Assignment 03

Julio Vargas

September 21, 2025

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as F
from pyspark.sql.functions import col, split, explode, regexp_replace, transform, when
from pyspark.sql.functions import col, monotonically_increasing_id
from pyspark.sql.types import StructType  # to/from JSON

import json
import re
import numpy as np
import pandas as pd

import plotly.express as px
import plotly.io as pio
import plotly.graph_objects as go


np.random.seed(30)  # set a fixed seed for reproducibility
pio.renderers.default = "vscode+notebook"   #
# Initialize Spark Session
spark = SparkSession.builder.appName("JobPostingsAnalysis").getOrCreate()
# Load schema from JSON file
with open("data/schema_lightcast.json") as f:
    schema = StructType.fromJson(json.load(f))

# Load Data
df = (spark.read
      .option("header", "true")
      .option("inferSchema", "false")
      .schema(schema)              # saved schema
      .option("multiLine", "true")
      .option("escape", "\"")
```

```
      .csv("data/lightcast_job_postings.csv")
      )

df.createOrReplaceTempView("job_postings")
# Show Schema and Sample Data
#df.printSchema()
df.show(5)
df.count()
```

```
+--------------------+----------------+---------------------+----------
+--------+---------+--------+--------------------+--------------------+--------------------
+----------+------------------+--------------------+--------------------
+---------------+----------------+--------+--------------------+-----------
+-------------------+---------------+--------------------+-------------
+-------------------+-------------+------------------+---------------+--------------------
+--------------------+--------------------+-------------+------+-----------
+----------------+------------------+---------+-----------+--------------------
+--------------------+-------------+------+--------------+-----+--------------------
+-----+----------+---------------+--------------------+---------------+--------------------
+------------+--------------------+------------+--------------------+------
+--------------------+------+--------------------+------+--------------------
+------+--------------------+------+--------------------+------------------
+--------------------+--------------------+--------------------+--------------------
+--------------------+----------------------+--------------------+--------------------
+--------------------+--------------------+--------------------+--------------------
+----------+--------------------+----------+--------------------+--------------------
+--------------------+--------------------+--------------------+--------------------
+--------------------+----------+--------------------+----------+--------------------
+----------+---------------+----------+---------------+---------------+--------------------
+--------------+--------------------+--------------------------+--------------------------
+--------------------+------------------------+----------------------------
+-----------------------------------+-----------------+----------------------
+------------------------+----------------------------+------------------
+-------+---------------+-------+---------------+-----------------+----------------------
+------------+--------------------+------------+--------------------+------------
+--------------------+------------+--------------------+------------+--------------------
+
|                  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
May-2024\n\nEn...|       6/8/2024|               6|  894731|          Murphy USA| Murphy USA
time (> 32 h...|                   2|                   2|        false|  NULL|          0|
2051.01|Business Intellig...|15-2051.01|Business Intellig...|[\n  "45.0601",\n...|[\n  "Econ
0000|Computer and Math...|   15-2000|Mathematical Scie...|   15-2050|Data Scientists|   15-
2051|Data Scientists|             23|Information Techn...|        231010|Business Intellig..
0000|Computer and Math...|15-2000|Mathematical Scie...|15-2050|Data Scientists|15-
2051|Data Scientists|        [\n  7\n]|  [\n  "Artificial ...|          44|        Retail Tra
|0cb072af26757b6c4...|         8/2/2024|  2024-08-02 17:08:...|         0|6/2/2024| 8/1/2024
time (> 32 h...|                   3|                   3|        false|  NULL|          1|
Watervill...|   23|     Maine|          23011|        Kennebec, ME|          23011|        K
Watervill...|       12300|Augusta-Watervill...|    56|Administrative an...|   561|Administrat
2051.01|Business Intellig...|15-2051.01|Business Intellig...|                  []|
0000|Computer and Math...|   15-2000|Mathematical Scie...|   15-2050|Data Scientists|   15-
2051|Data Scientists|             23|Information Techn...|        231010|Business Intellig..
0000|Computer and Math...|15-2000|Mathematical Scie...|15-2050|Data Scientists|15-
2051|Data Scientists|             NULL|                  NULL|          56|Administrative an
```

```
|85318b12b3331fa49...|         9/6/2024|  2024-09-06 20:32:...|         1|6/2/2024| 7/7/2024
time (> 32 h...|                   5|                NULL|        false|  NULL|          0|
Fort Worth...|   48|     Texas|          48113|          Dallas, TX|          48113|
Fort Worth...|       19100|Dallas-Fort Worth...|    52|Finance and Insur...|   524|Insurance
2051.01|Business Intellig...|15-2051.01|Business Intellig...|                  []|
0000|Computer and Math...|   15-2000|Mathematical Scie...|   15-2050|Data Scientists|   15-
2051|Data Scientists|             23|Information Techn...|        231113|Data / Data Minin..
0000|Computer and Math...|15-2000|Mathematical Scie...|15-2050|Data Scientists|15-
2051|Data Scientists|             NULL|                  NULL|          52|Finance and Insur
|1b5c3941e54a1889e...|         9/6/2024|  2024-09-06 20:32:...|         1|6/2/2024|7/20/2024
time (> 32 h...|                   3|                NULL|        false|  NULL|          0|
Mesa-Chan...|    4|   Arizona|           4013|        Maricopa, AZ|           4013|        Ma
Mesa-Chan...|       38060|Phoenix-Mesa-Chan...|    52|Finance and Insur...|   522|Credit Inte
2051.01|Business Intellig...|15-2051.01|Business Intellig...|                  []|
0000|Computer and Math...|   15-2000|Mathematical Scie...|   15-2050|Data Scientists|   15-
2051|Data Scientists|             23|Information Techn...|        231113|Data / Data Minin..
0000|Computer and Math...|15-2000|Mathematical Scie...|15-2050|Data Scientists|15-
2051|Data Scientists|        [\n  6\n]|  [\n  "Data Privac...|          52|Finance and Insur
|cb5ca25f02bdf25c1...|        6/19/2024|   2024-06-19 07:00:00|         0|6/2/2024|6/17/2024
time / full-...|                NULL|                NULL|        false| 92500|          0|
2051.01|Business Intellig...|15-2051.01|Business Intellig...|                  []|
0000|Computer and Math...|   15-2000|Mathematical Scie...|   15-2050|Data Scientists|   15-
2051|Data Scientists|             23|Information Techn...|        231010|Business Intellig..
0000|Computer and Math...|15-2000|Mathematical Scie...|15-2050|Data Scientists|15-
2051|Data Scientists|             NULL|                  NULL|          99|Unclassified Indu
+--------------------+-----------------+----------------------+----------
+--------+---------+--------+--------------------+--------------------+--------------------
+-----------+--------------------+--------------------+--------------------
+---------------+----------------+---------+--------------------+-----------
+-------------------+----------------+----------------------+-------------
+--------------------+-------------+-------------------+--------------+--------------------
+--------------------+--------------------+-------------+------+-----------
+----------------+-------------------+---------+-----------+--------------------
+--------------------+-------------+------+--------------+-----+--------------------
+-----+----------+---------------+--------------------+---------------+--------------------
+------------+--------------------+------------+--------------------+------
+--------------------+------+--------------------+------+--------------------
+------+--------------------+------+--------------------+------------------
+-------------------+--------------------+--------------------+--------------------
+--------------------+----------------------+--------------------+--------------------
+--------------------+--------------------+--------------------+--------------------
+----------+--------------------+----------+--------------------+--------------------
+--------------------+--------------------+--------------------+--------------------
```

```
+--------------------+---------+--------------------+---------+--------------------
+---------+--------------+---------+--------------+--------------+--------------------
+-------------+--------------------+------------------------+--------------------------
+--------------------+------------------------+---------------------------
+-----------------------------------+-----------------+----------------------
+-----------------------+---------------------------+-----------------
+-----------------------+-------+--------------------+-------+--------------------
+-------+--------------+-------+--------------+-----------------+----------------------
+------------+--------------------+------------+--------------------+------------
+--------------------+------------+--------------------+------------+--------------------
+
only showing top 5 rows

[Stage 37:>                                                         (0 + 1) / 1]

72498
```

1 1.1 Casting salary and experience columns

1.0.1 1.1 Computing medians

```
from pyspark.sql.functions import col, regexp_replace, trim
# 1.1 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"))
#df.select("SALARY", "SALARY_FROM", "SALARY_TO", "MIN_YEARS_EXPERIENCE", "MAX_YEARS_EXPERIEN
#df.select("SALARY", "SALARY_FROM", "SALARY_TO", "MIN_YEARS_EXPERIENCE", "MAX_YEARS_EXPERIEN
```

1.0.2 1.2 Computing medians

```
# 1.2 Computing medians
def compute_median(sdf, col_name):
    q = sdf.approxQuantile(col_name, [0.5], 0.01) #50 percentile 1% error
    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")
```

```
[Stage 40:>                                                         (0 + 1) / 1]
```

```
# 1.2 Output
#the median_from, median_to , median_salary respectively are:

print("- Median SALARY_FROM: $" + str(median_from))
print("- Median SALARY_TO: $" + str(median_to))
print("- Median SALARY: $" + str(median_salary))
```

```
- Median SALARY_FROM: $87295.0
- Median SALARY_TO: $130042.0
- Median SALARY: $115024.0
```

```
# 1.3 Imputing missing salaries
df = df.fillna({
    "SALARY_FROM": median_from,
    "SALARY_TO": median_to,
    "SALARY": median_salary
})

# 1.3 Add new column Average_Salary
df = df.withColumn("Average_Salary", (col("SALARY_FROM") + col("SALARY_TO")) / 2)

export_cols = ["Average_Salary","SALARY","EDUCATION_LEVELS_NAME","REMOTE_TYPE_NAME",
          "MAX_YEARS_EXPERIENCE","LOT_V6_SPECIALIZED_OCCUPATION_NAME"]

# 1.3 Output
df.select(*export_cols).show(5, truncate=False)
```

```
+--------------+--------+----------------------------+----------------
+--------------------+-----------------------------------+
```

```
|Average_Salary|SALARY  |EDUCATION_LEVELS_NAME        |REMOTE_TYPE_NAME|MAX_YEARS_EXPERIENCE
+--------------+--------+-----------------------------+----------------
+--------------------+---------------------------------+
|108668.5      |115024.0|[\n  "Bachelor's degree"\n]  |[None]          |2.0
|108668.5      |115024.0|[\n  "No Education Listed"\n]|Remote          |3.0
|108668.5      |115024.0|[\n  "Bachelor's degree"\n]  |[None]          |NULL
|108668.5      |115024.0|[\n  "No Education Listed"\n]|[None]          |NULL
|92500.0       |92500.0 |[\n  "No Education Listed"\n]|[None]          |NULL
+--------------+--------+-----------------------------+----------------
+--------------------+---------------------------------+
only showing top 5 rows
```

```
#1.4 Cleaning Education column
#remove the \n and \r
df1 = df.withColumn("EDUCATION_LEVELS_NAME",
    trim(
        regexp_replace(regexp_replace(col("EDUCATION_LEVELS_NAME"),r"\n|\r", ""), #remove \n
                       r"\[\s+\"", "[\"" ) #remove spaces.
    )
)
# 1.4 Output
df1.select(*export_cols).show(5, truncate=False)
```

```
+--------------+--------+----------------------+----------------+--------------------
+----------------------------------+
|Average_Salary|SALARY  |EDUCATION_LEVELS_NAME  |REMOTE_TYPE_NAME|MAX_YEARS_EXPERIENCE|LOT_V
+--------------+--------+----------------------+----------------+--------------------
+----------------------------------+
|108668.5      |115024.0|["Bachelor's degree"]  |[None]          |2.0                 |Genera
|108668.5      |115024.0|["No Education Listed"]|Remote          |3.0                 |Oracl
|108668.5      |115024.0|["Bachelor's degree"]  |[None]          |NULL                |Data
|108668.5      |115024.0|["No Education Listed"]|[None]          |NULL                |Data
|92500.0       |92500.0 |["No Education Listed"]|[None]          |NULL                |Oracl
+--------------+--------+----------------------+----------------+--------------------
+----------------------------------+
only showing top 5 rows
```

```
#1.5 Exporting Cleaned Data
# Export to CSV
df_selected=df1.select(*export_cols)
pdf = df_selected.toPandas()
pdf.to_csv("data/lightcast_cleaned.csv", index=False)
```

```
print("Data cleaning complete. Rows retained:", len(pdf))
```

```
[Stage 45:>                                                         (0 + 1) / 1]
```

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

2 2.0 TEMPLATE

```
import plotly.graph_objects as go
import plotly.io as pio

pio.templates["nike"] = go.layout.Template(
    # LAYOUT
    layout = {
        # Fonts and colors
        'title': {
            'font': {'family': 'HelveticaNeue-CondensedBold, Helvetica, Sans-serif',
                     'size': 30,
                     'color': '#13007c'}
        },
        'font': {'family': 'Helvetica Neue, Helvetica, Sans-serif',
                 'size': 16,
                 'color': '#3b3b3b'},

        'colorway': ['#fffb00', '#e010fc'],
        # Adding others
        'hovermode': 'x unified',
        'plot_bgcolor': '#E5ECF6',
        'paper_bgcolor': "#FFFFFF",

    },
    # DATA
    data = {
        # Default style applied to all bar charts
        'bar': [go.Bar(
            texttemplate = '%{value:$.2s}',
            textposition = 'outside',
            textfont = {'family': 'Helvetica Neue, Helvetica, Sans-serif',
```

```
                        'size': 20,
                        'color': '#ff6874'} # FFFFFF
        )]
    }
)
```

2.1 2.1 Salary Distribution by Industry and Employment Type

```
#your code for first query
import pandas as pd
import polars as pl
from IPython.display import display, HTML

#2.2 Filter the dataset - Remove records where salary is missing or zero.
df_valid_salaries = df.filter(df["SALARY"] > 0).select("NAICS2_NAME","EMPLOYMENT_TYPE_NAME",

#2.2 output - convert to pandas
pdf = df_valid_salaries.toPandas()
print("Data cleaning complete. Rows retained:", len(pdf))

#2.3 Aggregate data - NAICS industry codes, employment type and compute salary distribution.

# Clean employment type names for better readability
pdf["EMPLOYMENT_TYPE_NAME"] = (pdf["EMPLOYMENT_TYPE_NAME"].astype(str)
                               .str.replace(r"[^\x00-\x7F]+", "", regex=True)
)

#2.3 output
median_salaries_naics = pdf.groupby("NAICS2_NAME")["SALARY"].median()
median_salaries_employee = pdf.groupby("EMPLOYMENT_TYPE_NAME")["SALARY"].median()
display(median_salaries_naics.to_frame().head())
display(median_salaries_employee.to_frame().head())
```

```
[Stage 46:>                                                         (0 + 1) / 1]

Data cleaning complete. Rows retained: 72498
```

NAICS2_NAME	SALARY
Accommodation and Food Services	115024.0
Administrative and Support and Waste Management and Remediation Services	115024.0
Agriculture, Forestry, Fishing and Hunting	115024.0
Arts, Entertainment, and Recreation	115024.0
Construction	115024.0

EMPLOYMENT_TYPE_NAME	SALARY
Full-time (> 32 hours)	115024.0
None	115024.0
Part-time (32 hours)	115024.0
Part-time / full-time	115024.0

```
#2.4 Visualize results  box plot
# X-axis = NAICS2_NAME || Y-axis = SALARY_FROM || Group by EMPLOYMENT_TYPE_NAME.
pdf = df.select("NAICS2_NAME", "SALARY").toPandas()
fig = px.box(pdf, x="NAICS2_NAME", y="SALARY", title="Salary Distribution by Industry",
             color_discrete_sequence=["#EF553B"])
                 # add nike template
#fig.update_layout(template="nike")

#fig.update_xaxes(tickangle=45)

fig.update_layout(
    template="nike",
    height=700,
    xaxis=dict(
        title=dict(text="NAICS2_NAME", standoff=40),
        tickangle=45,
        tickfont=dict(size=10),
        automargin=True
    ),
    yaxis=dict(title=dict(text="Salary")),
    margin=dict(b=150)

)
```

```
fig.show()
```

```
[Stage 47:>                                                         (0 + 1) / 1]
```

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

```
Unable to display output for mime type(s): application/vnd.plotly.v1+json, text/html
```

3 3 Salary Analysis by ONET Occupation Type (Bubble Chart)

```
import plotly.express as px
#3.1 Analyze how salaries differ across LOT_OCCUPATION_NAME occupation types.
#ONET_NAME CHANGE TO LOT_OCCUPATION_NAME

#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
    WHERE LOT_OCCUPATION_NAME IS NOT NULL
    GROUP BY LOT_OCCUPATION_NAME
    ORDER BY Job_Postings DESC
    LIMIT 10
""") #the result only has 6 results and a null, limit to 10 is not necessary

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

#Simple plot to Analyze
figa = px.scatter(
    salary_pd,
    x="Occupation_name",
    y="Median_Salary",
    size="Job_Postings",
    title="Salary Analysis by  Occupation",
```

```
    color="Occupation_name"
)
figa.update_xaxes(tickangle=45, automargin=True)
figa.show()

#3.2 Visualize results  bubble chart
import plotly.express as px

fig = px.scatter(
    salary_pd,
    x="Occupation_name",
    y="Median_Salary",
    size="Job_Postings",
    title="Salary Analysis by LOT Occupation Type (Bubble Chart)",
    labels={
        "Occupation_name": "LOT Occupation",
        "Median_Salary": "Median Salary",
        "Job_Postings": "Number of Job Postings"
    },
    hover_name="Occupation_name",
    size_max=60,
    width=900,
    height=600,
    color="Job_Postings",
    color_continuous_scale="Plasma"
)
#customize layout
fig.update_layout(

    height=700,
    font_family="Arial",
    font_size=14,
    title_font_size=25,
    title_font_color="#13007c",
    font_color="#2e2e2e",
    xaxis_title="LOT Occupation",
    yaxis_title="Median Salary",
    plot_bgcolor="#FAFDFF",
    xaxis=dict(
        tickangle=-45,
        showline=True,
        linecolor="#444"
```

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


fig.show()
fig.write_image("output/Q3.svg", width=1000, height=600, scale=1)
```

```
[Stage 48:>                                                         (0 + 1) / 1]
```

	Occupation_name	Median_Salary	Job_Postings
0	Data / Data Mining Analyst	95250.0	30057
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

```
Unable to display output for mime type(s): application/vnd.plotly.v1+json, text/html
```

```
Unable to display output for mime type(s): application/vnd.plotly.v1+json, text/html
```

```
/tmp/ipykernel_2789/4211918895.py:80: DeprecationWarning:


Support for Kaleido versions less than 1.0.0 is deprecated and will be removed after Septembe
Please upgrade Kaleido to version 1.0.0 or greater (`pip install 'kaleido>=1.0.0'` or `pip in
```

4 4 Salary by Education Level

```
# Defining education level groupings
lower_deg = ["Bachelor's", "Associate", "GED", "No Education Listed", "High school"]
higher_deg = ["Master's degree", "PhD or professional degree"]
```

```
# Adding new column EDU_GROUP
df = df.withColumn(
    "EDU_GROUP",
    when(col("EDUCATION_LEVELS_NAME").rlike("|".join([f"(?i){deg}" for deg in lower_deg])), '
    .when(col("EDUCATION_LEVELS_NAME").rlike("|".join([f"(?i){deg}" for deg in higher_deg]))
    .otherwise("Other")
)

# Modyfying/Casting necessary columns to float
df = df.withColumn("MAX_YEARS_EXPERIENCE", col("MAX_YEARS_EXPERIENCE").cast("float"))
df = df.withColumn("Average_Salary", col("Average_Salary").cast("float"))

# df.select("MAX_YEARS_EXPERIENCE","Average_Salary","EDU_GROUP","EDUCATION_LEVELS_NAME").prin

# print(df.count()) #Total 72,498 after 8074

# Filtering for non-null and positive values
df = df.filter(
    col("MAX_YEARS_EXPERIENCE").isNotNull() & col("Average_Salary").isNotNull() &
    (col("MAX_YEARS_EXPERIENCE") > 0) &  (col("Average_Salary") > 0)
)

# Filtering for just the two EDU_GROUP groups
df_filtered = df.filter(col("EDU_GROUP").isin("Bachelor's or lower", "Master's or PhD"))

# Converting to Pandas for plotting
df_pd = df_filtered.toPandas()
pdf4=df.select("MAX_YEARS_EXPERIENCE","Average_Salary","EDU_GROUP","EDUCATION_LEVELS_NAME").t
display(pdf4.head())
```

```
[Stage 51:>                                                         (0 + 1) / 1]
```

	MAX_YEARS_EXPERIENCE	Average_Salary	EDU_GROUP	EDUCATION_LEVELS_NAM
0	2.0	108668.5	Bachelor's or lower	[\n "Bachelor's degree"\n]
1	3.0	108668.5	Bachelor's or lower	[\n "No Education Listed"\n]
2	7.0	108668.5	Bachelor's or lower	[\n "No Education Listed"\n]
3	2.0	92962.0	Bachelor's or lower	[\n "Bachelor's degree",\n "Mas
4	5.0	108668.5	Bachelor's or lower	[\n "Associate degree",\n "Bach

```
# Jittering and trimming
df_pd["MAX_EXPERIENCE_JITTER"] = df_pd["MAX_YEARS_EXPERIENCE"] + np.random.uniform(-0.25, 0.
df_pd["AVERAGE_SALARY_JITTER"] = df_pd["Average_Salary"] + np.random.uniform(-2500, 2500, siz
df_pd = df_pd.round(2)

# Remove outlier higher than 399K
df_pd = df_pd[df_pd["AVERAGE_SALARY_JITTER"] <= 399000]

df_pd.head()
```

	ID	LAST_UPDATED_DATE	LAST_UPDATED_TIMEST
0	1f57d95acf4dc67ed2819eb12f049f6a5c11782c	9/6/2024	2024-09-06 20:32:57.352
1	0cb072af26757b6c4ea9464472a50a443af681ac	8/2/2024	2024-08-02 17:08:58.838
2	5a843df632e1ff756fa19d80a0871262d51becc0	6/21/2024	2024-06-21 07:00:00.000
3	229620073766234e814e8add21db7dfaef69b3bd	10/9/2024	2024-10-09 18:07:44.758
4	138ce2c9453b47a9b33403c364d4fd80996caa4f	8/10/2024	2024-08-10 19:36:49.244

```
#jittering and triming
# Plot four groups
fig1 = px.scatter(
    df_pd,
    x="MAX_EXPERIENCE_JITTER",
    y="AVERAGE_SALARY_JITTER",
    color="EDU_GROUP",
    hover_data=["LOT_V6_SPECIALIZED_OCCUPATION_NAME"],
    title="<b>Experience vs Salary by Education Level</b>",
    opacity=1.0, #0.7
    color_discrete_sequence=[
        "#636EFA",  # Blue
        "#EF553B",  # Red
        "#00CC96",  # Green
        "#AB63FA"   # Purple
    ]
)

fig1.update_traces(
    marker=dict(size=10, line=dict(width=1, color="black"))
)

fig1.update_layout(
```

```
    plot_bgcolor="#fcfcf0",   # light grey chart background
    paper_bgcolor="#f5d9b2",  # soft blue frame
    font=dict(family="Segoe UI", size=14, color="#2b2b2b"),
    title_font=dict(size=22, color="#4b3832"),
    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, b=60, l=60, r=60),
    xaxis=dict(
        gridcolor="#e0e0e0",
        tickmode="linear",
        dtick=1  # show every integer year clearly
    ),
    yaxis=dict(gridcolor="#cccccc")
)

fig1.show()
fig1.write_html("output/q_1a_Experience_vs_Salary_by_Education_Level.html")
```

```
Unable to display output for mime type(s): application/vnd.plotly.v1+json, text/html
```

5 5 Salary by Remote Work Type

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

#5.1 Split into three groups based on REMOTE_TYPE_NAME
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")
)

#print(df.count())

#5.1 Filter valid values
```

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

#5.1 Pandas
df_pd = df.select(
    "MAX_YEARS_EXPERIENCE","Average_Salary","LOT_V6_SPECIALIZED_OCCUPATION_NAME","REMOTE_GROU
    ).toPandas()

df_pd.head()

# Jittering and trimming
df_pd["MAX_EXPERIENCE_JITTER"] = df_pd["MAX_YEARS_EXPERIENCE"] + np.random.uniform(-0.15, 0.1
df_pd["AVERAGE_SALARY_JITTER"] = df_pd["Average_Salary"] + np.random.uniform(-1000, 1000, siz
df_pd = df_pd.round(2)

# Remove outlier higher than 399K
df_pd = df_pd[df_pd["AVERAGE_SALARY_JITTER"] <= 399000]
```

```
[Stage 53:>                                                         (0 + 1) / 1]
```

```
# Plot four groups
fig5 = 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=1.0, #0.7
    color_discrete_sequence=[
        "#636EFA",  # Blue
        "#EF553B",  # Red
        "#00CC96",  # Green
        "#AB63FA"   # Purple
    ]
)

fig5.update_traces(
    marker=dict(size=10, line=dict(width=1, color="black"))
```

```
)

fig5.update_layout(
    plot_bgcolor="#fcfcf0",   # light grey chart background
    paper_bgcolor="#f5d9b2",  # soft blue frame
    font=dict(family="Segoe UI", size=14, color="#2b2b2b"),
    title_font=dict(size=22, color="#4b3832"),
    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, b=60, l=60, r=60),
    xaxis=dict(
        gridcolor="#e0e0e0",
        tickmode="linear",
        dtick=1  # show every integer year clearly
    ),
    yaxis=dict(gridcolor="#cccccc")
)

fig5.show()
```

```
Unable to display output for mime type(s): application/vnd.plotly.v1+json, text/html
```