Assignment 03

Ava Godsy

September 22, 2025

1 Load the 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 = "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)
```

```
WARNING: Using incubator modules: jdk.incubator.vector
Using Spark's default log4j profile: org/apache/spark/log4j2-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/09/24 03:46:52 WARN NativeCodeLoader: Unable to load native-hadoop library for your platf
java classes where applicable
25/09/24 03:47:07 WARN SparkStringUtils: Truncated the string representation of a plan since
```

2 Data Cleaning

```python
df = df.withColumn("SALARY_FROM", col("SALARY_FROM").cast("float")) \
       .withColumn("SALARY_TO", col("SALARY_TO").cast("float")) \
       .withColumn("SALARY", col("SALARY").cast("float")) \
       .withColumn("MAX_YEARS_EXPERIENCE", col("MAX_YEARS_EXPERIENCE").cast("float")) \
       .withColumn("MIN_YEARS_EXPERIENCE", col("MIN_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_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)
```

```
[Stage 4:>                                                          (0 + 1) / 1]

Medians: 87295.0 - 130042.0 - 115024.0
```

```python
df = df.fillna({
    "SALARY_FROM": median_from,
    "SALARY_TO": median_to
})

df = df.withColumn("Average_Salary", (col("SALARY_FROM") + col("SALARY_TO")) / 2)
df = df.withColumn("EDUCATION_LEVELS_NAME", regexp_replace("EDUCATION_LEVELS_NAME", "\n ", "
```

```
df = df.withColumn("EDUCATION_LEVELS_NAME", regexp_replace("EDUCATION_LEVELS_NAME", "\n", "")
df = df.withColumn("EDUCATION_LEVELS_NAME", regexp_replace("EDUCATION_LEVELS_NAME", "\r ", "
df = df.withColumn("EDUCATION_LEVELS_NAME", regexp_replace("EDUCATION_LEVELS_NAME", "\r", "")

export_cols = [
    "EDUCATION_LEVELS_NAME",
    "REMOTE_TYPE_NAME",
    "MAX_YEARS_EXPERIENCE",
    "Average_Salary",
    "LOT_V6_SPECIALIZED_OCCUPATION_NAME",
    "NAICS2_NAME",
    "EMPLOYMENT_TYPE_NAME",
    "ONET_NAME",
    "SALARY_FROM",
    "SALARY_TO",
    "SALARY"
]
df_selected = df.select (*export_cols)

pdf = df_selected.toPandas()
pdf.to_csv("lightcast_cleaned.csv", index=False)

print(" Data cleaning complete. Rows retained:", len(pdf))
# df_selected.show(5)
```

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

3 Salary Distribution by Industry and Employment Type

The following data suggests, as expected, that salaries tend to be higher in full-time positions. One notable outlier is part-time / full-time roles in Healthcare, which may reflect the high demand for healthcare professionals even in part-time capacities, and/or the high demand for travel nurses. The Information and Professional / Scientific / Technical Service industries have some of the highest average salaries for full-time work, as represented by their boxes.

```
import plotly.express as px

fig = px.box(
```

```
    df_selected,
    x="NAICS2_NAME",
    y="SALARY_FROM",
    color="EMPLOYMENT_TYPE_NAME",
    title="Salary Distribution by Industry and Employment Type",
    points="all",  # Show all points
    notched=True,  # Notched boxes
    height=1000,  # Taller figure
    color_discrete_sequence=["purple", "blue", "green"]  # Custom colors
)

fig.update_layout(
    title_font=dict(family="Garamond", size=24, color="black"),
    xaxis_title="Industry (NAICS2)",
    yaxis_title="Starting Salary",
    boxmode="group",  # Grouped box plots
    xaxis_tickangle=45,  # Rotate x-axis labels
    font=dict(
        family="Garamond, serif",  # Set font to Garamond
        size=12
    )
)

fig.show()
```

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

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

4 Salary Analysis by ONET Occupation Type (Bubble Chart)

By far, the Data Analyst role had the highest number of job postings in this dataset, but a low-to-medium salary offered. Enterprise Architects have the highest average salary as well as a solid representation of job postings, 3,321. Oracle Consultant / Analyst is a close second with median salary and has slightly more availability, although similar. Healthcare and Marketing Analysts have very few offerings, which may be due to their more niche role. It is also possible that the data is collected and classified in a way that loses some level of detail.

```python
from pyspark.sql import functions as F

df_filtered = df_selected.filter(F.col("SALARY").isNotNull())

lot_salary = df_filtered.groupBy("LOT_V6_SPECIALIZED_OCCUPATION_NAME").agg(
    F.expr("percentile_approx(SALARY, 0.5)").alias("Median Salary"),
    F.count("*").alias("Job_Postings")
)

lot_salary.show()
```

```
[Stage 7:>                                                          (0 + 1) / 1]

+----------------------------------+-------------+------------+
|LOT_V6_SPECIALIZED_OCCUPATION_NAME|Median Salary|Job_Postings|
+----------------------------------+-------------+------------+
|              Business Intellig...|     107500.0|        1800|
|              Business Analyst ...|      93650.0|        1640|
|                Healthcare Analyst|      89440.0|          94|
|              Oracle Consultant...|     138750.0|        3526|
|               SAP Analyst / Admin|     120640.0|        3373|
|                      Data Analyst|      96100.0|       12377|
|              General ERP Analy...|     125900.0|        3703|
|                 Marketing Analyst|      94500.0|          65|
|              Enterprise Architect|     157600.0|        3321|
|              Financial Data An...|      49920.0|         429|
|              Data Quality Analyst|      96600.0|         480|
+----------------------------------+-------------+------------+
```

```python
import plotly.express as px

fig = px.scatter(
    lot_salary,
    x="LOT_V6_SPECIALIZED_OCCUPATION_NAME",
    y="Median Salary",
    size="Job_Postings",
    color="Median Salary",
    hover_name="LOT_V6_SPECIALIZED_OCCUPATION_NAME",
```

```
    size_max=60,
    title="Salary Analysis by LOT Occupation Type",
)
fig.update_layout(
    title_font=dict(family="Garamond", size=24, color="black"),
    font=dict(family="Garamond", size=12, color="black"),
    plot_bgcolor="white",
    paper_bgcolor="#f7f7f7",
    xaxis=dict(title="Occupation Name", tickangle=45),
    yaxis=dict(title="Median Salary ($)", gridcolor="#e5e5e5"),
)

# Show the figure
fig.show()
```

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

5 Salary by Education Level

What might be suprising about this data is the similarities in salaries between each education level. While we do see that Doctoral degrees have the highest average salary, the difference between Bachelor's, Master's, and Doctoral is not as large as one might expect. This could suggest that in the tech industry, experience and skills may play a more significant role in determining salary than formal education level alone. This also might point to a trend in many industries, where education barriers are being lowered and skills and experience are being prioritized more. For example, those listings that only require an Associate degree or lower do not frequently also allow for 0-1 years of experience.

```
# df_selected.select("EDUCATION_LEVELS_NAME").distinct().show(truncate=False)


from pyspark.sql.functions import col, when

# Create the EDU_GROUP column based on EDUCATION_LEVELS_NAME
df_with_edu_group = df_selected.withColumn(
    "EDU_GROUP",
    when(
        col("EDUCATION_LEVELS_NAME").rlike("(?i)No Education Listed|GED|Associate"),
        "Associate's or lower"
```

```python
    ).when(
        col("EDUCATION_LEVELS_NAME").rlike("(?i)Bachelor"),
        "Bachelor's"
    ).when(
        col("EDUCATION_LEVELS_NAME").rlike("(?i)Master"),
        "Master's"
    ).when(
        col("EDUCATION_LEVELS_NAME").rlike("(?i)Ph\\.D\\.|professional degree"),
        "PhD"
    ).otherwise("Associate's or lower")  # Optional: handle unmatched entries
)

# Select required columns
final_df = df_with_edu_group.select(
    "EDU_GROUP",
    "LOT_V6_SPECIALIZED_OCCUPATION_NAME",
    "Average_Salary",
    "MAX_YEARS_EXPERIENCE"
)

final_df.show()
```

```
+--------------------+----------------------------------+--------------
+--------------------+
|           EDU_GROUP|LOT_V6_SPECIALIZED_OCCUPATION_NAME|Average_Salary|MAX_YEARS_EXPERIENCE
+--------------------+----------------------------------+--------------
+--------------------+
|          Bachelor's|              General ERP Analy...|      108668.5|                 2.0
|Associate's or lower|              Oracle Consultant...|      108668.5|                 3.0
|          Bachelor's|                      Data Analyst|      108668.5|                NULL
|Associate's or lower|                      Data Analyst|      108668.5|                NULL
|Associate's or lower|              Oracle Consultant...|       92500.0|                NULL
|          Bachelor's|                      Data Analyst|      110155.0|                NULL
|          Bachelor's|                      Data Analyst|      108668.5|                NULL
|          Bachelor's|                      Data Analyst|      108668.5|                NULL
|Associate's or lower|              General ERP Analy...|      108668.5|                 7.0
|          Bachelor's|                      Data Analyst|       92962.0|                 2.0
|Associate's or lower|                      Data Analyst|      107645.5|                NULL
|Associate's or lower|                      Data Analyst|      108668.5|                NULL
|          Bachelor's|                      Data Analyst|      108668.5|                NULL
|          Bachelor's|              General ERP Analy...|      192800.0|                NULL
|Associate's or lower|                Enterprise Architect|       81286.0|                NULL
```

```
|Associate's or lower|                      Data Analyst|      108668.5|                   5.0
|Associate's or lower|              General ERP Analy...|      125900.0|                  NULL
|Associate's or lower|              Oracle Consultant...|      108668.5|                   3.0
|          Bachelor's|              Enterprise Architect|      165000.0|                   8.0
|Associate's or lower|                      Data Analyst|      170000.0|                  NULL
+--------------------+----------------------------------+--------------
+--------------------+
only showing top 20 rows
```

```python
import plotly.express as px
import numpy as np

# Step 1: Convert PySpark DataFrame to Pandas
pdf = final_df.toPandas()

# Step 2: Add jitter to MAX_YEARS_EXPERIENCE
np.random.seed(0)
jitter_strength = 0.1
pdf['JITTERED_EXPERIENCE'] = pdf['MAX_YEARS_EXPERIENCE'] + np.random.uniform(
    -jitter_strength, jitter_strength, size=len(pdf)
)

# Step 3: Define custom color mapping
color_map = {
    "Associate's or lower": 'yellow',
    "Bachelor's": 'green',
    "Master's": 'blue',
    "PhD": 'purple'
}

# Step 4: Create the Plotly scatter plot
fig = px.scatter(
    pdf,
    x='JITTERED_EXPERIENCE',
    y='Average_Salary',
    color='EDU_GROUP',
    color_discrete_map=color_map,
    title="Salary by Education Level",
    labels={
        'JITTERED_EXPERIENCE': 'Max Years of Experience Required (jittered)',
        'Average_Salary': 'Average Salary',
        'EDU_GROUP': 'Minimum Education Level Required'
```

```
    },
    opacity=0.7
)

# Step 5: Update layout with Garamond font and sizes
fig.update_layout(
    title_font=dict(family='Garamond', size=24, color='black'),
    font=dict(family='Garamond', size=12, color='black'),
    legend_title_font=dict(family='Garamond', size=12, color='black'),
    legend_font=dict(family='Garamond', size=12, color='black')
)

# Step 6: Show the figure
fig.show()
```

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

6 Salary by Remote Work Type

The scatterplots reveal that there are severla outliers for onsite salary being offered at 7 years of experience, despite the fact that this group has a lower average salary than remote roles at 7 years of experience. As expected, higher experience leads to higher salary, and there are far more onsite roles than remote and hybrid roles. The spread of salaries for onsite roles is also much wider than for remote roles, which may reflect the wider variety of industries and roles that offer onsite work.

There are also several data points for Hybrid roles where the average salary is $108k; this may be a result of having replaced missing salary data with the median salary.

What is more easily seen in the histogram is the higher average salary for mid-level experience, a trend that is especially strong for remote roles. We can also see that hybrid roles pay the least for most of the experience levels.

```
# df_selected.select("REMOTE_TYPE_NAME").distinct().show(truncate=False)

from pyspark.sql.functions import col, when
```

```python
df_with_remote_group = df_selected.withColumn(
    "REMOTE_GROUP",
    when(
        col("REMOTE_TYPE_NAME") == "Remote", "Remote"
    ).when(
        col("REMOTE_TYPE_NAME") == "Hybrid Remote", "Hybrid"
    ).when(
        (col("REMOTE_TYPE_NAME").isNull()) |
        (col("REMOTE_TYPE_NAME") == "Not Remote") |
        (col("REMOTE_TYPE_NAME") == "[None]"),
        "Onsite"
    ).otherwise("Onsite"))

remote_df = df_with_remote_group.select(
    "REMOTE_GROUP",
    "LOT_V6_SPECIALIZED_OCCUPATION_NAME",
    "Average_Salary",
    "MAX_YEARS_EXPERIENCE"
)

remote_df.show()
```

```
+------------+----------------------------------+--------------+--------------------
+
|REMOTE_GROUP|LOT_V6_SPECIALIZED_OCCUPATION_NAME|Average_Salary|MAX_YEARS_EXPERIENCE|
+------------+----------------------------------+--------------+--------------------
+
|      Onsite|              General ERP Analy...|      108668.5|                 2.0|
|      Remote|              Oracle Consultant...|      108668.5|                 3.0|
|      Onsite|                      Data Analyst|      108668.5|                NULL|
|      Onsite|                      Data Analyst|      108668.5|                NULL|
|      Onsite|              Oracle Consultant...|       92500.0|                NULL|
|      Remote|                      Data Analyst|      110155.0|                NULL|
|      Onsite|                      Data Analyst|      108668.5|                NULL|
|      Onsite|                      Data Analyst|      108668.5|                NULL|
|      Onsite|              General ERP Analy...|      108668.5|                 7.0|
|      Onsite|                      Data Analyst|       92962.0|                 2.0|
|      Onsite|                      Data Analyst|      107645.5|                NULL|
|      Onsite|                      Data Analyst|      108668.5|                NULL|
|      Onsite|                      Data Analyst|      108668.5|                NULL|
|      Onsite|              General ERP Analy...|      192800.0|                NULL|
|      Remote|              Enterprise Architect|       81286.0|                NULL|
```

```
|      Remote|                     Data Analyst|      108668.5|                 5.0|
|      Onsite|              General ERP Analy...|      125900.0|                NULL|
|      Remote|              Oracle Consultant...|      108668.5|                 3.0|
|      Onsite|              Enterprise Architect|      165000.0|                 8.0|
|      Onsite|                     Data Analyst|      170000.0|                NULL|
+------------+----------------------------------+--------------+--------------------
+
only showing top 20 rows
```

```python
import plotly.express as px
import numpy as np

# Step 1: Convert PySpark DataFrame to Pandas
pdf2 = remote_df.toPandas()

# Step 2: Add jitter to MAX_YEARS_EXPERIENCE
np.random.seed(0)
jitter_strength = 0.1
pdf2['JITTERED_EXPERIENCE'] = pdf2['MAX_YEARS_EXPERIENCE'] + np.random.uniform(
    -jitter_strength, jitter_strength, size=len(pdf2)
)

# Step 3: Define custom color mapping
color_map = {
    "Remote": 'yellow',
    "Hybrid": 'green',
    "Onsite": 'blue',
}

fig = px.scatter(
    pdf2,
    x='JITTERED_EXPERIENCE',
    y='Average_Salary',
    color='REMOTE_GROUP',
    color_discrete_map=color_map,
    title="Salary by Remote Status",
    labels={
        'JITTERED_EXPERIENCE': 'Max Years of Experience Required',
        'Average_Salary': 'Average Salary',
        'REMOTE_GROUP': 'Remote Status'
    },
    opacity=0.7
```

```
)

# Step 5: Update layout with Garamond font and sizes
fig.update_layout(
    title_font=dict(family='Garamond', size=24, color='black'),
    font=dict(family='Garamond', size=12, color='black'),
    legend_title_font=dict(family='Garamond', size=12, color='black'),
    legend_font=dict(family='Garamond', size=12, color='black')
)

# Step 6: Show the figure
fig.show()
```

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

```
# Step 1: Convert to Pandas
pdf = remote_df.toPandas()

# Step 2: Create plot using Plotly
import plotly.express as px

color_map = {
    "Remote": 'yellow',
    "Hybrid": 'green',
    "Onsite": 'blue',
}

fig = px.histogram(
    pdf,
    x="MAX_YEARS_EXPERIENCE",
    y="Average_Salary",
    color="REMOTE_GROUP",
    color_discrete_map=color_map,
    histfunc="avg",
    nbins=int(pdf['MAX_YEARS_EXPERIENCE'].max()) + 1,
    barmode='group',
    title="Average Salary by Years of Experience and Remote Type",
    labels={
        'MAX_YEARS_EXPERIENCE': 'Max Years of Experience Required',
```

```
        'Average_Salary': 'Average Salary',
        'REMOTE_GROUP': 'Remote Status'
    }
)

fig.update_layout(
    title_font=dict(family='Garamond', size=24, color='black'),
    font=dict(family='Garamond', size=12, color='black'),
    legend_title_font=dict(family='Garamond', size=12, color='black'),
    legend_font=dict(family='Garamond', size=12, color='black'),
    xaxis=dict(dtick=1),
    yaxis_title="Average Salary",
    bargap=0.2
)

fig.show()
```

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