Assignment 03

Ava Godsy

September 22, 2025

# 1. Load the dataset

import pandas as pd  
import plotly.express as px  
import plotly.io as pio  
from pyspark.sql import SparkSession  
import re  
import numpy as np  
import plotly.graph\_objects as go  
from pyspark.sql.functions import col, split, explode, regexp\_replace, transform, when  
from pyspark.sql import functions as F  
from pyspark.sql.functions import col, monotonically\_increasing\_id  
  
np.random.seed(42)  
  
pio.renderers.default = "notebook"  
  
# Initialize Spark Session  
spark = SparkSession.builder.appName("LightcastData").getOrCreate()  
  
# Load Data  
df = spark.read.option("header", "true").option("inferSchema", "true").option("multiLine","true").option("escape", "\"").csv("lightcast\_job\_postings.csv")  
df.createOrReplaceTempView("job\_postings")  
  
# Show Schema and Sample Data  
# print("---This is Diagnostic check, No need to print it in the final doc---")  
  
# df.printSchema() # comment this line when rendering the submission  
# df.show(5)

WARNING: Using incubator modules: jdk.incubator.vector  
Using Spark's default log4j profile: org/apache/spark/log4j2-defaults.properties  
Setting default log level to "WARN".  
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).  
25/09/24 03:46:52 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable  
25/09/24 03:47:07 WARN SparkStringUtils: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.

# 2. Data Cleaning

df = df.withColumn("SALARY\_FROM", col("SALARY\_FROM").cast("float")) \  
 .withColumn("SALARY\_TO", col("SALARY\_TO").cast("float")) \  
 .withColumn("SALARY", col("SALARY").cast("float")) \  
 .withColumn("MAX\_YEARS\_EXPERIENCE", col("MAX\_YEARS\_EXPERIENCE").cast("float")) \  
 .withColumn("MIN\_YEARS\_EXPERIENCE", col("MIN\_YEARS\_EXPERIENCE").cast("float"))  
  
def compute\_median(sdf, col\_name):  
 q = sdf.approxQuantile(col\_name, [0.5], 0.01)  
 return q[0] if q else None  
  
median\_from = compute\_median(df, "SALARY\_FROM")  
median\_to = compute\_median(df, "SALARY\_TO")  
median\_salary = compute\_median(df, "SALARY")  
  
print("Medians:", median\_from, "-", median\_to, "-", median\_salary)

[Stage 4:> (0 + 1) / 1]

df = df.fillna({  
 "SALARY\_FROM": median\_from,  
 "SALARY\_TO": median\_to  
})  
  
df = df.withColumn("Average\_Salary", (col("SALARY\_FROM") + col("SALARY\_TO")) / 2)  
df = df.withColumn("EDUCATION\_LEVELS\_NAME", regexp\_replace("EDUCATION\_LEVELS\_NAME", "\n ", ""))  
df = df.withColumn("EDUCATION\_LEVELS\_NAME", regexp\_replace("EDUCATION\_LEVELS\_NAME", "\n", ""))  
df = df.withColumn("EDUCATION\_LEVELS\_NAME", regexp\_replace("EDUCATION\_LEVELS\_NAME", "\r ", ""))  
df = df.withColumn("EDUCATION\_LEVELS\_NAME", regexp\_replace("EDUCATION\_LEVELS\_NAME", "\r", ""))  
  
export\_cols = [  
 "EDUCATION\_LEVELS\_NAME",  
 "REMOTE\_TYPE\_NAME",  
 "MAX\_YEARS\_EXPERIENCE",  
 "Average\_Salary",  
 "LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME",  
 "NAICS2\_NAME",  
 "EMPLOYMENT\_TYPE\_NAME",  
 "ONET\_NAME",  
 "SALARY\_FROM",  
 "SALARY\_TO",  
 "SALARY"  
]  
df\_selected = df.select (\*export\_cols)  
  
pdf = df\_selected.toPandas()  
pdf.to\_csv("lightcast\_cleaned.csv", index=False)  
  
print(" Data cleaning complete. Rows retained:", len(pdf))  
# df\_selected.show(5)

Data cleaning complete. Rows retained: 72498

# 3. Salary Distribution by Industry and Employment Type

The following data

import plotly.express as px  
  
fig = px.box(  
 df\_selected,  
 x="NAICS2\_NAME",  
 y="SALARY\_FROM",  
 color="EMPLOYMENT\_TYPE\_NAME",  
 title="Salary Distribution by Industry and Employment Type",  
 points="all", # Show all points  
 notched=True, # Notched boxes  
 height=1000, # Taller figure  
 color\_discrete\_sequence=["purple", "blue", "green"] # Custom colors  
)  
  
fig.update\_layout(  
 title\_font=dict(family="Garamond", size=24, color="black"),  
 xaxis\_title="Industry (NAICS2)",  
 yaxis\_title="Starting Salary",  
 boxmode="group", # Grouped box plots  
 xaxis\_tickangle=45, # Rotate x-axis labels  
 font=dict(  
 family="Garamond, serif", # Set font to Garamond  
 size=12  
 )  
)  
  
fig.show()

Unable to display output for mime type(s): text/html

# 4. Salary Analysis by ONET Occupation Type (Bubble Chart)

from pyspark.sql import functions as F  
  
df\_filtered = df\_selected.filter(F.col("SALARY").isNotNull())  
  
lot\_salary = df\_filtered.groupBy("LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME").agg(  
 F.expr("percentile\_approx(SALARY, 0.5)").alias("Median Salary"),  
 F.count("\*").alias("Job\_Postings")  
)  
  
lot\_salary.show()

[Stage 8:> (0 + 1) / 1]

+----------------------------------+-------------+------------+  
|LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME|Median Salary|Job\_Postings|  
+----------------------------------+-------------+------------+  
| Business Intellig...| 107500.0| 1800|  
| Business Analyst ...| 93650.0| 1640|  
| Healthcare Analyst| 89440.0| 94|  
| Oracle Consultant...| 138750.0| 3526|  
| SAP Analyst / Admin| 120640.0| 3373|  
| Data Analyst| 96100.0| 12377|  
| General ERP Analy...| 125900.0| 3703|  
| Marketing Analyst| 94500.0| 65|  
| Enterprise Architect| 157600.0| 3321|  
| Financial Data An...| 49920.0| 429|  
| Data Quality Analyst| 96600.0| 480|  
+----------------------------------+-------------+------------+

import plotly.express as px  
  
fig = px.scatter(  
 lot\_salary,  
 x="LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME",  
 y="Median Salary",  
 size="Job\_Postings",  
 color="Median Salary",  
 hover\_name="LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME",  
 size\_max=60,  
 title="Salary Analysis by LOT Occupation Type",  
)  
fig.update\_layout(  
 title\_font=dict(family="Garamond", size=24, color="black"),  
 font=dict(family="Garamond", size=12, color="black"),  
 plot\_bgcolor="white",  
 paper\_bgcolor="#f7f7f7",  
 xaxis=dict(title="Occupation Name", tickangle=45),  
 yaxis=dict(title="Median Salary ($)", gridcolor="#e5e5e5"),  
)  
  
# Show the figure  
fig.show()

Unable to display output for mime type(s): text/html

# 5. Salary by Education Level

# df\_selected.select("EDUCATION\_LEVELS\_NAME").distinct().show(truncate=False)  
  
  
from pyspark.sql.functions import col, when  
  
# Create the EDU\_GROUP column based on EDUCATION\_LEVELS\_NAME  
df\_with\_edu\_group = df\_selected.withColumn(  
 "EDU\_GROUP",  
 when(  
 col("EDUCATION\_LEVELS\_NAME").rlike("(?i)No Education Listed|GED|Associate"),   
 "Associate's or lower"  
 ).when(  
 col("EDUCATION\_LEVELS\_NAME").rlike("(?i)Bachelor"),  
 "Bachelor's"  
 ).when(  
 col("EDUCATION\_LEVELS\_NAME").rlike("(?i)Master"),  
 "Master's"  
 ).when(  
 col("EDUCATION\_LEVELS\_NAME").rlike("(?i)Ph\\.D\\.|professional degree"),  
 "PhD"  
 ).otherwise("Associate's or lower") # Optional: handle unmatched entries  
)  
  
# Select required columns  
final\_df = df\_with\_edu\_group.select(  
 "EDU\_GROUP",  
 "LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME",  
 "Average\_Salary",  
 "MAX\_YEARS\_EXPERIENCE"  
)  
  
final\_df.show()

+--------------------+----------------------------------+--------------+--------------------+  
| EDU\_GROUP|LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME|Average\_Salary|MAX\_YEARS\_EXPERIENCE|  
+--------------------+----------------------------------+--------------+--------------------+  
| Bachelor's| General ERP Analy...| 108668.5| 2.0|  
|Associate's or lower| Oracle Consultant...| 108668.5| 3.0|  
| Bachelor's| Data Analyst| 108668.5| NULL|  
|Associate's or lower| Data Analyst| 108668.5| NULL|  
|Associate's or lower| Oracle Consultant...| 92500.0| NULL|  
| Bachelor's| Data Analyst| 110155.0| NULL|  
| Bachelor's| Data Analyst| 108668.5| NULL|  
| Bachelor's| Data Analyst| 108668.5| NULL|  
|Associate's or lower| General ERP Analy...| 108668.5| 7.0|  
| Bachelor's| Data Analyst| 92962.0| 2.0|  
|Associate's or lower| Data Analyst| 107645.5| NULL|  
|Associate's or lower| Data Analyst| 108668.5| NULL|  
| Bachelor's| Data Analyst| 108668.5| NULL|  
| Bachelor's| General ERP Analy...| 192800.0| NULL|  
|Associate's or lower| Enterprise Architect| 81286.0| NULL|  
|Associate's or lower| Data Analyst| 108668.5| 5.0|  
|Associate's or lower| General ERP Analy...| 125900.0| NULL|  
|Associate's or lower| Oracle Consultant...| 108668.5| 3.0|  
| Bachelor's| Enterprise Architect| 165000.0| 8.0|  
|Associate's or lower| Data Analyst| 170000.0| NULL|  
+--------------------+----------------------------------+--------------+--------------------+  
only showing top 20 rows

import plotly.express as px  
import numpy as np  
  
# Step 1: Convert PySpark DataFrame to Pandas  
pdf = final\_df.toPandas()  
  
# Step 2: Add jitter to MAX\_YEARS\_EXPERIENCE  
np.random.seed(0)  
jitter\_strength = 0.1  
pdf['JITTERED\_EXPERIENCE'] = pdf['MAX\_YEARS\_EXPERIENCE'] + np.random.uniform(  
 -jitter\_strength, jitter\_strength, size=len(pdf)  
)  
  
# Step 3: Define custom color mapping  
color\_map = {  
 "Associate's or lower": 'yellow',  
 "Bachelor's": 'green',  
 "Master's": 'blue',  
 "PhD": 'purple'  
}  
  
# Step 4: Create the Plotly scatter plot  
fig = px.scatter(  
 pdf,  
 x='JITTERED\_EXPERIENCE',  
 y='Average\_Salary',  
 color='EDU\_GROUP',  
 color\_discrete\_map=color\_map,  
 title="Salary by Education Level",  
 labels={  
 'JITTERED\_EXPERIENCE': 'Max Years of Experience Required (jittered)',  
 'Average\_Salary': 'Average Salary',  
 'EDU\_GROUP': 'Minimum Education Level Required'  
 },  
 opacity=0.7  
)  
  
# Step 5: Update layout with Garamond font and sizes  
fig.update\_layout(  
 title\_font=dict(family='Garamond', size=24, color='black'),  
 font=dict(family='Garamond', size=12, color='black'),  
 legend\_title\_font=dict(family='Garamond', size=12, color='black'),  
 legend\_font=dict(family='Garamond', size=12, color='black')  
)  
  
# Step 6: Show the figure  
fig.show()

Unable to display output for mime type(s): text/html

# 6. Salary by Remote Work Type

# df\_selected.select("REMOTE\_TYPE\_NAME").distinct().show(truncate=False)  
  
from pyspark.sql.functions import col, when  
  
df\_with\_remote\_group = df\_selected.withColumn(  
 "REMOTE\_GROUP",  
 when(  
 col("REMOTE\_TYPE\_NAME") == "Remote", "Remote"  
 ).when(  
 col("REMOTE\_TYPE\_NAME") == "Hybrid Remote", "Hybrid"  
 ).when(  
 (col("REMOTE\_TYPE\_NAME").isNull()) |   
 (col("REMOTE\_TYPE\_NAME") == "Not Remote") |  
 (col("REMOTE\_TYPE\_NAME") == "[None]"),  
 "Onsite"  
 ).otherwise("Onsite"))  
   
remote\_df = df\_with\_remote\_group.select(  
 "REMOTE\_GROUP",  
 "LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME",  
 "Average\_Salary",  
 "MAX\_YEARS\_EXPERIENCE"  
)  
  
remote\_df.show()

+------------+----------------------------------+--------------+--------------------+  
|REMOTE\_GROUP|LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME|Average\_Salary|MAX\_YEARS\_EXPERIENCE|  
+------------+----------------------------------+--------------+--------------------+  
| Onsite| General ERP Analy...| 108668.5| 2.0|  
| Remote| Oracle Consultant...| 108668.5| 3.0|  
| Onsite| Data Analyst| 108668.5| NULL|  
| Onsite| Data Analyst| 108668.5| NULL|  
| Onsite| Oracle Consultant...| 92500.0| NULL|  
| Remote| Data Analyst| 110155.0| NULL|  
| Onsite| Data Analyst| 108668.5| NULL|  
| Onsite| Data Analyst| 108668.5| NULL|  
| Onsite| General ERP Analy...| 108668.5| 7.0|  
| Onsite| Data Analyst| 92962.0| 2.0|  
| Onsite| Data Analyst| 107645.5| NULL|  
| Onsite| Data Analyst| 108668.5| NULL|  
| Onsite| Data Analyst| 108668.5| NULL|  
| Onsite| General ERP Analy...| 192800.0| NULL|  
| Remote| Enterprise Architect| 81286.0| NULL|  
| Remote| Data Analyst| 108668.5| 5.0|  
| Onsite| General ERP Analy...| 125900.0| NULL|  
| Remote| Oracle Consultant...| 108668.5| 3.0|  
| Onsite| Enterprise Architect| 165000.0| 8.0|  
| Onsite| Data Analyst| 170000.0| NULL|  
+------------+----------------------------------+--------------+--------------------+  
only showing top 20 rows

import plotly.express as px  
import numpy as np  
  
# Step 1: Convert PySpark DataFrame to Pandas  
pdf2 = remote\_df.toPandas()  
  
# Step 2: Add jitter to MAX\_YEARS\_EXPERIENCE  
np.random.seed(0)  
jitter\_strength = 0.1  
pdf2['JITTERED\_EXPERIENCE'] = pdf2['MAX\_YEARS\_EXPERIENCE'] + np.random.uniform(  
 -jitter\_strength, jitter\_strength, size=len(pdf2)  
)  
  
# Step 3: Define custom color mapping  
color\_map = {  
 "Remote": 'yellow',  
 "Hybrid": 'green',  
 "Onsite": 'blue',  
}  
  
fig = px.scatter(  
 pdf2,  
 x='JITTERED\_EXPERIENCE',  
 y='Average\_Salary',  
 color='REMOTE\_GROUP',  
 color\_discrete\_map=color\_map,  
 title="Salary by Remote Status",  
 labels={  
 'JITTERED\_EXPERIENCE': 'Max Years of Experience Required',  
 'Average\_Salary': 'Average Salary',  
 'REMOTE\_GROUP': 'Remote Status'  
 },  
 opacity=0.7  
)  
  
# Step 5: Update layout with Garamond font and sizes  
fig.update\_layout(  
 title\_font=dict(family='Garamond', size=24, color='black'),  
 font=dict(family='Garamond', size=12, color='black'),  
 legend\_title\_font=dict(family='Garamond', size=12, color='black'),  
 legend\_font=dict(family='Garamond', size=12, color='black')  
)  
  
# Step 6: Show the figure  
fig.show()

Unable to display output for mime type(s): text/html

# Step 1: Convert to Pandas  
pdf = remote\_df.toPandas()  
  
# Step 2: Create plot using Plotly  
import plotly.express as px  
  
color\_map = {  
 "Remote": 'yellow',  
 "Hybrid": 'green',  
 "Onsite": 'blue',  
}  
  
fig = px.histogram(  
 pdf,  
 x="MAX\_YEARS\_EXPERIENCE",  
 y="Average\_Salary",  
 color="REMOTE\_GROUP",  
 color\_discrete\_map=color\_map,  
 histfunc="avg",  
 nbins=int(pdf['MAX\_YEARS\_EXPERIENCE'].max()) + 1,  
 barmode='group',  
 title="Average Salary by Years of Experience and Remote Type",  
 labels={  
 'MAX\_YEARS\_EXPERIENCE': 'Max Years of Experience Required',  
 'Average\_Salary': 'Average Salary',  
 'REMOTE\_GROUP': 'Remote Status'  
 }  
)  
  
fig.update\_layout(  
 title\_font=dict(family='Garamond', size=24, color='black'),  
 font=dict(family='Garamond', size=12, color='black'),  
 legend\_title\_font=dict(family='Garamond', size=12, color='black'),  
 legend\_font=dict(family='Garamond', size=12, color='black'),  
 xaxis=dict(dtick=1),  
 yaxis\_title="Average Salary",  
 bargap=0.2  
)  
  
fig.show()

Unable to display output for mime type(s): text/html