Assignment 03

Fuhan Zhang

September 21, 2025

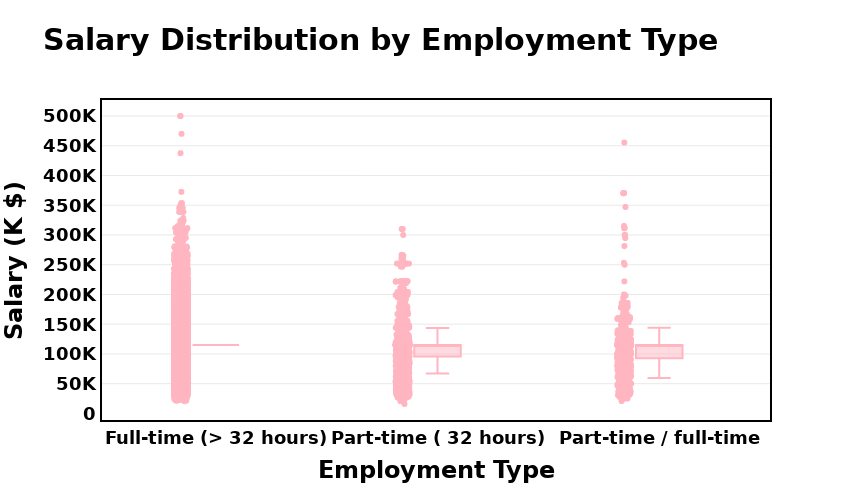
# 1. Load the Dataset

import os, sys  
os.environ["PYSPARK\_PYTHON"] = sys.executable  
os.environ["PYSPARK\_DRIVER\_PYTHON"] = sys.executable  
os.environ.pop("SPARK\_HOME", None)  
os.environ.pop("SPARK\_DIST\_CLASSPATH", None)  
os.makedirs("./output", exist\_ok=True)  
from pyspark.sql import SparkSession  
from pyspark.sql.functions import col, monotonically\_increasing\_id  
from pyspark.sql import functions as F  
import pandas as pd  
import numpy as np  
import plotly.io as pio  
from IPython.display import Image, display  
def show\_and\_save(fig, path, width=950, height=550, scale=2):  
 png\_bytes = fig.to\_image(format="png", width=width, height=height, scale=scale)  
 with open(path, "wb") as f:  
 f.write(png\_bytes)  
 display(Image(png\_bytes))   
np.random.seed(42)  
# 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)  
df = df.withColumn("SALARY", col("SALARY").cast("float")) \  
 .withColumn("MAX\_YEARS\_EXPERIENCE", col("MAX\_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\_salary = compute\_median(df, "SALARY")  
print("Median SALARY:", median\_salary)  
df = df.fillna({"SALARY": median\_salary})  
df = df.withColumn("Average\_Salary", col("SALARY"))  
export\_cols = [  
 "EDUCATION\_LEVELS\_NAME",  
 "REMOTE\_TYPE\_NAME",  
 "MAX\_YEARS\_EXPERIENCE",  
 "Average\_Salary",  
 "SALARY",  
 "LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME",  
 "EMPLOYMENT\_TYPE\_NAME"  
]  
df\_selected = df.select(\*export\_cols)  
pdf= df\_selected.toPandas()  
pdf.to\_csv("./output/cleaned\_subset.csv", index=False)  
print("Data Cleaning Complete. Rows Retained:", len(pdf))

root  
 |-- EDUCATION\_LEVELS\_NAME: string (nullable = true)  
 |-- REMOTE\_TYPE\_NAME: string (nullable = true)  
 |-- MAX\_YEARS\_EXPERIENCE: double (nullable = true)  
 |-- Average\_Salary: double (nullable = true)  
 |-- SALARY: double (nullable = true)  
 |-- LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME: string (nullable = true)  
 |-- EMPLOYMENT\_TYPE\_NAME: string (nullable = true)  
  
Median SALARY: 115024.0  
Data Cleaning Complete. Rows Retained: 72498

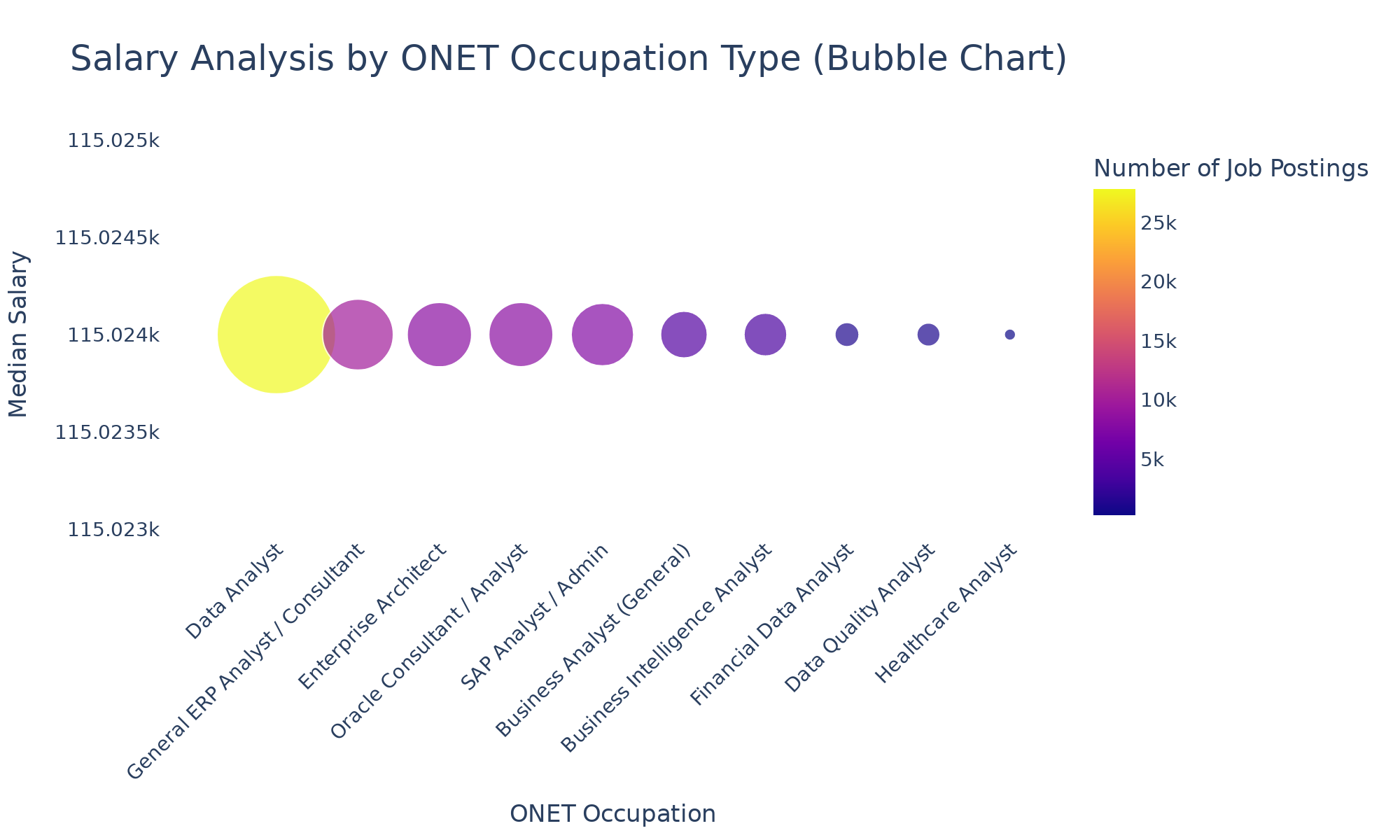
# 2. Salary Distribution by Employment Type

from pyspark.sql import SparkSession  
from pyspark.sql.functions import col  
import re  
import plotly.express as px  
import plotly.io as pio  
os.makedirs("output", exist\_ok=True)  
#Data Cleaning & Filtering  
pdf = df.filter(df["SALARY"] > 0).select("EMPLOYMENT\_TYPE\_NAME", "SALARY").toPandas()  
pdf["EMPLOYMENT\_TYPE\_NAME"] = pdf["EMPLOYMENT\_TYPE\_NAME"].apply(  
 lambda x: re.sub(r"[^\x00-\x7F]+", "", str(x)) if x is not None else x  
)  
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  
)  
#Creating the Boxplot  
fig = px.box(  
 pdf,  
 x="EMPLOYMENT\_TYPE\_NAME",  
 y="SALARY",  
 title="Salary Distribution by Employment Type",  
 color\_discrete\_sequence=["#ffb6c1", "#cb1a72ff", "#db7093", "#c71585"],  
 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")),  
 margin=dict(t=100, b=80, l=80, r=80),  
 xaxis=dict(  
 title=dict(text="Employment Type", font=dict(size=24, family="Arial", color="black", weight="bold")),  
 tickangle=0,  
 tickfont=dict(size=18, 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", "250K", "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="lightgrey", gridwidth=0.5  
 ),  
 font=dict(family="Arial", size=16, color="black"),  
 boxgap=0.7,  
 plot\_bgcolor="white",  
 paper\_bgcolor="white",  
 showlegend=False,  
 height=500,  
 width=850  
)  
show\_and\_save(fig, "output/Q1.png", width=850, height=500, scale=1)



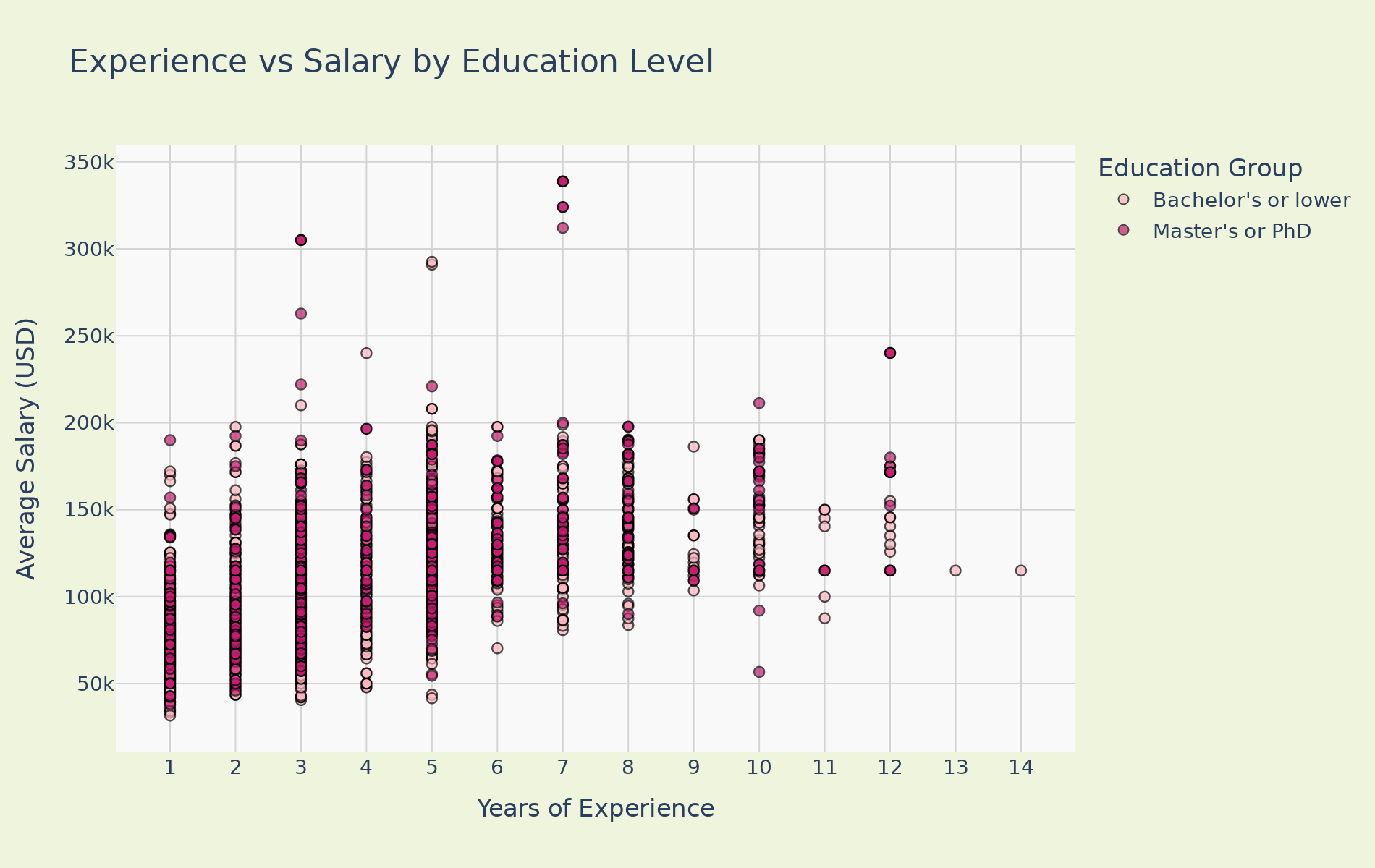
# 3. Salary Analysis by ONET Occupation Type (Bubble Chart)

from pyspark.sql import SparkSession  
from pyspark.sql.functions import col  
import plotly.express as px  
os.makedirs("output", exist\_ok=True)  
#Spark SQL to Converting  
salary\_analysis = 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  
""")  
salary\_pd = salary\_analysis.toPandas()  
#Creating Bubble Chart  
fig = px.scatter(  
 salary\_pd,  
 x="ONET\_NAME",  
 y="Median\_Salary",  
 size="Job\_Postings",  
 title="Salary Analysis by ONET Occupation Type (Bubble Chart)",  
 labels={  
 "ONET\_NAME": "ONET Occupation",  
 "Median\_Salary": "Median Salary",  
 "Job\_Postings": "Number of Job Postings"  
 },  
 hover\_name="ONET\_NAME",  
 size\_max=60,  
 width=1000,  
 height=600,  
 color="Job\_Postings",  
 color\_discrete\_sequence=["#ffe4e1", "#ffb6c1", "#ff69b4", "#db7093", "#c71585"],  
)  
fig.update\_layout(  
 font\_family="Arial",  
 font\_size=14,  
 title\_font\_size=25,  
 xaxis\_title="ONET Occupation",  
 yaxis\_title="Median Salary",  
 plot\_bgcolor="white",  
 xaxis=dict(tickangle=-45, showline=True),  
 yaxis=dict(showline=True),  
 margin=dict(t=100, b=80, l=80, r=80)  
)  
show\_and\_save(fig, "output/Q2.png", width=1000, height=600, scale=2)



# 4. Salary by Education Level

from pyspark.sql import SparkSession  
from pyspark.sql.functions import col, when, lit, trim, lower, regexp\_replace  
import plotly.express as px  
import plotly.io as pio  
os.makedirs("output", exist\_ok=True)  
#Building Educational Level Groups  
df = df.withColumn("EDU\_CLEAN", lower(trim(col("EDUCATION\_LEVELS\_NAME"))))  
df = df.withColumn(  
 "EDU\_GROUP",  
 when(col("EDU\_CLEAN").rlike("master|mba|msc"), lit("Master's or PhD"))  
 .when(col("EDU\_CLEAN").rlike("phd|doctor|professional"), lit("Master's or PhD"))  
 .when(col("EDU\_CLEAN").rlike("bachelor|associate|ged|high\\s\*school|no\\s\*education"), lit("Bachelor's or lower"))  
 .otherwise(lit("Other"))  
)  
df = df.withColumn("MAX\_YEARS\_EXPERIENCE", col("MAX\_YEARS\_EXPERIENCE").cast("double"))  
df = df.withColumn("Average\_Salary",  
 regexp\_replace(col("Average\_Salary"), "[$,]", "").cast("double"))  
df\_filtered = (  
 df.filter((col("MAX\_YEARS\_EXPERIENCE") > 0) & (col("Average\_Salary") > 0))  
 .filter(col("EDU\_GROUP").isin("Bachelor's or lower", "Master's or PhD"))  
 .select("MAX\_YEARS\_EXPERIENCE","Average\_Salary","EDU\_GROUP","LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME")  
)  
df\_pd = df\_filtered.toPandas()  
  
#Creating the Scatter Plot  
fig1 = px.scatter(  
 df\_pd,  
 x="MAX\_YEARS\_EXPERIENCE",  
 y="Average\_Salary",  
 color="EDU\_GROUP",  
 hover\_data=["LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME"],  
 opacity=0.7,  
 color\_discrete\_sequence=["#ffb6c1", "#cb1a72"],   
 title="Experience vs Salary by Education Level"  
)  
fig1.update\_traces(marker=dict(size=7, line=dict(width=1, color="black")))  
fig1.update\_layout(  
 plot\_bgcolor="#f9f9f9", paper\_bgcolor="#EFF5DC",  
 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"),  
 xaxis=dict(gridcolor="lightgrey", tickmode="linear", dtick=1),  
 yaxis=dict(gridcolor="lightgrey"),  
 margin=dict(t=100, b=80, l=80, r=80)  
)  
show\_and\_save(fig1, "output/Q3.png", width=950, height=600, scale=2)



# 5. Salary by Remote Work Type

from pyspark.sql import SparkSession  
from pyspark.sql.functions import col  
from pyspark.sql.functions import when, trim  
import numpy as np  
import plotly.express as px  
import plotly.io as pio  
os.makedirs("output", exist\_ok=True)  
np.random.seed(42)  
  
#Work Types Data Setting  
df = df.withColumn("REMOTE\_GROUP",  
 when(trim(col("REMOTE\_TYPE\_NAME")) == "Remote", "Remote")  
 .when(trim(col("REMOTE\_TYPE\_NAME")) == "Hybrid Remote", "Hybrid")  
 .when(trim(col("REMOTE\_TYPE\_NAME")) == "Not Remote", "Onsite")  
 .when(col("REMOTE\_TYPE\_NAME").isNull(), "Onsite")  
 .otherwise("Onsite")  
)  
df = df.filter(  
 (col("MAX\_YEARS\_EXPERIENCE").isNotNull()) &  
 (col("Average\_Salary").isNotNull()) &  
 (col("MAX\_YEARS\_EXPERIENCE") > 0) &  
 (col("Average\_Salary") > 0)  
)  
df\_pd = df.select(  
 "MAX\_YEARS\_EXPERIENCE",   
 "Average\_Salary",   
 "LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME",   
 "REMOTE\_GROUP"  
).toPandas()  
#Mathematical Adjusting  
df\_pd["MAX\_EXPERIENCE\_JITTER"] = df\_pd["MAX\_YEARS\_EXPERIENCE"] + np.random.uniform(-0.25, 0.25, size=len(df\_pd))  
df\_pd["AVERAGE\_SALARY\_JITTER"] = df\_pd["Average\_Salary"] + np.random.uniform(-2500, 2500, size=len(df\_pd))  
df\_pd = df\_pd.round(2)  
df\_pd.head()  
df\_pd = df\_pd[df\_pd["AVERAGE\_SALARY\_JITTER"] <= 390000]  
  
#Creating the Scatter Plot  
fig1 = px.scatter(  
 df\_pd,  
 x="MAX\_EXPERIENCE\_JITTER",  
 y="AVERAGE\_SALARY\_JITTER",  
 color="REMOTE\_GROUP",  
 hover\_data=["LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME"],  
 title="<b>Experience vs Salary by Remote Work Type</b>",  
 opacity=0.7,  
 color\_discrete\_sequence=["#f9acb7", "#96d8ee", "#f3f39a"]   
)  
  
fig1.update\_traces(marker=dict(size=7, line=dict(width=1, color="black")))  
  
fig1.update\_layout(  
 plot\_bgcolor="#f9f9f9",  
 paper\_bgcolor="#E9EAFF",  
 font=dict(family="Segoe UI", size=14),  
 title\_font=dict(size=22),  
 margin=dict(t=100, b=80, l=80, r=80),  
 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"),  
 xaxis=dict(  
 gridcolor="lightgrey",  
 tickmode="linear",  
 tick0=1,  
 dtick=1,  
 tickangle=0  
 ),  
 yaxis=dict(gridcolor="lightgrey"),  
 legend=dict(orientation="h", yanchor="bottom", y=-1.02, xanchor="right", x=1)  
)  
  
show\_and\_save(fig1, "output/Q4.png", width=950, height=550, scale=2)

