  
  
trend\_pd = trend.toPandas()  
  
fig = px.line(trend\_pd, x="month", y="count",  
 title="Job Postings Trend Over Time",  
 labels={"month": "Month", "count": "Number of Postings"})  
fig.update\_xaxes(type='category') # keep months readable  
fig.show()

# Top job titles by count  
title\_counts = df2.groupBy("JOB\_TITLE").count().orderBy(F.desc("count")).limit(15)  
title\_pd = title\_counts.toPandas()  
  
fig = px.bar(title\_pd, x="JOB\_TITLE", y="count",  
 title="Top 15 Job Titles by Frequency")  
fig.show()

# Top 15 States by job posting count  
state\_counts = df2.groupBy("STATE").count().orderBy(F.desc("count")).limit(15)  
state\_pd = state\_counts.toPandas()  
  
fig = px.bar(state\_pd, x="STATE", y="count",  
 title="Top 15 States by Job Postings")  
fig.show()

# Top 15 Cities by job posting count  
city\_counts = df2.groupBy("CITY").count().orderBy(F.desc("count")).limit(15)  
city\_pd = city\_counts.toPandas()  
  
fig = px.bar(city\_pd, x="CITY", y="count",  
 title="Top 15 Cities by Job Postings")  
fig.show()

# Top 20 most common skills in postings  
skill\_counts = df2.groupBy("SKILL\_NAME").count().orderBy(F.desc("count")).limit(20)  
skill\_pd = skill\_counts.toPandas()  
  
fig = px.bar(skill\_pd, x="SKILL\_NAME", y="count",  
 title="Top 20 Skills by Frequency")  
fig.show()

# Average salary by skill (filter out nulls)  
skill\_salary = df2.groupBy("SKILL\_NAME").agg(F.avg("SALARY").alias("avg\_salary")) \  
 .orderBy(F.desc("avg\_salary")).limit(15)  
skill\_salary\_pd = skill\_salary.toPandas()  
  
fig = px.bar(skill\_salary\_pd, x="SKILL\_NAME", y="avg\_salary",  
 title="Top 15 Skills by Average Salary")  
fig.show()

# Group by occupation: count of postings and average salary  
occ\_stats = df2.groupBy("OCCUPATION").agg(  
 F.count("\*").alias("posting\_count"),  
 F.avg("SALARY").alias("avg\_salary")  
).orderBy(F.desc("posting\_count")).limit(20)  
  
occ\_stats\_pd = occ\_stats.toPandas()  
  
fig = px.scatter(occ\_stats\_pd, x="posting\_count", y="avg\_salary", text="OCCUPATION",  
 title="Postings vs. Average Salary (Top 20 Occupations)",  
 labels={"posting\_count": "Number of Postings", "avg\_salary": "Average Salary"})  
fig.update\_traces(textposition="top center")  
fig.show()

# Count + average salary for skills  
skill\_stats = df2.groupBy("SKILL\_NAME").agg(  
 F.count("\*").alias("posting\_count"),  
 F.avg("SALARY").alias("avg\_salary")  
).orderBy(F.desc("posting\_count")).limit(30)  
  
skill\_stats\_pd = skill\_stats.toPandas()  
  
fig = px.scatter(skill\_stats\_pd, x="posting\_count", y="avg\_salary", text="SKILL\_NAME",  
 title="Skill Demand vs. Salary (Top 30 Skills)",  
 labels={"posting\_count": "Number of Postings", "avg\_salary": "Average Salary"})  
fig.update\_traces(textposition="top center")  
fig.show()

print("✅ Assignment 03 Summary")  
print("1. Salary distributions show large variance across industries and occupations.")  
print("2. Certain industries (e.g., Tech, Finance) consistently offer higher average salaries.")  
print("3. Some skills are extremely in-demand (like SQL, Python), but salary premiums vary.")  
print("4. High-demand skills don’t always equal high pay — strategic niche skills often pay more.")  
print("5. Location matters: states like California and New York dominate high-paying postings.")  
print("6. Trends over time show growth in postings for tech-heavy roles.")  
print("7. Correlation analysis reveals that many postings don’t guarantee top salaries — skills and specialization drive pay.")