Assignment 03

Fuhan Zhang

September 21, 2025

1 Load the Dataset

```
import os, sys
os.environ["PYSPARK_PYTHON"] = sys.executable
os.environ["PYSPARK_DRIVER_PYTHON"] = sys.executable
os.environ.pop("SPARK_HOME", None)
os.environ.pop("SPARK_DIST_CLASSPATH", None)
os.makedirs("./output", exist_ok=True)
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, monotonically_increasing_id
from pyspark.sql import functions as F
import pandas as pd
import numpy as np
import plotly.io as pio

np.random.seed(42)
pio.renderers.default = "notebook"

# Initialize Spark Session
spark = SparkSession.builder.appName("LightcastData").getOrCreate()
# Load Data
df = spark.read.option("header", "true").option("inferSchema", "true").option("multiLine","t
df.createOrReplaceTempView("job_postings")

# 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)
```

```
df = df.withColumn("SALARY", col("SALARY").cast("float")) \
       .withColumn("MAX_YEARS_EXPERIENCE", col("MAX_YEARS_EXPERIENCE").cast("float"))

def compute_median(sdf, col_name):
    q = sdf.approxQuantile(col_name, [0.5], 0.01)
    return q[0] if q else None

median_salary = compute_median(df, "SALARY")
print("Median SALARY:", median_salary)

df = df.fillna({"SALARY": median_salary})

df = df.withColumn("Average_Salary", col("SALARY"))

export_cols = [
    "EDUCATION_LEVELS_NAME",
    "REMOTE_TYPE_NAME",
    "MAX_YEARS_EXPERIENCE",
    "Average_Salary",
    "SALARY",
    "LOT_V6_SPECIALIZED_OCCUPATION_NAME",
    "EMPLOYMENT_TYPE_NAME"
]
df_selected = df.select(*export_cols)

pdf= df_selected.toPandas()
pdf.to_csv("./output/cleaned_subset.csv", index=False)
print("Data Cleaning Complete. Rows Retained:", len(pdf))
```

```
root
 |-- EDUCATION_LEVELS_NAME: string (nullable = true)
 |-- REMOTE_TYPE_NAME: string (nullable = true)
 |-- MAX_YEARS_EXPERIENCE: double (nullable = true)
 |-- Average_Salary: double (nullable = true)
 |-- SALARY: double (nullable = true)
 |-- LOT_V6_SPECIALIZED_OCCUPATION_NAME: string (nullable = true)
 |-- EMPLOYMENT_TYPE_NAME: string (nullable = true)

Median SALARY: 115024.0
Data Cleaning Complete. Rows Retained: 72498
```

2 Salary Distribution by Employment Type

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
import re
import plotly.express as px
import plotly.io as pio
pio.renderers.default = "iframe"
#Data Cleaning & Filtering
pdf = df.filter(df["SALARY"] > 0).select("EMPLOYMENT_TYPE_NAME", "SALARY").toPandas()
pdf["EMPLOYMENT_TYPE_NAME"] = pdf["EMPLOYMENT_TYPE_NAME"].apply(
    lambda x: re.sub(r"[^\x00-\x7F]+", "", str(x)) if x is not None else x
)
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
)

#Creating the Boxplot
fig = px.box(
    pdf,
    x="EMPLOYMENT_TYPE_NAME",
    y="SALARY",
    title="Salary Distribution by Employment Type",
    color_discrete_sequence=["#ffb6c1", "#cb1a72ff", "#db7093", "#c71585"],
    boxmode="group",
    points="all"
)
fig.update_layout(
    title=dict(text="Salary Distribution by Employment Type", font=dict(size=30, family="Aria
    xaxis=dict(
        title=dict(text="Employment Type", font=dict(size=24, family="Arial", color="black",
        tickangle=0,
        tickfont=dict(size=18, family="Arial", color="black", weight="bold"),
        showline=True, linewidth=2, linecolor="black", mirror=True,
        showgrid=False,
        categoryorder="array",
        categoryarray=sorted_employment_types.tolist()
```

```
    ),
    yaxis=dict(
        title=dict(text="Salary (K $)", font=dict(size=24, family="Arial", color="black", wei
        tickvals=[0, 50000, 100000, 150000, 200000, 250000, 300000, 350000, 400000, 450000, 5
        ticktext=["0", "50K", "100K", "150K", "200K", "250K", "300K", "350K", "400K", "450K"
        tickfont=dict(size=18, family="Arial", color="black", weight="bold"),
        showline=True, linewidth=2, linecolor="black", mirror=True,
        showgrid=True, gridcolor="lightgrey", gridwidth=0.5
    ),
  font=dict(family="Arial", size=16, color="black"),
    boxgap=0.7,
    plot_bgcolor="white",
    paper_bgcolor="white",
    showlegend=False,
    height=500,
    width=850
)
fig.show()
fig.write_image("output/Q1.png", width=850, height=500, scale=1)
![](output/Q1.png)
```

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

```
/bin/bash: -c: line 1: syntax error near unexpected token `output/Q1.png'
/bin/bash: -c: line 1: `[](output/Q1.png)'
```

3 Salary Analysis by ONET Occupation Type (Bubble Chart)

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
import plotly.express as px
#Spark SQL to Converting
salary_analysis = spark.sql("""
    SELECT
        LOT_V6_SPECIALIZED_OCCUPATION_NAME AS ONET_NAME,
        PERCENTILE(SALARY, 0.5) AS Median_Salary,
        COUNT(*) AS Job_Postings
    FROM job_postings
    GROUP BY LOT_V6_SPECIALIZED_OCCUPATION_NAME
```

```
    ORDER BY Job_Postings DESC
    LIMIT 10
""")
salary_pd = salary_analysis.toPandas()
#Creating Bubble Chart
fig = px.scatter(
    salary_pd,
    x="ONET_NAME",
    y="Median_Salary",
    size="Job_Postings",
    title="Salary Analysis by ONET Occupation Type (Bubble Chart)",
    labels={
        "ONET_NAME": "ONET Occupation",
        "Median_Salary": "Median Salary",
        "Job_Postings": "Number of Job Postings"
    },
    hover_name="ONET_NAME",
    size_max=60,
    width=1000,
    height=600,
    color="Job_Postings",
    color_discrete_sequence=["#ffe4e1", "#ffb6c1", "#ff69b4", "#db7093", "#c71585"],
)
fig.update_layout(
    font_family="Arial",
    font_size=14,
    title_font_size=25,
    xaxis_title="ONET Occupation",
    yaxis_title="Median Salary",
    plot_bgcolor="white",
    xaxis=dict(tickangle=-45, showline=True),
    yaxis=dict(showline=True)
)
fig.show()
fig.write_image("output/Q2.png", width=1000, height=600, scale=2)
![](output/Q2.png)
```

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

```
/bin/bash: -c: line 1: syntax error near unexpected token `output/Q2.png'
/bin/bash: -c: line 1: `[](output/Q2.png)'
```

#Salary by Education Level

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when, lit
import re
import plotly.io as pio
pio.renderers.default = "iframe"
#Educational Groups Data Setting
lower_deg  = ["Bachelor's", "Associate", "GED", "No Education Listed", "High school"]
higher_deg = ["Master's degree", "PhD or professional degree"]
df = df.withColumn(
    "EDU_GROUP",
    when(col("EDUCATION_LEVELS_NAME").isin(lower_deg), lit("Bachelor's or lower"))
    .when(col("EDUCATION_LEVELS_NAME").isin(higher_deg), lit("Master's or PhD"))
    .otherwise(lit("Other"))
)
df = df.withColumn("MAX_YEARS_EXPERIENCE", col("MAX_YEARS_EXPERIENCE").cast("float"))
df = df.withColumn("Average_Salary", col("Average_Salary").cast("float"))
df_filtered = df.filter(
    col("MAX_YEARS_EXPERIENCE").isNotNull() & (col("MAX_YEARS_EXPERIENCE") > 0) &
    col("Average_Salary").isNotNull() & (col("Average_Salary") > 0))
pdf_scatter = df_filtered.toPandas()
df_filtered = df.filter(col("EDU_GROUP").isin("Bachelor's or lower", "Master's or PhD"))
df_pd = df_filtered.toPandas()

#Creating the Scatter Plot
fig1 = px.scatter(
    df_pd, x="MAX_YEARS_EXPERIENCE", y="Average_Salary",
    color="EDU_GROUP",
    hover_data=["LOT_V6_SPECIALIZED_OCCUPATION_NAME"],
    opacity=0.7,
    color_discrete_sequence=["#ffb6c1", "#cb1a72ff"],
    title="Experience vs Salary by Education Level"
)
fig1.update_traces(marker=dict(size=7, line=dict(width=1, color="black")))
fig1.update_layout(
    plot_bgcolor="#f9f9f9", paper_bgcolor="#EFF5DC",
    font=dict(family="Segoe UI", size=14),
    title_font=dict(size=22),
    xaxis_title="Years of Experience",
    yaxis_title="Average Salary (USD)",
    legend_title="Education Group",
    hoverlabel=dict(bgcolor="white", font_size=13, font_family="Arial"),
```

```
    margin=dict(t=70, r=80, b=60, l=60),
    xaxis=dict(gridcolor="lightgrey", tickmode="linear", dtick=1),
    yaxis=dict(gridcolor="lightgrey")
)
fig1.show()
fig1.write_image("output/Q3.png", width=950, height=550, scale=2)
![](output/Q3.png)
```

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

```
/bin/bash: -c: line 1: syntax error near unexpected token `output/Q3.png'
/bin/bash: -c: line 1: `[](output/Q3.png)'
```

#Salary by Remote Work Type

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
from pyspark.sql.functions import when, trim
import numpy as np
import plotly.express as px
import plotly.io as pio
pio.renderers.default = "iframe"
np.random.seed(42)

#Work Types Data Setting
df = df.withColumn("REMOTE_GROUP",
    when(trim(col("REMOTE_TYPE_NAME")) == "Remote", "Remote")
    .when(trim(col("REMOTE_TYPE_NAME")) == "Hybrid Remote", "Hybrid")
    .when(trim(col("REMOTE_TYPE_NAME")) == "Not Remote", "Onsite")
    .when(col("REMOTE_TYPE_NAME").isNull(), "Onsite")
    .otherwise("Onsite")
)
df = df.filter(
    (col("MAX_YEARS_EXPERIENCE").isNotNull()) &
    (col("Average_Salary").isNotNull()) &
    (col("MAX_YEARS_EXPERIENCE") > 0) &
    (col("Average_Salary") > 0)
)
df_pd = df.select(
    "MAX_YEARS_EXPERIENCE",
    "Average_Salary",
```

```
    "LOT_V6_SPECIALIZED_OCCUPATION_NAME",
    "REMOTE_GROUP"
).toPandas()
#Mathematical Adjusting
df_pd["MAX_EXPERIENCE_JITTER"] = df_pd["MAX_YEARS_EXPERIENCE"] + np.random.uniform(-0.25, 0.2
df_pd["AVERAGE_SALARY_JITTER"] = df_pd["Average_Salary"] + np.random.uniform(-2500, 2500, siz
df_pd = df_pd.round(2)
df_pd.head()
df_pd = df_pd[df_pd["AVERAGE_SALARY_JITTER"] <= 390000]

#Creating the Scatter Plot
fig1 = px.scatter(
    df_pd,
    x="MAX_EXPERIENCE_JITTER",
    y="AVERAGE_SALARY_JITTER",
    color="REMOTE_GROUP",
    hover_data=["LOT_V6_SPECIALIZED_OCCUPATION_NAME"],
    title="<b>Experience vs Salary by Remote Work Type</b>",
    opacity=0.7,
    color_discrete_sequence=["#f9acb7", "#96d8ee", "#f3f39a"]
)

fig1.update_traces(marker=dict(size=7, line=dict(width=1, color="black")))

fig1.update_layout(
    plot_bgcolor="#f9f9f9",
    paper_bgcolor="#E9EAFF",
    font=dict(family="Segoe UI", size=14),
    title_font=dict(size=22),
    xaxis_title="Years of Experience",
    yaxis_title="Average Salary (USD)",
    legend_title="Remote Work Type",
    hoverlabel=dict(bgcolor="white", font_size=13, font_family="Arial"),
    margin=dict(t=70, b=60, l=60, r=60),
    xaxis=dict(
        gridcolor="lightgrey",
        tickmode="linear",
        tick0=1,
        dtick=1,
        tickangle=0
    ),
    yaxis=dict(gridcolor="lightgrey"),
```

```
    legend=dict(orientation="h", yanchor="bottom", y=-1.02, xanchor="right", x=1)
)

fig1.show()
fig1.write_image("output/Q4.png", width=950, height=550, scale=2)
![](output/Q4.png)
```

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

```
/bin/bash: -c: line 1: syntax error near unexpected token `output/Q4.png'
/bin/bash: -c: line 1: `[](output/Q4.png)'
```