Assignment 03

Sabrina Minaya Vasquez

September 24, 2025

# 1. Import Packages

import pandas as pd  
import plotly.express as px  
import plotly.io as pio  
#pio.renderers.default = "svg"  
  
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  
import os  
  
  
  
# Set random seed  
np.random.seed(42)  
  
# Change Plotly renderer for notebooks  
pio.renderers.default = "notebook"

# 2. Plotly Templete

pio.templates["nike"] = go.layout.Template(  
 # LAYOUT  
 layout = {  
 # Fonts  
 # Note - 'family' must be a single string, NOT a list or dict!  
 'title':  
 {'font': {'family': 'HelveticaNeue-CondensedBold, Helvetica, Sans-serif',  
 'size':30,  
 'color': '#333'}  
 },  
 'font': {'family': 'Helvetica Neue, Helvetica, Sans-serif',  
 'size':16,  
 'color': '#333'},  
 # Colorways  
 'colorway': ['#ec7424', '#a4abab'],  
 # Keep adding others as needed below  
 'hovermode': 'x unified'  
 },  
 # DATA  
 data = {  
 # Each graph object must be in a tuple or list for each trace  
 'bar': [go.Bar(texttemplate = '%{value:$.2s}',  
 textposition='outside',  
 textfont={'family': 'Helvetica Neue, Helvetica, Sans-serif',  
 'size': 20,  
 'color': '#FFFFFF'  
 })]  
 }  
)  
# Make Nike the default for BOTH PX and GO:  
px.defaults.template = "nike"  
pio.templates.default = "nike"

# 3. Load Dataset

# 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("/home/ubuntu/assignment-03-Sabrina1211/data/lightcast\_job\_postings.csv")  
  
df.createOrReplaceTempView("job\_postings")  
  
#df.printSchema() # comment this line when rendering the submission  
#df.show(5)

[Stage 27:> (0 + 1) / 1]

# 4. Data Preparation

# Step 1: Casting Salary and experience columns  
df = df.withColumn("SALARY", col("SALARY").cast("float")) \  
 .withColumn("SALARY\_FROM", col("SALARY\_FROM").cast("float")) \  
 .withColumn("SALARY\_TO", col("SALARY\_TO").cast("float")) \  
 .withColumn("MIN\_YEARS\_EXPERIENCE", col("MIN\_YEARS\_EXPERIENCE").cast("float")) \  
 .withColumn("MAX\_YEARS\_EXPERIENCE", col("MAX\_YEARS\_EXPERIENCE").cast("float"))  
  
  
# Step 2: Computing medians for salary columns  
  
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)  
  
# Step 4: Imputing missing salaries, but no experience  
df = df.fillna({  
 "SALARY\_FROM": median\_from,  
 "SALARY\_TO": median\_to,  
   
})  
  
# Step 5: Computing Average Salary  
df = df.withColumn("Average\_Salary", (col("SALARY\_FROM") + col("SALARY\_TO")) /2)  
  
# Step 6: Selecting required columns  
export\_cols = [  
 "EDUCATION\_LEVELS\_NAME",  
 "REMOTE\_TYPE\_NAME",  
 "MAX\_YEARS\_EXPERIENCE",  
 "Average\_Salary",  
 "SALARY",  
 "LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME"  
]  
  
df\_selected = df.select(\*export\_cols)  
  
# Step 7: Saving to csv  
pdf = df\_selected.toPandas()  
pdf.to\_csv("./data/lightcast\_cleaned.csv", index=False)  
  
print("Data cleaning complete. Rows retained:", len(pdf))

[Stage 28:> (0 + 1) / 1] [Stage 29:> (0 + 1) / 1] [Stage 30:> (0 + 1) / 1]

Medians: 87295.0 130042.0 115024.0

[Stage 31:> (0 + 1) / 1]

Data cleaning complete. Rows retained: 72498

# 5. Salary Distribution Employment Type

# Salary Distribution Employment Type  
  
os.makedirs("figures", exist\_ok=True)  
  
pdf = (  
 df.select("EMPLOYMENT\_TYPE\_NAME", F.col("SALARY").cast("double").alias("SALARY"))  
 .filter((F.col("SALARY").isNotNull()) & (F.col("SALARY") > 0))  
 .toPandas()  
)  
  
pdf["EMPLOYMENT\_TYPE\_NAME"] = (  
 pdf["EMPLOYMENT\_TYPE\_NAME"].astype("string").fillna("Unknown")  
 .str.replace(r"[^\x00-\x7F]+", "", regex=True).str.strip()  
)  
  
sorted\_employment\_types = (  
 pdf.groupby("EMPLOYMENT\_TYPE\_NAME")["SALARY"].median()  
 .sort\_values(ascending=False).index  
)  
pdf["EMPLOYMENT\_TYPE\_NAME"] = pd.Categorical(  
 pdf["EMPLOYMENT\_TYPE\_NAME"],  
 categories=sorted\_employment\_types,  
 ordered=True  
)  
  
fig = px.box(  
 pdf,  
 x="EMPLOYMENT\_TYPE\_NAME",  
 y="SALARY",  
 title="Salary Distribution by Employment Type",  
 points="outliers" # template will set color/style  
)  
  
fig.update\_layout(  
 xaxis=dict(title="Employment Type",  
 categoryorder="array",  
 categoryarray=sorted\_employment\_types.tolist(),  
 tickfont=dict(size=18)),  
 yaxis=dict(title="Salary (K $)", range=[0, 500000],  
 tickvals=[0, 50\_000, 100\_000, 150\_000, 200\_000, 250\_000, 300\_000, 350\_000, 400\_000, 450\_000, 500\_000],  
 ticktext=["0","50K","100K","150K","200K","250K","300K","350K","400K","450K","500K"]),  
 font=dict(family="Arial", size=16),  
 showlegend=False,  
 height=500, width=850  
)  
  
fig.write\_html("figures/DistributionEmploymentType.html")  
fig.write\_image("figures/DistributionEmploymentType.svg", width=850, height=500, scale=1)

[Stage 32:> (0 + 1) / 1]

The graph shows how full-time jobs are more likely to have higher pay than part-time work. Yet, there are some part-time jobs that have a relatively wide wage range, where some people earn as much as full-time employees.

# 6. Salary Distribution by Industry

pdf = df.select("NAICS2\_NAME", "SALARY").toPandas()  
  
  
fig = px.box(  
 pdf,  
 x="NAICS2\_NAME",  
 y="SALARY",  
 title="Salary Distribution by Industry",  
 color\_discrete\_sequence=["#EF553B"]  
)  
  
fig.update\_layout(template="nike")   
  
# rotate x-axis labels for readability  
fig.update\_xaxes(tickangle=45)  
  
# fig.show()  
fig.write\_html("figures/DistributionIndustry.html")  
fig.write\_image("figures/DistributionIndustry.svg", width=1000, height=600, scale=2)

[Stage 33:> (0 + 1) / 1]

This chart highlights the large differences in pay between industries. Industries like Information and Finance/Insurance have higher median pay and a wide range of pay. On the other hand, industries like education and retail normally have lower pay which are more concentrated.

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

# Step 1: Spark SQL - Median salary and job count per TITLE\_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  
""")  
  
  
# Step 2: Convert to Pandas DataFrame  
salary\_pd = salary\_analysis.toPandas()  
salary\_pd.head()

[Stage 34:> (0 + 1) / 1]

|  | Occupation\_Name | Median\_Salary | Job\_Postings |
| --- | --- | --- | --- |
| 0 | Data / Data Mining Analyst | 95250.0 | 30057 |
| 1 | Business Intelligence Analyst | 125900.0 | 29445 |
| 2 | Computer Systems Engineer / Architect | 157600.0 | 8212 |
| 3 | Business / Management Analyst | 93650.0 | 4326 |
| 4 | Clinical Analyst / Clinical Documentation and ... | 89440.0 | 261 |

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="Plasma",  
)  
# Step 4: Layout customization  
fig.update\_layout(  
 font\_family="Arial",  
 font\_size=14,  
 title\_font\_size=25,  
 xaxis\_title="LOT Occupation",  
 yaxis\_title="Median Salary",  
 plot\_bgcolor="white",  
 xaxis=dict(  
 tickangle=-45,  
 showline=True,  
 linecolor="black",  
 ),  
 yaxis=dict(  
 showline=True,  
 linecolor="black",  
 ),  
)  
  
  
# Step 5: Show and export  
#fig.show()  
fig.write\_html("figures/salaryAnalsisbyLotOccupation.html")  
fig.write\_image("figures/salaryAnalysisbyLotOccupation.svg", width=1000, height=600, scale=1)

The bubble chart shows that occupations like Data Analysts and Business Intelligence Analysts have the highest number of job postings. However, Computer Systems Engineers/Architects stand out with the highest median salary, even though there are fewer postings compared to other roles.

# 8. Salary by Education Level

# Map raw education text into 4 groups  
df\_edu = (  
 df.withColumn(  
 "EDU\_GROUP",  
 F.when(F.col("EDUCATION\_LEVELS\_NAME").rlike("(?i)Associate|GED|No Education Listed|High school"), "Associate or Lower")  
 .when(F.col("EDUCATION\_LEVELS\_NAME").rlike("(?i)Bachelor"), "Bachelor")  
 .when(F.col("EDUCATION\_LEVELS\_NAME").rlike("(?i)Master"), "Master’s")  
 .when(F.col("EDUCATION\_LEVELS\_NAME").rlike("(?i)PhD|Doctorate|professional degree"), "PhD")  
 .otherwise(None)  
 )  
 .filter(  
 F.col("EDU\_GROUP").isNotNull()  
 & F.col("MAX\_YEARS\_EXPERIENCE").isNotNull()  
 & F.col("Average\_Salary").isNotNull()  
 & (F.col("MAX\_YEARS\_EXPERIENCE") > 0)  
 & (F.col("Average\_Salary") > 0)  
 )  
)  
  
  
pdf\_edu = df\_edu.select(  
 F.col("MAX\_YEARS\_EXPERIENCE").alias("Experience"),  
 F.col("Average\_Salary").alias("Average\_Salary"),  
 F.col("EDU\_GROUP").alias("Education Group"),  
 F.col("LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME").alias("Occupation Name")  
).toPandas()  
  
  
# Show table first  
from IPython.display import display  
tbl = (  
 pdf\_edu[["Experience","Average\_Salary","Education Group","Occupation Name"]]  
 .rename(columns={"Experience":"MAX\_EXPERIENCE","Average\_Salary":"AVERAGE\_SALARY",  
 "Education Group":"EDU\_GROUP","Occupation Name":"OCCUPATION\_NAME"})  
 .head(5)  
)  
display(  
 tbl.style  
 .format({"AVERAGE\_SALARY": "${:,.0f}", "MAX\_EXPERIENCE": "{:.1f}"})  
 .hide(axis="index")  
)  
  
  
# Plot (nike template)  
  
  
order = ["Associate or Lower","Bachelor","Master’s","PhD"]  
fig1 = px.scatter(  
 pdf\_edu,  
 x="Experience",  
 y="Average\_Salary",  
 color="Education Group",  
 category\_orders={"Education Group": order},  
 title="Experience vs Salary by Education Level",  
 labels={"Experience":"Years of Experience", "Average\_Salary":"Average Salary (USD)"},  
 template="nike",  
 opacity=0.8  
)  
# Soft beige frame + light plotting area + subtle grid  
fig1.update\_layout(  
 paper\_bgcolor="#FFF5DC", # outer background (around the plot)  
 plot\_bgcolor="#f9f9f9", # inner plotting area  
 margin=dict(l=60, r=60, t=80, b=70),  
 xaxis=dict(  
 tickmode="linear", dtick=1,  
 gridcolor="rgba(0,0,0,0.12)", # light grid  
 zeroline=False  
 ),  
 yaxis=dict(  
 gridcolor="rgba(0,0,0,0.12)",  
 zeroline=False  
 ),  
 legend=dict(  
 bgcolor="rgba(255,245,220,0.7)", # optional: match the frame  
 bordercolor="#E6D9B6", borderwidth=1  
 )  
)  
  
  
fig1.update\_traces(marker=dict(size=6, opacity=0.8))  
  
  
# Show the chart   
#fig1.show()  
fig1.write\_html("figures/SalaryEducationLevel.html")  
fig1.write\_image("figures/SalaryEducationLevel.svg", width=1000, height=600, scale=1)

[Stage 37:> (0 + 1) / 1]

| MAX\_EXPERIENCE | AVERAGE\_SALARY | EDU\_GROUP | OCCUPATION\_NAME |
| --- | --- | --- | --- |
| 2.0 | $108,668 | Bachelor | General ERP Analyst / Consultant |
| 3.0 | $108,668 | Associate or Lower | Oracle Consultant / Analyst |
| 7.0 | $108,668 | Associate or Lower | General ERP Analyst / Consultant |
| 2.0 | $92,962 | Bachelor | Data Analyst |
| 5.0 | $108,668 | Associate or Lower | Data Analyst |

The chart shows that salaries generally increase with more years of experience across all education levels. Higher education levels, like Master’s and PhD, tend to earn slightly more, but there is overlap with Bachelor’s and Associate degrees.

# 9. Salary by Remote Work Type

# Normalize strings and map to 3 buckets  
# Build Remote / Hybrid / Onsite groups   
df\_remote = (  
 df.withColumn("REMOTE\_NORM", F.lower(F.trim(F.col("REMOTE\_TYPE\_NAME"))))  
 .withColumn(  
 "REMOTE\_GROUP",  
 F.when(F.col("REMOTE\_NORM").rlike(r"hyb|mix|both|partial|split|combo"), "Hybrid")  
 .when(F.col("REMOTE\_NORM").rlike(r"remote|wfh|home|tele"), "Remote")  
 .otherwise("Onsite")  
 )  
 .drop("REMOTE\_NORM")  
 .filter(  
 F.col("MAX\_YEARS\_EXPERIENCE").isNotNull() &  
 F.col("Average\_Salary").isNotNull() &  
 (F.col("MAX\_YEARS\_EXPERIENCE") > 0) &  
 (F.col("Average\_Salary") > 0)  
 )  
)  
  
pdf\_remote = df\_remote.select(  
 F.col("MAX\_YEARS\_EXPERIENCE").alias("Experience"),  
 F.col("Average\_Salary").alias("Average\_Salary"),  
 F.col("REMOTE\_GROUP").alias("Remote Work Type"),  
 F.col("LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME").alias("Occupation Name")  
).toPandas()  
  
# Preview table (top 5)  
from IPython.display import display  
display(  
 (pdf\_remote[["Experience","Average\_Salary","Remote Work Type","Occupation Name"]]  
 .rename(columns={"Experience":"MAX\_YEARS\_EXPERIENCE",  
 "Average\_Salary":"AVERAGE\_SALARY",  
 "Remote Work Type":"REMOTE\_GROUP",  
 "Occupation Name":"OCCUPATION\_NAME"})  
 .head(5)  
 .style.format({"AVERAGE\_SALARY":"${:,.0f}", "MAX\_YEARS\_EXPERIENCE":"{:.1f}"})  
 .hide(axis="index"))  
)  
  
order = ["Onsite", "Remote", "Hybrid"]  
color\_map = {"Onsite":"#1f77b4", "Remote":"#ff7f0e", "Hybrid":"#2ca02c"}   
  
fig\_remote = px.scatter(  
 pdf\_remote,  
 x="Experience",  
 y="Average\_Salary",  
 color="Remote Work Type",  
 category\_orders={"Remote Work Type": order},  
 color\_discrete\_map=color\_map,  
 title="Experience vs Salary by Remote Work Type",  
 labels={"Experience":"Years of Experience", "Average\_Salary":"Average Salary (USD)"},  
 template="nike",  
 opacity=0.8  
)  
  
# match your education layout exactly  
fig\_remote.update\_layout(  
 paper\_bgcolor="#E8F5E9",   
 plot\_bgcolor="#F1FAF1",   
 margin=dict(l=60, r=60, t=80, b=70),  
 xaxis=dict(  
 tickmode="linear", dtick=1,  
 gridcolor="rgba(0,0,0,0.12)",  
 zeroline=False  
 ),  
 yaxis=dict(  
 gridcolor="rgba(0,0,0,0.12)",  
 zeroline=False  
 ),  
 legend=dict(  
 bgcolor="rgba(232,245,233,0.8)",   
 bordercolor="#A5D6A7", borderwidth=1,  
 title\_text="Remote Work Type"  
 )  
)  
  
fig\_remote.update\_traces(marker=dict(size=6, opacity=0.8, line=dict(width=0)))  
  
# Show   
# fig\_remote.show()  
fig\_remote.write\_html("figures/RemoteWorkType\_scatter.html")  
fig\_remote.write\_image("figures/RemoteWorkType\_scatter.svg", width=1000, height=600, scale=1)

[Stage 38:> (0 + 1) / 1]

| MAX\_YEARS\_EXPERIENCE | AVERAGE\_SALARY | REMOTE\_GROUP | OCCUPATION\_NAME |
| --- | --- | --- | --- |
| 2.0 | $108,668 | Onsite | General ERP Analyst / Consultant |
| 3.0 | $108,668 | Remote | Oracle Consultant / Analyst |
| 7.0 | $108,668 | Onsite | General ERP Analyst / Consultant |
| 2.0 | $92,962 | Onsite | Data Analyst |
| 5.0 | $108,668 | Remote | Data Analyst |

The chart shows that salaries increase slightly with more years of experience, regardless of remote work type. Onsite, remote, and hybrid roles have similar salary patterns, though onsite jobs appear more common across the data.