# Assignment 03

Bhargavi Manyala

September 24, 2025

## 1 Import 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(42)
#pio.renderers.default = "vscode+notebook+svg"

spark = SparkSession.builder.appName("LightcastData").getOrCreate()
df = (
    spark.read
    .option("header", "true")
    .option("inferSchema", "true")
    .option("multiLine", "true")
    .option("escape", "\"")
    .csv("lightcast_job_postings.csv")
)
```

```
df.createOrReplaceTempView("job_postings")
#df.show(5)
```

```
[Stage 12:>                                                         (0 + 1) / 1]
```

# 2 Casting salary and experience columns

## 2.1 Computing medians and Imputing missing salaries

```
from pyspark.sql.functions import col

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"))

# Computing medians for salary columns
def compute_median(sdf, col_name):
    return sdf.approxQuantile(col_name, [0.5], 0.01)[0]

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)

# Imputing missing salaries, but not experience
df = df.fillna({
    "SALARY_FROM": median_from,
    "SALARY_TO": median_to
})

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

```
)

# Selecting required columns
export_cols = [
    "Average_Salary",
    "SALARY",
    "EDUCATION_LEVELS_NAME",
    "REMOTE_TYPE_NAME",
    "MAX_YEARS_EXPERIENCE",
    "LOT_V6_SPECIALIZED_OCCUPATION_NAME"
]

df_selected = df.select(*export_cols)

pdf_selected = df_selected.toPandas()
pdf_selected.head()
```

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

```
Medians: 87295.0 130042.0 115024.0
```

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

| | Average_Salary | SALARY | EDUCATION_LEVELS_NAME | REMOTE_TYPE_NAME | MAX_YEARS_EXPERIENCE | LOT_V6_SPECIALIZ |
|---|---|---|---|---|---|---|
| 0 | 108668.5 | NaN | [\n "Bachelor's degree"\n] | [None] | 2.0 | General ERP Analyst / |
| 1 | 108668.5 | NaN | [\n "No Education Listed"\n] | Remote | 3.0 | Oracle Consultant / An |
| 2 | 108668.5 | NaN | [\n "Bachelor's degree"\n] | [None] | NaN | Data Analyst |
| 3 | 108668.5 | NaN | [\n "No Education Listed"\n] | [None] | NaN | Data Analyst |
| 4 | 92500.0 | 92500.0 | [\n "No Education Listed"\n] | [None] | NaN | Oracle Consultant / An |

## 2.2 Cleaning Education column and Exporting Cleaned Data

*I referred to Claude Sonnet 4 for prompts and sample code ideas, but I wrote and adapted the final implementation myself

```
# To remove \n and \r
pdf_selected["EDUCATION_LEVELS_NAME"] = (
    pdf_selected["EDUCATION_LEVELS_NAME"]
    .astype(str)
    .str.replace(r"[\n\r]", "", regex=True)
    .str.strip()
)
pdf_selected.to_csv("data/lightcast_cleaned.csv", index=False)
pdf_selected.head()
```

| | Average_Salary | SALARY | EDUCATION_LEVELS_NAME | REMOTE_TYPE_NAME | MAX_YEARS_EXPERIENCE | LOT_V6_SPECIALIZ |
|---|---|---|---|---|---|---|
| 0 | 108668.5 | NaN | [ "Bachelor's degree"] | [None] | 2.0 | General ERP Analyst / |
| 1 | 108668.5 | NaN | [ "No Education Listed"] | Remote | 3.0 | Oracle Consultant / An |
| 2 | 108668.5 | NaN | [ "Bachelor's degree"] | [None] | NaN | Data Analyst |
| 3 | 108668.5 | NaN | [ "No Education Listed"] | [None] | NaN | Data Analyst |
| 4 | 92500.0 | 92500.0 | [ "No Education Listed"] | [None] | NaN | Oracle Consultant / An |

### 2.3 Exporting Cleaned Data

```
print("Data cleaning complete. Rows retained:", len(pdf_selected))
```

```
Data cleaning complete. Rows retained: 72498
```

## 3 Salary Distribution by Industry and Employment Type

### 3.1 Salary Distribution by Employment Type

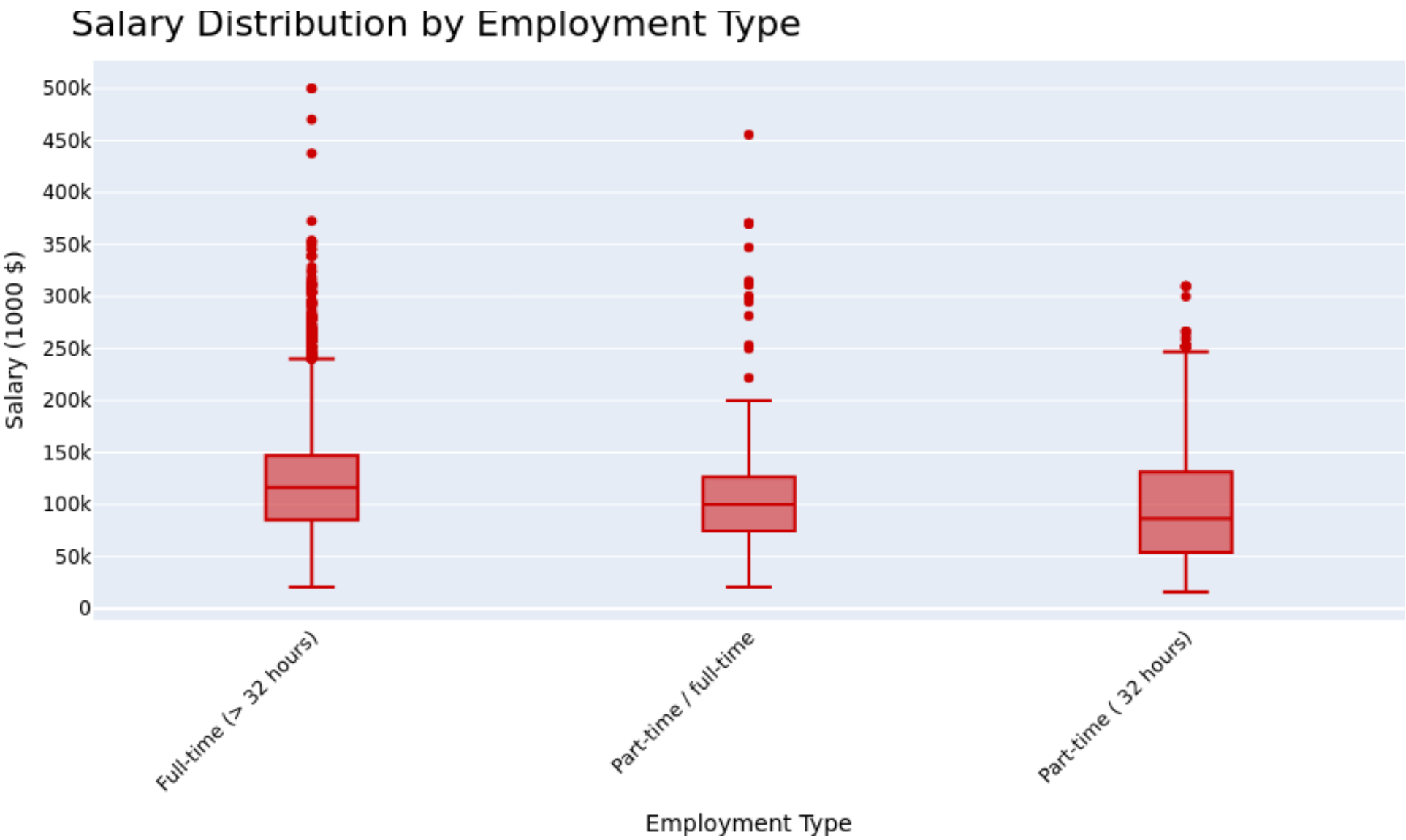
```
pdf = df.select(
    "EMPLOYMENT_TYPE_NAME",
    "NAICS2_NAME",
    "SALARY"
).toPandas()
```

```
pdf = pdf[pdf["SALARY"] > 0]

pdf["EMPLOYMENT_TYPE_NAME"] = (
    pdf["EMPLOYMENT_TYPE_NAME"]
    .astype(str)
    .apply(lambda x: re.sub(r"[^\x00-\x7F]+", "", x))
    .str.strip()
)

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
)
```

```
[Stage 17:>                                                 (0 + 1) / 1]
```

```
fig = px.box(
    pdf,
    x="EMPLOYMENT_TYPE_NAME",
    y="SALARY",
    title="Salary Distribution by Employment Type",
    color_discrete_sequence=["#CC0000"],
    boxmode="group",
    points="outliers"
)


fig.update_layout(
    xaxis=dict(
        title=dict(
            text="Employment Type",
            font=dict(size=14, family="Arial Black", color="black")
        ),
```

```
        tickangle=-45,
        tickfont=dict(size=12, family="Arial Black", color="black"),
        categoryorder="array",
        categoryarray=sorted_employment_types.tolist()
    ),
    yaxis=dict(
        title=dict(
            text="Salary (1000 $)",
            font=dict(size=14, family="Arial Black", color="black")
        ),
        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=12, family="Arial Black", color="black"),
    ),
    font=dict(family="Arial", size=16, color="black"),
    boxgap=0.7,
    showlegend=False,
    height=500,
    width=850,
)




fig.write_html("./output/Q1.html")
fig.write_image("./output/Q1.png", width = 850, height = 500, scale=1)
```



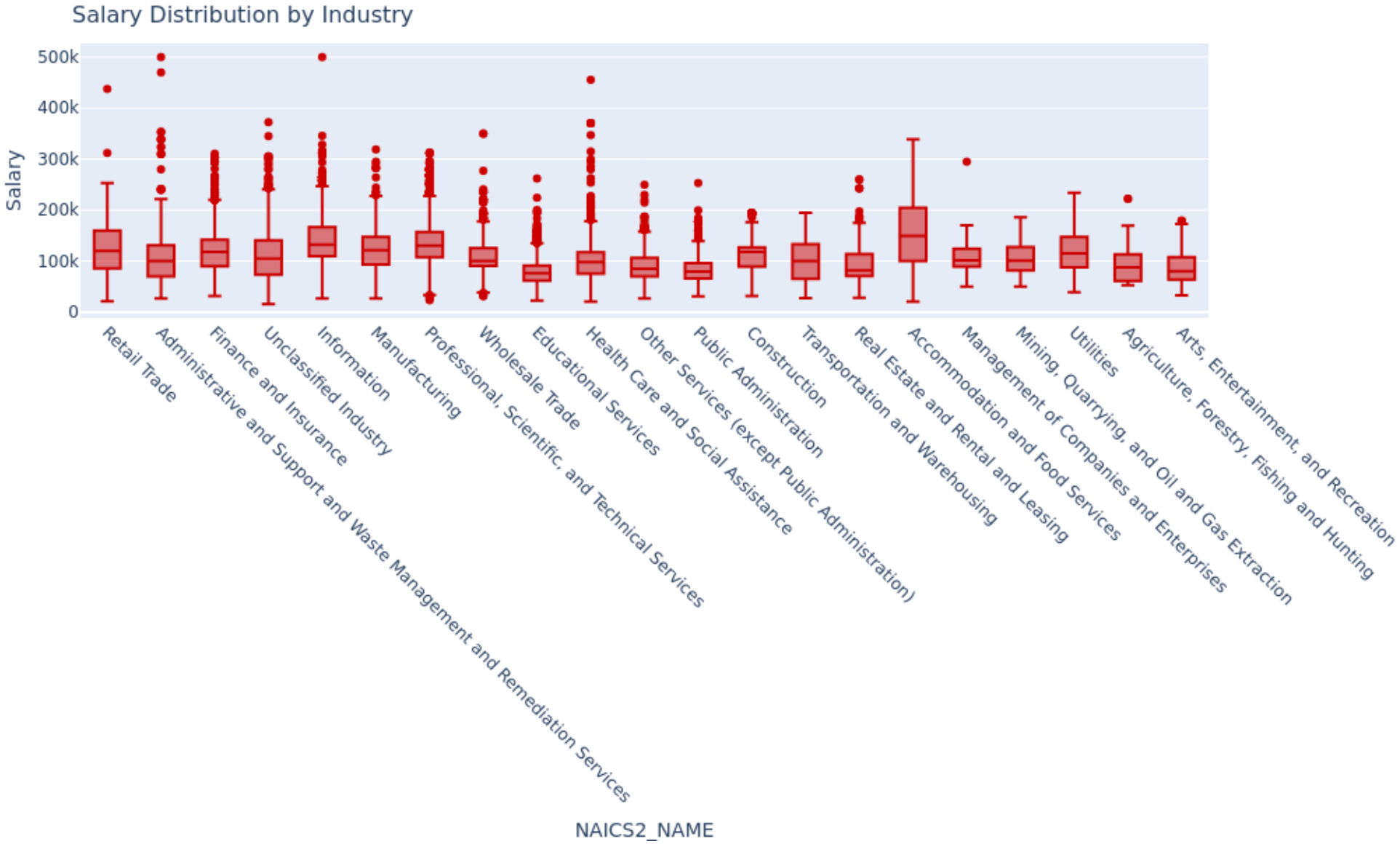

### 3.1.1 Interpretation

- The boxplot indicates that full-time employees (>32 hours) receive higher median salaries with greater variability.
- In contrast, part-time employees indicates lower and more consistent salary ranges, with occasional high outliers

### 3.2 Salary Distribution by Industry

```
pdf = df.select(
    "NAICS2_NAME",
    "SALARY"
).toPandas()



fig = px.box(
    pdf,
    x="NAICS2_NAME",
    y="SALARY",
    title="Salary Distribution by Industry",
    color_discrete_sequence=["#CC0000"],
    points="outliers"
)


fig.update_layout(
    height=500,
    font_family="Arial",
    title_font_size=16,
    xaxis_title="NAICS2_NAME",
    yaxis_title="Salary",
    xaxis_tickangle=45,

)


fig.write_html("./output/Q2.html")
fig.write_image("./output/Q2.png", width=1000, height=600, scale=1)
```

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




### 3.2.1 Interpretation

- The boxplot indicates that salary levels differ across industries, with Information, Finance, and Professional Services showing higher median salaries. *Education and Public Admiration have lower median salaries with narrower ranges.
- Most industries center around $100k, though some display wider variation and notable outliers.

## 4 Salary Analysis by ONET Occupation Type (Bubble Chart)

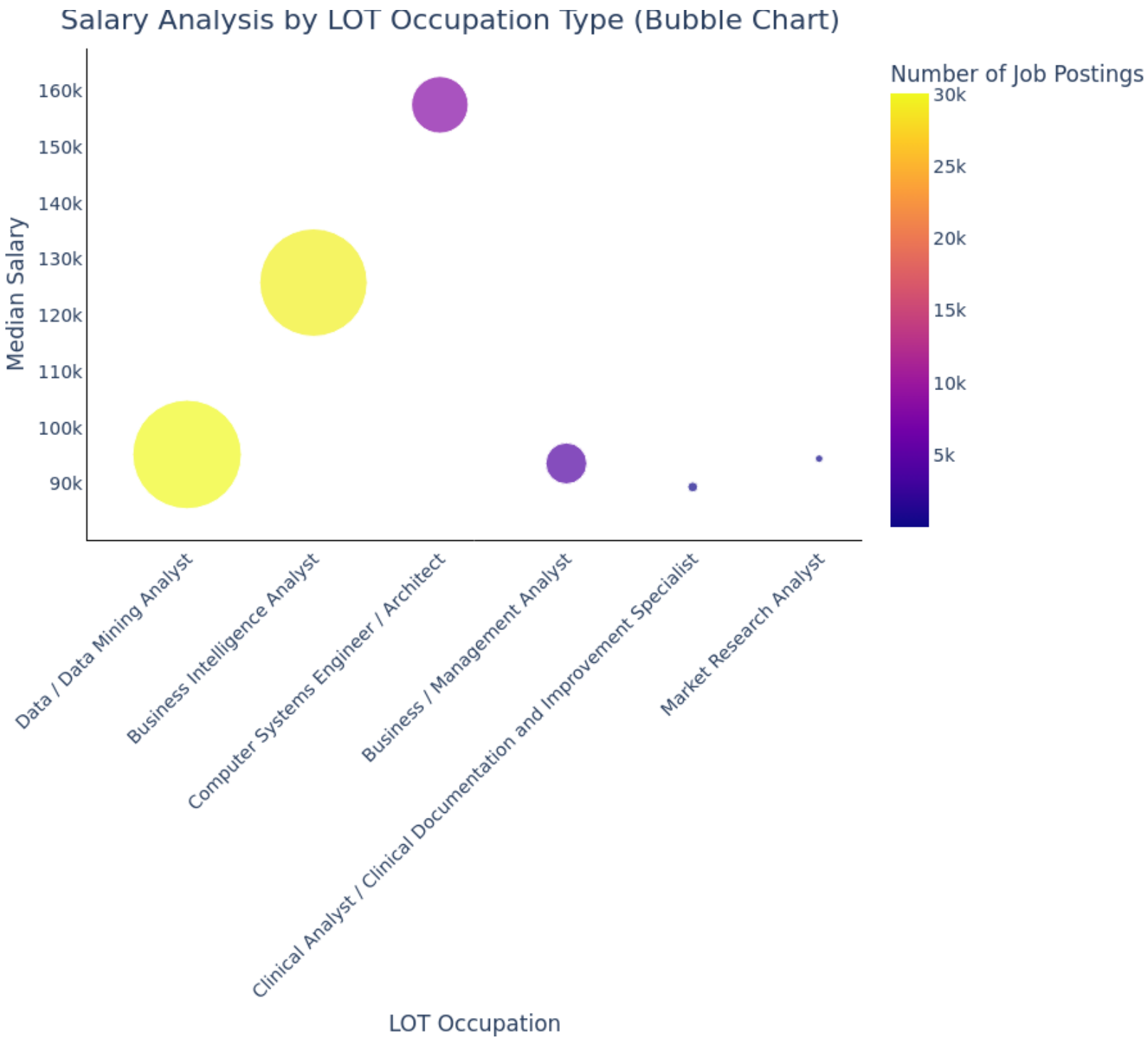
```
salary_analysis = spark.sql("""
SELECT
    LOT_OCCUPATION_NAME AS Occupation_Name,
    PERCENTILE(SALARY, 0.5) AS Median_Salary,
    COUNT(*) AS Job_Postings
FROM job_postings
GROUP BY LOT_OCCUPATION_NAME
ORDER BY Job_Postings DESC
LIMIT 10
""")

salary_pd = salary_analysis.toPandas()
salary_pd.head()
```

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

| | Occupation_Name | Median_Salary | Job_Postings |
|---|---|---|---|
| 0 | Data / Data Mining Analyst | 95250.0 | 30057 |
| 1 | Business Intelligence Analyst | 125900.0 | 29445 |
| 2 | Computer Systems Engineer / Architect | 157600.0 | 8212 |
| 3 | Business / Management Analyst | 93650.0 | 4326 |
| 4 | Clinical Analyst / Clinical Documentation and ... | 89440.0 | 261 |

```
fig = px.scatter(
   salary_pd,
   x="Occupation_Name",
   y="Median_Salary",
   size="Job_Postings",
   title="Salary Analysis by LOT Occupation Type (Bubble Chart)",
   labels={
       "Occupation_Name": "LOT Occupation",
       "Median_Salary": "Median Salary",
       "Job_Postings": "Number of Job Postings",
   },
```

```
    hover_name="Occupation_Name",
    size_max=60,
    width=900,
    height=800,
    color="Job_Postings",
    color_continuous_scale="Plasma",
)

fig.update_layout(
    font_family="Arial",
    font_size=14,
    title_font_size=22,
    xaxis_title="LOT Occupation",
    yaxis_title="Median Salary",
    plot_bgcolor="white",
    xaxis=dict(
        tickangle=-45,
        showline=True,
        linecolor="black",
    ),
    yaxis=dict(
        showline=True,
        linecolor="black",
    ),
)

fig.write_html("./output/Q3.html")
fig.write_image("./output/Q3.png", width=900, height=800, scale=1)
```


Salary Analysis by LOT Occupation Type (Bubble Chart)
Median Salary
160k
150k
140k
130k
120k
110k
100k
90k
Number of Job Postings
30k
25k
20k
15k
10k
5k
Data / Data Mining Analyst
Business Intelligence Analyst
Computer Systems Engineer / Architect
Business / Management Analyst
Clinical Analyst / Clinical Documentation and Improvement Specialist
Market Research Analyst
LOT Occupation