Assignment 03

Julio Vargas

September 21, 2025

```python
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))
# Show Schema and Sample Data
#df.printSchema()
#df.show(5)
```

```
# 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).

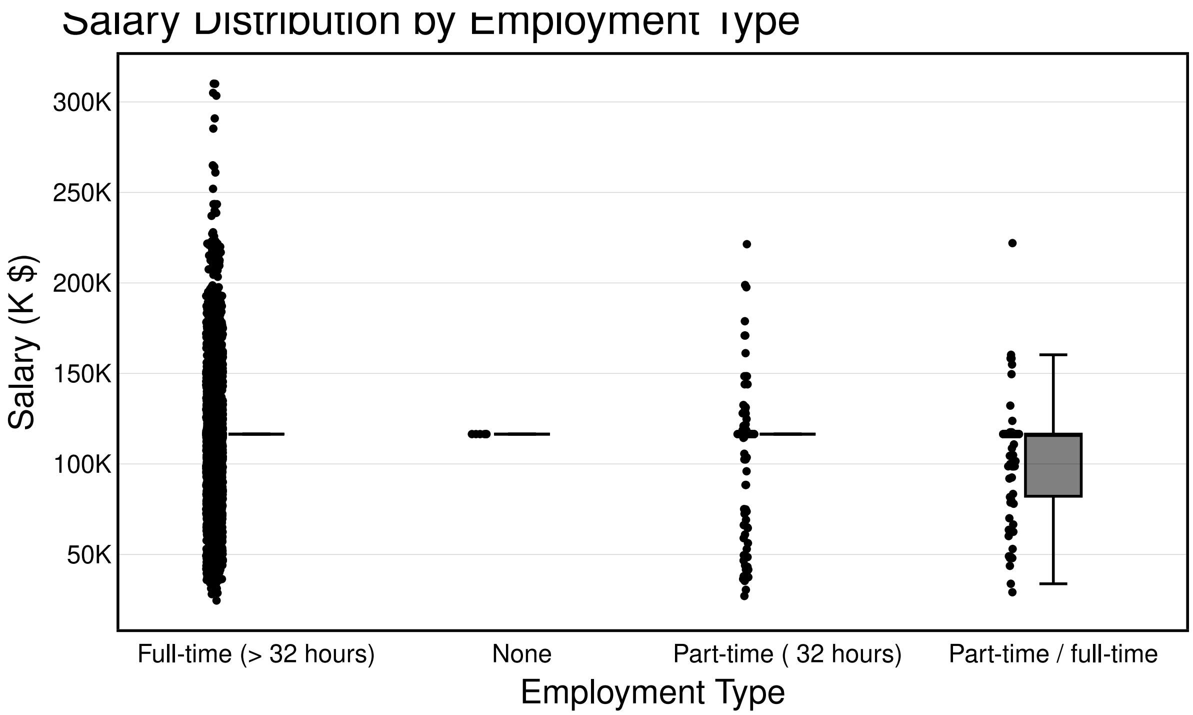
```
#your code for first query
import pandas as pd
import polars as pl
from IPython.display import display, HTML

# 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="blac
        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)
```

	SALARY
EMPLOYMENT_TYPE_NAME	
Full-time (> 32 hours)	116490.0
None	116490.0
Part-time (32 hours)	116490.0
Part-time / full-time	116490.0

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



2 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.

```python
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"])
fig.update_layout(font_family="Arial", title_font_size=16,
                height=900,
                width=1100)
# rotate x-axis label for readability
fig.update_xaxes(tickangle=45, tickfont=dict(size=12))
fig.show()
```

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