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 15:>                                                         (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_salary = compute_median(df,"SALARY")

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

```
[Stage 17:>                                                         (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_salary
})
```

2.4 Cleaning Education Column

```
df = df.withColumn("EDUCATION_LEVELS_NAME", regexp_replace(col("EDUCATION_LEVELS_NAME"), "[\r
ed = df.select("EDUCATION_LEVELS_NAME")
ed.show(15)
```

```
+---------------------+
|EDUCATION_LEVELS_NAME|
+---------------------+
| [  "Bachelor's de...|
| [  "No Education ...|
| [  "Bachelor's de...|
| [  "No Education ...|
| [  "No Education ...|
| [  "Bachelor's de...|
| [  "Bachelor's de...|
| [  "Bachelor's de...|
| [  "No Education ...|
| [  "Bachelor's de...|
| [  "High school o...|
| [  "No Education ...|
| [  "Bachelor's de...|
| [  "Bachelor's de...|
| [  "No Education ...|
+---------------------+
only showing top 15 rows
```

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 21:>                                                         (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"].fillna("Unknown")
pdf["EMPLOYMENT_TYPE_NAME"] = pdf["EMPLOYMENT_TYPE_NAME"].apply(lambda x: re.sub(r"[^\x00-\x

pdf.head()
```

```
[Stage 22:>                                                 (0 + 1) / 1]
```

	EMPLOYMENT_TYPE_NAME	SALARY	NAICS2_NAME
0	Full-time (> 32 hours)	115024.0	Retail Trade
1	Full-time (> 32 hours)	115024.0	Administrative and Support and Waste Managemen...
2	Full-time (> 32 hours)	115024.0	Finance and Insurance
3	Full-time (> 32 hours)	115024.0	Finance and Insurance
4	Part-time / full-time	92500.0	Unclassified Industry

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",
  title = "Salary Distribution by NAICS2 Name",
  color_discrete_sequence = ["peru"],
  boxmode = "group",
  points = "outliers"
)
```

```
fig.update_layout(
  margin=dict(t=150),
  width=2000,
  height=800,
  title=dict(
    text = "Salary Distribution By NAICS2 Name",
    font=dict(size=30, family="Montserrat", color = "saddlebrown", weight="bold")
  ),

  xaxis=dict(
    title=dict(text="NAICS2 Name", font=dict(size=14, family="Montserrat", color="saddlebrow
    tickangle=50,
    showline=True,
    linewidth=2,
    linecolor="saddlebrown",
    mirror=True,
    showgrid=False,
    categoryorder="array",
    categoryarray=sorted_employment_types.tolist()
  ),

  yaxis=dict(
    title=dict(text="Salary (K $)", font=dict(size=14, family="Montserrat", color="saddlebro
    tickvals=[0,50000,100000,150000,200000,250000,300000,350000,400000,450000,500000],
    ticktext=["0","50K","100K","150K","200K","250K","300K","350K","400K","450K","500K"],
    tickfont=dict(size=12, family="Montserrat", color="saddlebrown",weight="bold"),
    showline=True,
    linewidth=2,
    linecolor="saddlebrown",
    mirror=True,
    showgrid=True,
    gridcolor="tan",
    gridwidth=0.5
  )
)

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.

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

fig = px.box(
  pdf,
  x = "EMPLOYMENT_TYPE_NAME",
  y = "SALARY",
  title = "Salary Distribution by Employment Type",
  color_discrete_sequence = ["peru"],
  boxmode = "group",
  points = "outliers"
)
```

```
fig.update_layout(
  margin=dict(t=150),
  width=2000,
  height=800,
  title=dict(
    text = "Salary Distribution By Employment Type",
    font=dict(size=30, family="Montserrat", color = "saddlebrown", weight="bold")
  ),

  xaxis=dict(
    title=dict(text="Employment Type", font=dict(size=14, family="Montserrat", color="saddleb
    tickangle=50,
    showline=True,
    linewidth=2,
    linecolor="saddlebrown",
    mirror=True,
    showgrid=False,
    categoryorder="array",
    categoryarray=sorted_employment_types.tolist()
  ),

  yaxis=dict(
    title=dict(text="Salary (K $)", font=dict(size=14, family="Montserrat", color="saddlebrow
    tickvals=[0,50000,100000,150000,200000,250000,300000,350000,400000,450000,500000],
    ticktext=["0","50K","100K","150K","200K","250K","300K","350K","400K","450K","500K"],
```

```
    tickfont=dict(size=12, family="Montserrat", color="saddlebrown",weight="bold"),
    showline=True,
    linewidth=2,
    linecolor="saddlebrown",
    mirror=True,
    showgrid=True,
    gridcolor="tan",
    gridwidth=0.5
  )
)

fig.show()
```

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

4 Salary Analysis by ONET Occupation Type

4.1 Aggregate Data

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

salary_pd = salary_analysis.toPandas()
salary_pd.head()
```

```
[Stage 23:>                                                         (0 + 1) / 1]
```

	Occupation_Name	Median_Salary	Job_Postings
0	Data / Data Mining Analyst	95250.0	30057

	Occupation_Name	Median_Salary	Job_Postings
1	Business Intelligence Analyst	125900.0	29445
2	Computer Systems Engineer / Architect	157600.0	8212
3	Business / Management Analyst	93650.0	4326
4	Clinical Analyst / Clinical Documentation and ...	89440.0	261

4.2 Bubble Plot

```
fig = px.scatter(
  salary_pd,
  x="Occupation_Name",
  y="Median_Salary",
  size="Job_Postings",
  title = "Salary Analysis by Occupation (Bubble Chart)",
  labels={
    "Occupation_Name": "Occupation",
    "Median_Salary" : "Median Salary",
    "Job_Postings" : "Number of Job Postings"
  },
  hover_name = "Occupation_Name",
  size_max=60,
  width=1000,
  height=1000,
  color="Job_Postings",
  color_continuous_scale="Sunsetdark"
)
```

```
fig.update_layout(
  margin=dict(t=150),
  font_family="Montserrat",
  font_size = 14,
  title_font_size = 25,
  xaxis_title = "Occupation",
  yaxis_title = "Median Salary",
  plot_bgcolor = "white",
  xaxis=dict(
    tickangle=-45,
    showline=True,
    linecolor="black"
  ),
```

```
  yaxis=dict(
    showline=True,
    linecolor="black"
  )
)

fig.show()
```

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

This figure shows us that, although data analysts and business analysts have roughly the same median salary, data analysts have far more job postings than their business analyst counterparts.

5 Salary by Education Level

5.1 Create Education Groups

```
lower_degree = ["Bachelor's", "Associate","GED", "No Education Listed", "High School"]

higher_degree = ["Master's Degree", "PhD or professional degree"]

df = df.withColumn(
  "EDUCATION_GROUP",
  when(col("EDUCATION_LEVELS_NAME").rlike("|".join([f"(?i){deg}"for deg in lower_degree])), '
  .when(col("EDUCATION_LEVELS_NAME").rlike("|".join([f"(?i){deg}"for deg in higher_degree]))
  .otherwise("Other")
)

df = df.filter(
  col("MAX_YEARS_EXPERIENCE").isNotNull() &
  col("AVG_SALARY").isNotNull() &
  (col("MAX_YEARS_EXPERIENCE")>0) &
  (col("AVG_SALARY") > 0)
)

df_filtered = df.filter(col("EDUCATION_GROUP").isin("Bachelor's or lower","Master's or PhD"))

df_pd = df_filtered.toPandas()
```

```
[Stage 26:>                                                    (0 + 1) / 1]
```

5.2 Scatter Plot

```
fig = px.scatter(
  df_pd,
  x = "MAX_YEARS_EXPERIENCE",
  y = "AVG_SALARY",
  color = "EDUCATION_GROUP",
  hover_data=["LOT_V6_SPECIALIZED_OCCUPATION_NAME"],
  title = "Experience vs Salary by Education Level",
  opacity=0.7,
  color_discrete_sequence= ["palevioletred", "mediumseagreen"]
)

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

fig.update_layout(
  plot_bgcolor = "papayawhip",
  font=dict(family = "Montserrat", size = 14),
  title_font = dict(size = 22, weight = "bold"),
  xaxis_title = "Years of Experience",
  yaxis_title = "Average Salary (USD)",
  legend_title = "Education Group",
  hoverlabel = dict(bgcolor = "white", font_size = 13, font_family = "Montserrat"),
  margin=dict(t=70, l=60, r=60),
  xaxis=dict(
    gridcolor="peru",
    tickmode = 'linear',
    dtick=1
  ),
  yaxis=dict(gridcolor = "peru")
)

fig.show()
```

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

This graph is showing that as years of experience increases, the floor for average salary also increases. Additionally, it shows that Bachelor's or lower appears to have significantly more outliers while Master's or PhD is more consistent.

6 Salary By Remote Work Type

6.1 Split Work Type Groups

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

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("AVG_SALARY").isNotNull() &
  (col("MAX_YEARS_EXPERIENCE") > 0) &
  (col("AVG_SALARY") > 0 )
)

df_pd = df.select("MAX_YEARS_EXPERIENCE", "AVG_SALARY", "LOT_V6_SPECIALIZED_OCCUPATION_NAME"
```

```
[Stage 27:>                                                         (0 + 1) / 1]
```

6.2 Scatter Plot

```
df_pd["MAX_EXPERIENCE_JITTER"] = df_pd["MAX_YEARS_EXPERIENCE"] + np.random.uniform(-0.3, 0.3
df_pd["AVG_SALARY_JITTER"] = df_pd["AVG_SALARY"] + np.random.uniform(-1500, 1500, size = len

fig = px.scatter(
  df_pd,
  width = 2000,
  height = 800,
  x = "MAX_EXPERIENCE_JITTER",
  y = "AVG_SALARY_JITTER",
  color = "REMOTE_GROUP",
  hover_data=["LOT_V6_SPECIALIZED_OCCUPATION_NAME"],
  title = "Experience vs Salary by Remote Work Type",
```

```
  opacity = 0.7,
  color_discrete_sequence=["palevioletred", "mediumseagreen","lightblue"]
)

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

fig.update_layout(
  plot_bgcolor = "papayawhip",
  font=dict(family = "Montserrat", size = 14),
  title_font = dict(size = 22, weight = "bold"),
  xaxis_title = "Years of Experience",
  yaxis_title = "Average Salary (USD)",
  legend_title = "Remote Work Type",
  hoverlabel = dict(bgcolor = "white", font_size = 13, font_family = "Montserrat"),
  margin=dict(t=70, l=60, r=60),
  xaxis=dict(
    gridcolor="peru",
    tickmode = 'linear',
    dtick=1
  ),
  yaxis=dict(gridcolor = "peru")
)

fig.show()
```

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

This graph shows us that there isn't a significant difference in average salary across the three different remote work types but those outliers with the highest paying job postings are almost entirely onsite.

6.3 Histograms

6.3.1 Onsite

```
fig = px.histogram(
    df_pd[df_pd["REMOTE_GROUP"] == "Onsite"],
    x="AVG_SALARY",
    nbins=20,
```

```
    title=f"Salary Distribution - Onsite",
    color_discrete_sequence=["palevioletred"]
)

fig.update_layout(
    plot_bgcolor="papayawhip",
    xaxis_title="Average Salary (USD)",
    yaxis_title="Count of Job Postings",
    font=dict(family="Montserrat", size=14)
)
fig.show()
```

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

This histogram demonstrates that onsite job postings have an average salary of just over $100K with a few outliers ranging from roughly $300K - $800K.

6.3.2 Remote

```
fig = px.histogram(
    df_pd[df_pd["REMOTE_GROUP"] == "Remote"],
    x="AVG_SALARY",
    nbins=20,
    title=f"Salary Distribution - Remote",
    color_discrete_sequence=["mediumseagreen"]
)

fig.update_layout(
    plot_bgcolor="papayawhip",
    xaxis_title="Average Salary (USD)",
    yaxis_title="Count of Job Postings",
    font=dict(family="Montserrat", size=14)
)
fig.show()
```

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

This histogram paints a similar picture as the onsite histogram with the average salary being just over $100K. The cap for remote positions however seems to be around $300K.

6.3.3 Hybrid

```
fig = px.histogram(
    df_pd[df_pd["REMOTE_GROUP"] == "Hybrid"],
    x="AVG_SALARY",
    nbins=20,
    title=f"Salary Distribution - Hybrid",
    color_discrete_sequence=["lightblue"]
)

fig.update_layout(
    plot_bgcolor="papayawhip",
    xaxis_title="Average Salary (USD)",
    yaxis_title="Count of Job Postings",
    font=dict(family="Montserrat", size=14)
)
fig.show()
```

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

The Hybrid histogram looks very similar to the remote histogram with less lucrative outliers but there is a large difference in the drastic decline in number of job postings after ~$125K. It appears that after that point (or so), the number of job postings dwindles rapidly. It's also worth noting that there are significantly less overall job postings for hybrid postions than fully remote or fully onsite.