Assignment 03

Joshua Lawrence

September 21, 2025

import kaleido  
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(123)  
  
pio.renderers.default = "notebook\_connected"  
  
# 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).  
25/09/24 07:22:10 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] 25/09/24 07:22:24 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'.

## 1 Data Prep / Cleaning

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

[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]

Data cleaning complete. Rows retained: 72498

## 2 Salary Distribution by Industry and Employment Type

### 2.1 Data Filtering

pdf = df\_selected.filter(df["SALARY"] > 0).select("EMPLOYMENT\_TYPE\_NAME", "SALARY", "NAICS2\_NAME").toPandas()  
pdf = pdf.dropna()  
  
pdf["EMPLOYMENT\_TYPE\_NAME"] = pdf["EMPLOYMENT\_TYPE\_NAME"].apply(lambda x: re.sub(r"[^\x00-\x7F]+", "", x))  
  
median\_salaries\_naics = pdf.groupby("NAICS2\_NAME")["SALARY"].median()  
  
top\_20\_naics = median\_salaries\_naics.sort\_values(ascending=False).head(20).index  
pdf = pdf[pdf["NAICS2\_NAME"].isin(top\_20\_naics)]  
  
pdf["NAICS2\_NAME"] = pd.Categorical(  
 pdf["NAICS2\_NAME"],  
 categories=top\_20\_naics,  
 ordered=True  
)  
pdf = pdf[pdf["NAICS2\_NAME"].isin(top\_20\_naics)]  
  
median\_salaries = pdf.groupby("EMPLOYMENT\_TYPE\_NAME")["SALARY"].median()  
sorted\_employment\_types = median\_salaries.sort\_values(ascending=False).index  
  
pdf["EMPLOYMENT\_TYPE\_NAME"] = pd.Categorical(  
 pdf["EMPLOYMENT\_TYPE\_NAME"],  
 categories=sorted\_employment\_types,  
 ordered=True  
)

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

### 2.2 Chart

fig = px.box(  
 pdf,  
 x="NAICS2\_NAME",  
 y="SALARY",  
 labels={  
 "NAICS2\_NAME": "NAICS Name",  
 "SALARY": "Salary"  
 },  
 color\_discrete\_sequence=["#4682B4"],  
 boxmode="group",  
 points="all",  
)  
fig.update\_layout(  
 title=dict(  
 text="Salary Distribution by Employment Type",  
 font=dict(size=30, family="Arial", color="black", weight="bold")  
 ),  
 xaxis=dict(  
 title=dict(text="Employment Type", font=dict(size=24, family="Arial", color="black", weight="bold")),  
 tickangle=70,  
 tickfont=dict(size=10, family="Arial", color="black", weight="bold"),  
 showline=True,  
 linewidth=2,  
 linecolor="black",  
 mirror=True,  
 showgrid=False,  
 categoryorder="array",  
 categoryarray=sorted\_employment\_types.tolist()  
 ),  
 yaxis=dict(  
 title=dict(text="Salary (K $)", font=dict(size=24, family="Arial", color="black", weight="bold")),  
 tickvals=[0, 50000, 100000, 150000, 200000, 250000, 300000, 350000, 400000, 450000, 500000],  
 ticktext=["0", "50K", "100K", "150K", "200K", "300K", "350K", "400K", "450K", "500K"],  
 tickfont=dict(size=18, family="Arial", color="black", weight="bold"),  
 showline=True,  
 linewidth=2,  
 linecolor="black",  
 mirror=True,  
 showgrid=True,  
 gridcolor="lightgray",  
 gridwidth=0.5  
 ),  
 font=dict(family="Arial", size=16, color="black"),  
 boxgap=0.7,  
 plot\_bgcolor="#F5F5F5",  
 paper\_bgcolor="#DCDCDC",  
 showlegend=True,  
 height=850,  
 width=1300  
)  
fig.show()  
fig.write\_html("output/naics\_salary.html")

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

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

### 2.3 Explanation // Chart Analysis

Salaries vary widely across employment types, with fields like Information and Manufacturing showing higher median salaries compared to sectors like Accommodation and Food Services. Outliers exist in almost all industries, with some individuals earning well above the median levels.

## 3 Salary Analysis by ONET Occupation Type

### 3.1 Data Filtering // Setup for Chart

bubble\_chart\_df = spark.sql("""  
 SELECT   
 LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME AS ONET\_NAME,  
 PERCENTILE(SALARY, 0.5) AS Median\_Salary,  
 COUNT(\*) AS Job\_Postings  
 FROM job\_postings  
 GROUP BY LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME  
 ORDER BY Job\_Postings DESC  
 LIMIT 10  
""")  
  
bubble\_chart\_df\_pd = bubble\_chart\_df.toPandas()  
bubble\_chart\_df\_pd.head()

[Stage 7:> (0 + 1) / 1]

|  | ONET\_NAME | Median\_Salary | Job\_Postings |
| --- | --- | --- | --- |
| 0 | Data Analyst | 96100.0 | 27832 |
| 1 | General ERP Analyst / Consultant | 125900.0 | 9931 |
| 2 | Enterprise Architect | 157600.0 | 8212 |
| 3 | Oracle Consultant / Analyst | 138750.0 | 8141 |
| 4 | SAP Analyst / Admin | 120640.0 | 7734 |

### 3.2 Bubble Chart

import plotly.express as px  
  
fig = px.scatter(  
 bubble\_chart\_df\_pd,  
 x="ONET\_NAME",  
 y="Median\_Salary",  
 size="Job\_Postings",  
 title="Salary Analysis by LOT Occupation Type",  
 labels={  
 "LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME": "LOT Occupation",  
 "Median\_Salary": "Median Salary",  
 "Job\_Postings": "Number of Job Postings"  
 },  
 hover\_name= "ONET\_NAME",  
 size\_max=60,  
 color="Job\_Postings",  
 color\_continuous\_scale="emrld"  
)  
fig.update\_layout(  
 title=dict(  
 text="Median Salary by Occupation Type",  
 font=dict(size=24, family="Arial", color="black", weight="bold")  
 ),  
 xaxis=dict(  
 title=dict(text="Occupation Name", font=dict(size=24, family="Arial", color="black", weight="bold")),  
 tickangle=60,  
 tickfont=dict(size=12, family="Arial", color="black", weight="bold"),  
 showline=True,  
 linewidth=2,  
 linecolor="black",  
 mirror=True,  
 showgrid=False,  
 ),  
 yaxis=dict(  
 title=dict(text="Median Salary (k $)", font=dict(size=24, family="Arial", color="black", weight="bold")),  
 tickvals=[40000, 50000, 60000, 70000, 80000, 90000, 100000, 110000, 120000, 130000, 140000, 150000, 160000, 170000],  
 ticktext=["40k", "50k", "60k", "70k", "80k", "90K", "100K", "110K", "120K", "130K", "140k", "150k", "160k", "170k"],  
 tickfont=dict(size=12, family="Arial", color="black", weight="bold"),  
 showline=True,  
 linewidth=2,  
 linecolor="black",  
 mirror=True,  
 showgrid=True,  
 gridcolor="lightgray",  
 gridwidth=0.5  
 ),  
 font=dict(family="Arial", size=16, color="black"),  
 boxgap=0.7,  
 plot\_bgcolor="white",  
 paper\_bgcolor="white",  
 showlegend=True,  
 height=850,  
 width=1000  
)  
fig.show()  
fig.write\_html("output/bubblechart.html")

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

### 3.3 Explanation // Chart Analysis

Enterprise Architect and Oracle Consultant roles have the highest median salaries, while Data Analyst positions have the lowest but also the largest number of job postings. Specialized roles generally command higher pay despite fewer postings.

## 4 Salary by Education Level

### 4.1 Data Filtering / Setup

lower\_degree = ["Associate", "GED", "No Education Listed", "High School"]  
bachelors\_degree = ["Bachelor's"]  
higher\_degree = ["Master's degree", "PhD or professional degree"]  
  
df = df.withColumn(  
 "EDU\_GROUP",  
 when(col("EDUCATION\_LEVELS\_NAME").rlike("|".join([f"(?i){deg}" for deg in lower\_degree])), "Associate or lower")  
 .when(col("EDUCATION\_LEVELS\_NAME").rlike("|".join([f"(?i){deg}" for deg in bachelors\_degree])), "Bachelor's")  
 .when(col("EDUCATION\_LEVELS\_NAME").rlike("|".join([f"(?i){deg}" for deg in higher\_degree])), "Master's or PhD")  
 .otherwise("Other")  
)  
  
df = df.filter(  
 col("MAX\_YEARS\_EXPERIENCE").isNotNull() &  
 col("Average\_Salary").isNotNull() &  
 (col("MAX\_YEARS\_EXPERIENCE") > 0) &  
 (col("Average\_Salary") > 0)  
)  
  
df\_filtered = df.filter(col("EDU\_GROUP").isin("Associate or lower", "Bachelor's", "Master's or PhD"))  
  
saledu\_df = df\_filtered.toPandas()

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

### 4.2 Scatter Plot

saledufig = px.scatter(  
 saledu\_df,  
 x="MAX\_YEARS\_EXPERIENCE",  
 y="Average\_Salary",  
 color="EDU\_GROUP",  
 hover\_data=["LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME"],  
 opacity=0.7,  
 labels = {  
 "EDU\_GROUP": "Level of Education",  
 "LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME": "Occupation Title",  
 "Average\_Salary": "Average Salary",  
 "MAX\_YEARS\_EXPERIENCE": "Years of Experience"  
 },  
 color\_discrete\_sequence=["#B4464B", "#4682B4", "#B4AF46"]  
)  
saledufig.update\_layout(  
 title=dict(  
 text="Experience vs Salary by Education Level",  
 font=dict(size=30, family="Arial", color="black", weight="bold")  
 ),  
 xaxis=dict(  
 title=dict(text="Years of Experience", font=dict(size=24, family="Arial", color="black", weight="bold")),  
 tickangle=0,  
 tickfont=dict(size=12, family="Arial", color="black", weight="bold"),  
 showline=True,  
 linewidth=2,  
 linecolor="black",  
 mirror=True,  
 showgrid=True,  
 ),  
 yaxis=dict(  
 title=dict(text="Average Salary (K $)", font=dict(size=24, family="Arial", color="black", weight="bold")),  
 tickvals=[0, 50000, 100000, 150000, 200000, 250000, 300000, 350000, 400000, 450000, 500000, 550000, 600000, 650000, 700000, 750000, 800000],  
 ticktext=["0", "50k", "100k", "150k", "200k", "250k", "300k", "350k", "400k", "450k", "500k", "550k", "600k", "650k", "700k", "750k", "800k"],  
 tickfont=dict(size=12, family="Arial", color="black", weight="bold"),  
 showline=True,  
 linewidth=2,  
 linecolor="black",  
 mirror=True,  
 showgrid=True,  
 gridcolor="lightgray",  
 gridwidth=0.5  
 ),  
 font=dict(family="Arial", size=16, color="black"),  
 boxgap=0.7,  
 plot\_bgcolor="#F5F5F5",  
 paper\_bgcolor="#DCDCDC",  
 showlegend=True,  
 height=850,  
 width=1000  
)  
saledufig.show()  
saledufig.write\_html("output/saledufig.html")

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

### 4.3 Explanation // Chart Analysis

Higher education levels, particularly Master’s or PhD, show some instances of higher salaries, though the overall pattern across experience years remains mixed. Many high-earning outliers appear at lower experience levels, suggesting factors beyond experience and education influence pay.

## 5 Salary by Remote Work Type

### 5.1 Data Filtering / Setup

Onsite = ["None", "Not Remote"]  
Remote = ["Remote"]  
Hybrid = ["Hybrid"]  
  
df = df.withColumn(  
 "WORK\_LOC",  
 when(col("REMOTE\_TYPE\_NAME").rlike("|".join([f"(?i){deg}" for deg in Onsite])), "Onsite")  
 .when(col("REMOTE\_TYPE\_NAME").rlike("|".join([f"(?i){deg}" for deg in Hybrid])), "Hybrid")  
 .when(col("REMOTE\_TYPE\_NAME").rlike("|".join([f"(?i){deg}" for deg in Remote])), "Remote")  
 .otherwise("Other")  
)  
  
df\_filtered = df.filter(col("WORK\_LOC").isin("Onsite", "Remote", "Hybrid"))  
  
remote\_df = df\_filtered.toPandas()

[Stage 11:> (0 + 1) / 1]

### 5.2 Scatter Plot

remotefig = px.scatter(  
 remote\_df,  
 x="MAX\_YEARS\_EXPERIENCE",  
 y="Average\_Salary",  
 color="WORK\_LOC",  
 hover\_data=["LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME"],  
 opacity=0.7,  
 labels = {  
 "WORK\_LOC": "Remote Type",  
 "LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME": "Occupation Title",  
 "Average\_Salary": "Average Salary",  
 "MAX\_YEARS\_EXPERIENCE": "Years of Experience"  
 },  
 color\_discrete\_sequence=["#B4464B", "#4682B4", "#B4AF46"]  
)  
remotefig.update\_layout(  
 title=dict(  
 text="Experience vs Salary by Remote Type",  
 font=dict(size=30, family="Arial", color="black", weight="bold")  
 ),  
 xaxis=dict(  
 title=dict(text="Years of Experience", font=dict(size=24, family="Arial", color="black", weight="bold")),  
 tickangle=0,  
 tickfont=dict(size=12, family="Arial", color="black", weight="bold"),  
 showline=True,  
 linewidth=2,  
 linecolor="black",  
 mirror=True,  
 showgrid=True,  
 ),  
 yaxis=dict(  
 title=dict(text="Average Salary (k $)", font=dict(size=24, family="Arial", color="black", weight="bold")),  
 tickvals=[0, 50000, 100000, 150000, 200000, 250000, 300000, 350000, 400000, 450000, 500000, 550000, 600000, 650000, 700000, 750000, 800000],  
 ticktext=["0", "50k", "100k", "150k", "200k", "250k", "300k", "350k", "400k", "450k", "500k", "550k", "600k", "650k", "700k", "750k", "800k"],  
 tickfont=dict(size=12, family="Arial", color="black", weight="bold"),  
 showline=True,  
 linewidth=2,  
 linecolor="black",  
 mirror=True,  
 showgrid=True,  
 gridcolor="lightgray",  
 gridwidth=0.5  
 ),  
 font=dict(family="Arial", size=16, color="black"),  
 boxgap=0.7,  
 plot\_bgcolor="#F5F5F5",  
 paper\_bgcolor="#DCDCDC",  
 showlegend=True,  
 height=850,  
 width=1000  
)  
remotefig.show()  
remotefig.write\_html("output/remotefig.html")

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

### 5.3 Salary Histograms

salhist = px.histogram(  
 remote\_df,   
 x='SALARY',   
 facet\_col='WORK\_LOC',  
 title='Salary Distribution by Work Location Type',  
 labels={'SALARY': 'Salary', 'count': 'Frequency'},  
 nbins=20)  
  
salhist.update\_layout(height=400, showlegend=False)  
salhist.show()  
salhist.write\_html("output/salhist.html")

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

### 5.4 Explanation // Chart Analysis

Salaries tend to increase slightly with more years of experience, regardless of remote type, but there is significant variability among individual cases. Onsite roles appear to have the highest salary outliers compared to Remote and Hybrid positions.

### 5.5 Salary Histogram Analysis

Onsite: The onsite salary distribution shows the largest concentration of salaries between $90K–$120K, with a sharp peak around $100K and relatively few outliers above $150K. This indicates that onsite roles dominate the dataset and generally cluster tightly around a consistent mid-range salary.

Remote: Remote roles show a smaller but noticeable concentration around $100K–$120K, with fewer postings overall compared to onsite work. While the distribution shape is similar, the lower counts suggest fewer remote opportunities or less data relative to onsite jobs.

Hybrid: Hybrid roles have the smallest sample size, with only a modest cluster around $100K–$120K and minimal variation beyond $150K. This limited data may indicate either fewer hybrid postings or less reporting of salaries for such roles.