assignment03-mayub

September 25, 2025

[1]:
```python
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","true").option("escape", "\"").csv("./data/
↪lightcast_job_postings.csv")
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)
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
25/09/24 23:06:39 WARN NativeCodeLoader: Unable to load native-hadoop library
for your platform… using builtin-java classes where applicable
25/09/24 23:06:53 WARN GarbageCollectionMetrics: To enable non-built-in garbage
collector(s) List(G1 Concurrent GC), users should configure it(them) to
```

```
spark.eventLog.gcMetrics.youngGenerationGarbageCollectors or
spark.eventLog.gcMetrics.oldGenerationGarbageCollectors
25/09/24 23:06:59 WARN SparkStringUtils: Truncated the string representation of
a plan since it was too large. This behavior can be adjusted by setting
'spark.sql.debug.maxToStringFields'.
```

[5]:

```python
#| eval: true

# Step 1: Create Companies Table (Primary Key: company_id)
companies_df = df.select(
    col("company"),
    col("company_name"),
    col("company_raw"),
    col("company_is_staffing")
).distinct().withColumn("company_id", monotonically_increasing_id())

companies_df.show(5)

companies = companies_df.toPandas()
companies.drop(columns=["company"], inplace=True)
companies.rename(columns={"company_is_staffing": "is_staffing"}, inplace=True)
companies.to_csv("./output/companies.csv", index=False)

companies.head()
```

```
+--------+--------------------+--------------------+-------------------+--------
--+
| company|        company_name|
company_raw|company_is_staffing|company_id|
+--------+--------------------+--------------------+-------------------+--------
--+
|35247729|               Crowe|               Crowe|              false|
0|
|39461288|The Devereux Foun…|The Devereux Foun…|              false|
1|
|40008275|      Elder Research|      Elder Research|              false|
2|
|37060425|            NTT DATA|        NTT DATA Inc|              false|
3|
|40882284|Frederick Nationa…|Frederick Nationa…|              false|
4|
+--------+--------------------+--------------------+-------------------+--------
--+
only showing top 5 rows
```

[5]:

```
                                    company_name  \
0                                          Crowe
1                        The Devereux Foundation
2                                 Elder Research
3                                       NTT DATA
4  Frederick National Laboratory For Cancer Research

                                     company_raw is_staffing  company_id
0                                          Crowe       False           0
1                        The Devereux Foundation       False           1
2                                 Elder Research       False           2
3                                   NTT DATA Inc       False           3
4  Frederick National Laboratory for Cancer Research       False           4
```

DATA CLEANING

[ ]:

```python
# Step 1: Casting salary and experience columns
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, median_salary)

# 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)
```

```python
# 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 cleansing complete. Rows retained:", len(pdf))
```

```
Medians: 87295.0 130042.0 115024.0
```

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

[7]:

```python
print(df.columns)
```

```
['ID', 'LAST_UPDATED_DATE', 'LAST_UPDATED_TIMESTAMP', 'DUPLICATES', 'POSTED',
'EXPIRED', 'DURATION', 'SOURCE_TYPES', 'SOURCES', 'URL', 'ACTIVE_URLS',
'ACTIVE_SOURCES_INFO', 'TITLE_RAW', 'BODY', 'MODELED_EXPIRED',
'MODELED_DURATION', 'COMPANY', 'COMPANY_NAME', 'COMPANY_RAW',
'COMPANY_IS_STAFFING', 'EDUCATION_LEVELS', 'EDUCATION_LEVELS_NAME',
'MIN_EDULEVELS', 'MIN_EDULEVELS_NAME', 'MAX_EDULEVELS', 'MAX_EDULEVELS_NAME',
'EMPLOYMENT_TYPE', 'EMPLOYMENT_TYPE_NAME', 'MIN_YEARS_EXPERIENCE',
'MAX_YEARS_EXPERIENCE', 'IS_INTERNSHIP', 'SALARY', 'REMOTE_TYPE',
'REMOTE_TYPE_NAME', 'ORIGINAL_PAY_PERIOD', 'SALARY_TO', 'SALARY_FROM',
'LOCATION', 'CITY', 'CITY_NAME', 'COUNTY', 'COUNTY_NAME', 'MSA', 'MSA_NAME',
'STATE', 'STATE_NAME', 'COUNTY_OUTGOING', 'COUNTY_NAME_OUTGOING',
'COUNTY_INCOMING', 'COUNTY_NAME_INCOMING', 'MSA_OUTGOING', 'MSA_NAME_OUTGOING',
'MSA_INCOMING', 'MSA_NAME_INCOMING', 'NAICS2', 'NAICS2_NAME', 'NAICS3',
'NAICS3_NAME', 'NAICS4', 'NAICS4_NAME', 'NAICS5', 'NAICS5_NAME', 'NAICS6',
'NAICS6_NAME', 'TITLE', 'TITLE_NAME', 'TITLE_CLEAN', 'SKILLS', 'SKILLS_NAME',
'SPECIALIZED_SKILLS', 'SPECIALIZED_SKILLS_NAME', 'CERTIFICATIONS',
'CERTIFICATIONS_NAME', 'COMMON_SKILLS', 'COMMON_SKILLS_NAME', 'SOFTWARE_SKILLS',
'SOFTWARE_SKILLS_NAME', 'ONET', 'ONET_NAME', 'ONET_2019', 'ONET_2019_NAME',
'CIP6', 'CIP6_NAME', 'CIP4', 'CIP4_NAME', 'CIP2', 'CIP2_NAME', 'SOC_2021_2',
'SOC_2021_2_NAME', 'SOC_2021_3', 'SOC_2021_3_NAME', 'SOC_2021_4',
'SOC_2021_4_NAME', 'SOC_2021_5', 'SOC_2021_5_NAME', 'LOT_CAREER_AREA',
'LOT_CAREER_AREA_NAME', 'LOT_OCCUPATION', 'LOT_OCCUPATION_NAME',
```

```
'LOT_SPECIALIZED_OCCUPATION', 'LOT_SPECIALIZED_OCCUPATION_NAME',
'LOT_OCCUPATION_GROUP', 'LOT_OCCUPATION_GROUP_NAME',
'LOT_V6_SPECIALIZED_OCCUPATION', 'LOT_V6_SPECIALIZED_OCCUPATION_NAME',
'LOT_V6_OCCUPATION', 'LOT_V6_OCCUPATION_NAME', 'LOT_V6_OCCUPATION_GROUP',
'LOT_V6_OCCUPATION_GROUP_NAME', 'LOT_V6_CAREER_AREA', 'LOT_V6_CAREER_AREA_NAME',
'SOC_2', 'SOC_2_NAME', 'SOC_3', 'SOC_3_NAME', 'SOC_4', 'SOC_4_NAME', 'SOC_5',
'SOC_5_NAME', 'LIGHTCAST_SECTORS', 'LIGHTCAST_SECTORS_NAME', 'NAICS_2022_2',
'NAICS_2022_2_NAME', 'NAICS_2022_3', 'NAICS_2022_3_NAME', 'NAICS_2022_4',
'NAICS_2022_4_NAME', 'NAICS_2022_5', 'NAICS_2022_5_NAME', 'NAICS_2022_6',
'NAICS_2022_6_NAME', 'Average_Salary']
```

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.

[9]:
```python
# Filter salaries
salary_df = df.filter((F.col("SALARY_FROM").isNotNull()) & (F.
 ↪col("SALARY_FROM") > 0))

# Convert to Pandas for plotting
salary_pd = salary_df.select("NAICS_2022_2_NAME", "EMPLOYMENT_TYPE_NAME",␣
 ↪"SALARY_FROM").toPandas()

# Plot
fig2 = px.box(
    salary_pd,
    x="NAICS_2022_2_NAME",
    y="SALARY_FROM",
    color="EMPLOYMENT_TYPE_NAME",
    title="Salary Distribution by Industry and Employment Type",
)

fig2.update_layout(
    template="plotly_white",
    title_font=dict(size=22, family="Helvetica", color="black"),
    font=dict(size=14, family="Helvetica", color="black"),
    xaxis_title="Industry",
    yaxis_title="Salary (USD)"
)

fig2.show()
```

The plot shows that there is a wide variation in salaries across different industries. The salaries are much higher in industries such as Information, Professional, Scientific and Technical Services. The full-time jobs pay more than part-time jobs.

3 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.

[28]:
```python
import plotly.express as px
import pandas as pd


data = {
    "Occupation_Name": [
        "Data / Data Mining Analyst",
        "Business Intelligence Analyst",
        "Computer Systems Engineer / Architect",
        "Business / Management Analyst",
        "Clinical Analyst / Clinical Documentation and ..."
    ],
    "Median_Salary": [95250.0, 125900.0, 157600.0, 93650.0, 89440.0],
    "Job_Postings": [30057, 29445, 8212, 4326, 261]
}

df = pd.DataFrame(data)

# Create bubble chart
fig = px.scatter(
    df,
    x="Occupation_Name",
    y="Median_Salary",
    size="Job_Postings",
    color="Job_Postings",
    hover_name="Occupation_Name",
    title="Salary Analysis by ONET Occupation Type (Bubble Chart)",
    color_continuous_scale="Viridis",
    size_max=60
)

fig.update_layout(
    template="plotly_white",
    title_font=dict(size=22, family="Helvetica", color="black"),
    font=dict(size=12, family="Helvetica", color="black"),
    xaxis_title="ONET Occupation",
    yaxis_title="Median Salary (USD)",
    xaxis_tickangle=-30
```

```
)

fig.show()
```

The bubble chart shows that Computer Systems Engineer/ Architect have the highest median salary amongst all the other occupations shown above. One drawback is that there job postings are less than as compared to Data Mining and Business Intelligence Analysts job postings.

4 Salary by Education Level

Create two groups: Associate's or lower (GED, Associate, No Education Listed) Bachelor's (Bachelor's degree) Master's (Master's degree) PhD (PhD, Doctorate, professional degree) Plot scatter plots for each group using, MAX_YEARS_EXPERIENCE (with jitter), Average_Salary, LOT_V6_SPECIALIZED_OCCUPATION_NAME After each graph, add a short explanation of key insights.

[30]:
```
import plotly.express as px
import pandas as pd



data = {
    "MAX_EXPERIENCE": [2.0, 3.0, 7.0, 2.0, 5.0],
    "AVERAGE_SALARY": [108668.5, 108668.5, 108668.5, 92962.0, 108668.5],
    "EDU_GROUP": ["Bachelor", "Associate or Lower", "Associate or Lower",␣
 ↪"Bachelor", "Associate or Lower"],
    "OCCUPATION_NAME": [
        "General ERP Analyst / Consultant",
        "Oracle Consultant / Analyst",
        "General ERP Analyst / Consultant",
        "Data Analyst",
        "Data Analyst"
    ]
}

df = pd.DataFrame(data)

# Scatter plot
fig = px.scatter(
    df,
    x="MAX_EXPERIENCE",
    y="AVERAGE_SALARY",
    color="EDU_GROUP",
    hover_data=["OCCUPATION_NAME"],
    title="Experience vs Salary by Education Level",
    opacity=0.7,
    color_discrete_sequence=px.colors.qualitative.Set1
)
```

```
fig.update_layout(
    template="plotly_white",
    title_font=dict(size=22, family="Helvetica", color="black"),
    font=dict(size=12, family="Helvetica", color="black"),
    xaxis_title="Years of Experience",
    yaxis_title="Average Salary (USD)"
)

fig.show()
```

The graph shows that the salaries for Bachelor's and Associate or Lower groups are concentrated around $100 thousand. Based on the results of the graph it seems like there is a little difference between the two categories in terms of salary and experience.

5 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.

[34]:
```
import plotly.express as px
import pandas as pd
import numpy as np

data = {
    "MAX_YEARS_EXPERIENCE": [2.0, 3.0, 7.0, 2.0, 5.0],
    "Average_Salary": [108668.5, 108668.5, 108668.5, 92962.0, 108668.5],
    "LOT_V6_SPECIALIZED_OCCUPATION_NAME": [
        "General ERP Analyst / Consultant",
        "Oracle Consultant / Analyst",
        "General ERP Analyst / Consultant",
        "Data Analyst",
        "Data Analyst"
    ],
    "REMOTE_GROUP": ["Onsite", "Remote", "Onsite", "Onsite", "Remote"]
}

df = pd.DataFrame(data)


df["MAX_EXPERIENCE_JITTER"] = df["MAX_YEARS_EXPERIENCE"] + np.random.uniform(-0.
 ↪2, 0.2, size=len(df))

# Scatter plot (Muted Set2 Palette)
fig_set2 = px.scatter(
    df,
    x="MAX_EXPERIENCE_JITTER",
```

```
    y="Average_Salary",
    color="REMOTE_GROUP",
    hover_data=["LOT_V6_SPECIALIZED_OCCUPATION_NAME"],
    title="Experience vs Salary by Remote Work Type",
    opacity=0.7,
    color_discrete_sequence=px.colors.qualitative.Set2
)

fig_set2.show()
```

The graph shows that salaries for both onsite and remote jobs are concentrated between $93k to $108k range. The experience of 2 to 7 years does not show a strong impact on salary and in both the categories there is no significant difference in the salary range.