Assignment 03

Devin Blanchard

September 24, 2025

## 1 Data Loading and Inspection

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("./data/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).  
25/10/18 02:27:38 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable  
[Stage 0:> (0 + 1) / 1] [Stage 1:> (0 + 1) / 1] 25/10/18 02:27:53 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

from pyspark.sql.functions import col  
  
df = df.withColumn("SALARY", col("SALARY").cast("float"))  
df = df.withColumn("SALARY\_FROM", col("SALARY\_FROM").cast("float"))  
df = df.withColumn("SALARY\_TO", col("SALARY\_TO").cast("float"))  
df = df.withColumn("MIN\_YEARS\_EXPERIENCE", col("MIN\_YEARS\_EXPERIENCE").cast("float"))  
df = df.withColumn("MAX\_YEARS\_EXPERIENCE", col("MAX\_YEARS\_EXPERIENCE").cast("float"))  
  
# Compute median salary  
median\_from = df.approxQuantile("SALARY\_FROM", [0.5], 0.01)[0]  
median\_to = df.approxQuantile("SALARY\_TO", [0.5], 0.01)[0]  
median\_salary = df.approxQuantile("SALARY", [0.5], 0.01)[0]  
  
print("Medians:",median\_from, median\_to, median\_salary)  
  
# Impute missing 'SALARY\_FROM' and 'SALARY\_TO' with their medians  
df = df.fillna({  
 "SALARY\_FROM": median\_from,  
 "SALARY\_TO": median\_to,  
 "SALARY": median\_salary  
})  
  
# Compute 'AVERAGE\_SALARY'  
df = df.withColumn(  
 "AVERAGE\_SALARY", (col("SALARY\_FROM") + col("SALARY\_TO")) / 2  
)  
  
# Impute missing 'SALARY' with AVERAGE\_SALARY, and if that's missing, with the median salary  
from pyspark.sql.functions import when  
  
df = df.withColumn(  
 "SALARY",  
 when(  
 col("SALARY").isNull(),  
 when(col("AVERAGE\_SALARY").isNotNull(), col("AVERAGE\_SALARY"))  
 .otherwise(median\_salary)  
 ).otherwise(col("SALARY"))  
)  
  
from pyspark.sql.functions import regexp\_replace  
  
df = df.withColumn(  
 "EDUCATION\_LEVELS\_NAME",  
 regexp\_replace(col("EDUCATION\_LEVELS\_NAME"), r'[\n\r]', '')  
)  
  
# Overwrite  
df.write.option("header", True).mode("overwrite").csv("data/lightcast\_job\_postings\_cleaned.csv")  
  
# Display row count  
print(f"Rows retained after cleaning: {df.count()}")

[Stage 2:> (0 + 1) / 1] [Stage 3:> (0 + 1) / 1] [Stage 4:> (0 + 1) / 1]

Medians: 87295.0 130042.0 115024.0

[Stage 5:> (0 + 1) / 1] [Stage 6:> (0 + 1) / 1]

Rows retained after cleaning: 72498

## 3 Salary Distribution by Industry and Employment Type

# Filter for non-missing, nonzero salaries (use SALARY, not SALARY\_FROM)  
  
plot\_df = df.select("NAICS2\_NAME", "SALARY").filter(col("SALARY") > 0).toPandas()  
  
fig = px.box(  
 plot\_df,  
 x="NAICS2\_NAME",  
 y="SALARY",  
 points="all",  
 title="Salary Distribution by Industry",  
)  
  
fig.update\_traces(  
 marker=dict(color='rgb(52,152,219)', opacity=0.5, size=4),  
 line=dict(color='rgb(41,128,185)'),  
 fillcolor='rgba(41,128,185,0.3)',  
 jitter=0,  
 pointpos=0  
)  
  
fig.update\_layout(  
 xaxis\_title="Industry",  
 yaxis\_title="SALARY",  
 font=dict(size=14, family="Arial"),  
 plot\_bgcolor="#F4F8FF",  
 paper\_bgcolor="#F4F8FF",  
 xaxis\_tickangle=-90,  
 height=1200,  
 width=700,  
 showlegend=False,  
 boxmode='overlay'  
)  
  
fig.show()  
fig.write\_image("figures/salary\_distro\_by\_ind.png", scale=2)

[Stage 9:> (0 + 1) / 1]

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

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

Salaries vary widely between industries, with sectors like Information and Finance & Insurance generally showing higher salary ranges than industries such as Accommodation and Food Services. Full-time positions tend to have higher median salaries across most industries compared to part-time or other employment types.

## 4 Salary Analysis by Occupation (Bubble Chart)

# Lot Occupation Name  
salary\_analysis = spark.sql("""  
 SELECT  
 LOT\_OCCUPATION\_NAME AS OCCUPATION\_NAME,  
 PERCENTILE(SALARY, 0.5) AS Median\_Salary,  
 COUNT(\*) AS Job\_Postings  
 FROM job\_postings  
 GROUP BY LOT\_OCCUPATION\_NAME  
 ORDER BY Job\_Postings DESC  
 LIMIT 10  
""")  
  
salary\_pd = salary\_analysis.toPandas()  
salary\_pd.head()  
  
import plotly.express as px  
  
fig = px.scatter(  
 salary\_pd,  
 x="OCCUPATION\_NAME",  
 y="Median\_Salary",  
 size="Job\_Postings",  
 title="Salary Analysis by LOT Occupation Type (Bubble Chart)",  
 labels={  
 "OCCUPATION\_NAME": "LOT Occupation",  
 "Median\_Salary": "Median Salary",  
 "Job\_Postings": "Number of Job Postings"  
 },  
 hover\_name="OCCUPATION\_NAME",  
 size\_max=60,  
 width=1000,  
 height=600,  
 color="Job\_Postings",  
 color\_continuous\_scale="Viridis",  
)  
  
# Layout Customization  
fig.update\_layout(  
 title={  
 'text': "Salary Analysis by Occupation Type (Bubble Chart)",  
 'x': 0.5,  
 'xanchor': 'center',  
 },  
 font\_family="Arial",  
 font\_size=14,  
 title\_font\_size=25,  
 xaxis\_title="LOT Occupation",  
 yaxis\_title="Median Salary",  
 plot\_bgcolor="#f6f9fa",  
 width=1000,  
 height=600,  
 xaxis=dict(  
 tickangle=-15,  
 showline=True,  
 linecolor="#444"  
 ),  
 yaxis=dict(  
 showline=True,  
 linecolor="#444"  
 ),  
 xaxis\_title\_font=dict(size=17),  
 yaxis\_title\_font=dict(size=17),  
)  
  
fig.show()  
  
fig.write\_image("figures/salary\_by\_occupation.png", scale=2)

[Stage 10:> (0 + 1) / 1]

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

The bubble chart reveals that certain occupations have much higher job posting volumes, while also revealing a broad range of median salaries. High-demand roles tend to cluster at higher salary levels, indicating strong competition for talent in these areas.

## 5 Salary by Education Level

# Education levels  
lower\_deg = ["Bachelor", "Associate", "GED", "No Education Listed", "High school"]  
higher\_deg = ["Master", "PhD", "Doctorate", "professional degree"]  
  
# Add EDU\_GROUP column  
df = df.withColumn(  
 "EDU\_GROUP",  
 when(  
 col("EDUCATION\_LEVELS\_NAME").rlike("|".join([f"(?i){deg}" for deg in lower\_deg])),  
 "Bachelor's or lower"  
 ).when(  
 col("EDUCATION\_LEVELS\_NAME").rlike("|".join([f"(?i){deg}" for deg in higher\_deg])),  
 "Master's or PhD"  
 ).otherwise("Other")  
)  
  
# Cast columns  
df = df.withColumn("MAX\_YEARS\_EXPERIENCE", col("MAX\_YEARS\_EXPERIENCE").cast("float"))  
df = df.withColumn("Average\_Salary", col("Average\_Salary").cast("float"))  
  
# Filter for non-null and positive values  
df = df.filter(  
 (col("MAX\_YEARS\_EXPERIENCE").isNotNull()) &  
 (col("Average\_Salary").isNotNull()) &  
 (col("MAX\_YEARS\_EXPERIENCE") > 0) &  
 (col("Average\_Salary") > 0)  
)  
  
# Filter for education groups  
df\_filtered = df.filter(  
 col("EDU\_GROUP").isin("Bachelor's or lower", "Master's or PhD"))  
  
# Convert to Pandas  
df\_pd = df\_filtered.toPandas()  
df\_pd.head()  
  
# Scatter plot: Experience vs. Salary by Education Group  
  
import plotly.express as px  
  
# Plot  
fig1 = px.scatter(  
 df\_pd,  
 x="MAX\_YEARS\_EXPERIENCE",  
 y="Average\_Salary",  
 color="EDU\_GROUP",  
 hover\_data=["LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME"],  
 title="<b>Experience vs Salary by Education Level</b>",  
 opacity=0.7,  
 color\_discrete\_sequence=["#36B37E", "#A259EC"] # Custom green & purple  
)  
  
# Add borders  
fig1.update\_traces(marker=dict(size=7, line=dict(width=1, color="black")))  
  
# Update layout  
fig1.update\_layout(  
 plot\_bgcolor="#f9f9f9",  
 paper\_bgcolor="#EAF7FF", # Softer blue background  
 font=dict(family="Segoe UI", size=14),  
 title\_font=dict(size=22),  
 xaxis\_title="Years of Experience",  
 yaxis\_title="Average Salary (USD)",  
 legend\_title="Education Group",  
 hoverlabel=dict(bgcolor="white", font\_size=13, font\_family="Arial"),  
 margin=dict(t=70, b=60, l=60, r=60),  
 xaxis=dict(  
 gridcolor="lightgrey",  
 tickmode="linear",  
 dtick=1,  
 zeroline=False  
 ),  
 yaxis=dict(  
 gridcolor="lightgrey",  
 zeroline=False  
 )  
)  
  
fig1.show()  
fig.write\_image("figures/salary\_by\_education.png", scale=2)

[Stage 13:> (0 + 1) / 1]

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

Salaries generally rise with years of experience, regardless of education level. In this dataset though, the “bachelor’s or lower” group actually shows a wider spread and sometimes even higher salaries than the master’s or PhD group. This could be because some top-paying roles in fields like tech, sales, or management don’t always require advanced degrees, or because the “bachelor’s or lower” category includes a lot of jobs where the education isn’t specifically listed, so there’s a lot of variation in pay.

## 6 Salary by Remote Work Type

# Categorize remote work types  
  
df = df.withColumn(  
"REMOTE\_GROUP",  
when(col("REMOTE\_TYPE\_NAME") == "Remote", "Remote")  
.when(col("REMOTE\_TYPE\_NAME") == "Hybrid", "Hybrid")  
.otherwise("Onsite")  
)  
  
# Ensure correct types  
  
df = df.withColumn("MAX\_YEARS\_EXPERIENCE", col("MAX\_YEARS\_EXPERIENCE").cast("float"))  
df = df.withColumn("Average\_Salary", col("Average\_Salary").cast("float"))  
  
# Filter out invalid rows  
  
df\_remote = df.filter(  
(col("MAX\_YEARS\_EXPERIENCE").isNotNull()) &  
(col("Average\_Salary").isNotNull()) &  
(col("MAX\_YEARS\_EXPERIENCE") > 0) &  
(col("Average\_Salary") > 0)  
)  
  
# Convert to pandas  
  
df\_remote\_pd = df\_remote.select(  
"MAX\_YEARS\_EXPERIENCE",  
"Average\_Salary",  
"LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME",  
"REMOTE\_GROUP"  
).toPandas()  
  
# Define colors  
  
remote\_palette = {  
"Onsite": "#60A5FA", # blue  
"Remote": "#E15759", # red  
"Hybrid": "#7CCBA2" # green  
}  
  
import plotly.express as px  
  
# Scatter plot: Experience vs Salary by Remote Work Type  
  
fig\_remote = px.scatter(  
df\_remote\_pd,  
x="MAX\_YEARS\_EXPERIENCE",  
y="Average\_Salary",  
color="REMOTE\_GROUP",  
color\_discrete\_map=remote\_palette,  
opacity=0.7,  
hover\_data=["LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME"],  
title="<b>Experience vs Salary by Remote Work Type</b>"  
)  
  
fig\_remote.update\_traces(marker=dict(size=7, line=dict(width=1, color="black")))  
fig\_remote.update\_layout(  
plot\_bgcolor="#F7FFF9",  
paper\_bgcolor="#E9FFF0",  
font=dict(family="Segoe UI", size=14),  
title\_font=dict(size=22),  
xaxis\_title="Years of Experience",  
yaxis\_title="Average Salary (USD)",  
legend\_title="Remote Work Type",  
hoverlabel=dict(bgcolor="white", font\_size=13, font\_family="Arial"),  
margin=dict(t=70, b=60, l=60, r=60),  
xaxis=dict(  
gridcolor="lightgrey",  
tickmode="linear",  
dtick=1,  
zeroline=False  
),  
yaxis=dict(  
gridcolor="lightgrey",  
zeroline=False  
)  
)  
fig\_remote.show()  
fig.write\_image("figures/salary\_by\_remote\_type.png", scale=2)

[Stage 14:> (0 + 1) / 1]

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

The scatter plot shows that average salary generally increases with years of experience for both remote and onsite positions. There is no dramatic separation between the two work types, but remote roles appear somewhat more concentrated at mid-level experience, while onsite roles are distributed more broadly across the experience range.

# Onsite salary histogram  
group\_df1 = df\_remote\_pd[df\_remote\_pd["REMOTE\_GROUP"] == "Onsite"]  
fig\_hist1 = px.histogram(  
 group\_df1,  
 x="Average\_Salary",  
 nbins=40,  
 title="Salary Distribution: Onsite",  
 color\_discrete\_sequence=["#1f77b4"]  
)  
fig\_hist1.update\_layout(  
 plot\_bgcolor="#F9F9FF",  
 paper\_bgcolor="#FCFFF8",  
 font=dict(family="Segoe UI", size=13),  
 title\_font=dict(size=20),  
 xaxis\_title="Average Salary (USD)",  
 yaxis\_title="Number of Job Postings",  
 margin=dict(t=70, b=60, l=60, r=60),  
)  
fig\_hist1.show()  
fig\_hist1.write\_image("figures/onsite\_salary\_hist.png", scale=2)

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

The salary distribution for onsite roles is tightly clustered, with most job postings offering salaries around the $100,000 mark. There are relatively few onsite positions at significantly higher salary levels, and the distribution drops off sharply past $150,000.

# Remote salary histogram  
  
group\_df2 = df\_remote\_pd[df\_remote\_pd["REMOTE\_GROUP"] == "Remote"]  
fig\_hist2 = px.histogram(  
group\_df2,  
x="Average\_Salary",  
nbins=40,  
title="Salary Distribution: Remote",  
color\_discrete\_sequence=["#ef553b"]  
)  
fig\_hist2.update\_layout(  
plot\_bgcolor="#F9F9FF",  
paper\_bgcolor="#FCFFF8",  
font=dict(family="Segoe UI", size=13),  
title\_font=dict(size=20),  
xaxis\_title="Average Salary (USD)",  
yaxis\_title="Number of Job Postings",  
margin=dict(t=70, b=60, l=60, r=60),  
)  
fig\_hist2.show()  
fig\_hist2.write\_image("figures/remote\_salary\_hist.png", scale=2)

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

The salary distribution for remote roles is also centered near $100,000, but it shows a broader spread of salaries compared to onsite roles. There are slightly more remote postings with higher and lower salary extremes, suggesting greater variability in remote job pay.