Assignment 03

Julio Vargas

September 23, 2025

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as F
from pyspark.sql.functions import col, split, explode, regexp_replace, transform, when
from pyspark.sql.functions import col, monotonically_increasing_id
from pyspark.sql.types import StructType  # to/from JSON

import json
import re
import numpy as np
import pandas as pd

import plotly.express as px
import plotly.io as pio
import plotly.graph_objects as go


np.random.seed(30)  # set a fixed seed for reproducibility
pio.renderers.default = "vscode+notebook"   #
# Initialize Spark Session
spark = SparkSession.builder.appName("JobPostingsAnalysis").getOrCreate()
# Load schema from JSON file
with open("data/schema_lightcast.json") as f:
    schema = StructType.fromJson(json.load(f))

# Load Data
df = (spark.read
      .option("header", "true")
      .option("inferSchema", "false")
      .schema(schema)              # saved schema
      .option("multiLine", "true")
      .option("escape", "\"")
```

```
      .csv("data/lightcast_job_postings.csv")
      )

df.createOrReplaceTempView("job_postings")
# Show Schema and Sample Data
#df.printSchema()
df.show(5)
```

```
+--------------------+----------------+----------------------+----------
+--------+---------+--------+--------------------+--------------------+--------------------
+-----------+--------------------+--------------------+--------------------
+---------------+----------------+--------+--------------------+-----------
+--------------------+----------------+----------------------+-------------
+--------------------+-------------+------------------+--------------+--------------------
+--------------------+--------------------+-------------+------+-----------
+----------------+--------------------+---------+-----------+--------------------
+--------------------+-------------+------+--------------+-----+--------------------
+-----+----------+---------------+--------------------+---------------+--------------------
+------------+--------------------+------------+--------------------+------
+--------------------+------+--------------------+------+--------------------
+------+--------------------+------+--------------------+------------------
+--------------------+--------------------+--------------------+--------------------
+--------------------+----------------------+--------------------+--------------------
+--------------------+--------------------+--------------------+--------------------
+----------+--------------------+----------+--------------------+--------------------
+--------------------+--------------------+--------------------+--------------------
+--------------------+----------+--------------------+----------+--------------------
+----------+---------------+----------+---------------+---------------+--------------------
+--------------+--------------------+----------------------------+-------------------------------
+--------------------+--------------------------+-----------------------------
+----------------------------------+-----------------+---------------------
+------------------------+----------------------------+------------------
+-------------------------+-------+--------------------+-------+--------------------
+-------+---------------+-------+---------------+-----------------+----------------------
+------------+--------------------+------------+--------------------+------------
+--------------------+------------+--------------------+------------+--------------------
+
|                  ID|LAST_UPDATED_DATE|LAST_UPDATED_TIMESTAMP|DUPLICATES|  POSTED|  EXPIRED
+--------------------+-----------------+----------------------+----------
+--------+---------+--------+--------------------+--------------------+--------------------
+-----------+--------------------+--------------------+--------------------
+---------------+----------------+--------+--------------------+-----------
```

```
+------------------+--------------+--------------------+-------------
+------------------+-------------+------------------+--------------+--------------------
+--------------------+--------------------+-------------+------+-----------
+----------------+------------------+---------+-----------+--------------------
+--------------------+-------------+------+--------------+-----+--------------------
+-----+----------+--------------+--------------------+--------------+--------------------
+-----------+--------------------+------------+--------------------+------
+--------------------+------+--------------------+------+--------------------
+------+--------------------+------+--------------------+------------------
+------------------+--------------------+--------------------+--------------------
+--------------------+----------------------+--------------------+--------------------
+--------------------+--------------------+--------------------+--------------------
+---------+--------------------+---------+--------------------+--------------------
+--------------------+--------------------+--------------------+--------------------
+---------+---------------+----------+--------------+---------------+--------------------
+-------------+--------------------+--------------------------+----------------------------
+------------------+------------------------+----------------------------
+----------------------------------+-----------------+----------------------
+----------------------+----------------------------+------------------
+-------+--------------+-------+--------------+-----------------+----------------------
+-----------+--------------------+------------+--------------------+------------
+------------------+------------+--------------------+-----------+--------------------
+
|1f57d95acf4dc67ed...|         9/6/2024|  2024-09-06 20:32:...|         0|6/2/2024| 6/8/2024
May-2024\n\nEn...|       6/8/2024|                6|  894731|          Murphy USA| Murphy USA
time (> 32 h...|                   2|                   2|        false|  NULL|          0|
2051.01|Business Intellig...|15-2051.01|Business Intellig...|[\n  "45.0601",\n...|[\n  "Econ
0000|Computer and Math...|   15-2000|Mathematical Scie...|   15-2050|Data Scientists|   15-
2051|Data Scientists|             23|Information Techn...|        231010|Business Intellig..
0000|Computer and Math...|15-2000|Mathematical Scie...|15-2050|Data Scientists|15-
2051|Data Scientists|        [\n  7\n]|  [\n  "Artificial ...|          44|        Retail Tra
|0cb072af26757b6c4...|         8/2/2024|  2024-08-02 17:08:...|         0|6/2/2024| 8/1/2024
time (> 32 h...|                   3|                   3|        false|  NULL|          1|
Watervill...|   23|     Maine|          23011|        Kennebec, ME|          23011|        Ke
Watervill...|       12300|Augusta-Watervill...|    56|Administrative an...|   561|Administrat
2051.01|Business Intellig...|15-2051.01|Business Intellig...|                  []|
0000|Computer and Math...|   15-2000|Mathematical Scie...|   15-2050|Data Scientists|   15-
2051|Data Scientists|             23|Information Techn...|        231010|Business Intellig..
0000|Computer and Math...|15-2000|Mathematical Scie...|15-2050|Data Scientists|15-
2051|Data Scientists|             NULL|                  NULL|          56|Administrative an
|85318b12b3331fa49...|         9/6/2024|  2024-09-06 20:32:...|         1|6/2/2024| 7/7/2024
```

```
time (> 32 h...|                   5|                NULL|        false|  NULL|          0|
Fort Worth...|   48|     Texas|          48113|          Dallas, TX|          48113|
Fort Worth...|       19100|Dallas-Fort Worth...|    52|Finance and Insur...|   524|Insurance
2051.01|Business Intellig...|15-2051.01|Business Intellig...|                  []|
0000|Computer and Math...|   15-2000|Mathematical Scie...|   15-2050|Data Scientists|   15-
2051|Data Scientists|             23|Information Techn...|        231113|Data / Data Minin..
0000|Computer and Math...|15-2000|Mathematical Scie...|15-2050|Data Scientists|15-
2051|Data Scientists|             NULL|                  NULL|          52|Finance and Insur
|1b5c3941e54a1889e...|         9/6/2024|  2024-09-06 20:32:...|         1|6/2/2024|7/20/2024
time (> 32 h...|                   3|                NULL|        false|  NULL|          0|
Mesa-Chan...|    4|   Arizona|           4013|        Maricopa, AZ|           4013|        Ma
Mesa-Chan...|       38060|Phoenix-Mesa-Chan...|    52|Finance and Insur...|   522|Credit Inte
2051.01|Business Intellig...|15-2051.01|Business Intellig...|                  []|
0000|Computer and Math...|   15-2000|Mathematical Scie...|   15-2050|Data Scientists|   15-
2051|Data Scientists|             23|Information Techn...|        231113|Data / Data Minin..
0000|Computer and Math...|15-2000|Mathematical Scie...|15-2050|Data Scientists|15-
2051|Data Scientists|        [\n  6\n]|  [\n  "Data Privac...|          52|Finance and Insur
|cb5ca25f02bdf25c1...|        6/19/2024|   2024-06-19 07:00:00|         0|6/2/2024|6/17/2024
time / full-...|                NULL|                NULL|        false| 92500|          0|
2051.01|Business Intellig...|15-2051.01|Business Intellig...|                  []|
0000|Computer and Math...|   15-2000|Mathematical Scie...|   15-2050|Data Scientists|   15-
2051|Data Scientists|             23|Information Techn...|        231010|Business Intellig..
0000|Computer and Math...|15-2000|Mathematical Scie...|15-2050|Data Scientists|15-
2051|Data Scientists|             NULL|                  NULL|          99|Unclassified Indu
+--------------------+----------------+---------------------+----------
+--------+---------+--------+--------------------+--------------------+--------------------
+-----------+-------------------+--------------------+--------------------
+---------------+----------------+--------+--------------------+-----------
+--------------------+----------------+---------------------+-------------
+--------------------+-------------+-------------------+---------------+--------------------
+---------------------+---------------------+-------------+------+-----------
+----------------+------------------+---------+-----------+--------------------
+--------------------+-------------+------+--------------+-----+--------------------
+-----+----------+---------------+--------------------+---------------+--------------------
+------------+--------------------+------------+---------------------+------
+--------------------+------+--------------------+------+--------------------
+------+--------------------+------+--------------------+------------------
+--------------------+---------------------+--------------------+--------------------
+--------------------+----------------------+--------------------+--------------------
+--------------------+--------------------+--------------------+--------------------
+----------+--------------------+----------+--------------------+--------------------
+--------------------+--------------------+--------------------+--------------------
+--------------------+----------+--------------------+----------+--------------------
```

```
+---------+--------------+---------+--------------+--------------+------------------
+--------------+--------------------+-------------------------+-----------------------------
+--------------------+-------------------------+----------------------------
+-----------------------------------+-----------------+----------------------
+-----------------------+-----------------------------+------------------
+-----------------------+-------+--------------------+-------+--------------------
+-------+---------------+-------+---------------+-----------------+----------------------
+-----------+--------------------+-----------+--------------------+------------
+--------------------+-----------+--------------------+-----------+--------------------
+
only showing top 5 rows
```

```
# Histogram of SALARY distribution (cast + filter)
salary_df = (
    df.select(col("SALARY").cast("float"))
      .filter(col("SALARY").isNotNull() & (col("SALARY") > 0))
)

fig = px.histogram(
    salary_df.toPandas(),
    x="SALARY",
    nbins=50,
    title="Salary Distribution"
)
fig.update_layout(bargap=0.1)
fig
```

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

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

```
Unable to display output for mime type(s): application/vnd.plotly.v1+json, text/html
```

1 Data Preparation

```
# Step 1: Casting salary and experience columns
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"))

# Step 2: Computing medians 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)

# Step 3: Imputing missing salaries, but not experience
df = df.fillna({
    "SALARY_FROM": median_from,
    "SALARY_TO": median_to,
    "SALARY": median_salary
})

# Step 5: Computing average salary
df = df.withColumn("Average_Salary",
                   (col("SALARY_FROM") + col("SALARY_TO")) / 2)

# Step 6: 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 7: 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 28:>                                                         (0 + 1) / 1]
```

```
Medians: 87295.0 130042.0
```

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

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

1.1 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** (e.g., `NAICS2_NAME`). - Group by **employment type** (`EMPLOYMENT_TYPE_NAME`) and compute salary distribution. - Calculate **salary percentiles** (25th, 50th, 75th) for each group.

Visualize results - Create a **box plot** where: - **X-axis** = `NAICS2_NAME` - **Y-axis** = `SALARY_FROM`, or `SALARY_TO`, or `SALARY` - **Group/color** = `EMPLOYMENT_TYPE_NAME` - Customize colors, fonts, and styles.

Explanation: Write two sentences about what the graph reveals (e.g., median differences across industries and dispersion by employment type).

2 Set up plotly template

```
import plotly.graph_objects as go
import plotly.io as pio

pio.templates["nike"] = go.layout.Template(
    # LAYOUT
    layout = {
        # Fonts
        # Note - 'family' must be a single string, NOT a list or dict!
        'title': {
            'font': {'family': 'HelveticaNeue-CondensedBold, Helvetica, Sans-serif',
```

```
                    'size': 30,
                    'color': '#13007c'} # 333
        },
        'font': {'family': 'Helvetica Neue, Helvetica, Sans-serif',
                 'size': 16,
                 'color': '#3b3b3b'}, # 333

        # Colorways ec7424
        'colorway': ['#fffb00', '#e010fc'], # CC0000, a4adab
        # Keep adding others as needed below
        'hovermode': 'x unified'
    },
    # DATA
    data = {
        # Each graph object must be in a tuple or list for each trace
        'bar': [go.Bar(
            texttemplate = '%{value:$.2s}',
            textposition = 'outside',
            textfont = {'family': 'Helvetica Neue, Helvetica, Sans-serif',
                        'size': 20,
                        'color': '#ff6874'} # FFFFFF
        )]
    }
)
```

```
#your code for first query
import pandas as pd
import polars as pl
from IPython.display import display, HTML

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


# Clean employment type names for better readability
pdf["EMPLOYMENT_TYPE_NAME"] = (
    pdf["EMPLOYMENT_TYPE_NAME"]
      .astype(str)
      .str.replace(r"[^\x00-\x7F]+", "", regex=True)
)

#display(HTML(f"<div style='height:300px; overflow:auto'>{pdf.iloc[:10].to_html(index=False)
```

```
# Compute median salary for sorting
median_salaries = pdf.groupby("EMPLOYMENT_TYPE_NAME")["SALARY"].median()
display(median_salaries.to_frame().head())


# Sort employment types based on median salary in descending order
sorted_employment_types = median_salaries.sort_values(ascending=False).index

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

# Create box plot with horizontal grid lines
fig = px.box(
    pdf,
    x="EMPLOYMENT_TYPE_NAME",
    y="SALARY",
    title="Salary Distribution by Employment Type",
    color_discrete_sequence=["#CC0000"],  # Single neutral color
    boxmode="group",
    points="all"  # Show all outliers
)
fig



# Improve layout, font styles, and axis labels
fig.update_layout(
    title=dict(
        text="Salary Distribution by Employment Type",
        font=dict(size=16, family="Helvetica", color="black")  # Bigger & Bold Title
    ),
    xaxis=dict(
        title=dict(text="Employment Type", font=dict(size=14, family="Helvetica", color="bla
        tickangle=0,  # Rotate X-axis labels for readability
        tickfont=dict(size=12, family="Helvetica", color="black"),  # Bigger & Bold X-ticks
        showline=True,  # Show axis lines
        linewidth=2,    # Thicker axis lines
        linecolor="black",
```

```
        mirror=True,
        showgrid=False,  # Remove vertical grid lines
        categoryorder="array",
        categoryarray=sorted_employment_types.tolist()
    ),
    yaxis=dict(
        title=dict(text="Salary (in $1000)", font=dict(size=14, family="Helvetica", color="bl
        tickvals=[0, 50000, 100000, 150000, 200000, 250000, 300000, 350000, 400000, 450000, 5
        ticktext=["0", "50", "100", "150", "200", "250", "300", "350", "400", "450", "500"],
        tickfont=dict(size=12, family="Helvetica", color="black"),  # Bigger & Bold Y-ticks
        showline=True,
        linewidth=2,
        linecolor="black",
        mirror=True,
        showgrid=True,       # Enable light horizontal grid lines
        gridcolor="lightgray",  # Light shade for the horizontal grid
        gridwidth=0.5        # Thin grid lines
    ),
    font=dict(family="Helvetica", size=12, color="black"),
    boxgap=0.7,
    plot_bgcolor="white",
    paper_bgcolor="white",
    showlegend=False,
    height=500,
    width=850
)

# Show & export
fig.show()
fig.write_html("output/Q1.html")
fig.write_image("output/Q1.svg", width=850, height=500, scale=1)
```

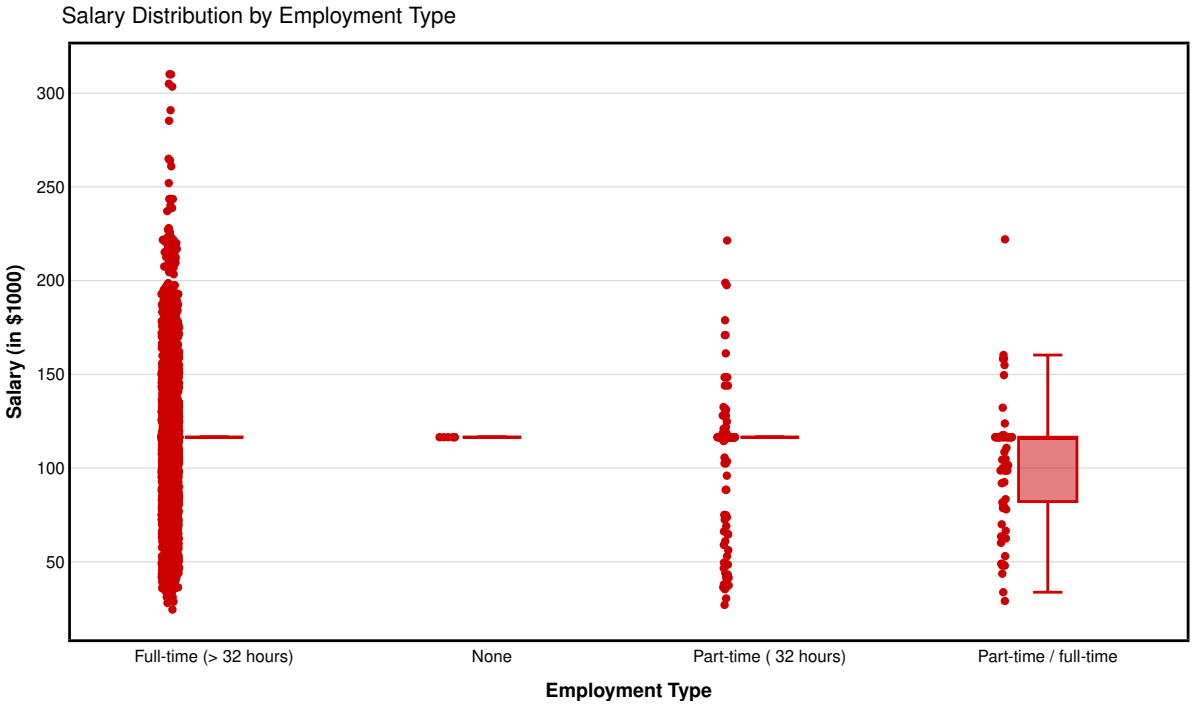
```
[Stage 32:>                                                 (0 + 1) / 1]
```

EMPLOYMENT_TYPE_NAME	SALARY
Full-time (> 32 hours)	115024.0
None	115024.0
Part-time (32 hours)	115024.0
Part-time / full-time	115024.0

```
Unable to display output for mime type(s): application/vnd.plotly.v1+json, text/html
```

```
/tmp/ipykernel_17899/1235387320.py:91: DeprecationWarning:


Support for Kaleido versions less than 1.0.0 is deprecated and will be removed after Septembe
Please upgrade Kaleido to version 1.0.0 or greater (`pip install 'kaleido>=1.0.0'` or `pip in
```



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.

- **Explanation:** Write two sentences about what the graph reveals.

```
pdf = df.select("NAICS2_NAME", "SALARY").toPandas()
fig = px.box(pdf, x="NAICS2_NAME", y="SALARY", title="Salary Distribution by Industry",
             color_discrete_sequence=["#EF553B"])
                 # add nike template
fig.update_layout(template="nike")

fig.update_xaxes(tickangle=45)
fig.show()
```

```
[Stage 33:>                                                         (0 + 1) / 1]
```

```
Unable to display output for mime type(s): application/vnd.plotly.v1+json, text/html
```

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.
- **Explanation:** Write two sentences about what the graph reveals.

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

df_onet = pdf.groupby(["ONET_NAME"]).agg({"SALARY": "median", "ID": "count"}).reset_index()
fig = px.scatter(
    df_onet,
    x="ONET_NAME",
    y="SALARY",
    size="ID",
    title="Salary Analysis by ONET Occupation",
    color="ONET_NAME"
)
fig.update_layout(font_family="Arial", title_font_size=16)
fig.show()
```

```
[Stage 34:>                                                         (0 + 1) / 1]
```

```
Unable to display output for mime type(s): application/vnd.plotly.v1+json, text/html
```

5 Salary by Education Level

- Create two groups:
  - **Bachelor's or lower** (Bachelor's, GED, Associate, No Education Listed)
  - **Master's or PhD** (Master's degree, Ph.D. or professional degree)
- Plot scatter plots for each group using `MAX_YEARS_EXPERIENCE` (with jitter), `Average_Salary`, `LOT_V6_SPECIALIZED_OCCUPATION_NAME`.
- Then, plot histograms overlaid with KDE curves for each group.
  - This would generate two scatter plots and two histograms.
- **After each graph, add a short explanation** of key insights.

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.**

Submission Instructions

- Submit the Word Document (part of git repo) containing:
  - The **HTTPS URL** of your GitHub repository.
  - Answer to the questions.
  - **Visualizations** created using matplotlib, Seaborn or plotly (preferred).
  - Answers to the questions below.

```
# Step 1: Spark SQL - Median salary and job count per TITLE_NAME
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
""")

# Step 2: Convert to Pandas DataFrame
salary_pd = salary_analysis.toPandas()
salary_pd.head()



# Step 3: Bubble chart using Plotly

import plotly.express as px

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=1000,
    height=600,
    color="Job_Postings",
    color_continuous_scale="Plasma"
)



# Step 4: Layout customization
fig.update_layout(
    font_family="Arial",
    font_size=14,
    title_font_size=25,
    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"
    )
)

# Step 5: Show and export
fig.show()
fig.write_image("output/Q7.svg", width=1000, height=600, scale=1)
```

```
[Stage 35:>                                                         (0 + 1) / 1]
```

```
Unable to display output for mime type(s): application/vnd.plotly.v1+json, text/html
```

```
/tmp/ipykernel_17899/2086897635.py:65: DeprecationWarning:


Support for Kaleido versions less than 1.0.0 is deprecated and will be removed after Septembe
Please upgrade Kaleido to version 1.0.0 or greater (`pip install 'kaleido>=1.0.0'` or `pip in
```

```
# Defining education level groupings
lower_deg = ["Bachelor's", "Associate", "GED", "No Education Listed", "High school"]
higher_deg = ["Master's degree", "PhD or professional degree"]

# Adding EDU_GROUP column
df = df.withColumn(
    "EDU_GROUP",
    when(col("EDUCATION_LEVELS_NAME").rlike("|".join([f"(?i){deg}" for deg in lower_deg])), '
    .when(col("EDUCATION_LEVELS_NAME").rlike("|".join([f"(?i){deg}" for deg in higher_deg]))
    .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"))

# 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)
)

# Filtering for just the two education groups
df_filtered = df.filter(col("EDU_GROUP").isin("Bachelor's or lower", "Master's or PhD"))

# Converting to Pandas for plotting
df_pd = df_filtered.toPandas()
df_pd.head()
```

```
[Stage 38:>                                                         (0 + 1) / 1]
```

	ID	LAST_UPDATED_DATE	LAST_UPDATED_TIMEST
0	1f57d95acf4dc67ed2819eb12f049f6a5c11782c	9/6/2024	2024-09-06 20:32:57.352
1	0cb072af26757b6c4ea9464472a50a443af681ac	8/2/2024	2024-08-02 17:08:58.838
2	5a843df632e1ff756fa19d80a0871262d51becc0	6/21/2024	2024-06-21 07:00:00.000
3	229620073766234e814e8add21db7dfaef69b3bd	10/9/2024	2024-10-09 18:07:44.758
4	138ce2c9453b47a9b33403c364d4fd80996caa4f	8/10/2024	2024-08-10 19:36:49.244