Assignment 03

Emily Sundberg

September 24, 2025

1 Load the Data Set

```
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"

spark = SparkSession.builder.appName("LightcastData").getOrCreate()

df = spark.read.option("header", "true").option("inferSchema", "true").option("multiLine","t
df.createOrReplaceTempView("job_postings")

#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)
```

```
[Stage 12:>                                                         (0 + 1) / 1]
```

```
+--------------------+----------------+----------------------+----------+--------+---------
|                  ID|LAST_UPDATED_DATE|LAST_UPDATED_TIMESTAMP|DUPLICATES|  POSTED|  EXPIRED
+--------------------+----------------+----------------------+----------+--------+---------
|1f57d95acf4dc67ed...|         9/6/2024|  2024-09-06 20:32:...|         0|6/2/2024| 6/8/2024
|0cb072af26757b6c4...|         8/2/2024|  2024-08-02 17:08:...|         0|6/2/2024| 8/1/2024
|85318b12b3331fa49...|         9/6/2024|  2024-09-06 20:32:...|         1|6/2/2024| 7/7/2024
|1b5c3941e54a1889e...|         9/6/2024|  2024-09-06 20:32:...|         1|6/2/2024|7/20/2024
|cb5ca25f02bdf25c1...|        6/19/2024|   2024-06-19 07:00:00|         0|6/2/2024|6/17/2024
+--------------------+----------------+----------------------+----------+--------+---------
only showing top 5 rows
```

2 Data Cleaning

2.1 Casting salary and experience columns

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

2.2 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_sal = compute_median(df,"SALARY")

print("Medians:", median_from, median_to, median_sal)
```

```
[Stage 14:>                                                         (0 + 1) / 1]

Medians: 87295.0 130042.0 115024.0
```

2.3 Impute missing salaries

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

2.4 Cleaning Education Column

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

2.5 Compute Average Salary

```
df = df.withColumn("AVG_SALARY", (col("SALARY_FROM")+col("SALARY_TO"))/2)
```

2.6 Exporting Cleaned Data

```
df_selected = df.select(
  "EDUCATION_LEVELS_NAME",
  "REMOTE_TYPE_NAME",
  "MAX_YEARS_EXPERIENCE",
  "AVG_SALARY",
  "LOT_V6_SPECIALIZED_OCCUPATION_NAME",
  "NAICS2_NAME")
```

2.7 Save to CSV

```
pdf = df_selected.toPandas()

pdf.to_csv("./data/lightcast_cleaned.csv", index=False)

print("Data Cleaning Complete. Rows retained:", len(pdf))
```

```
[Stage 17:>                                                         (0 + 1) / 1]

Data Cleaning Complete. Rows retained: 72498
```

3 Salary Distribution by Industry and Employment Type

3.1 Remove records where salary is missing or zero

```
pdf = df.filter(df["SALARY"] >0).select("EMPLOYMENT_TYPE_NAME", "SALARY","NAICS2_NAME").toPa

#pdf["EMPLOYMENT_TYPE_NAME"] = pdf["EMPLOYMENT_TYPE_NAME"].apply(lambda x: re.sub(r"[^\x00-\
```

```
[Stage 18:>                                                         (0 + 1) / 1]
```

3.2 Aggregate Data

```
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
)
```

3.3 Visualize Results

```
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

fig = px.box(
  pdf,
  x = "NAICS2_NAME",
```

```
  y = "SALARY",
  color = "EMPLOYMENT_TYPE_NAME",
  title = "Salary Distribution by Employment Type",
  #color_discrete_sequence = ["orange"],
  boxmode = "group",
  points = "all"
)

fig.show()
```

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

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

The plot above shows us that the medians of these industries average around 100K but there are several career fields with the opportunity to make significantly more. Both admin support and information have outliers in the $500K range.

4 Salary Analysis by ONET Occupation Type

4.1 Aggregate Data

```
salary_analysis = spark.sql("""
  SELECT
    TITLE_NAME AS ONET_NAME,
    PERCENTILE(SALARY, 0.5) AS Median_Salary,
    Count(*) AS Job_Postings
  FROM job_postings
  GROUP BY TITLE_NAME
  ORDER BY Job_Postings DESC
  LIMIT 10
""")

salary_pd = salary_analysis.toPandas()
```

```
[Stage 19:>                                                         (0 + 1) / 1]
```

4.2 Bubble Plot

```
fig = px.scatter(
  salary_pd,
  x="ONET_NAME",
  y="Median_Salary",
  size="Job_Postings",
  title ="Salary Analysis by ONET Occupation Type (Bubble Chart)"
)

fig.show()
```

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

This tells us that data analysts have a significant amount of job postings in the less than $100K range.