Assignment 03 AI Prompts

Ava Godsy

September 22, 2025

# 1. Cleaning Education column

## 1.1 Prompt

using python remove “/n” from values in column EDUCATION\_LEVELS\_NAME in dataframe “df”

## 1.2 Response

To remove “” (newline characters) from the values in the column EDUCATION\_LEVELS\_NAME in a DataFrame df, you can use the .str.replace() method from pandas.

Here’s the code:

df['EDUCATION\_LEVELS\_NAME'] = df['EDUCATION\_LEVELS\_NAME'].str.replace('\n', '', regex=False)

Explanation:

str.replace(‘’, ’‘, regex=False) replaces all newline characters () with an empty string (’’), effectively removing them.

regex=False ensures that is treated as a literal string (which is fine in this case since is a special character anyway).

If you also want to remove any leading or trailing spaces (common after removing ), you can chain .str.strip():

df['EDUCATION\_LEVELS\_NAME'] = df['EDUCATION\_LEVELS\_NAME'].str.replace('\n', '', regex=False).str.strip()

## 1.3 Additional Prompt

i got an error TypeError: ‘Column’ object is not callable

## 1.4 Additional Response

Thanks — the error:

TypeError: 'Column' object is not callable

means you’re likely using PySpark rather than pandas. In PySpark, DataFrame columns are represented by Column objects, and they behave differently than pandas Series.

✅ Fix for PySpark

If you’re using PySpark and want to remove from a column (like EDUCATION\_LEVELS\_NAME), use withColumn() and regexp\_replace() from pyspark.sql.functions.

Here’s the correct code:

from pyspark.sql.functions import regexp\_replace  
  
df = df.withColumn("EDUCATION\_LEVELS\_NAME", regexp\_replace("EDUCATION\_LEVELS\_NAME", "\n", ""))

# 2. Salary Distribution by Industry and Employment Type

## 2.1 Prompt

Using Plotly, Create a box plot where: X-axis = NAICS2\_NAME Y-axis = SALARY\_FROM Group by EMPLOYMENT\_TYPE\_NAME. dataframe = data\_selected

## 2.2 Response

Here’s how you can create a box plot using Plotly in Python, with:

X-axis = NAICS2\_NAME

Y-axis = SALARY\_FROM

Grouped by = EMPLOYMENT\_TYPE\_NAME

Dataframe = data\_selected

✅ Code using Plotly Express:

import plotly.express as px  
  
fig = px.box(  
 data\_selected,  
 x="NAICS2\_NAME",  
 y="SALARY\_FROM",  
 color="EMPLOYMENT\_TYPE\_NAME",  
 title="Salary Distribution by Industry and Employment Type",  
 points="all", # Optional: shows all data points  
 notched=True # Optional: adds notches to box for median CI  
)  
  
fig.update\_layout(  
 xaxis\_title="Industry (NAICS2)",  
 yaxis\_title="Starting Salary",  
 boxmode="group", # Groups boxes side by side  
 xaxis\_tickangle=45 # Rotate x-axis labels for better readability  
)  
  
fig.show()

## 2.3 Additional Prompt

make the height taller, change the colors to purple, blue and green. make the font garamond

## 2.4 Additional Response

You can customize the height and colors of your Plotly box plot using the height parameter and the color\_discrete\_sequence argument.

Here’s the updated code with:

Increased height

Custom colors: purple, blue, green

To set the font to Garamond in your Plotly chart, you can specify it in the update\_layout() method using the font parameter.

✅ Final Updated Code

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=700, # Taller figure  
 color\_discrete\_sequence=["purple", "blue", "green"] # Custom colors  
)  
  
fig.update\_layout(  
 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=14 # Optional: adjust font size  
 )  
)  
  
fig.show()

# 3. Salary Analysis by ONET Occupation Type Table

## 3.1 Prompt

Using Python, create a table of median “SALARY” by ONET\_NAME. also include a count of records for each ONET\_NAME, call the column Job\_Postings

## 3.2 Response

To create a table of the median “SALARY” grouped by “ONET\_NAME”, and include a count of records (as “Job\_Postings”), you can use pandas in Python. Here’s how you can do it:

✅ Sample Code

import pandas as pd  
  
# Sample: Load your DataFrame  
# df = pd.read\_csv('your\_data.csv') # Or however your DataFrame is created  
  
# Group by ONET\_NAME, calculate median salary and count of job postings  
result = df.groupby('ONET\_NAME').agg(  
 SALARY\_Median=('SALARY', 'median'),  
 Job\_Postings=('SALARY', 'count')  
).reset\_index()  
  
# Optional: Sort by Job\_Postings or SALARY\_Median if needed  
# result = result.sort\_values(by='Job\_Postings', ascending=False)  
  
# Display the result  
print(result)

NameError: name 'df' is not defined  
[31m---------------------------------------------------------------------------[39m  
[31mNameError[39m Traceback (most recent call last)  
[36mCell[39m[36m [39m[32mIn[2][39m[32m, line 10[39m  
[32m 4[39m [38;5;28;01mimport[39;00m[38;5;250m [39m[34;01mpandas[39;00m[38;5;250m [39m[38;5;28;01mas[39;00m[38;5;250m [39m[34;01mpd[39;00m  
[32m 6[39m [38;5;66;03m# Sample: Load your DataFrame[39;00m  
[32m 7[39m [38;5;66;03m# df = pd.read\_csv('your\_data.csv') # Or however your DataFrame is created[39;00m  
[32m 8[39m   
[32m 9[39m [38;5;66;03m# Group by ONET\_NAME, calculate median salary and count of job postings[39;00m  
[32m---> [39m[32m10[39m result = [43mdf[49m.groupby([33m'[39m[33mONET\_NAME[39m[33m'[39m).agg(  
[32m 11[39m SALARY\_Median=([33m'[39m[33mSALARY[39m[33m'[39m, [33m'[39m[33mmedian[39m[33m'[39m),  
[32m 12[39m Job\_Postings=([33m'[39m[33mSALARY[39m[33m'[39m, [33m'[39m[33mcount[39m[33m'[39m)  
[32m 13[39m ).reset\_index()  
[32m 15[39m [38;5;66;03m# Optional: Sort by Job\_Postings or SALARY\_Median if needed[39;00m  
[32m 16[39m [38;5;66;03m# result = result.sort\_values(by='Job\_Postings', ascending=False)[39;00m  
[32m 17[39m   
[32m 18[39m [38;5;66;03m# Display the result[39;00m  
[32m 19[39m [38;5;28mprint[39m(result)  
  
[31mNameError[39m: name 'df' is not defined

## 3.3 Additional Prompt

I got the error TypeError: GroupedData.agg() got an unexpected keyword argument ‘SALARY\_Median’ ## Additional Response

Ah, the error message you’re getting:

TypeError: GroupedData.agg() got an unexpected keyword argument 'SALARY\_Median'

suggests you’re not using pandas, but PySpark instead. In PySpark, the syntax for .agg() is different — it doesn’t accept keyword arguments with new column names directly.

✅ Correct PySpark Version

Here’s how you can do what you need in PySpark:

from pyspark.sql import functions as F  
  
# Group by ONET\_NAME and calculate median salary and count of postings  
  
# First, compute Job\_Postings (count) and SALARY\_Median (approximate median using percentile\_approx)  
result = df.groupBy("ONET\_NAME").agg(  
 F.expr("percentile\_approx(SALARY, 0.5)").alias("SALARY\_Median"),  
 F.count("\*").alias("Job\_Postings")  
)  
  
# Show the result  
result.show()

# 4. Salary Analysis by ONET Occupation Type Plot

## 4.1 Prompt

Using Plotly Create a bubble chart where dataframe = onet\_salary X-axis = ONET\_NAME Y-axis = Median Salary Size = Job\_Postings Apply custom colors and font styles.

## 4.2 Response

ChatGPT said:

Here’s how you can create a bubble chart using Plotly with the following settings:

DataFrame: onet\_salary

X-axis: ONET\_NAME

Y-axis: Median Salary

Bubble Size: Job\_Postings

Custom Colors & Font Styles: Applied using Plotly’s update\_layout

✅ Prerequisites

Ensure you have Plotly installed:

pip install plotly

✅ Sample Code

import plotly.express as px  
  
# Sample usage (assumes 'onet\_salary' is a DataFrame with relevant columns)  
fig = px.scatter(  
 onet\_salary,  
 x="ONET\_NAME",  
 y="Median Salary",  
 size="Job\_Postings",  
 color="Median Salary", # You can change this to another column if desired  
 hover\_name="ONET\_NAME",  
 size\_max=60,  
 title="Bubble Chart of ONET Median Salaries vs Job Postings",  
)  
  
# Customize layout: fonts, colors, background  
fig.update\_layout(  
 title\_font=dict(family="Arial Black", size=24, color="darkblue"),  
 font=dict(family="Verdana", size=14, color="black"),  
 plot\_bgcolor="white",  
 paper\_bgcolor="#f7f7f7",  
 xaxis=dict(title="ONET Job Titles", tickangle=45),  
 yaxis=dict(title="Median Salary ($)", gridcolor="#e5e5e5"),  
)  
  
# Show the figure  
fig.show()

# 5. Salary by Education Level Table

## 5.1 Prompt

Create a table using pyspark from dataframe df\_selected where there is an EDU\_GROUP column, which is a grouping of column “EDUCATION\_LEVELS\_NAME”. If EDUCATION\_LEVELS\_NAME contains No Education Listed, GED, or Associate degree then “Associate’s or lower”. If EDUCATION\_LEVELS\_NAME contains Bachelor’s degree then Bachelor’s. If EDUCATION\_LEVELS\_NAME contains Master’s degree then Master’s. If EDUCATION\_LEVELS\_NAME contains Ph.D. or professional degree then PhD. ALSO Include LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME Average\_Salary and MAX\_YEARS\_EXPERIENCE in the table

## 5.2 Response

To create a table in PySpark from df\_selected with the specified logic for EDU\_GROUP, and include the requested columns, follow these steps:

✅ Steps Overview

Define EDU\_GROUP using when and col from pyspark.sql.functions

Select required columns

Create a temporary view or save as a table if needed

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"  
 ).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("Other") # 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"  
)  
  
# Optional: Create a temporary view or save as a table  
final\_df.createOrReplaceTempView("education\_summary\_view")  
  
# Or save as permanent Hive table (optional and requires Hive support)  
# final\_df.write.mode("overwrite").saveAsTable("education\_summary")

# 6. Salary by Education Level Plot

## 6.1 Prompt

Plot scatter plot using plotly where colors are EDU\_GROUP (Associate’s or lower is yellow, Bachelor’s is green, Master’s is blue, and PhD is purple), dataframe = final\_df, x-axis is MAX\_YEARS\_EXPERIENCE (with jitter), y-axis is Average\_Salary. Title is Garamond size 24 and all other text is Garamond size 12. All text is black.

## 6.2 Response

Here’s how you can create a Plotly scatter plot from a PySpark DataFrame (final\_df) with the following requirements:

✅ Requirements Recap:

x-axis: MAX\_YEARS\_EXPERIENCE with jitter

y-axis: Average\_Salary

Color by: EDU\_GROUP

“Associate’s or lower” → yellow

“Bachelor’s” → green

“Master’s” → blue

“PhD” → purple

Title: font Garamond, size 24, black

All other text: Garamond, size 12, black

✅ Full Code (with Explanation)

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 = 0.3  
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="Average Salary vs. Max Years of Experience by Education Group",  
 labels={  
 'JITTERED\_EXPERIENCE': 'Max Years of Experience (jittered)',  
 'Average\_Salary': 'Average Salary',  
 'EDU\_GROUP': 'Education Group'  
 },  
 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()

# 7. Salary by Remote Work Type Histograms

## 7.1 Prompt

create a plotly histogram where the x-axis is “MAX\_YEARS\_EXPERIENCE” split into groups of 1, and the y axis is “Average\_Salary”, dataframe is remote\_df, and group by REMOTE\_TYPE\_NAME

## 7.2 Response

To create a Plotly histogram where:

x = MAX\_YEARS\_EXPERIENCE (binned by 1 year),

y = Average\_Salary,

grouped by = REMOTE\_TYPE\_NAME,

you’ll want to use Plotly Express with histfunc=‘avg’ so that it computes the average salary per bin, and use nbins or manually bin the experience column.

✅ Option 1: Use Plotly’s Built-in Histogram Aggregation

This is the simplest approach:

import plotly.express as px  
  
fig = px.histogram(  
 remote\_df,  
 x="MAX\_YEARS\_EXPERIENCE",  
 y="Average\_Salary",  
 color="REMOTE\_TYPE\_NAME",  
 histfunc="avg", # compute average salary per bin  
 nbins=int(remote\_df['MAX\_YEARS\_EXPERIENCE'].max()) + 1,  
 title="Average Salary by Years of Experience and Remote Type",  
 labels={  
 'MAX\_YEARS\_EXPERIENCE': 'Max Years of Experience',  
 'Average\_Salary': 'Average Salary',  
 'REMOTE\_TYPE\_NAME': 'Remote Work Type'  
 },  
 barmode='group' # group bars side by side  
)  
  
fig.update\_layout(  
 xaxis\_title="Max Years of Experience",  
 yaxis\_title="Average Salary",  
 bargap=0.2  
)  
  
fig.show()

## 7.3 Additional Prompt

I got the error TypeError: ‘Column’ object is not callable

## 7.4 Additional Response

Thanks — that error suggests you’re using PySpark DataFrame (remote\_df) directly with Plotly, which doesn’t work because Plotly expects a Pandas DataFrame, not a PySpark one.

The error likely came from this line:

nbins=int(remote\_df['MAX\_YEARS\_EXPERIENCE'].max()) + 1

In PySpark, remote\_df[‘MAX\_YEARS\_EXPERIENCE’] returns a Column object, not a value — which leads to the TypeError: ‘Column’ object is not callable.

✅ Fix: Convert to Pandas First

You must convert your PySpark DataFrame to Pandas before plotting with Plotly:

✔️ Step-by-step Fix

# Step 1: Convert to Pandas  
pdf = remote\_df.toPandas()  
  
# Step 2: Create plot using Plotly  
import plotly.express as px  
  
fig = px.histogram(  
 pdf,  
 x="MAX\_YEARS\_EXPERIENCE",  
 y="Average\_Salary",  
 color="REMOTE\_TYPE\_NAME",  
 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',  
 'Average\_Salary': 'Average Salary',  
 'REMOTE\_TYPE\_NAME': 'Remote Work Type'  
 }  
)  
  
fig.update\_layout(  
 xaxis=dict(dtick=1),  
 yaxis\_title="Average Salary",  
 bargap=0.2  
)  
  
fig.show()