Assignment 03 - AI Prompts

Gavin Boss

September 20, 2025

# 1. Appendix 1

|  |
| --- |
| Tip |
| You’re getting the same median salary across all ONET occupations because your SALARY column was overwritten with a constant value during data cleaning. Here’s how to fix it step-by-step: |

|  |
| --- |
| Tip |
| You’re getting the same median salary across all ONET occupations because your SALARY column was overwritten with a constant value during data cleaning. Here’s how to fix it step-by-step: 1.0.1 ✅ 1. **Avoid Overwriting SALARY with a Constant** In your current code:  df = df.fillna({  "SALARY\_FROM": median\_from,  "SALARY\_TO": median\_to })  df = df.withColumn("SALARY", (col("SALARY\_FROM") + col("SALARY\_TO")) / 2)  This fills missing SALARY\_FROM and SALARY\_TO with the same median, so any row missing salary data ends up with the same computed value — flattening the distribution. 1.0.2 ✅ 2. **Use Conditional Logic to Preserve Original SALARY** Replace the overwrite with this:  df = df.withColumn("SALARY", when(  col("SALARY").isNotNull() & (col("SALARY") > 0),  col("SALARY") ).otherwise((col("SALARY\_FROM") + col("SALARY\_TO")) / 2))  This keeps the original SALARY if it’s valid, and only computes it when missing. 1.0.3 ✅ 3. **Double-Check Aggregation Logic** Make sure you’re grouping correctly and using percentile\_approx with enough accuracy:  agg\_df = df.filter(  col("SALARY").isNotNull() & (col("SALARY") > 0) &  col("LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME").isNotNull() &  (col("LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME") != "") ).groupBy("LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME") \  .agg(  expr("percentile\_approx(SALARY, 0.5, 100)").alias("Median\_Salary"),  count("\*").alias("Job\_Postings") ).withColumnRenamed("LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME", "Occupation\_Name")  If you still see duplicates, try increasing the accuracy:  expr("percentile\_approx(SALARY, 0.5, 1000)") 1.0.4 ✅ 4. **Validate SALARY Distribution Before Aggregating** Run this to confirm variation:  df.select("SALARY").distinct().show(20)  If you see only one or two values, the issue is upstream in your fill logic.  Once you fix the conditional logic, your bubble chart will reflect real differences in median salary across occupations. Let me know if you want to filter out low-volume occupations or sort by salary for a cleaner visual. |

# 2. Appendix 2

|  |
| --- |
| Tip |
| how do I fix a plot in px.scatter that is stacked vertically |

|  |
| --- |
| Tip |
| If your px.scatter plot is stacked vertically, it usually means your x-axis values are discrete or repeated—like integers or categories—so all the points with the same x-value align in vertical lines. Here’s how to fix it:  ✅ Add Jitter to the X-Axis This spreads the points horizontally so they cluster more naturally: import numpy as np 3. Add jitter to your x-axis column df[“x\_jittered”] = df[“your\_x\_column”] + np.random.uniform(-0.3, 0.3, size=len(df)) 4. Use the jittered column in your plot fig = px.scatter( df, x=“x\_jittered”, y=“your\_y\_column”, … )   This works especially well if your x-values are integers like years of experience.  🧠 Why It Happens - If x = 1, 2, 3…, all points with x=1 stack vertically. - Jitter breaks that up by nudging each point slightly left or right.  🛠 Optional Enhancements - Add a trend line to show central tendency: fig = px.scatter(…, trendline=“ols”)   * Use opacity to reveal dense areas: fig.update\_traces(opacity=0.6) * Reduce marker size for better spacing: fig.update\_traces(marker=dict(size=6))   Let me know what your x-axis represents—I can help you fine-tune the jitter or suggest alternatives like violin plots or density contours if clustering is key. |

|  |
| --- |
| Tip |
| Appendix 3 Prompt: if I have all the data in a dataset, but want individual histographs using plotly how can i do that |

|  |
| --- |
| Tip |
| To create individual histograms for each column in your dataset using Plotly, you can loop through the columns and generate a separate histogram for each one. Here’s a clean and efficient way to do it using Plotly Express: ✅ Example: Plotly Histograms for Each Column import plotly.express as px import pandas as pd 5. Load your dataset df = pd.read\_csv(‘your\_dataset.csv’) # or use any DataFrame 6. Loop through each column and plot histogram for col in df.select\_dtypes(include=‘number’).columns: fig = px.histogram(df, x=col, title=f’Histogram of {col}’) fig.show()  🔍 Notes: - select\_dtypes(include=‘number’) ensures you only plot numeric columns. - fig.show() will display each histogram one by one. - You can customize bins, colors, and layout using additional parameters in px.histogram. 🛠 Optional Enhancements: - If you’re working in Quarto or Jupyter and want to save each plot: fig.write\_image(f”{col}\_histogram.png”) - To display all histograms in one dashboard-like layout, consider using plotly.subplots. Would you like help customizing bin sizes, colors, or exporting these for a report? |

|  |
| --- |
| Tip |
| prompt: i have three colums, and already have it jittered |

|  |
| --- |
| Tip |
| Perfect — if you’ve already jittered your data and want to create individual histograms for each of the three columns using Plotly, here’s how you can do it cleanly and efficiently: ✅ Plotly Histograms for Jittered Columns Assuming your DataFrame is called df and the three jittered columns are named col1, col2, and col3: import plotly.express as px 7. List of your jittered columns columns = [‘col1’, ‘col2’, ‘col3’] 8. Loop through each column and plot histogram for col in columns: fig = px.histogram(df, x=col, title=f’Histogram of {col}’, nbins=30) fig.update\_layout(bargap=0.1) fig.show()  🔧 Optional Enhancements: - Customize bins: Use nbins=30 or adjust based on your data spread. - Overlay KDE (density): Plotly doesn’t support KDE directly in px.histogram, but you can simulate it using go.Figure if needed. - Color by category: If you have a categorical column like group, you can do: fig = px.histogram(df, x=col, color=‘group’, barmode=‘overlay’)  📊 If You Want All Histograms in One Figure: You can use make\_subplots from plotly.subplots to stack them vertically or arrange them in a grid. Let me know if you want that layout and I’ll show you how. Would you like to export these as static images for your report or embed them in Quarto? |