

Assignment 03

Julio Vargas

September 21, 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")
.limit(5000))

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

```

ID	LAST_UPDATED_DATE	LAST_UPDATED_TIMESTAMP	DUPLICATES	POSTED	EXPIRED
1	2023-01-01	2023-01-01 12:00:00	1	1	1
2	2023-01-02	2023-01-02 13:00:00	1	1	1
3	2023-01-03	2023-01-03 14:00:00	1	1	1
4	2023-01-04	2023-01-04 15:00:00	1	1	1
5	2023-01-05	2023-01-05 16:00:00	1	1	1
6	2023-01-06	2023-01-06 17:00:00	1	1	1
7	2023-01-07	2023-01-07 18:00:00	1	1	1
8	2023-01-08	2023-01-08 19:00:00	1	1	1
9	2023-01-09	2023-01-09 20:00:00	1	1	1
10	2023-01-10	2023-01-10 21:00:00	1	1	1
11	2023-01-11	2023-01-11 22:00:00	1	1	1
12	2023-01-12	2023-01-12 23:00:00	1	1	1
13	2023-01-13	2023-01-13 00:00:00	1	1	1
14	2023-01-14	2023-01-14 01:00:00	1	1	1
15	2023-01-15	2023-01-15 02:00:00	1	1	1
16	2023-01-16	2023-01-16 03:00:00	1	1	1
17	2023-01-17	2023-01-17 04:00:00	1	1	1
18	2023-01-18	2023-01-18 05:00:00	1	1	1
19	2023-01-19	2023-01-19 06:00:00	1	1	1
20	2023-01-20	2023-01-20 07:00:00	1	1	1
21	2023-01-21	2023-01-21 08:00:00	1	1	1
22	2023-01-22	2023-01-22 09:00:00	1	1	1
23	2023-01-23	2023-01-23 10:00:00	1	1	1
24	2023-01-24	2023-01-24 11:00:00	1	1	1
25	2023-01-25	2023-01-25 12:00:00	1	1	1
26	2023-01-26	2023-01-26 13:00:00	1	1	1
27	2023-01-27	2023-01-27 14:00:00	1	1	1
28	2023-01-28	2023-01-28 15:00:00	1	1	1
29	2023-01-29	2023-01-29 16:00:00	1	1	1
30	2023-01-30	2023-01-30 17:00:00	1	1	1
31	2023-01-31	2023-01-31 18:00:00	1	1	1
32	2023-02-01	2023-02-01 19:00:00	1	1	1
33	2023-02-02	2023-02-02 20:00:00	1	1	1
34	2023-02-03	2023-02-03 21:00:00	1	1	1
35	2023-02-04	2023-02-04 22:00:00	1	1	1
36	2023-02-05	2023-02-05 23:00:00	1	1	1
37	2023-02-06	2023-02-06 00:00:00	1	1	1
38	2023-02-07	2023-02-07 01:00:00	1	1	1
39	2023-02-08	2023-02-08 02:00:00	1	1	1
40	2023-02-09	2023-02-09 03:00:00	1	1	1
41	2023-02-10	2023-02-10 04:00:00	1	1	1
42	2023-02-11	2023-02-11 05:00:00	1	1	1
43	2023-02-12	2023-02-12 06:00:00	1	1	1
44	2023-02-13	2023-02-13 07:00:00	1	1	1
45	2023-02-14	2023-02-14 08:00:00	1	1	1
46	2023-02-15	2023-02-15 09:00:00	1	1	1
47	2023-02-16	2023-02-16 10:00:00	1	1	1
48	2023-02-17	2023-02-17 11:00:00	1	1	1
49	2023-02-18	2023-02-18 12:00:00	1	1	1
50	2023-02-19	2023-02-19 13:00:00	1	1	1
51	2023-02-20	2023-02-20 14:00:00	1	1	1
52	2023-02-21	2023-02-21 15:00:00	1	1	1
53	2023-02-22	2023-02-22 16:00:00	1	1	1
54	2023-02-23	2023-02-23 17:00:00	1	1	1
55	2023-02-24	2023-02-24 18:00:00	1	1	1
56	2023-02-25	2023-02-25 19:00:00	1	1	1
57	2023-02-26	2023-02-26 20:00:00	1	1	1
58	2023-02-27	2023-02-27 21:00:00	1	1	1
59	2023-02-28	2023			

```

+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+
|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 Tr
|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|      K
Watervill...|      12300|Augusta-Watervill...| 56|Administrative an...| 561|Administra
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

```



```

+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+

```

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

```

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))

```

Medians: 89565.0 131400.0
Data cleaning complete. Rows retained: 5000

1.1 Salary Distribution by Industry and Employment Type

- Compare salary variations across industries.

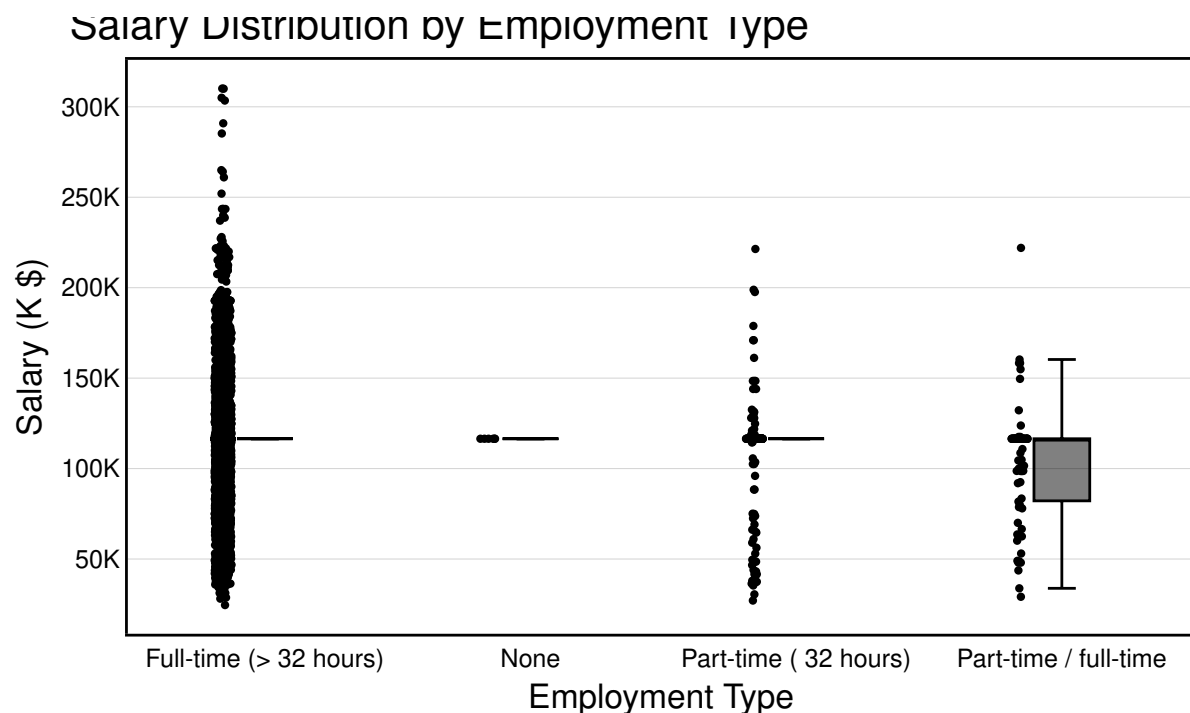
Filter the dataset - Remove records where **salary** is missing or zero.

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



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.

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.

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)
```

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