Assignment 03

Wei Wang

September 20, 2025

1 Load the Dataset

```
from pyspark.sql import SparkSession
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 monotonically_increasing_id

np.random.seed(42)

pio.renderers.default = "vscode+notebook+png"

# Initialize Spark Session
spark = SparkSession.builder.appName("LightcastData").getOrCreate()

# Load Data
df = spark.read.option("header", "true").option("inferSchema", "true").option("multiLine","t

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

2 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 median 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,
  "SALARY": median_salary
})

# Step 4 Computing average salary
df = df.withColumn("Average_Salary",(col("SALARY_FROM")+col("SALARY_TO"))/2)

# Step 5 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 6 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 18:>                                                         (0 + 1) / 1]

Medians: 87295.0 130042.0 115024.0

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

Data cleaning complete. Rows retained: 72498
```

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`
    - Group by `EMPLOYMENT_TYPE_NAME`.
  - Customize colors, fonts, and styles.

3.1 Salary Distribution by Industry

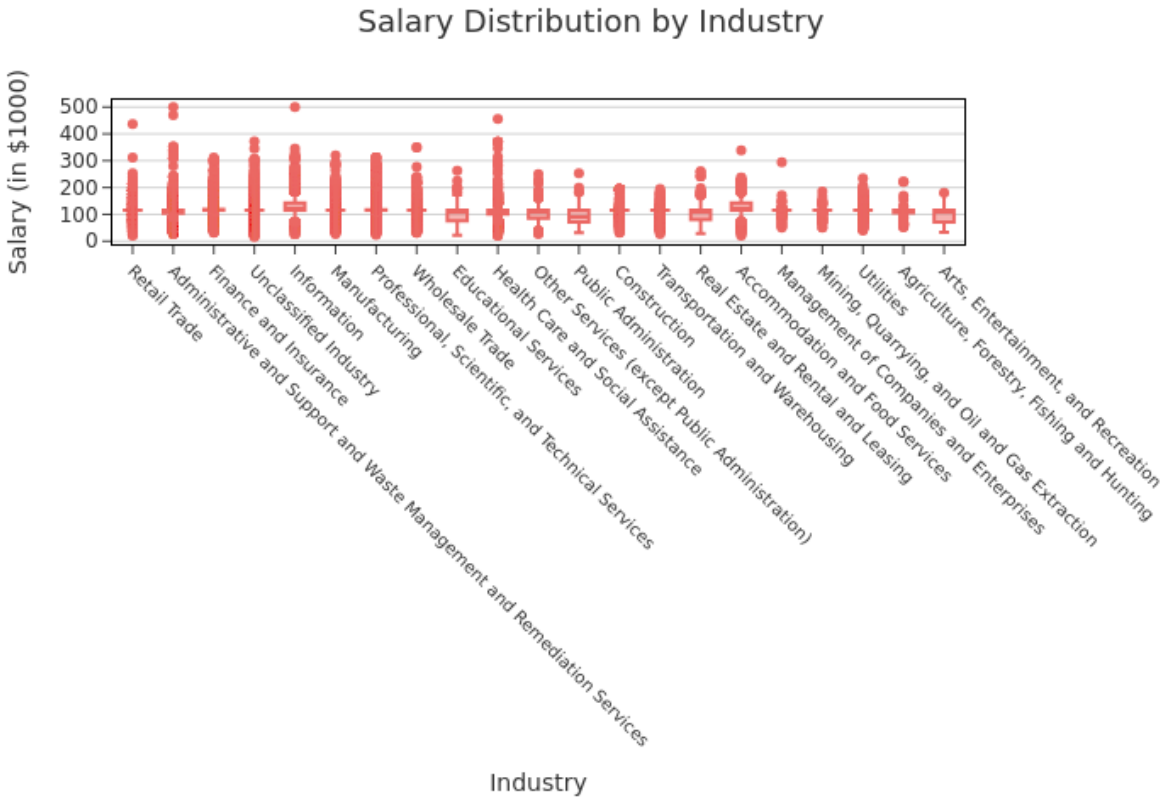
- **Explanation**: This graph reveals that the Information industry offers the highest median salaries (around $140K), while most other industries cluster between $100K and $120K with relatively similar distributions. It also highlights that all industries exhibit substantial salary ranges and numerous outliers, with some positions reaching $400K–$500K, indicating significant intra-industry variation in compensation.

```
# Filtering out missing or zero salary values
pdf = df.filter(df["SALARY"] > 0).select("NAICS2_NAME", "SALARY").toPandas()

# Cleaning industry names
pdf["NAICS2_NAME"] = (
    pdf["NAICS2_NAME"]
    .fillna("")
    .apply(lambda x: re.sub(r"[^\x00-\x7F]+", "", x).strip())
)
pdf = pdf[pdf["NAICS2_NAME"].str.len() > 0]

# Converting salary to $1000 units
pdf["SALARY"] = pdf["SALARY"] / 1000

# Computing median salary by industry for sorting
median_salaries = pdf.groupby("NAICS2_NAME")["SALARY"].median()
sorted_industries = median_salaries.sort_values(ascending=False).index

# Applying sorted categories
pdf["NAICS2_NAME"] = pd.Categorical(
    pdf["NAICS2_NAME"],
    categories=sorted_industries,
    ordered=True
)

# Creating box plot
fig = px.box(
    pdf,
    x="NAICS2_NAME",
    y="SALARY",
    points="outliers",
    title="Salary Distribution by Industry",
    labels={
        "NAICS2_NAME": "Industry",
```

```
        "SALARY": "Salary (in $1000)"
    },
    color_discrete_sequence=["#eb6864"],
    height=600
)

fig.update_layout(
    font=dict(family="Arial", size=12, color="#333333"),
    plot_bgcolor="white",
    paper_bgcolor="white",
    title=dict(x=0.5, xanchor="center", font=dict(size=18)),
    xaxis=dict(
        tickangle=45,
        tickfont=dict(size=10),
        showgrid=False,
        zeroline=False,
        linecolor='black',
        ticks='outside',
        showline=True,
        mirror=True
    ),
    yaxis=dict(
        tick0=0,
        dtick=100,
        showgrid=True,
        gridcolor='lightgray',
        zeroline=False,
        linecolor='black',
        ticks='outside',
        showline=True,
        mirror=True
    ),
    margin=dict(l=60, r=40, t=80, b=200),
    boxmode="group",
    hovermode="x unified"
)

fig.show()
fig.write_image("output/Q2_Industry_BoxPlot.svg", width=3500, height=600, scale=1)
```

```
[Stage 22:>                                                         (0 + 1) / 1]
```

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



3.2 Salary Distribution by Employment Type

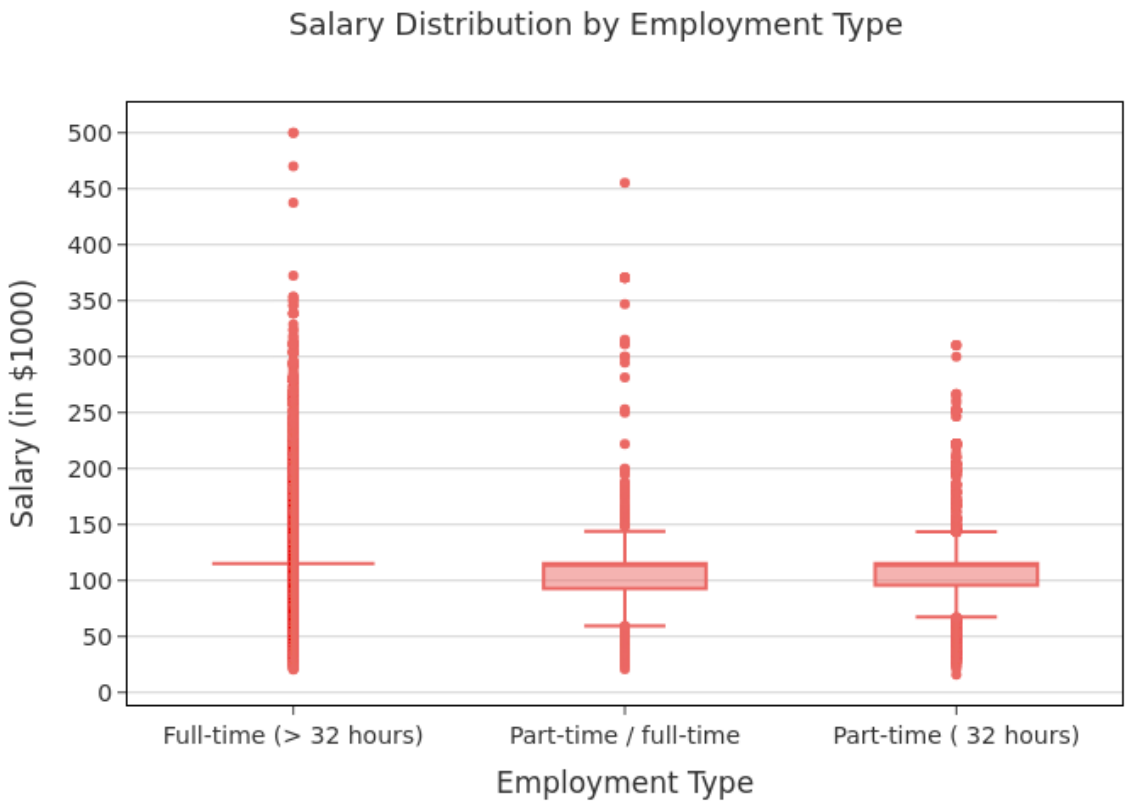
- **Explanation**: The box plot reveals that full-time employees tend to earn higher median salaries compared to part-time and mixed employment types. However, full-time roles also exhibit a wider salary range and more extreme outliers, indicating greater variability in compensation.

```
# Filtering out missing or zero salary values
pdf = df.filter(df["SALARY"]>0).select("EMPLOYMENT_TYPE_NAME","SALARY").toPandas()

# Cleaning employment type names for better readability
pdf["EMPLOYMENT_TYPE_NAME"] = (
    pdf["EMPLOYMENT_TYPE_NAME"]
    .fillna("")
    .apply(lambda x: re.sub(r"[^\x00-\x7F]+", "", x).strip())
```

```
)
pdf = pdf[pdf["EMPLOYMENT_TYPE_NAME"].str.len() > 0]

# Converting salary to $1000 units
pdf["SALARY"] = pdf["SALARY"] / 1000

# Computing media salary for sorting
median_salaries = pdf.groupby("EMPLOYMENT_TYPE_NAME")["SALARY"].median()

# Sorting employment types based on median salary
sorted_employment_types = median_salaries.sort_values(ascending=False).index

# Applying sorted categories
pdf["EMPLOYMENT_TYPE_NAME"] = pd.Categorical(
    pdf["EMPLOYMENT_TYPE_NAME"],
    categories=sorted_employment_types,
    ordered=True
)

# Creating box plot
fig = px.box(
    pdf,
    x="EMPLOYMENT_TYPE_NAME",
    y="SALARY",
    points="outliers",
    title="Salary Distribution by Employment Type",
    labels={
        "EMPLOYMENT_TYPE_NAME": "Employment Type",
        "SALARY": "Salary (in $1000)"
    },
    color_discrete_sequence=["#eb6864"],
    height=500
)

fig.update_layout(
    font=dict(family="Arial", size=14, color="#333333"),
    plot_bgcolor="white",
    paper_bgcolor="white",
    title=dict(x=0.5, xanchor="center", font=dict(size=18)),
    xaxis=dict(
        showgrid=False,
        zeroline=False,
```

```
        linecolor='black',
        ticks='outside',
        showline=True,
        mirror=True
    ),
    yaxis=dict(
        showgrid=True,
        gridcolor='lightgray',
        zeroline=False,
        linecolor='black',
        ticks='outside',
        showline=True,
        mirror=True,
        tick0=0,
        dtick=50
    ),
    margin=dict(l=60, r=40, t=80, b=60),
    boxmode="group",
    hovermode="x unified"
)

fig.show()
fig.write_image("output/Q1_EMPLOYMENT_TYPE_BoxPlot.svg", width=3000, height=500, scale=1)
```

```
[Stage 23:>                                                         (0 + 1) / 1]
```



4 Salary Analysis by ONET Occupation Type (Bubble Chart)

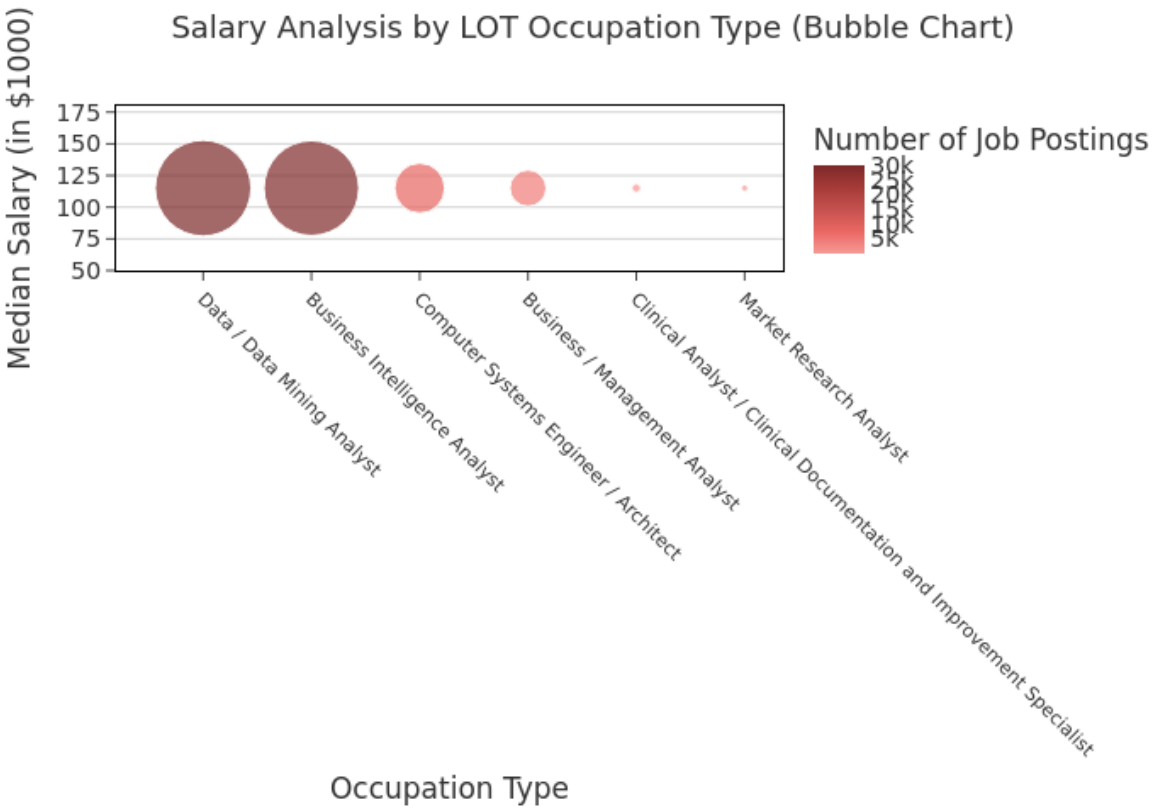
- Analyze how salaries differ across ONET occupation types.
- **Aggregate Data**
  - Compute **median salary** for each occupation in the **ONET taxonomy**.
- **Visualize results**
  - Create a **bubble chart** where:
    - **X-axis** = `ONET_NAME`
    - **Y-axis** = `Median Salary`
    - **Size** = Number of job postings
  - Apply custom colors and font styles.

```
# Spark SQL -  Median salary and job count per LOT_OCCUPATION_NAME
df.createOrReplaceTempView("Job_Postings")
```

```
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
""")

# Converting to Pandas Data Frame
salary_pd = salary_analysis.toPandas()

# Converting salary to $1000 units
salary_pd["Median_Salary"] = salary_pd["Median_Salary"] / 1000

# Creating Bubble Chart
custom_coral_scale = [
    [0.0, "#f79a96"],
    [0.25, "#eb6864"],
    [0.5, "#c6524f"],
    [0.75, "#a63b39"],
    [1.0, "#7c2a29"]
]
fig = px.scatter(
    salary_pd,
    x="Occupation_Name",
    y="Median_Salary",
    size="Job_Postings",
    color="Job_Postings",
    color_continuous_scale=custom_coral_scale,
    title="Salary Analysis by LOT Occupation Type (Bubble Chart)",
    labels={
        "Occupation_Name": "Occupation Type",
        "Median_Salary": "Median Salary (in $1000)",
        "Job_Postings": "Number of Job Postings"
    },
    hover_name="Occupation_Name",
    width=1000,
    height=500,
    size_max=40
```

```
)

fig.update_layout(
    font=dict(family="Arial", size=14, color="#333333"),
    title=dict(x=0.5, xanchor="center", font=dict(size=18)),
    plot_bgcolor="white",
    paper_bgcolor="white",
    xaxis=dict(
        tickangle=45,
        tickfont=dict(size=11),
        showline=True,
        linecolor='black',
        title_standoff=10,
        showgrid=False,
        zeroline=False,
        ticks='outside',
        mirror=True
    ),
    yaxis=dict(
        tick0=0,
        dtick=25,
        range=[50, 180],
        showline=True,
        linecolor='black',
        title_standoff=10,
        showgrid=True,
        gridcolor='lightgray',
        zeroline=False,
        ticks='outside',
        mirror=True
    ),
    coloraxis_colorbar=dict(
        tickvals=[5000, 10000, 15000, 20000, 25000, 30000],
        ticktext=["5k", "10k", "15k", "20k", "25k", "30k"],
        title="Number of Job Postings"
    ),
    margin=dict(l=60, r=40, t=80, b=140),
    hovermode="closest"
)

fig.show()
fig.write_image("output/Q3_BubbleChart_V1.svg", width=1500, height=600, scale=1)
```

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



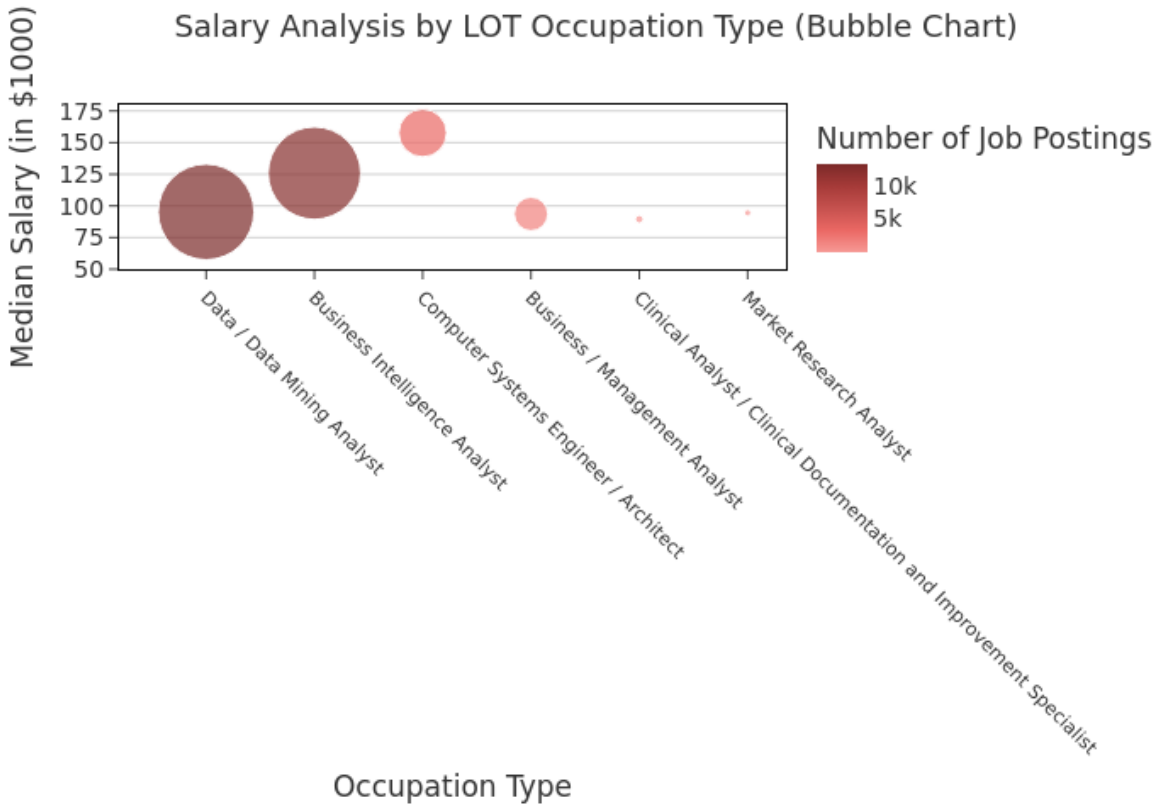
- **Explanation:** The bubble chart reveals that while job posting volumes vary widely across LOT occupation types, the median salaries appear uniformly clustered. This pattern is likely due to an earlier step where missing salary values were imputed with the overall median, flattening natural variation. Therefore, we take a further step to filter out the most common imputed value to improve accuracy.

```
# Spark SQL -  Median salary and job count per LOT_OCCUPATION_NAME
df.createOrReplaceTempView("Job_Postings")
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
    WHERE SALARY IS NOT NULL
        AND SALARY > 0
```

```
        AND SALARY NOT IN (115024.0)
        AND LOT_OCCUPATION_NAME IS NOT NULL
    GROUP BY LOT_OCCUPATION_NAME
    ORDER BY Job_Postings DESC
    LIMIT 10
""")

# Converting to Pandas Data Frame
salary_pd = salary_analysis.toPandas()

# Converting salary to $1000 units
salary_pd["Median_Salary"] = salary_pd["Median_Salary"] / 1000

# Creating Bubble Chart
custom_coral_scale = [
    [0.0, "#f79a96"],
    [0.25, "#eb6864"],
    [0.5, "#c6524f"],
    [0.75, "#a63b39"],
    [1.0, "#7c2a29"]
]
fig = px.scatter(
    salary_pd,
    x="Occupation_Name",
    y="Median_Salary",
    size="Job_Postings",
    color="Job_Postings",
    color_continuous_scale=custom_coral_scale,
    title="Salary Analysis by LOT Occupation Type (Bubble Chart)",
    labels={
        "Occupation_Name": "Occupation Type",
        "Median_Salary": "Median Salary (in $1000)",
        "Job_Postings": "Number of Job Postings"
    },
    hover_name="Occupation_Name",
    width=1000,
    height=500,
    size_max=40
)

fig.update_layout(
    font=dict(family="Arial", size=14, color="#333333"),
```

```
    title=dict(x=0.5, xanchor="center", font=dict(size=18)),
    plot_bgcolor="white",
    paper_bgcolor="white",
    xaxis=dict(
        tickangle=45,
        tickfont=dict(size=11),
        showline=True,
        linecolor='black',
        title_standoff=10,
        showgrid=False,
        zeroline=False,
        ticks='outside',
        mirror=True
    ),
    yaxis=dict(
        tick0=0,
        dtick=25,
        range=[50, 180],
        showline=True,
        linecolor='black',
        title_standoff=10,
        showgrid=True,
        gridcolor='lightgray',
        zeroline=False,
        ticks='outside',
        mirror=True
    ),
    coloraxis_colorbar=dict(
        tickvals=[5000, 10000, 15000, 20000, 25000, 30000],
        ticktext=["5k", "10k", "15k", "20k", "25k", "30k"],
        title="Number of Job Postings"
    ),
    margin=dict(l=60, r=40, t=80, b=140),
    hovermode="closest"
)

fig.show()
fig.write_image("output/Q3_BubbleChart_V2.svg", width=1500, height=600, scale=1)
```

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



- **Explanation:** The bubble chart shows that job posting volumes vary significantly across LOT occupation types, with roles like Data/Data Mining Analyst and Business Intelligence Analyst dominating in demand. Median salaries, however, differ more clearly after filtering out previously imputed values,revealing that roles in system architecture and engineering tend to command higher median pay.

5 Salary by Education Level

- Create four 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`
- **Short explanation** of key insights for each graph:

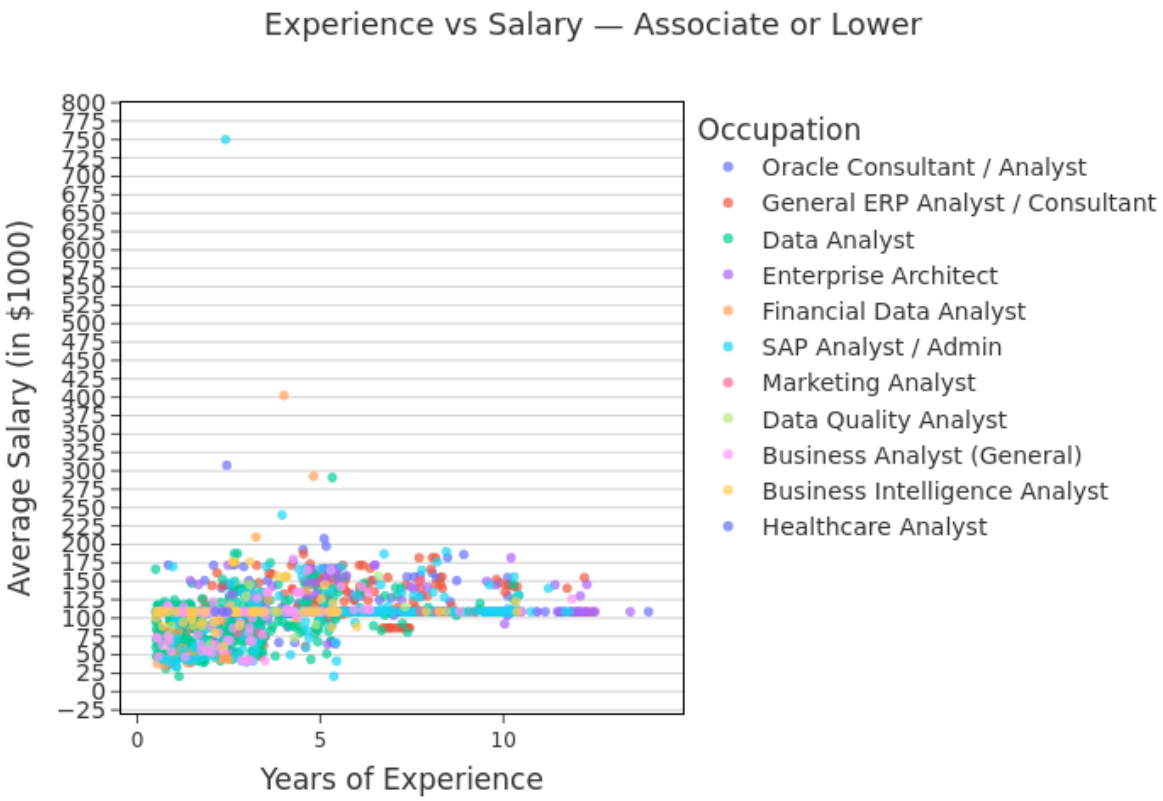
- **Associate or Lower**: Most salaries cluster between $50K–$150K. A few outliers exceed $300K–$700K, but they are extremely rare and likely exceptional cases. Increasing years of experience does not significantly improve salary for this group, indicating a potential ceiling for career growth without further education.
- **Bachelor's**: There is a noticeable upward trend—salaries increase with experience more consistently than the Associate group. Most salaries fall within $70K–$150K, with the median visibly higher than the Associate group. Broader range of occupations are represented, potentially offering better career mobility.
- **Master's**: Few roles pay below $100K, indicating a stronger starting point compared to the lower education groups. Despite the smaller sample size, salaries for experienced professionals (5+ years) often exceed $150K. More linear progression, suggesting Master's degree offers good return on investment for salary advancement.
- **PhD**: Limited sample, but high value. Even with low years of experience, this group commands higher salaries—likely due to specialized roles or niche expertise. This suggests deep specialization can offset lack of experience when entering the job market.

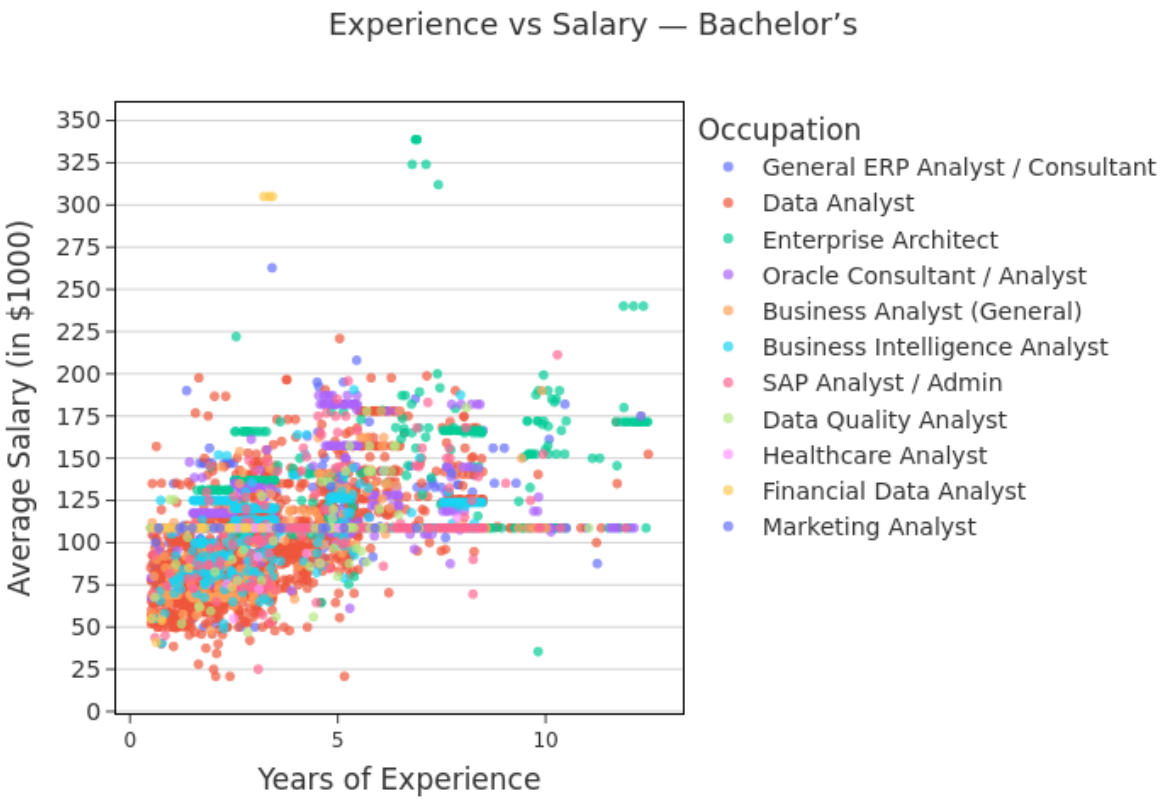
```
from pyspark.sql.functions import col, when, rand

# Defining education level groupings
associate_or_lower = ["GED", "Associate", "No Education Listed", "High school"]
bachelor = ["Bachelor"]
master = ["Master"]
phd = ["Ph.D", "Doctorate", "professional degree"]

# Adding EDU_GROUP column
df = df.withColumn(
    "EDU_GROUP",
    when(col("EDUCATION_LEVELS_NAME").rlike("|".join([f"(?i){deg}" for deg in associate_or_lo
    .when(col("EDUCATION_LEVELS_NAME").rlike("|".join([f"(?i){deg}" for deg in bachelor])), "
    .when(col("EDUCATION_LEVELS_NAME").rlike("|".join([f"(?i){deg}" for deg in master])), "Ma
    .when(col("EDUCATION_LEVELS_NAME").rlike("|".join([f"(?i){deg}" for deg in phd])), "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"))

# Adding jitter to avoid overlapping dots in scatter plot
df = df.withColumn("Jittered_Experience", col("MAX_YEARS_EXPERIENCE") + (rand() - 0.5))

# Filtering for non-null and positive values
```

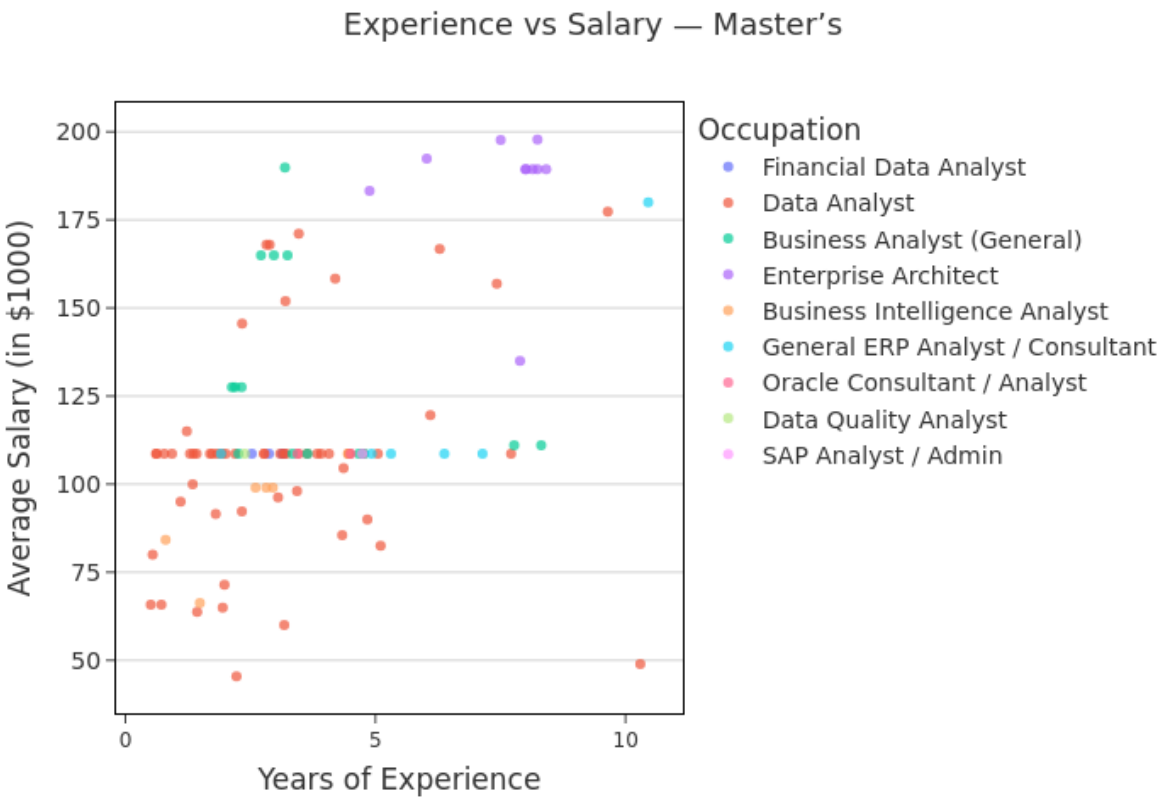
```
df = df.filter(
    col("MAX_YEARS_EXPERIENCE").isNotNull() &
    col("Average_Salary").isNotNull() &
    (col("MAX_YEARS_EXPERIENCE") > 0) &
    (col("Average_Salary") > 0)
)

# Keeping only four major groups
df_filtered = df.filter(
    col("EDU_GROUP").isin("Associate or Lower", "Bachelor's", "Master's", "PhD")
)

# Converting to Pandas for plotting
df_pd = df_filtered.select(
    "EDU_GROUP",
    "Jittered_Experience",
    "Average_Salary",
    "LOT_V6_SPECIALIZED_OCCUPATION_NAME"
).toPandas()

# Creating scatter plots for each group

edu_groups = ["Associate or Lower", "Bachelor's", "Master's", "PhD"]

for group in edu_groups:
    subset = df_pd[df_pd["EDU_GROUP"] == group].copy()

    subset["Salary_K"] = subset["Average_Salary"] / 1000

    fig = px.scatter(
        subset,
        x="Jittered_Experience",
        y="Salary_K",
        color="LOT_V6_SPECIALIZED_OCCUPATION_NAME",
        title=f"Experience vs Salary - {group}",
        labels={
            "Jittered_Experience": "Years of Experience",
            "Salary_K": "Average Salary (in $1000)",
            "LOT_V6_SPECIALIZED_OCCUPATION_NAME": "Occupation"
        },
        opacity=0.7,
        width=1000,
```

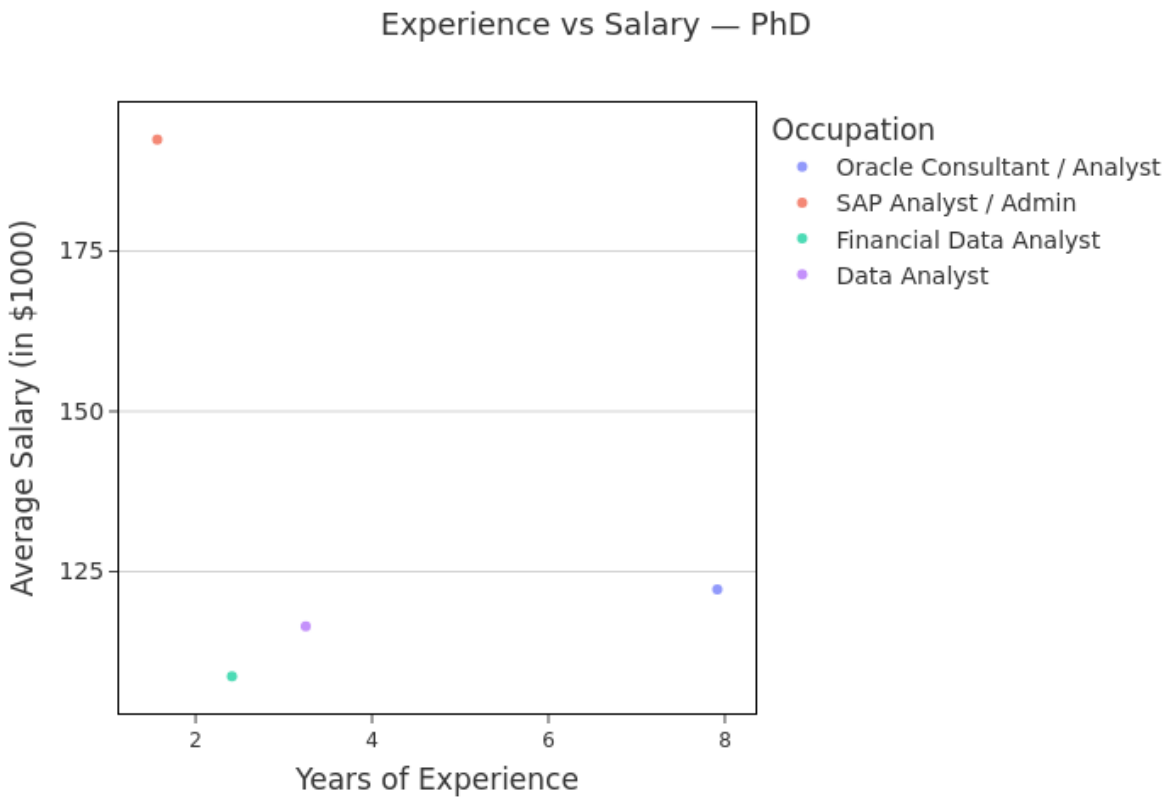
```
        height=600
    )

    fig.update_layout(
        font=dict(family="Arial", size=14, color="#333333"),
        title=dict(x=0.5, xanchor="center", font=dict(size=18)),
        plot_bgcolor="white",
        paper_bgcolor="white",
        xaxis=dict(
            tickangle=0,
            tickfont=dict(size=12),
            showline=True,
            linecolor='black',
            title_standoff=10,
            showgrid=False,
            zeroline=False,
            ticks='outside',
            mirror=True
        ),
        yaxis=dict(
            title="Average Salary (in $1000)",
            tick0=0,
            dtick=25,
            showline=True,
            linecolor='black',
            title_standoff=10,
            showgrid=True,
            gridcolor='lightgray',
            zeroline=False,
            ticks='outside',
            mirror=True
        ),
        margin=dict(l=60, r=40, t=80, b=60),
        legend_title_text="Occupation",
        hovermode="closest"
    )

    fig.show()
    safe_group = group.replace("'", "").replace(" ", "_")
    fig.write_image(f"output/Q4_experience_salary_{safe_group}.svg", width=1500, height=600,
```

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


Experience vs Salary — Associate or Lower
Average Salary (in $1000)
Years of Experience
Occupation
Oracle Consultant / Analyst
General ERP Analyst / Consultant
Data Analyst
Enterprise Architect
Financial Data Analyst
SAP Analyst / Admin
Marketing Analyst
Data Quality Analyst
Business Analyst (General)
Business Intelligence Analyst
Healthcare Analyst


Experience vs Salary — Bachelor's
Average Salary (in $1000)
350
325
300
275
250
225
200
175
150
125
100
75
50
25
0
0
5
10
Years of Experience
Occupation
General ERP Analyst / Consultant
Data Analyst
Enterprise Architect
Oracle Consultant / Analyst
Business Analyst (General)
Business Intelligence Analyst
SAP Analyst / Admin
Data Quality Analyst
Healthcare Analyst
Financial Data Analyst
Marketing Analyst


Experience vs Salary — Master's
Average Salary (in $1000)
200
175
150
125
100
75
50
0
5
10
Years of Experience
Occupation
Financial Data Analyst
Data Analyst
Business Analyst (General)
Enterprise Architect
Business Intelligence Analyst
General ERP Analyst / Consultant
Oracle Consultant / Analyst
Data Quality Analyst
SAP Analyst / Admin



6 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.**

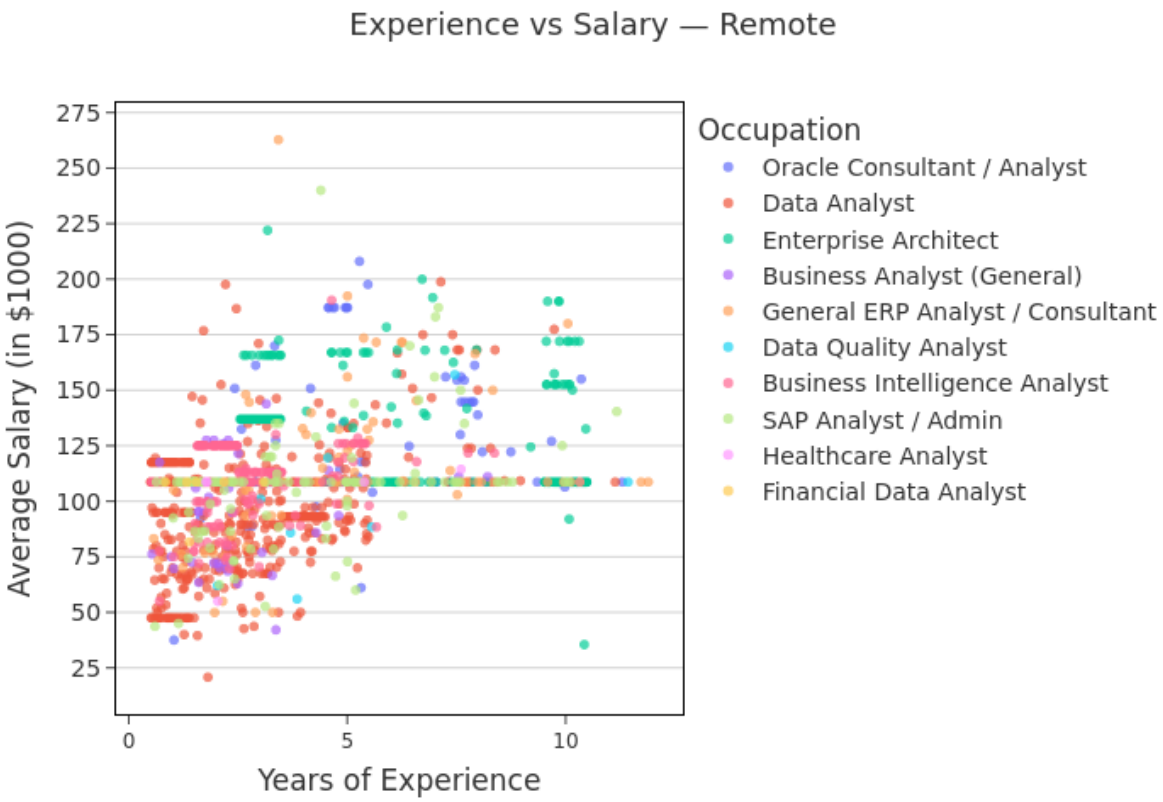
6.1 Scatter Plots by Remote Group

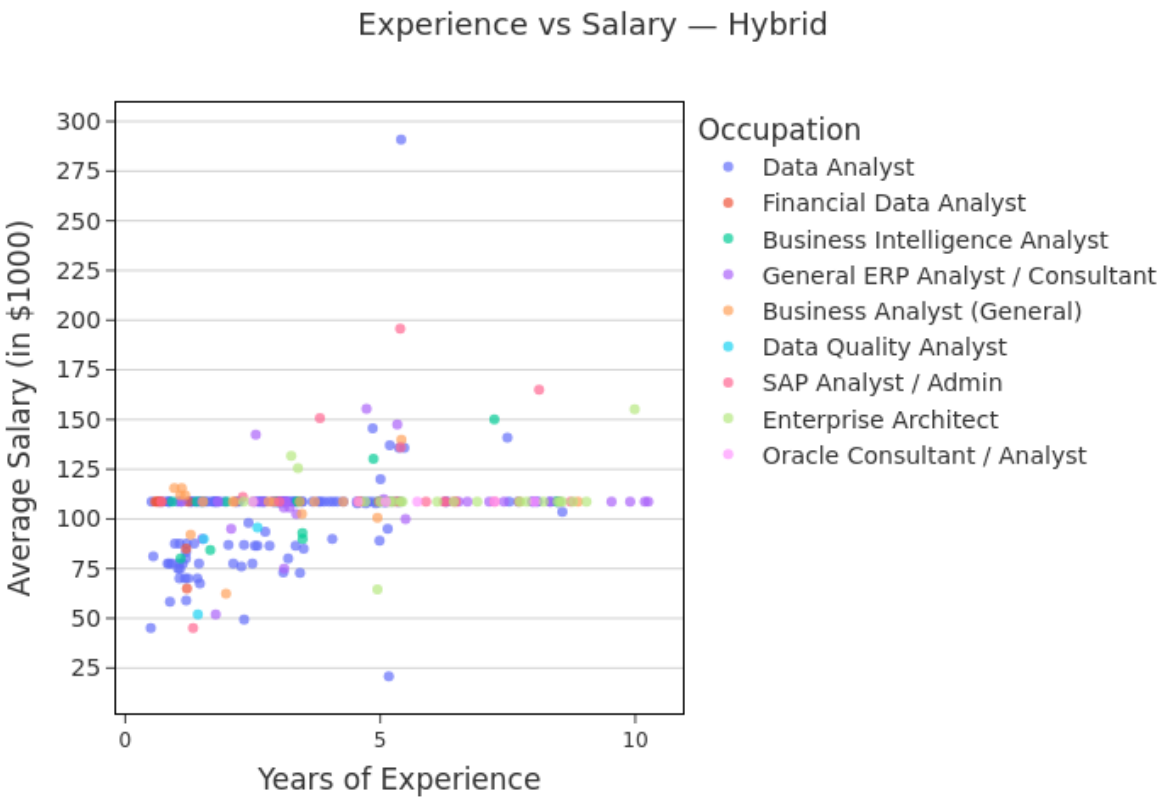
```
from pyspark.sql.functions import col, when, rand

# Creating a new column REMOTE_GROUP
df = df.withColumn(
    "REMOTE_GROUP",
    when(col("REMOTE_TYPE_NAME") == "Remote", "Remote")
    .when(col("REMOTE_TYPE_NAME") == "Hybrid Remote", "Hybrid")
    .otherwise("Onsite")
)

# Adding jitter to avoid overlapping dots
df = df.withColumn("Jittered_Experience", col("MAX_YEARS_EXPERIENCE") + (rand() - 0.5))

# Filtering for clean values
df_filtered = df.filter(
    col("MAX_YEARS_EXPERIENCE").isNotNull() &
    col("Average_Salary").isNotNull() &
    (col("MAX_YEARS_EXPERIENCE") > 0) &
    (col("Average_Salary") > 0)
)

# Converting to Pandas for Plotly
df_pd = df_filtered.select(
    "REMOTE_GROUP",
    "Jittered_Experience",
    "Average_Salary",
    "LOT_V6_SPECIALIZED_OCCUPATION_NAME"
).toPandas()

# Creating scatter plots for each group
remote_groups = ["Remote", "Hybrid", "Onsite"]

for group in remote_groups:
    subset = df_pd[df_pd["REMOTE_GROUP"] == group].copy()

    if subset.shape[0] < 10:
        print(f"Skipping plot for {group} (not enough data)")
        continue

    subset["Salary_K"] = subset["Average_Salary"] / 1000
```

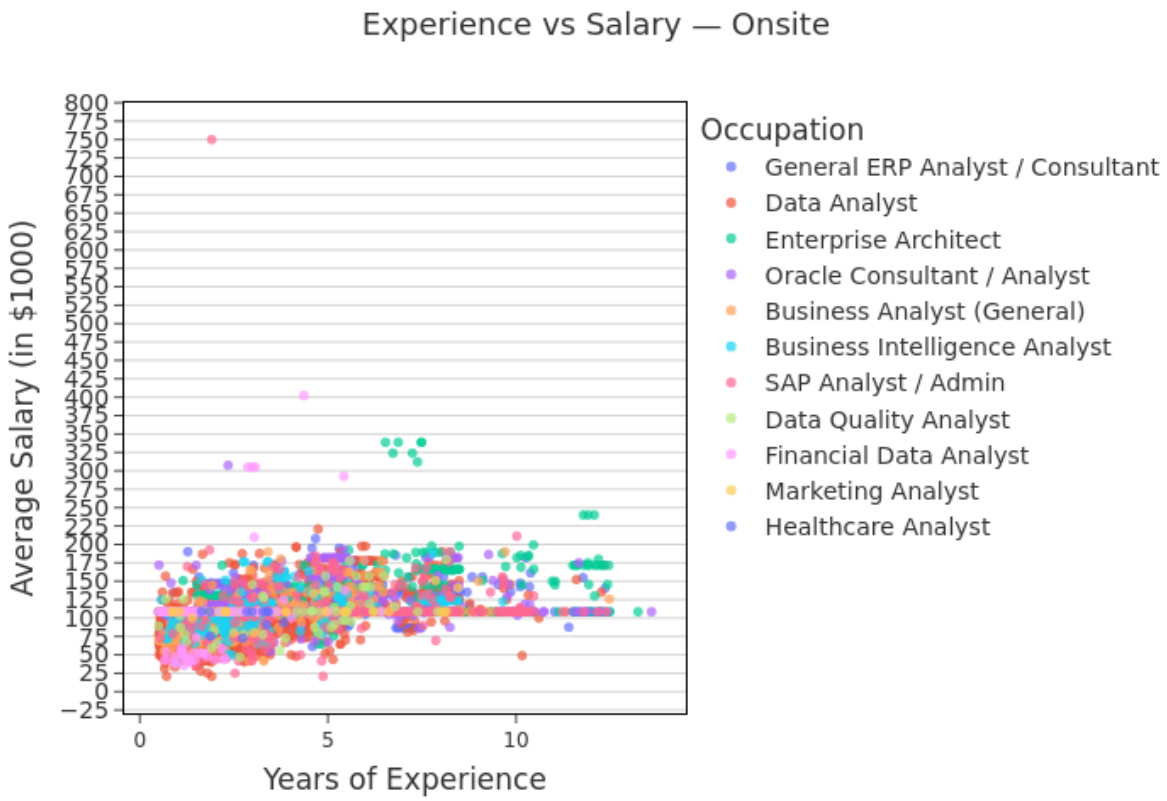
```
    fig = px.scatter(
        subset,
        x="Jittered_Experience",
        y="Salary_K",
        color="LOT_V6_SPECIALIZED_OCCUPATION_NAME",
        title=f"Experience vs Salary - {group}",
        labels={
            "Jittered_Experience": "Years of Experience",
            "Salary_K": "Average Salary (in $1000)",
            "LOT_V6_SPECIALIZED_OCCUPATION_NAME": "Occupation"
        },
        opacity=0.7,
        width=1000,
        height=600
    )

    fig.update_layout(
        font=dict(family="Arial", size=14, color="#333333"),
        title=dict(x=0.5, xanchor="center", font=dict(size=18)),
        plot_bgcolor="white",
        paper_bgcolor="white",
        xaxis=dict(
            tickangle=0,
            tickfont=dict(size=12),
            showline=True,
            linecolor='black',
            title_standoff=10,
            showgrid=False,
            zeroline=False,
            ticks='outside',
            mirror=True
        ),
        yaxis=dict(
            title="Average Salary (in $1000)",
            tick0=0,
            dtick=25,
            showline=True,
            linecolor='black',
            title_standoff=10,
            showgrid=True,
            gridcolor='lightgray',
            zeroline=False,
```

```
            ticks='outside',
            mirror=True
        ),
        margin=dict(l=60, r=40, t=80, b=60),
        legend_title_text="Occupation",
        hovermode="closest"
    )

    fig.show()
    safe_name = group.replace(" ", "_").lower()
    fig.write_image(f"output/Q5_remote_salary_{safe_name}.svg", width=1200, height=600)
```

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




Experience vs Salary — Hybrid
Average Salary (in $1000)
300
275
250
225
200
175
150
125
100
75
50
25
0
5
10
Years of Experience
Occupation
Data Analyst
Financial Data Analyst
Business Intelligence Analyst
General ERP Analyst / Consultant
Business Analyst (General)
Data Quality Analyst
SAP Analyst / Admin
Enterprise Architect
Oracle Consultant / Analyst



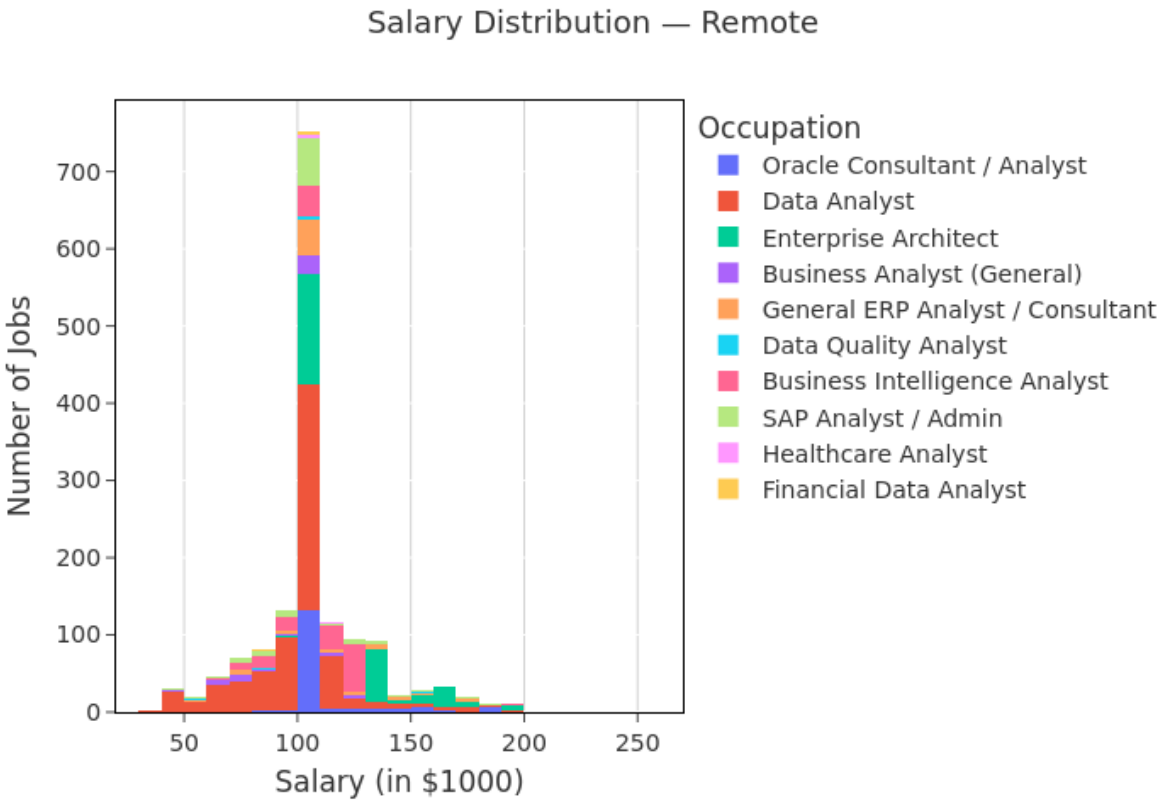
- **Explanation:**
  - **Onsite** roles show the widest salary range, with outliers >$700K, especially among senior roles like Enterprise Architects.
  - **Remote** roles have a more moderate salary distribution, mostly under $250K.
  - **Hybrid** roles show the narrowest salary range, mostly $50K–$200K.
  - **All three show a positive trend**: more experience generally leads to higher salary.
  - **Enterprise Architects** tends to earn the highest salaries, especially in onsite roles.

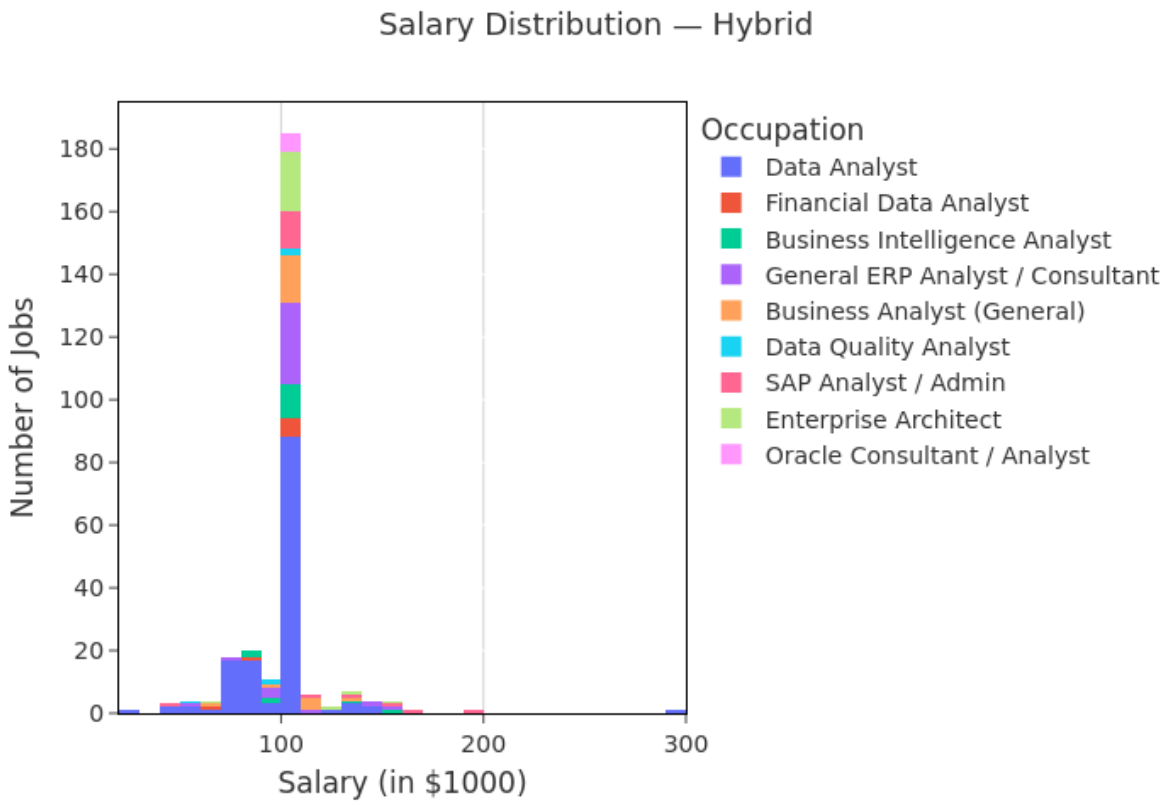
6.2 Histograms of Salary by Remote Group

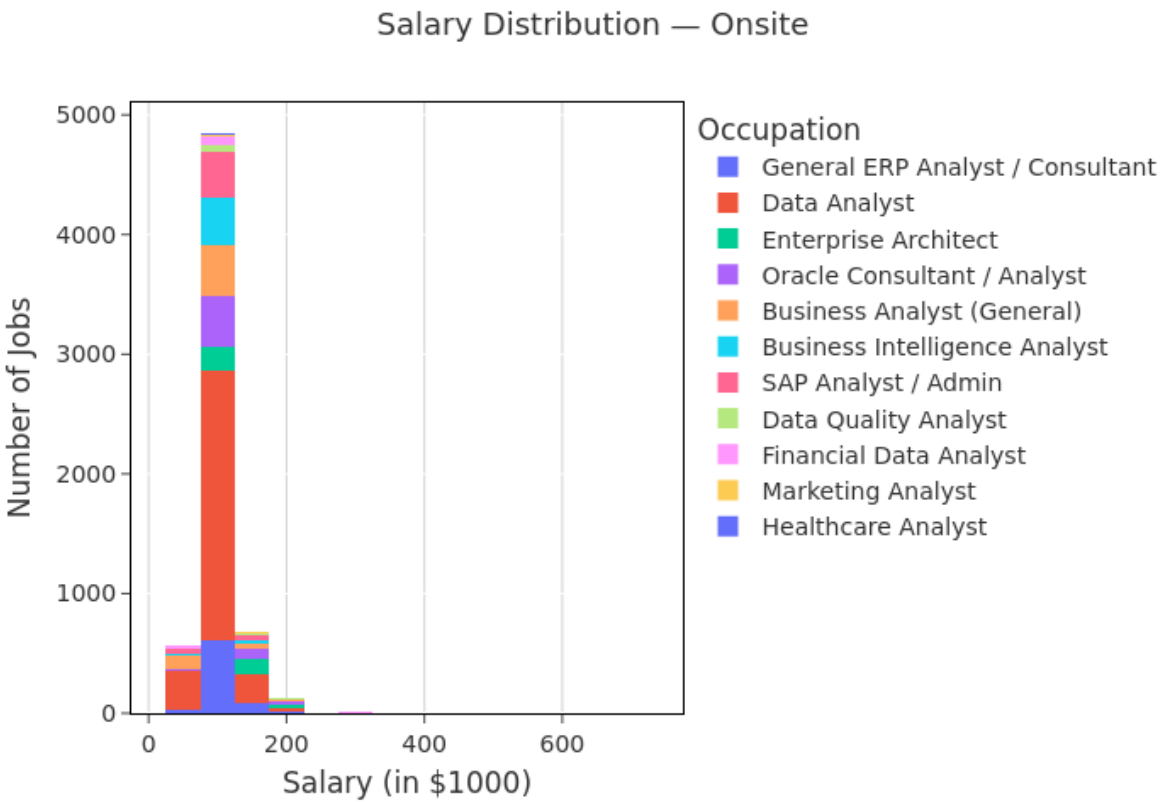
```
# Converting salary to $1000 units
df_pd["Salary_K"] = df_pd["Average_Salary"] / 1000

# Filtering out rows with missing salary or REMOTE_GROUP
filtered_df = df_pd.dropna(subset=["Average_Salary", "REMOTE_GROUP"])
```

```
# Ploting histogram for each remote group
for remote_type in ["Remote", "Hybrid", "Onsite"]:
    subset = filtered_df[filtered_df["REMOTE_GROUP"] == remote_type]

    fig = px.histogram(
        subset,
        x="Salary_K",
        color="LOT_V6_SPECIALIZED_OCCUPATION_NAME",
        nbins=30,
        title=f"Salary Distribution - {remote_type}",
        labels={
            "Salary_K": "Average Salary (in $1000)",
            "LOT_V6_SPECIALIZED_OCCUPATION_NAME": "Occupation"
        },
        barmode="stack",
        width=1000,
        height=500
    )

    fig.update_layout(
        font=dict(family="Arial", size=14, color="#333333"),
        title=dict(x=0.5, xanchor="center", font=dict(size=18)),
        xaxis=dict(
            title="Salary (in $1000)",
            tickformat=",d",
            showline=True,
            linecolor='black',
            title_standoff=10,
            showgrid=True,
            gridcolor='lightgray',
            zeroline=False,
            ticks='outside',
            mirror=True
        ),
        yaxis=dict(
            title="Number of Jobs",
            showline=True,
            linecolor='black',
            title_standoff=10,
            ticks='outside',
            mirror=True
        ),
```

```
        plot_bgcolor="white",
        paper_bgcolor="white",
        legend_title="Occupation",
        margin=dict(l=60, r=40, t=80, b=60)
    )

    fig.show()
    fig.write_image(f"output/Q5_salary_histogram_{remote_type.lower()}.svg", width=1500, hei
```




Salary Distribution — Hybrid
Number of Jobs
0
20
40
60
80
100
120
140
160
180
100
200
300
Salary (in $1000)
Occupation
Data Analyst
Financial Data Analyst
Business Intelligence Analyst
General ERP Analyst / Consultant
Business Analyst (General)
Data Quality Analyst
SAP Analyst / Admin
Enterprise Architect
Oracle Consultant / Analyst



- **Explanation:**
  - **Remote** roles offer consistency in pay but fewer outliers or variation.
    - **Onsite** roles are much more numerous and include extreme high-salary cases.
    - **Hybrid** roles are the least common but demonstrate a more balanced mix.
    - **Data Analyst** is the most common role across all remote types, underscoring its cross-location demand.