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

```python
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","t
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 12:>                                                         (0 + 1) / 1]
```

```
## 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"[\

## 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 co
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 t

## 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_YEA
```

```python
## 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\
print(pdf.columns.tolist())

print("Data cleaning complete. Row retained:", len(pdf))
```

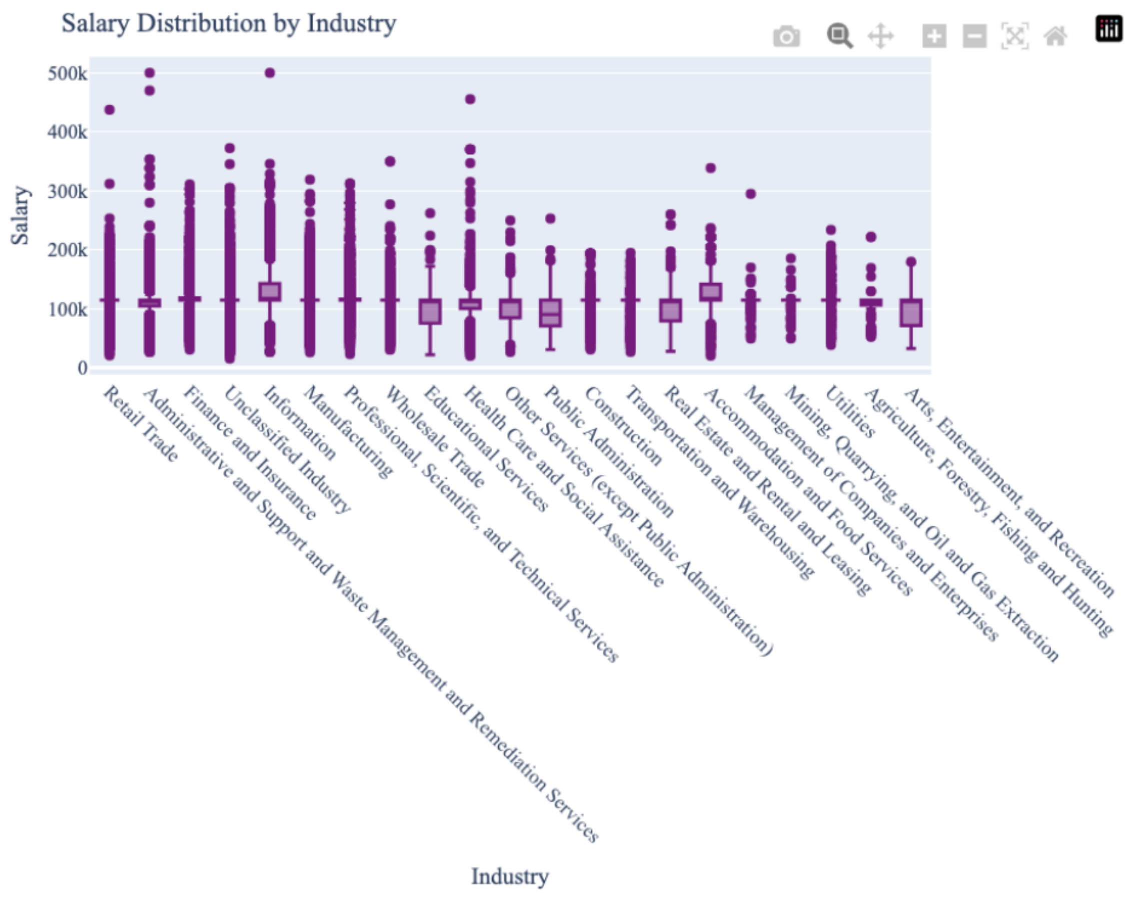
```
[Stage 13:>                                                         (0 + 1) / 1]

Medians: 87295.0 130042.0 115024.0

[Stage 16:>                                                         (0 + 1) / 1]

['EDUCATION_LEVELS_NAME', 'REMOTE_TYPE_NAME', 'MAX_YEARS_EXPERIENCE', 'Average_Salary', 'SAL
Data cleaning complete. Row retained: 72454
```

2 Question 1a - Salary Distribution by Industry

```python
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
```

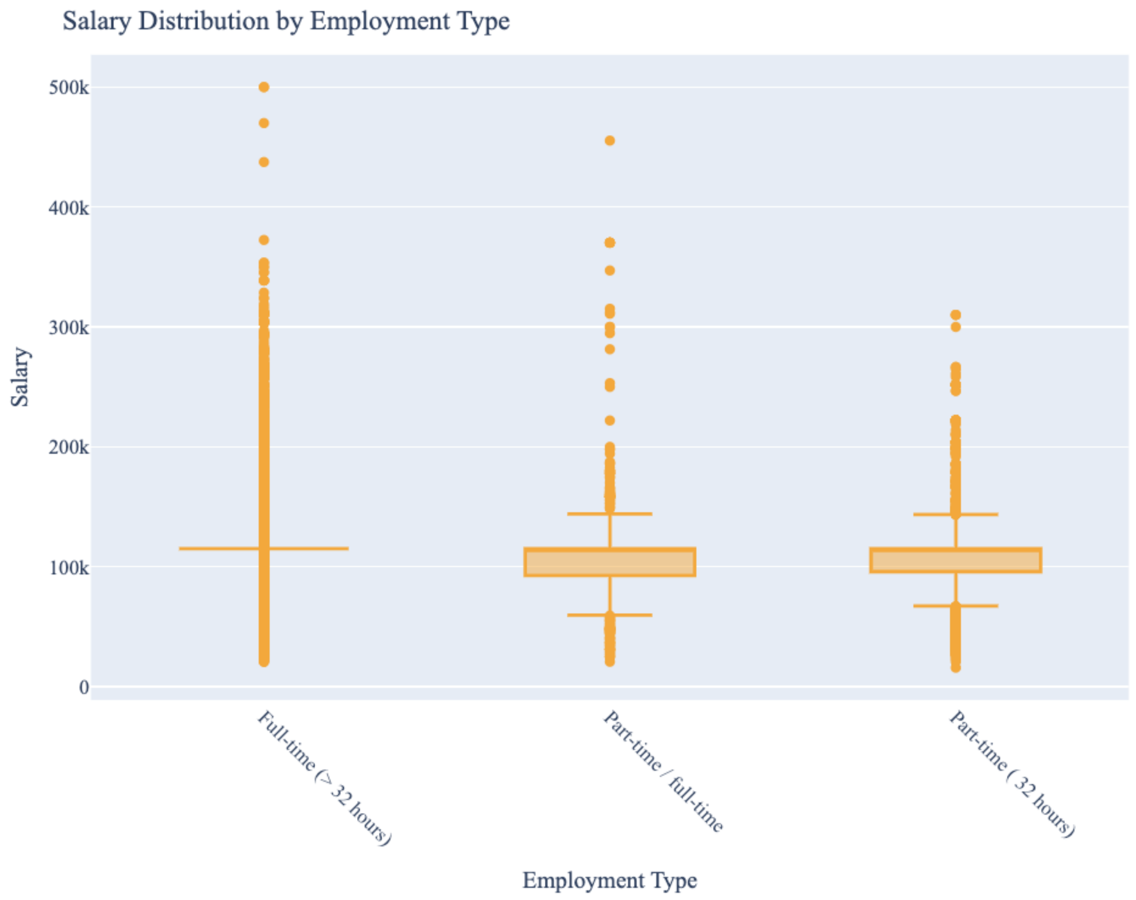


#Analysis: This box plot depicts how salaries vary across industries. Industries like Arts, Entertainment, and Recreation have a wider box, highlighting more variabilty in average salary, but with fewer outliers, most salaries stay within a predictable range. Unlike in Health Care and Social services, 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
```

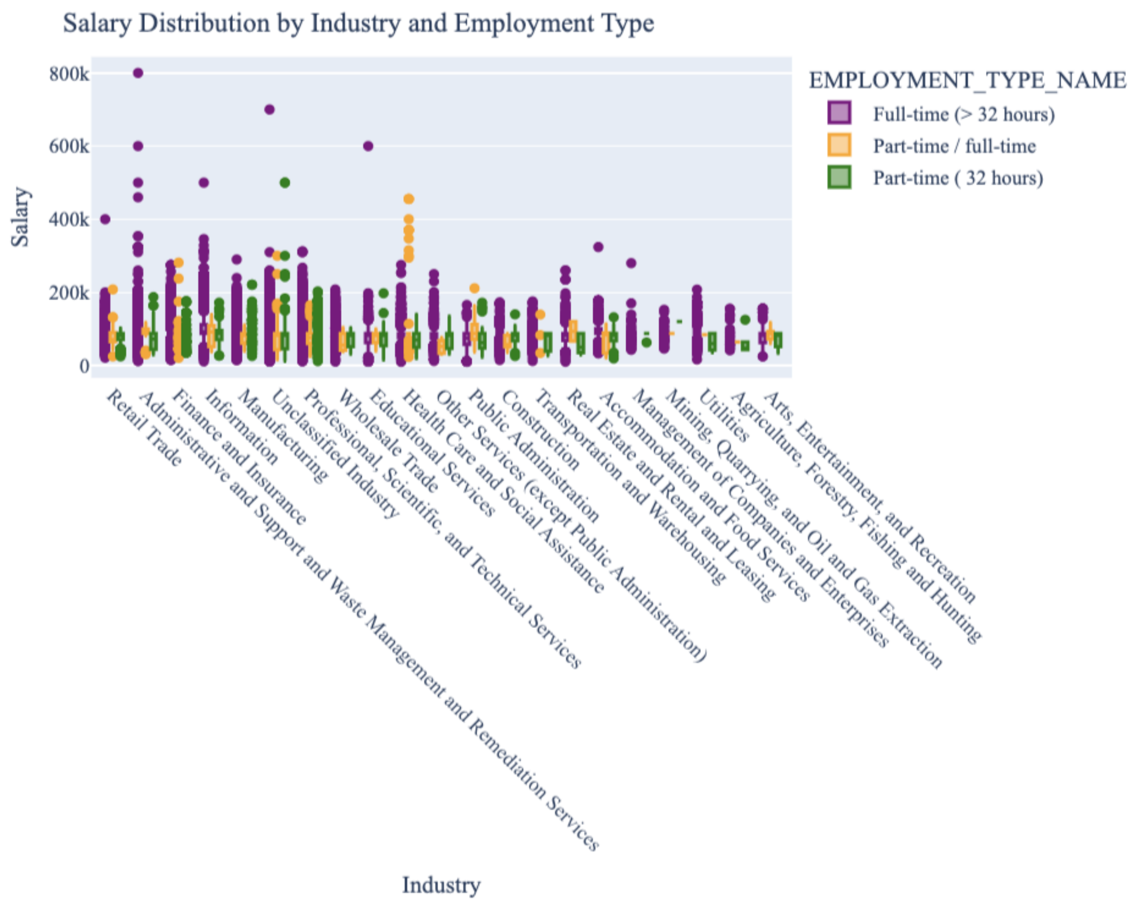


#Analysis: Despite all three having outliers, Full-time employees have more consistency in salary expectations than part-time employees as their hours are set while parttime might fluctate based on the needs of the business. Because of the increased variability, there is more job security and stability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

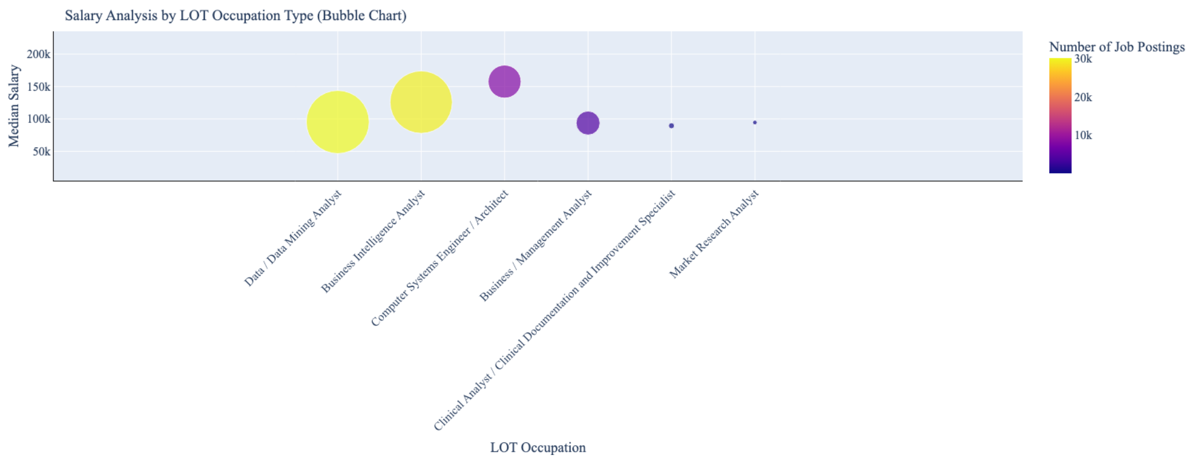
```
Unable to display output for mime type(s): text/html
```



#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 17:>                                                         (0 + 1) / 1]

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

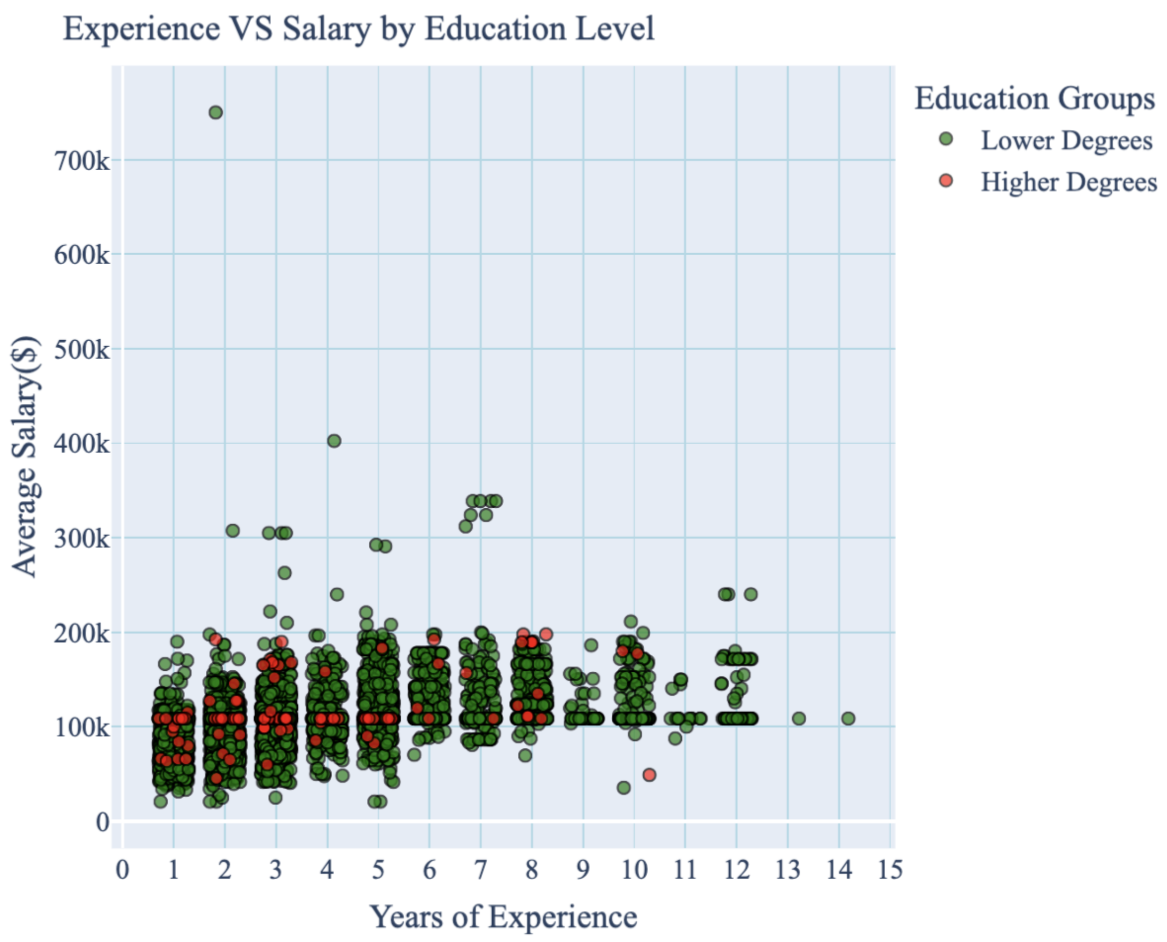


#Analysis: The chart depicts how variation in median salary and earning potential can be narrow in some occupations and widespread in others. For example Computer Systems Engineer/ Architect have the highest median salary but has a small range for salary variation, while Data/ Data Mining Analyst have 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 Educa
  .when(col("EDUCATION_LEVELS_NAME").rlike("(?i)Master'?s?|PhD|Doctorate|professional"), "Hig
  .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, si

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 20:>                                                          (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.

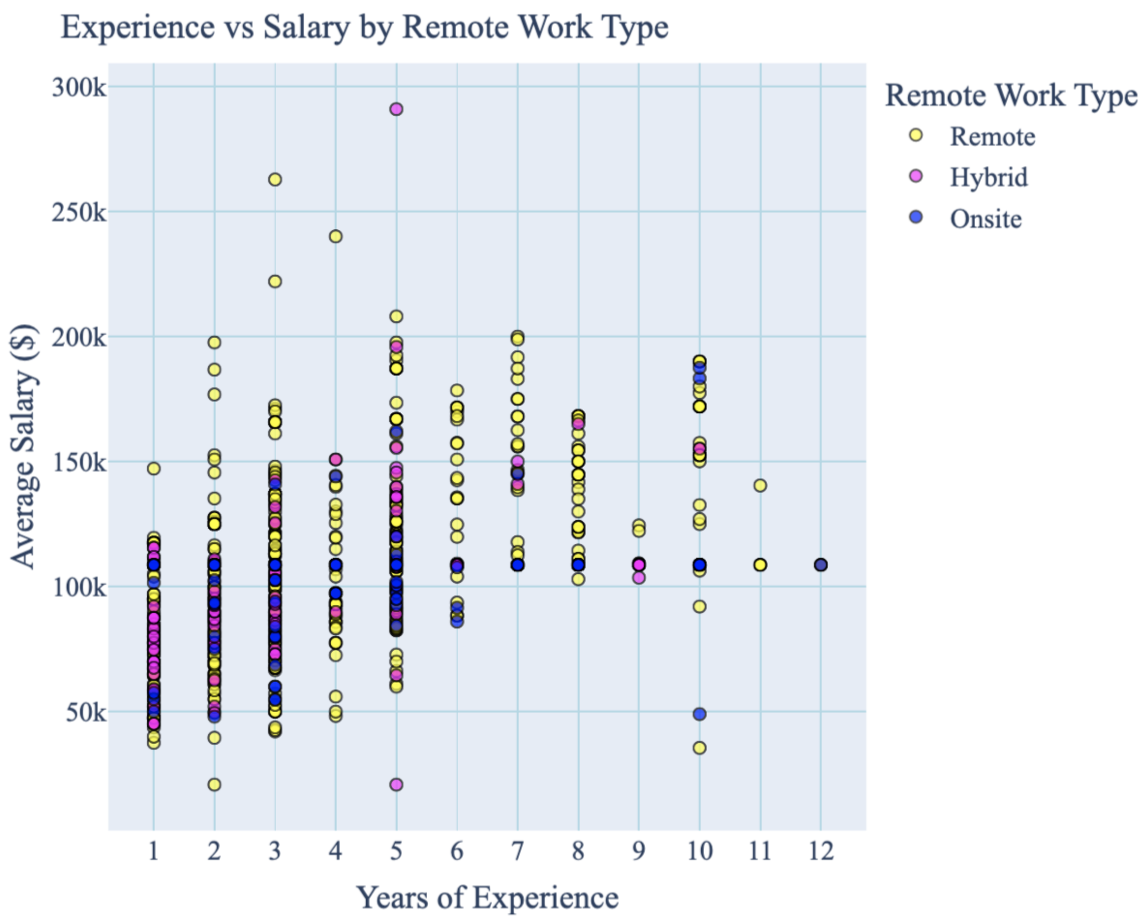
#Question 4 - Salary by Remote Work type

```
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$"
    .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 21:>                                                         (0 + 1) / 1]
```

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

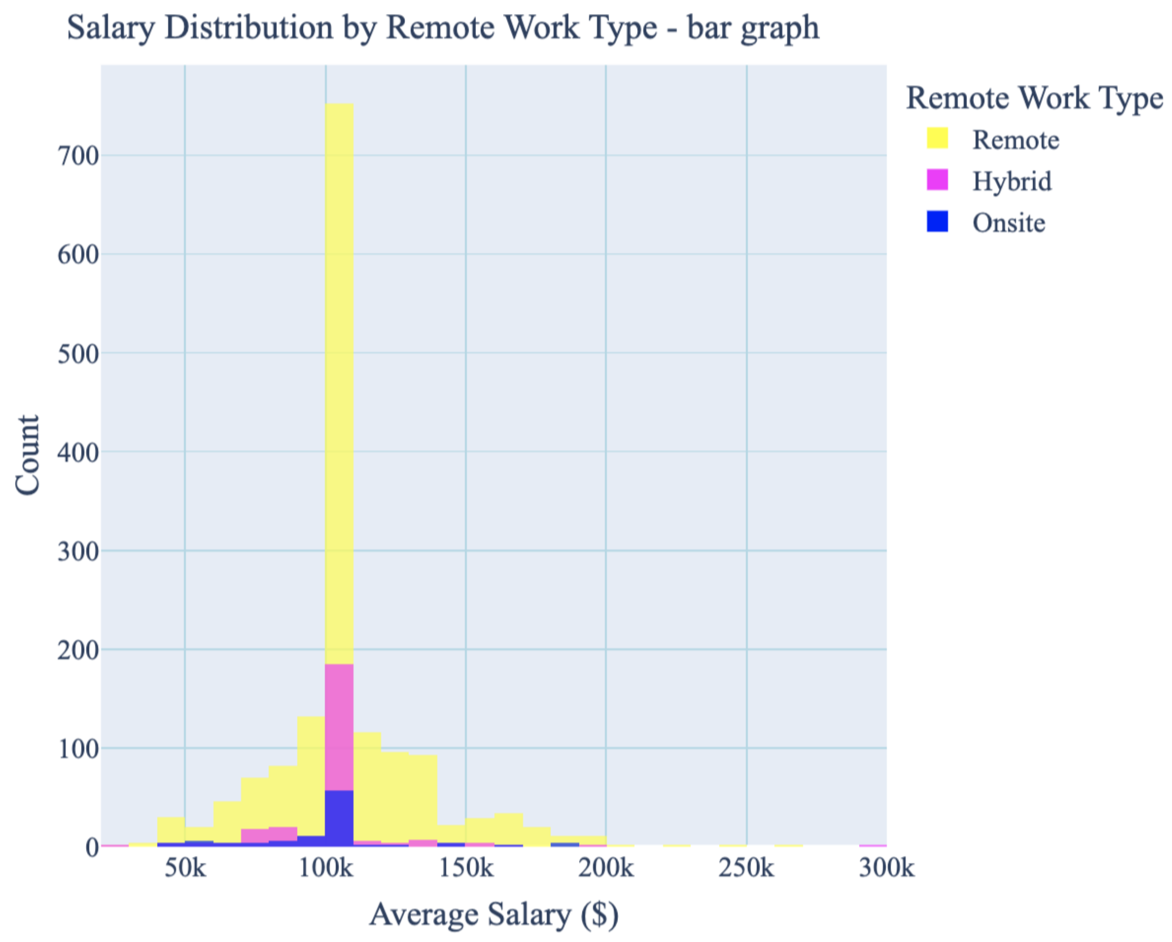


#Analysis: Both Remote and Hybrid roles have good average salaries that are similar and sometimes exceed those of onsite positions, particularly in more mid level and senior roles. This shows that there is no decuction in salary from working remote, and that with more years of expereince, the roles lean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

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

```python
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
```


Salary Distribution by Remote Work Type - bar graph
Count
700
600
500
400
300
200
100
0
50k
100k
150k
200k
250k
300k
Average Salary ($)
Remote Work Type
Remote
Hybrid
Onsite