Assignment 03

Thomas Primero

September 23, 2025

# 1. Load the Dataset

**Load the Raw Dataset:** Use Pyspark to the lightcast\_data.csv file into a DataFrame: You can reuse the previous code. Copying code from your friend constitutes plagiarism. DO NOT DO THIS.

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 = "png"  
  
# 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-t-primero/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)

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

# 2. Data Preparation

We will be converting numerical columns to floats - this is so we can perform functions on it such as average.

# 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 3: Imputing missing salaries, but not 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",  
 "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 15:> (0 + 1) / 1] [Stage 16:> (0 + 1) / 1] [Stage 17:> (0 + 1) / 1]

Medians: 87295.0 130042.0 115024.0

[Stage 18:> (0 + 1) / 1]

Data cleaning complete. Rows retained: 72498

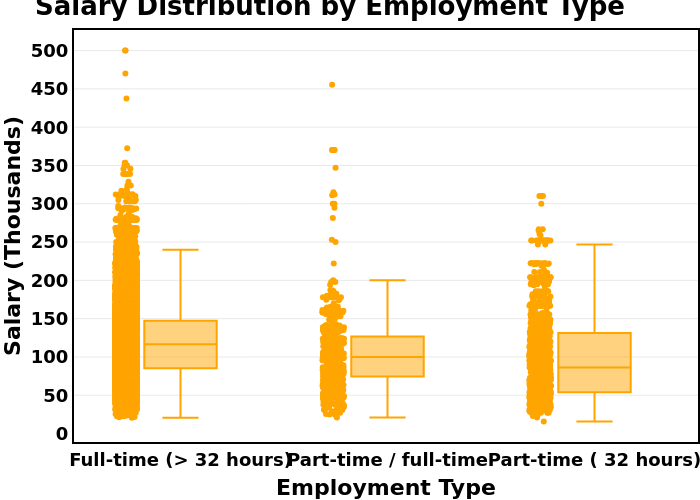
# Histogram of SALARY distribution  
# salary\_df = df.filter(col("SALARY").isNotNull() & (col("SALARY") > 0))  
#fig = px.histogram(salary\_df.toPandas(), x="SALARY", nbins=50, title="Salary Distribution")  
#fig.update\_layout(bargap=0.1)

# 3. Salary Distribution by Industry and Employment Type

Compare salary variations across industries. **Filter the dataset** Remove records where salary is missing or zero. **Aggregate Data** Group by NAICS industry codes. Group by employment type and compute salary distribution. **Visualize results** Create a box plot where: **X-axis** = ‘NAICS2\_NAME’ **Y-axis** = ‘SALARY\_FROM’, or ‘SALARY\_TO’, or ‘SALARY’ Group by ‘EMPLOYMENT\_TYPE\_NAME’. Customize colors, fonts, and styles. **Explanation:** Write two sentences about what the graph reveals.

# Your Code for 1st question here  
import pandas as pd  
import polars as pl  
  
# Filter out missing or zero salary values  
pdf = df.filter(df["SALARY"] > 0).select("EMPLOYMENT\_TYPE\_NAME", "SALARY").toPandas()  
# pdf.head()  
  
# Clean employment type names for better readability  
# This Basically looks for symbols numbers (which were incorrectly added into data name)  
pdf["EMPLOYMENT\_TYPE\_NAME"] = pdf["EMPLOYMENT\_TYPE\_NAME"].apply(lambda x: re.sub(r"[^\x00-\x7f]+", "", x))  
# pdf.head()  
  
# Compute median salary for sorting  
median\_salaries = pdf.groupby("EMPLOYMENT\_TYPE\_NAME")["SALARY"].median()  
# median\_salaries.head()  
  
# Sort employment types based on median salary in descending order  
sorted\_employment\_types = median\_salaries.sort\_values(ascending=False).index  
  
# Apply sorted categories  
pdf["EMPLOYMENT\_TYPE\_NAME"] = pd.Categorical(  
 pdf["EMPLOYMENT\_TYPE\_NAME"],  
 categories=sorted\_employment\_types,  
 ordered=True  
)  
  
# Create box plot with horizontal grid lines  
fig = px.box(  
 pdf,  
 x="EMPLOYMENT\_TYPE\_NAME",  
 y="SALARY",  
 title="Salary Distribution by Employment Type",  
 color\_discrete\_sequence=["orange"], # Single neutral color  
 boxmode="group",  
 points="all", # Show all outliers  
)  
# Improve layout, font styles, and axis labels  
fig.update\_layout(  
 title=dict(  
 text="Salary Distribution by Employment Type",  
 font=dict(size=26, family="Verdana", color="black", weight="bold") # Bigger & Bold Title  
 ),  
 xaxis=dict(  
 title=dict(text="Employment Type", font=dict(size=22, family="Verdana", color="black", weight="bold")), # Bigger X-label  
 tickangle=0, # Rotate X-axis labels for readability  
 tickfont=dict(size=18, family="Verdana", color="black", weight="bold"), # Bigger & Bold X-ticks  
 showline=True, # Show axis lines  
 linewidth=2, # Thicker axis lines  
 linecolor="black",  
 mirror=True,  
 showgrid=False, # Remove vertical grid lines  
 categoryorder="array",  
 categoryarray=sorted\_employment\_types.tolist()  
 ),  
 yaxis=dict(  
 title=dict(text="Salary (Thousands)", font=dict(size=22, family="Verdana", color="black", weight="bold")), # Bigger Y-label  
 tickvals=[0, 50000, 100000, 150000, 200000, 250000, 300000, 350000, 400000, 450000, 500000],  
 ticktext=["0", "50", "100", "150", "200", "250", "300", "350", "400", "450", "500"],  
 tickfont=dict(size=18, family="Verdana", color="black", weight="bold"), # Bigger & Bold Y-ticks  
 showline=True,  
 linewidth=2,  
 linecolor="black",  
 mirror=True,  
 showgrid=True, # Enable light horizontal grid lines  
 gridcolor="lightgray", # Light shade for the horizontal grid  
 gridwidth=0.5 # Thin grid lines  
 ),  
 font=dict(family="Verdana", size=16, color="black"),  
 boxgap=0.5,  
 plot\_bgcolor="white",  
 paper\_bgcolor="white",  
 showlegend=False,  
 height=800,  
 width=900  
)  
  
# Show the figure  
fig.show()

[Stage 19:> (0 + 1) / 1]

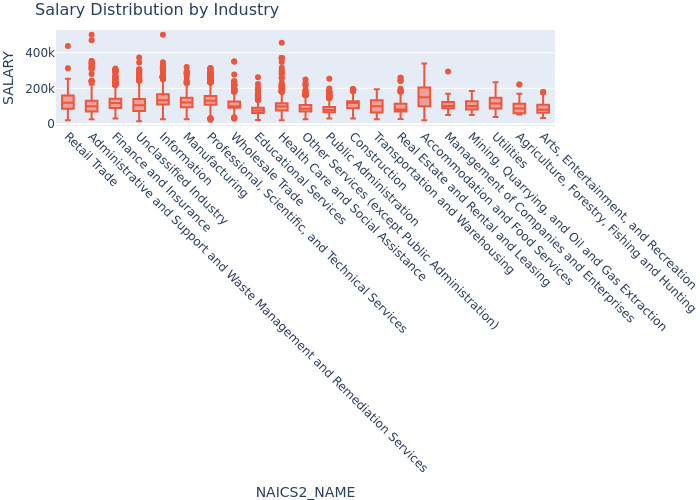


# 4. Salary Distribution by Industry and Employment Type

Compare salary variations across industries. Filter the dataset Remove records where salary is missing or zero. Aggregate Data Group by NAICS industry codes. Group by employment type and compute salary distribution. Visualize results Create a box plot where: X-axis = NAICS2\_NAME Y-axis = SALARY\_FROM Group by EMPLOYMENT\_TYPE\_NAME. Customize colors, fonts, and styles. Explanation: Write two sentences about what the graph reveals. # The graph reveals that fields like educational services, real estate, public administration, and arts have the largest median salary distributions, which means that salaries are more likely to vary. Salaries like Information,and finance have more accurate and higher paying median salaries, since their distribtion is lower.

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(font\_family="Arial", title\_font\_size=16,  
 height=1000,  
 width=1200)  
# Rotate x-axis labels for readability  
fig.update\_xaxes(tickangle=45, tickfont=dict(size=12))  
fig.show()

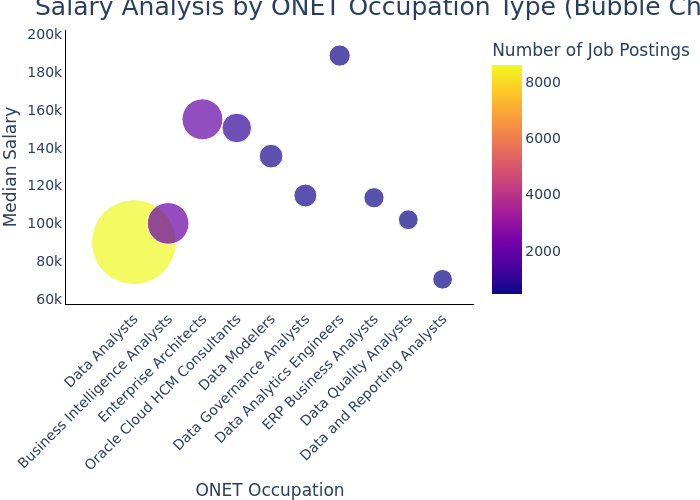
[Stage 20:> (0 + 1) / 1]



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

Analyze how salaries differ across ONET occupation types. **Aggregate Data** Compute median salary for each occupation in the ONET taxonomy. **Visualize results** Create a bubble chart where: **X-axis** = ONET\_NAME **Y-axis** = Median Salary **Size** = Number of job postings Apply custom colors and font styles. **Explanation**: Write two sentences about what the graph reveals. # The graph reveals that Enterprise Architects have the highest median salary and most number of job postings. It seems that this occupation is in high demand in the industry. Analysts and consultants have lower median salaries and lower job postings.

[Stage 21:> (0 + 1) / 1]



/bin/bash: -c: line 1: syntax error near unexpected token `output/Q7.svg'  
/bin/bash: -c: line 1: `[](output/Q7.svg)'

# 6. Salary by Education Level

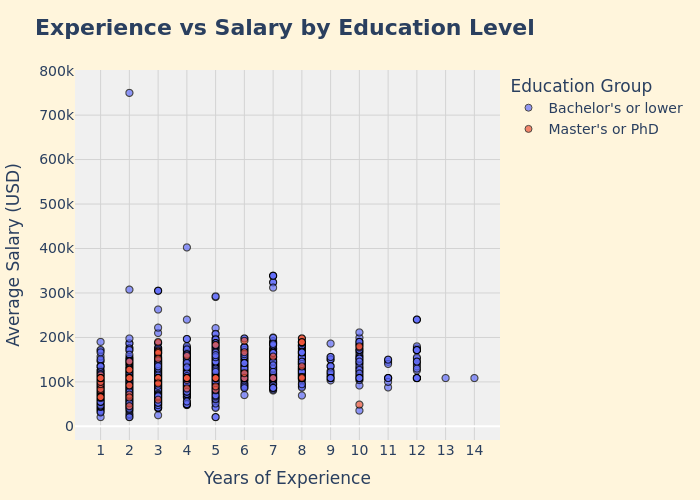
Create two groups: Associate’s or lower (GED, Associate, No Education Listed) Bachelor’s (Bachelor’s degree) Master’s (Master’s degree) PhD (PhD, Doctorate, professional degree) Plot scatter plots for each group using, MAX\_YEARS\_EXPERIENCE (with jitter), Average\_Salary, LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME After each graph, add a short explanation of key insights. # The results show us that a lot of as experience increases, a degree like a master’s degree or PhD isn’t necessarily more popular. This supports the idea that experience is better at a certain point than education.

# Defining education level groupings  
lower\_deg = ["Bachelor's", "Associate", "GED", "No Education Listed", "High school"]  
higher\_deg = ["Master's degree", "PhD or professional degree"]  
  
# Adding 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")  
)  
  
# Casting necessary columns to float  
df = df.withColumn("MAX\_YEARS\_EXPERIENCE", col("MAX\_YEARS\_EXPERIENCE").cast("float"))  
df = df.withColumn("Average\_Salary", col("Average\_Salary").cast("float"))  
  
# Filtering 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)  
)  
  
# Filtering for just the two education groups  
df\_filtered = df.filter(col("EDU\_GROUP").isin("Bachelor's or lower", "Master's or PhD"))  
  
# Converting to Pandas for plotting  
df\_pd = df\_filtered.toPandas()

[Stage 24:> (0 + 1) / 1]

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=["#636EFA", "#EF553B"] # Blue, Red  
)  
  
fig1.update\_traces(marker=dict(size=7, line=dict(width=1, color="black")))  
  
fig1.update\_layout(  
 plot\_bgcolor="#f0f0f0", # light grey chart background  
 paper\_bgcolor="#FFF5DC", # soft blue frame  
 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 #show every integer year clearly  
 ),  
 yaxis=dict(gridcolor="lightgrey")  
)  
  
fig1.show()  
fig1.write\_html("output\q\_1a\_Experience\_vs\_Salary\_by\_Education\_Level")

<>:33: SyntaxWarning:  
  
invalid escape sequence '\q'  
  
<>:33: SyntaxWarning:  
  
invalid escape sequence '\q'  
  
/tmp/ipykernel\_11772/50348896.py:33: SyntaxWarning:  
  
invalid escape sequence '\q'



# 7. Salary by Remote Work Type

Split into three groups based on ‘REMOTE\_TYPE\_NAME’: Remote Hybrid Onsite (includes [None] and blank) Plot scatter plots for each group using, ‘MAX\_YEARS\_EXPERIENCE’ (with jitter), ‘Average\_Salary’, ‘LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME’ Also, create salary histograms for all three groups. After each graph, briefly describe any patterns or comparisons. # My Plot shows that Remote workers tend to have less years of experience, while most onsite workers have been at the company for longer periods of time.

# Step 1: Create the Average\_Salary column using SQL  
remote\_salary\_data = spark.sql("""  
SELECT   
 MAX\_YEARS\_EXPERIENCE,  
 (SALARY\_FROM + SALARY\_TO) / 2 AS Average\_Salary,  
 LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME,  
 CASE   
 WHEN REMOTE\_TYPE\_NAME = 'Remote' THEN 'Remote'  
 WHEN REMOTE\_TYPE\_NAME = 'Hybrid' THEN 'Hybrid'  
 ELSE 'Onsite'  
 END AS REMOTE\_GROUP  
FROM job\_postings  
WHERE SALARY\_FROM IS NOT NULL   
 AND SALARY\_TO IS NOT NULL  
 AND SALARY\_FROM > 0  
 AND SALARY\_TO > 0  
 AND MAX\_YEARS\_EXPERIENCE IS NOT NULL  
""")  
  
# Step 2: Convert to pandas for plotting  
df\_viz = remote\_salary\_data.toPandas()  
  
# Step 3: Create scatter plot (rest of code same as before)  
import plotly.express as px  
import numpy as np  
  
np.random.seed(42)  
df\_viz['EXPERIENCE\_JITTER'] = df\_viz['MAX\_YEARS\_EXPERIENCE'] + np.random.uniform(-0.3, 0.3, len(df\_viz))  
  
fig\_scatter = px.scatter(  
 df\_viz,  
 x='EXPERIENCE\_JITTER',  
 y='Average\_Salary',  
 color='REMOTE\_GROUP',  
 hover\_data=['LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME'],  
 title='Salary by Experience and Remote Work Type'  
)  
  
fig\_scatter.show()

[Stage 25:> (0 + 1) / 1]

