Assignment 03

Tracy Anyasi

September 23, 2025

#Tracy’s Github Repo - https://github.com/met-ad-688/assignment-03-tanyasiii #Note: There are double images in the qmd but not the word doc because kaleido (no matter how many times i download it), won’t covert my pictures to be shown in the word doc so i had to do it manually.

# 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(51)  
  
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).  
25/09/24 23:10:38 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable  
[Stage 1:> (0 + 1) / 1] 25/09/24 23:10:52 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'.

## Casting Salaries  
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("MIN\_YEARS\_EXPERIENCE", col("MIN\_YEARS\_EXPERIENCE").cast("float")) \  
 .withColumn("MAX\_YEARS\_EXPERIENCE", col("MAX\_YEARS\_EXPERIENCE").cast("float")) \  
 .withColumn("EDUCATION\_LEVELS\_NAME",regexp\_replace(col("EDUCATION\_LEVELS\_NAME"), r"[\n\r]", "")) \  
  
## Computing Medians  
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") #calculates median of median\_from salaries column  
median\_to = compute\_median(df, "SALARY\_TO")  
median\_salary = compute\_median(df, "SALARY")  
  
print("Medians:", median\_from, median\_to, median\_salary)  
  
## Imput missing using the 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) #calculates the average  
  
## removing null values in Employmet type column  
df = df.na.drop(subset=["EMPLOYMENT\_TYPE\_NAME"])  
  
# df.select("Average\_Salary", "SALARY", "EDUCATION\_LEVELS\_NAME", "REMOTE\_TYPE\_NAME", "MAX\_YEARS\_EXPERIENCE", "LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME").show(5, truncate=False)  
  
  
## selecting required columns & exporting data/ saving to csv  
export\_cols = [  
 "EDUCATION\_LEVELS\_NAME",  
 "REMOTE\_TYPE\_NAME",  
 "MAX\_YEARS\_EXPERIENCE",  
 "Average\_Salary",  
 "SALARY\_TO",  
 "SALARY\_FROM",  
 "SALARY",  
 "LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME",  
 "LOT\_OCCUPATION\_NAME",  
 "NAICS2\_NAME",  
 "EMPLOYMENT\_TYPE\_NAME",  
 "MIN\_YEARS\_EXPERIENCE"  
  
] #selects these columns to be inputted into a new csv file  
  
df\_selected = df.select(\*export\_cols)  
  
## export  
pdf = df\_selected.toPandas()  
pdf.to\_csv("lightcast\_cleaned.csv", index=False)  
  
#removing random characters from these columns  
pdf["EMPLOYMENT\_TYPE\_NAME"] = pdf["EMPLOYMENT\_TYPE\_NAME"].astype(str).apply(  
 lambda x: re.sub(r"[^\x00-\x7F]+", "", x)  
)  
pdf["EDUCATION\_LEVELS\_NAME"] = pdf["EDUCATION\_LEVELS\_NAME"].astype(str).str.replace(r"[\n\r\\\"\[\]]", "", regex=True)  
print(pdf.columns.tolist())  
  
print("Data cleaning complete. Row 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]

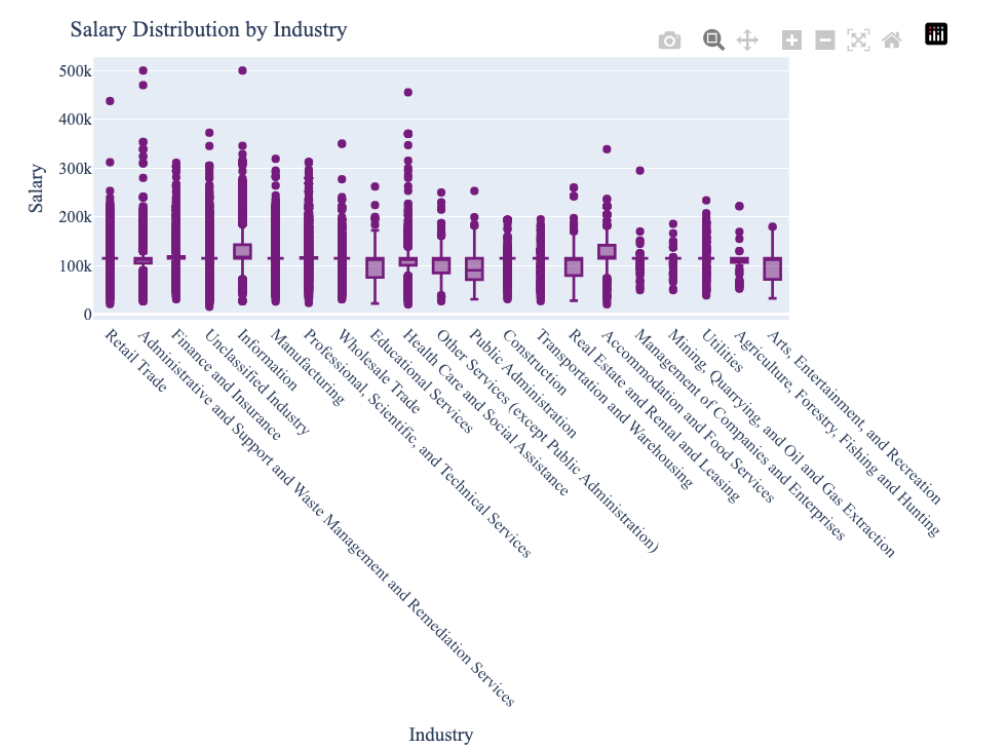
['EDUCATION\_LEVELS\_NAME', 'REMOTE\_TYPE\_NAME', 'MAX\_YEARS\_EXPERIENCE', 'Average\_Salary', 'SALARY\_TO', 'SALARY\_FROM', 'SALARY', 'LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME', 'LOT\_OCCUPATION\_NAME', 'NAICS2\_NAME', 'EMPLOYMENT\_TYPE\_NAME', 'MIN\_YEARS\_EXPERIENCE']  
Data cleaning complete. Row retained: 72454

# 2. Question 1a - Salary Distribution by Industry

fig = px.box(  
 pdf,  
 x="NAICS2\_NAME",  
 y="SALARY",  
 title="Salary Distribution by Industry",  
 color\_discrete\_sequence=["purple"],  
 points="outliers",  
)  
  
fig.update\_layout(  
 font\_family="Times New Roman",  
 title\_font\_size=16,  
 xaxis\_title="Industry",  
 yaxis\_title="Salary",  
 xaxis\_tickangle=45,  
)  
  
fig.show()  
fig.write\_html("Q1a.html") #makes picture in html link

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

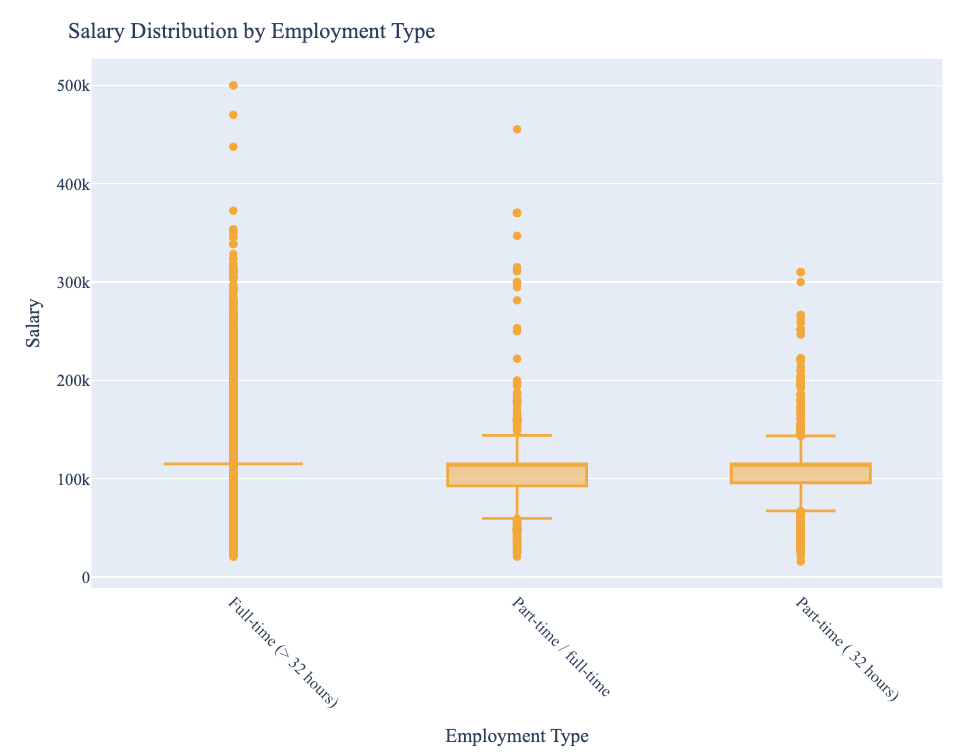
Unable to display output for mime type(s): text/html

 #Analysis: This box plot depicts how salaries vary across industries. Industries like Arts, Entertainment, and Recreation have a wider box, highlighting more variabilty in average salary, but with fewer outliers, most salaries stay within a predictable range. Unlike in Health Care and Social services, the smaller box shows consistency in typical salaries, but the presence of several outlies (above and below the median), indicate some employees make more or less than typical. It remarks on how different industries and their subsectors can affect the chances of making around average salary or not.

# 3. Question 1b - Salary Distribution by Employment Type

fig = px.box(  
 pdf,  
 x="EMPLOYMENT\_TYPE\_NAME",  
 y="SALARY",  
 title="Salary Distribution by Employment Type",  
 color\_discrete\_sequence=["orange"],  
 points="outliers",  
)  
  
fig.update\_layout(  
 font\_family="Times New Roman",  
 title\_font\_size=16,  
 xaxis\_title="Employment Type",  
 yaxis\_title="Salary",  
 xaxis\_tickangle=45,  
)  
  
fig.show()  
fig.write\_html("Q1b.html")  
#fig.write\_image("Q1b.png")

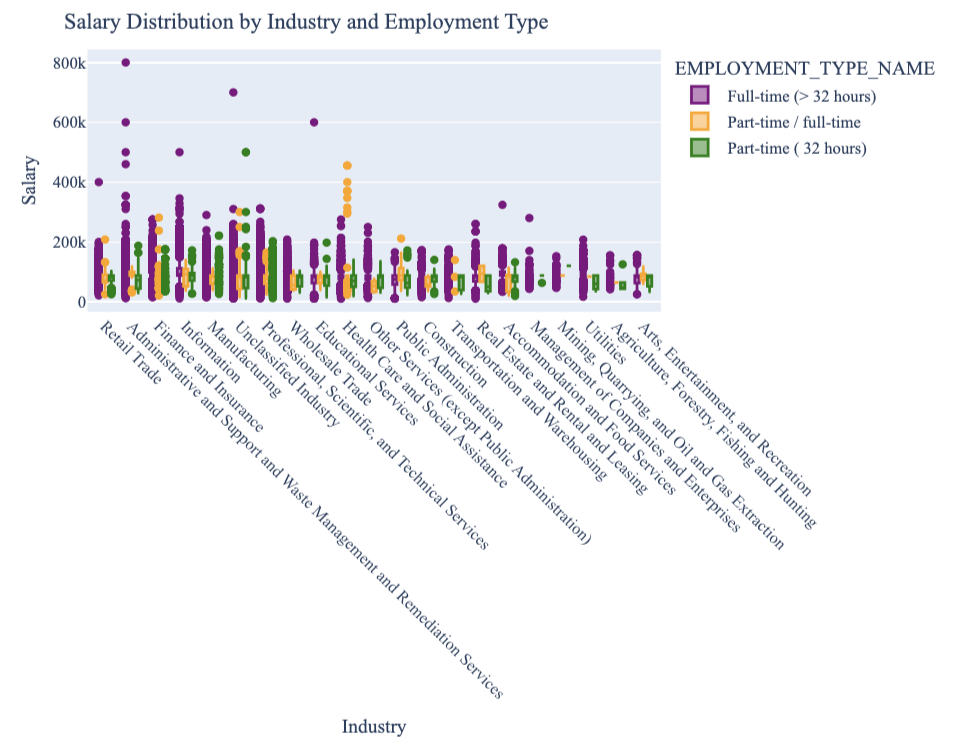
Unable to display output for mime type(s): text/html

 #Analysis: Despite all three having outliers, Full-time employees have more consistency in salary expectations than part-time employees as their hours are set while parttime might fluctate based on the needs of the business. Because of the increased variability, there is more job security and stability in full-time roles over part-time roles.

# 4. Question 1a and b together

fig = px.box(  
 pdf,  
 x="NAICS2\_NAME",  
 y="SALARY\_FROM",  
 color="EMPLOYMENT\_TYPE\_NAME", #groups the naics2\_names by employment type  
 points="outliers",  
 title="Salary Distribution by Industry and Employment Type",  
 color\_discrete\_sequence=["purple", "orange", "green"]  
)  
  
fig.update\_layout(  
 font\_family="Times New Roman",  
 title\_font\_size=16,  
 xaxis\_title="Industry",  
 yaxis\_title="Salary",  
 xaxis\_tickangle=45,  
 boxmode="group" # groups by employment type  
)  
  
fig.show()

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

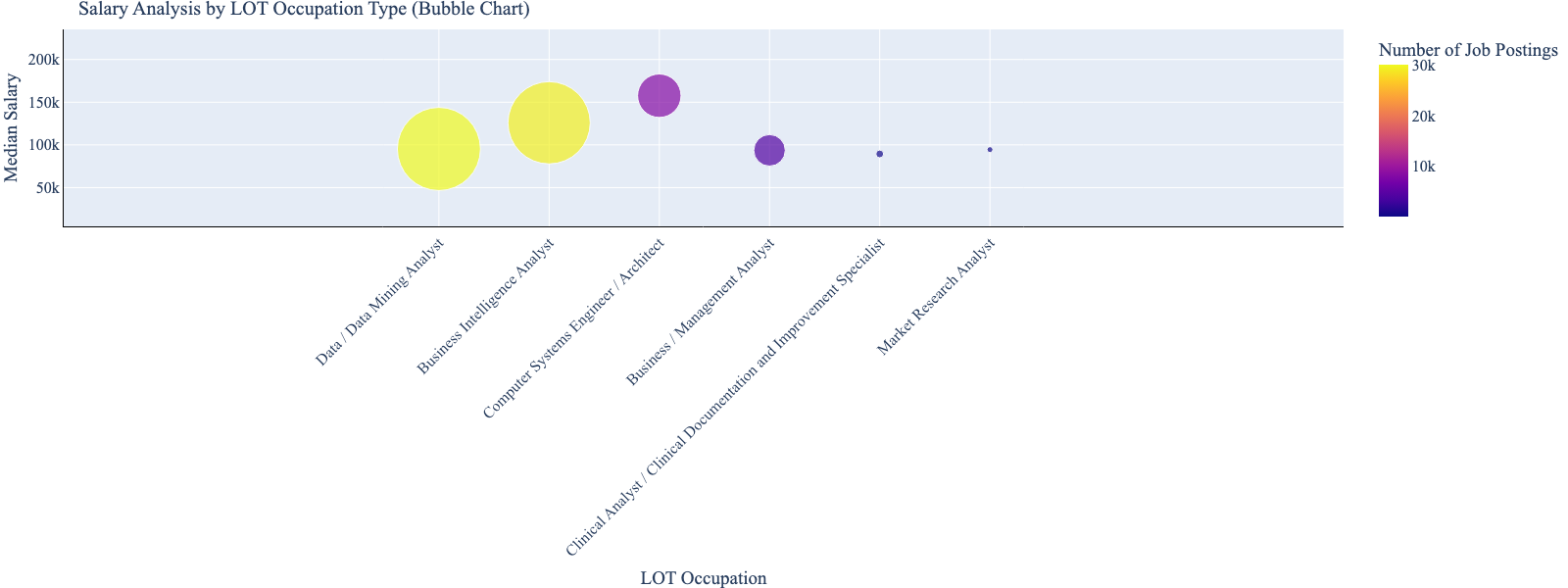


#Question 2 - Salary Analysis by ONET Occupation Type

#calculating median for each occupation  
saonet = 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  
"""  
)  
  
saonet\_pd = saonet.toPandas()  
saonet\_pd.head()  
  
  
fig = px.scatter(  
 saonet\_pd,  
 x="Occupation\_Name",  
 y="Median\_Salary",  
 size="Job\_Postings",  
 title="Salary Analysis by LOT Occupation Type (Bubble Chart)",  
 labels={  
 "LOT\_OCCUPATION\_NAME": "LOT Occupation",  
 "Median\_Salary": "Median Salary",  
 "Job\_Postings": "Number of Job Postings"  
 },  
 hover\_name="Occupation\_Name",  
 size\_max=60,  
 width=1600,  
 height=600,  
 color="Job\_Postings",  
 color\_continuous\_scale="Plasma"  
)  
  
fig.update\_layout(  
 font\_family="Times New Roman",  
 font\_size=16,  
 title\_font\_size=20,  
 xaxis\_title="LOT Occupation",  
 yaxis\_title="Median Salary",  
 xaxis=dict(  
 tickangle=-45,  
 showline=True,  
 linecolor="black"  
 ),  
 yaxis=dict(  
 showline=True,  
 linecolor="black"  
 )  
)  
  
fig.show()  
fig.write\_html("Q2.html")

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

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

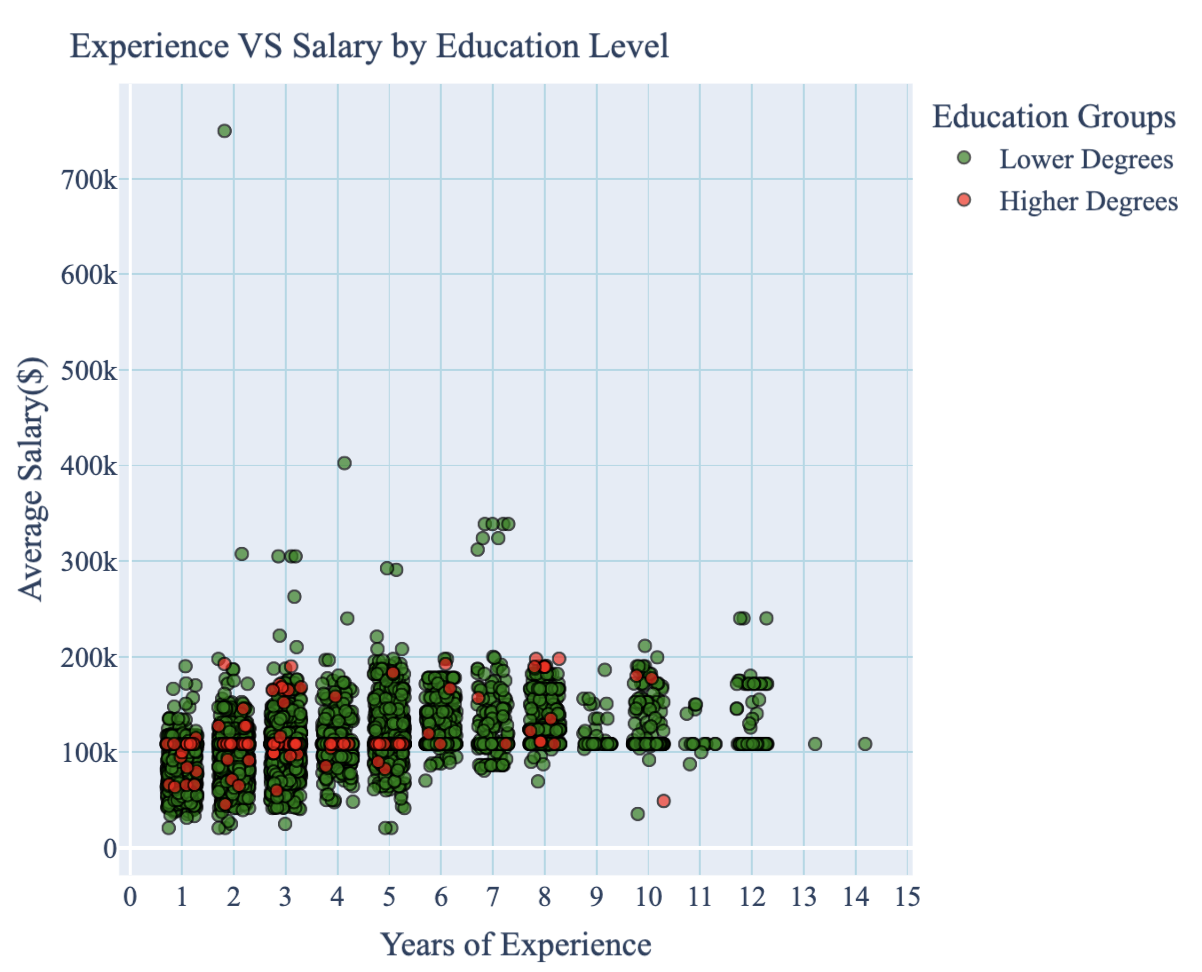
 #Analysis: The chart depicts how variation in median salary and earning potential can be narrow in some occupations and widespread in others. For example Computer Systems Engineer/ Architect have the highest median salary but has a small range for salary variation, while Data/ Data Mining Analyst have a lower median salary but wider range for salary variation.

#Question 3 - Salary by Educational Level

df = df.withColumn(  
 "EDU\_GROUP",  
 when(col("EDUCATION\_LEVELS\_NAME").rlike("(?i)Bachelor'?s?|Associate|GED|Highschool|No Education"), "Lower Degrees")  
 .when(col("EDUCATION\_LEVELS\_NAME").rlike("(?i)Master'?s?|PhD|Doctorate|professional"), "Higher Degrees")  
 .otherwise("Other")  
) #selects these items in the education column, splits them into two groups  
  
df = df.withColumn("MAX\_YEARS\_EXPERIENCE", col("MAX\_YEARS\_EXPERIENCE").cast("float"))  
df = df.withColumn("Average\_Salary", col("Average\_Salary").cast("float"))  
  
# cleaning up data  
df = df.filter(  
 col("MAX\_YEARS\_EXPERIENCE").isNotNull() &  
 (col("MAX\_YEARS\_EXPERIENCE")>0) &  
 col("Average\_Salary").isNotNull() &  
 (col("Average\_Salary")>0)  
)  
  
df\_filtered = df.filter(col("EDU\_GROUP").isin(["Lower Degrees", "Higher Degrees"]))  
  
df\_pd = df\_filtered.toPandas()  
  
# Add jitter readability  
np.random.seed(51)  
df\_pd["Experience\_Jitter"] = df\_pd["MAX\_YEARS\_EXPERIENCE"] + np.random.uniform(-0.3, 0.3, size=len(df\_pd))  
  
fig = px.scatter(  
 df\_pd,  
 x="Experience\_Jitter",  
 y="Average\_Salary",  
 color="EDU\_GROUP",  
 hover\_data=["LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME"],  
 title="Experience VS Salary by Education Level",  
 opacity = 0.7, #how translucent each dot is  
 color\_discrete\_sequence=["green", "red"]  
)  
  
fig.update\_traces(marker=dict(size=7, line=dict(width=1, color="black")))  
  
fig.update\_layout(  
 font\_family="Times New Roman",  
 title\_font=dict(size=20),  
 font=dict(size=16),  
 xaxis\_title = "Years of Experience",  
 yaxis\_title = "Average Salary($)",  
 legend\_title = "Education Groups",  
 xaxis=dict(  
 gridcolor="lightblue",  
 tickmode="linear",  
 dtick=1  
 ),  
 yaxis=dict(  
 gridcolor = "lightblue"  
 )  
  
)  
fig.show()  
fig.write\_html("Q3.html")

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

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

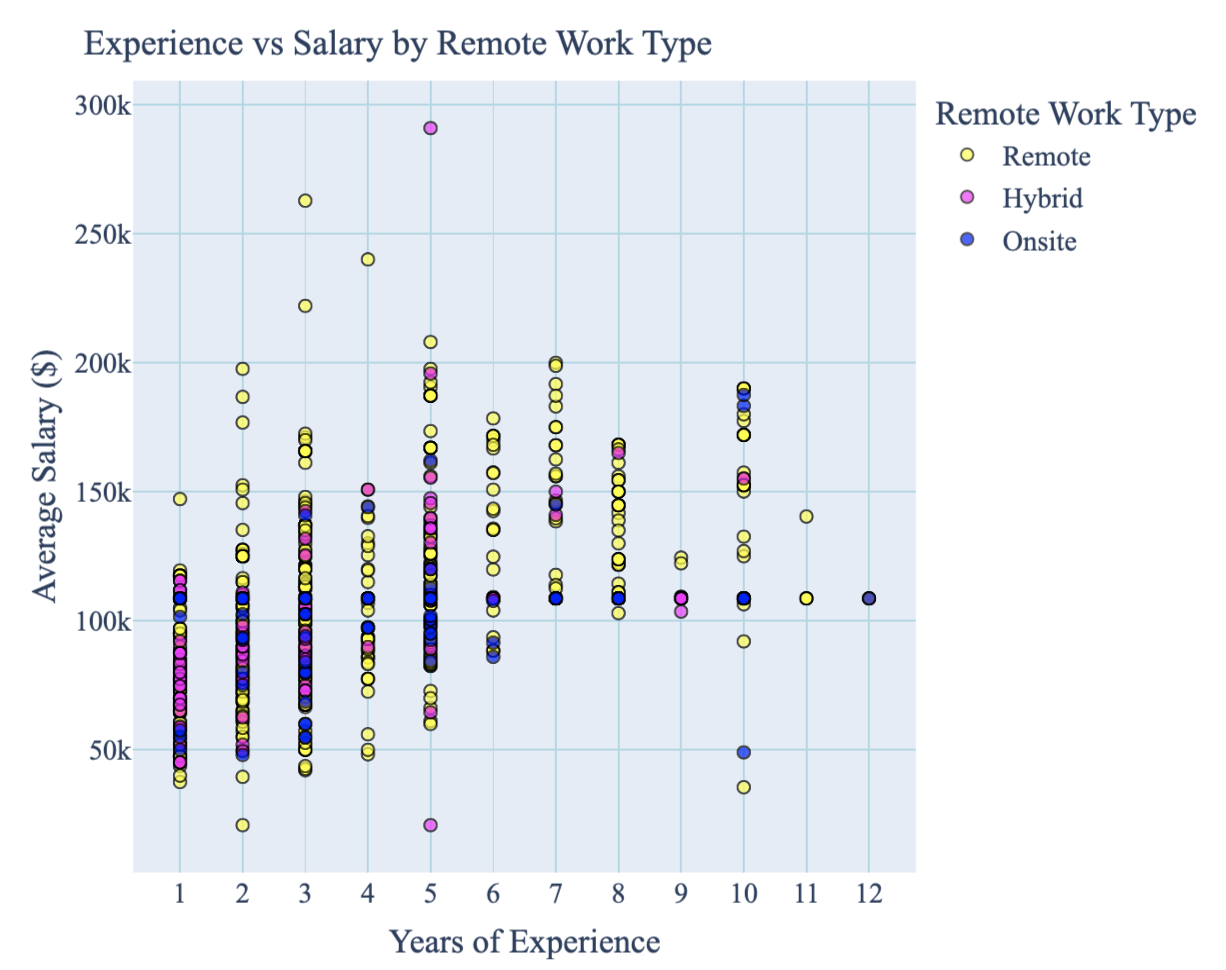
 #Analysis: Although there is a positive relationship between the years of experience (YOE) and salary for both education groups, it is evident that those with Higher degrees (Master’s and PhDs) at all YOEs make more than the Lower degrees (GED, Associate, Bachelor’s). This highlights how advanced education correlates to higher pay.

#Question 4 - Salary by Remote Work type

df = df.withColumn(  
 "REMOTE\_GROUP",  
 when(col("REMOTE\_TYPE\_NAME").rlike("(?i)^Remote$"), "Remote")  
 .when(col("REMOTE\_TYPE\_NAME").rlike("(?i)^Hybrid Remote$"), "Hybrid")  
 .when(col("REMOTE\_TYPE\_NAME").isNull() | col("REMOTE\_TYPE\_NAME").rlike("(?i)^Not Remote$"), "Onsite")  
 .otherwise("Other")  
) #selects these values from remote type name column  
  
# --- Step 2: Keep numeric columns as float & filter valid rows ---  
df = df.withColumn("MAX\_YEARS\_EXPERIENCE", col("MAX\_YEARS\_EXPERIENCE").cast("float"))  
df = df.withColumn("Average\_Salary", col("Average\_Salary").cast("float"))  
  
#cleaning data  
df = df.filter(  
 col("MAX\_YEARS\_EXPERIENCE").isNotNull() &  
 (col("MAX\_YEARS\_EXPERIENCE")>0) &  
 col("Average\_Salary").isNotNull() &  
 (col("Average\_Salary")>0)  
)  
  
# Filter only the main three remote types  
df\_filtered = df.filter(col("REMOTE\_GROUP").isin(["Remote", "Hybrid", "Onsite"]))  
  
df\_pd = df\_filtered.toPandas()  
  
  
fig = px.scatter(  
 df\_pd,  
 x="MAX\_YEARS\_EXPERIENCE",  
 y="Average\_Salary",  
 color="REMOTE\_GROUP",  
 hover\_data=["LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME"],  
 title="Experience vs Salary by Remote Work Type",  
 opacity=0.7,  
 color\_discrete\_sequence=["yellow", "magenta", "blue"]  
)  
  
fig.update\_traces(marker=dict(size=7, line=dict(width=1, color="black")))  
  
fig.update\_layout(  
 font\_family="Times New Roman",  
 title\_font=dict(size=20),  
 font=dict(size=16),  
 xaxis\_title="Years of Experience",  
 yaxis\_title="Average Salary ($)",  
 legend\_title="Remote Work Type",  
 xaxis=dict(gridcolor="lightblue", tickmode="linear", dtick=1),  
 yaxis=dict(gridcolor="lightblue")  
)  
  
fig.show()  
fig.write\_html("Q4.html")

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

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

 #Analysis: Both Remote and Hybrid roles have good average salaries that are similar and sometimes exceed those of onsite positions, particularly in more mid level and senior roles. This shows that there is no decuction in salary from working remote, and that with more years of expereince, the roles lean towards remote expectations.

#Question 4 - Salary by Remote Work type (histogram)

fig = px.histogram(  
 df\_pd,  
 x="Average\_Salary",  
 color="REMOTE\_GROUP",  
 barmode="overlay", #bars are layed unto of each other  
 nbins=30, #each bin for bar is $10K  
 opacity=0.7,  
 color\_discrete\_sequence=["yellow", "magenta", "blue"],  
 title="Salary Distribution by Remote Work Type - bar graph"  
)  
  
# Update layout  
fig.update\_layout(  
 font\_family="Times New Roman",  
 title\_font=dict(size=20),  
 font=dict(size=16),  
 xaxis\_title="Average Salary ($)",  
 yaxis\_title="Count",  
 legend\_title="Remote Work Type",  
 xaxis=dict(gridcolor="lightblue"),  
 yaxis=dict(gridcolor="lightblue")  
)  
  
fig.show()  
fig.write\_html("Q4Histogram.html")

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

