Assignment 03

Jonathan Leon

September 24, 2025

1 Load the Dataset Load the Raw Dataset: Use Pyspark to the lightcast\_data.csv file into a DataFrame: You can reuse the previous code. Copying code from your friend constitutes plagiarism. DO NOT DO THIS.

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 = "vscode+notebook+png"  
  
# Initialize Spark Session  
spark = SparkSession.builder.appName("LightcastData").getOrCreate()  
  
# Load Data  
df = spark.read.option("header", "true").option("inferSchema", "true").option("multiLine","true").option("escape", "\"").csv("data/lightcast\_job\_postings.csv")  
df.createOrReplaceTempView("job\_postings")  
  
# Show Schema and Sample Data  
#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 29:> (0 + 1) / 1]

# 1. Data Preparation

#Step 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"))  
  
# Step 2: Computing median for salary colums  
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)  
  
# Step 3: Imputing missing salaries, but not experience  
df = df.fillna({  
 "SALARY\_FROM": median\_from,  
 "SALARY\_TO": median\_to,  
 "SALARY": median\_salary  
})  
  
  
# Step 4: Computing average salary  
df = df.withColumn("Average\_Salary", (col("SALARY\_FROM") + col("SALARY\_TO")) /   
2)  
  
# Step 5: Selecting required columns  
export\_cols = [  
 "EDUCATION\_LEVELS\_NAME",  
 "REMOTE\_TYPE\_NAME",  
 "MAX\_YEARS\_EXPERIENCE",  
 "Average\_Salary",  
 "SALARY",  
 "LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME",  
 "EMPLOYMENT\_TYPE\_NAME",  
]  
  
  
df\_selected = df.select(\*export\_cols)  
  
  
# Step 6: Saving to CSV  
pdf = df\_selected.toPandas()  
pdf.to\_csv("./data/lightcast\_cleaned.csv", index=False)  
  
print("Data cleaning complete. Rows retained:", len(pdf))

[Stage 30:> (0 + 1) / 1] [Stage 31:> (0 + 1) / 1] [Stage 32:> (0 + 1) / 1]

Medians: 87295.0 130042.0 115024.0

[Stage 33:> (0 + 1) / 1]

Data cleaning complete. Rows retained: 72498

# 2. Set Up Style Template

import plotly.graph\_objects as go  
import plotly.io as pio  
  
pio.templates["nike"] = go.layout.Template(  
 # LAYOUT  
 layout = {  
 # Fonts  
 # Note - 'family' must be a single string, NOT a list or dict!  
 'title':  
 {'font': {'family': 'HelveticaNeue-CondensedBold, Helvetica, Sans-serif',  
 'size':30,  
 'color': '#333'}  
 },  
 'font': {'family': 'Helvetica Neue, Helvetica, Sans-serif',  
 'size':16,  
 'color': '#333'},  
 # Colorways  
 'colorway': ['#ec7424', '#a4abab'],  
 # Keep adding others as needed below  
 'hovermode': 'x unified'  
 },  
 # DATA  
 data = {  
 # Each graph object must be in a tuple or list for each trace  
 'bar': [go.Bar(texttemplate = '%{value:$.2s}',  
 textposition='outside',  
 textfont={'family': 'Helvetica Neue, Helvetica, Sans-serif',  
 'size': 20,  
 'color': '#FFFFFF'  
 })]  
 }  
)

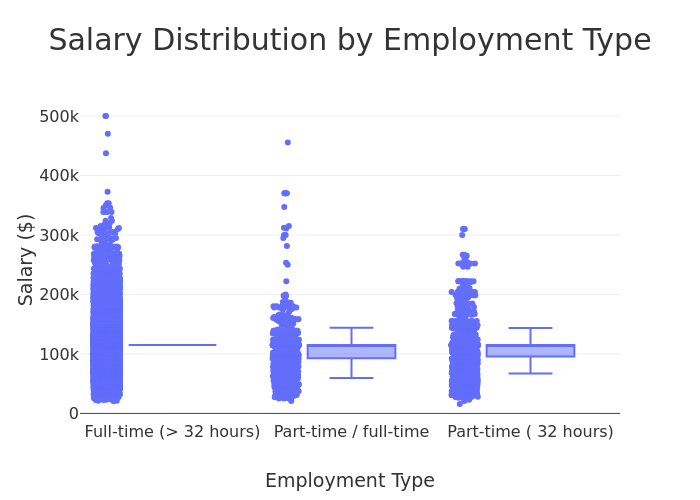
# 3. Question 1: Salary Distribution by Industry and Employment Type

Compare salary variations across industries. Filter the dataset Remove records where salary is missing or zero. Aggregate Data Group by NAICS industry codes. Group by employment type and compute salary distribution. Visualize results Create a box plot where: X-axis = NAICS2\_NAME Y-axis = SALARY\_FROM Group by EMPLOYMENT\_TYPE\_NAME. Customize colors, fonts, and styles. Explanation: Write two sentences about what the graph reveals.

# Code for 1st question  
import pandas as pd  
  
# filter out missing or zero salary values  
pdf = df.filter(df["SALARY"] > 0).select("EMPLOYMENT\_TYPE\_NAME", "SALARY").toPandas()  
pdf = pdf[pdf["EMPLOYMENT\_TYPE\_NAME"].notnull()]  
  
# Clean employment type names for better readability  
  
pdf["EMPLOYMENT\_TYPE\_NAME"] = pdf["EMPLOYMENT\_TYPE\_NAME"].apply(lambda x: re.sub(r"[^\x00-\x7F]+", "", x))  
  
# Compute median salary for sorting  
median\_salaries = pdf.groupby("EMPLOYMENT\_TYPE\_NAME")["SALARY"].median()  
  
#sort employment types based on median salary in descending order  
sorted\_employment\_types = median\_salaries.sort\_values(ascending=False).index  
  
# Apply sorted categories  
pdf["EMPLOYMENT\_TYPE\_NAME"] = pd.Categorical(  
 pdf["EMPLOYMENT\_TYPE\_NAME"],  
 categories=sorted\_employment\_types,  
 ordered=True  
)  
  
# Create box plot with horizontal grid lines  
fig = px.box(  
 pdf,  
 x="EMPLOYMENT\_TYPE\_NAME",  
 y="SALARY",  
 title="Salary Distribution by Employment Type",  
 boxmode="group",  
 points="all",  
 width=800 #show all outliers  
)  
  
# Improve layout, font styles, and axis labels  
fig.update\_layout(  
 template="nike",  
 xaxis\_title="Employment Type",  
 yaxis\_title="Salary ($)",  
 )  
   
# Show the figure  
fig.show()  
fig.write\_html("output/Q1.html")  
fig.write\_image("output/Q1.svg", width=850, height=500, scale=1)

[Stage 34:> (0 + 1) / 1]

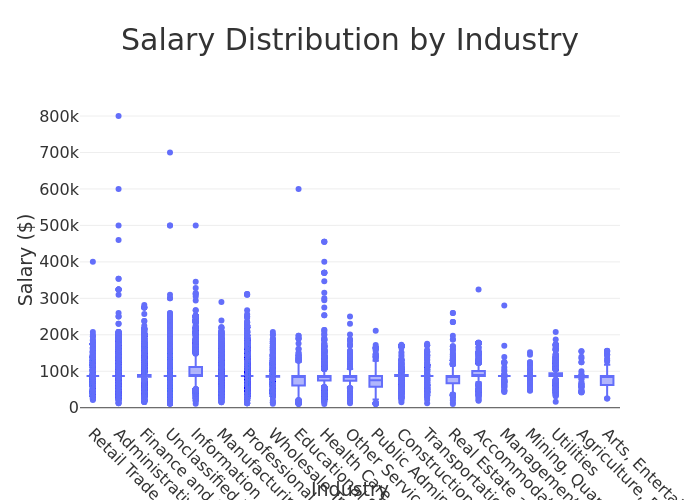
Unable to display output for mime type(s): text/html



The output shows there are significantly more full-time positions than part time positions and most fall in a range that goes up to just under $300k. While there are outliers for all 3 categories it is likely that the part-time outliers are due to annualized salary being calculated incorrectly while the full time outliers are likely legitimate.

pdf = df.select("NAICS2\_NAME", "SALARY\_FROM").toPandas()  
fig = px.box(pdf, x="NAICS2\_NAME", y="SALARY\_FROM", title="Salary Distribution by Industry")  
fig.update\_layout(  
 template="nike",  
 xaxis\_title="Industry",  
 yaxis\_title="Salary ($)",  
 )  
  
fig.update\_xaxes(tickangle=45)  
  
fig.show()  
  
fig.write\_html("output/Q1part2.html")  
fig.write\_image("output/Q1part2.svg", width=850, height=500, scale=1)

[Stage 35:> (0 + 1) / 1]



The output shows most industries observed have relatively similar salary ranges for the most part and outliers are rare. The only indsutries to seemingly have consistent outliers are administrative services and health care.

# 4. Question 2: Salary Analysis by ONET Occupation Type (Bubble Chart)

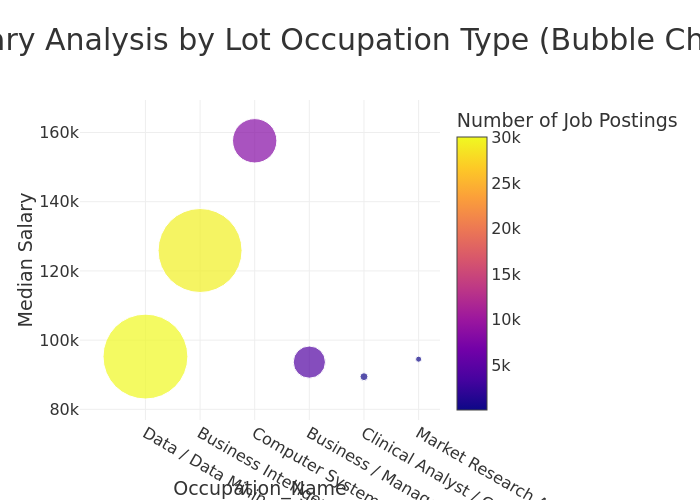
Analyze how salaries differ across ONET occupation types. Aggregate Data Compute median salary for each occupation in the ONET taxonomy. Visualize results Create a bubble chart where: X-axis = ONET\_NAME Y-axis = Median Salary Size = Number of job postings Apply custom colors and font styles. Explanation: Write two sentences about what the graph reveals.

#Step 1: Spark SQL  
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  
 """)  
  
#Step 2: Convert to Pandas DataFrame  
salary\_pd = salary\_analysis.toPandas()  
salary\_pd.head()

[Stage 36:> (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 |

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={  
 "LOT\_OCCUPATION\_NAME" : "Lot Occupation",  
 "Median\_Salary" : "Median Salary",  
 "Job\_Postings" : "Number of Job Postings"  
 },  
 hover\_name="Occupation\_Name",  
 size\_max=60,  
 width=1000,  
 height=600,  
 color="Job\_Postings",  
 color\_continuous\_scale="Plasma"  
)  
  
fig.update\_layout(template="nike")  
  
fig.show()  
  
fig.write\_html("output/Q2.html")  
fig.write\_image("output/Q2.svg", width=850, height=500, scale=1)



This output shows of the industries observed most have salaries in the 90k-100k range with the exception of Business Intellegence Analyst and Computer Systems Engingeer/Architect. Computer Systems Engineer/Architect jobs have the 3rd highest amount of job postings observed despite the highest median salary suggesting a healthy need for these well paid professionals.

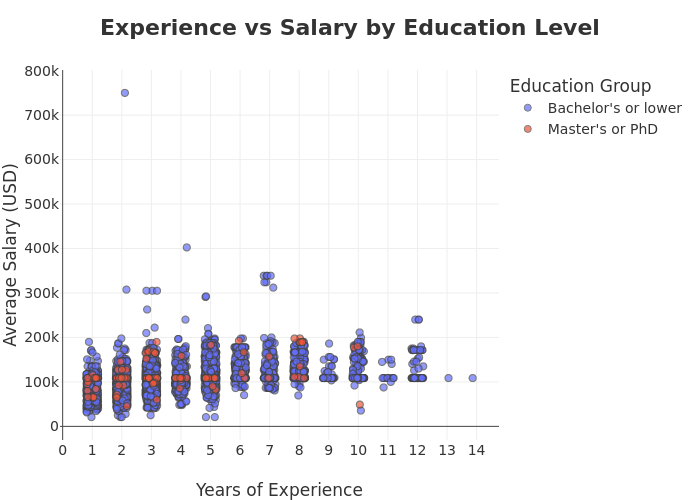
# 5. Question 3: Salary by Education Level

Create two groups: Associate’s or lower (GED, Associate, No Education Listed) Bachelor’s (Bachelor’s degree) Master’s (Master’s degree) PhD (PhD, Doctorate, professional degree) Plot scatter plots for each group using, MAX\_YEARS\_EXPERIENCE (with jitter), Average\_Salary, LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME After each graph, add a short explanation of key insights.

#Create 2 Groups  
lower\_deg = ["Bachelor's", "Associate", "GED", "No Education Listed", "High school"]  
higher\_deg = ["Master's degree", "PhD or professional degree"]  
  
# Add EDU\_Group column  
df = df.withColumn(  
 "EDU\_GROUP",  
 when(col("EDUCATION\_LEVELS\_NAME").rlike("|".join([f"(?i){deg}" for deg in lower\_deg])), "Bachelor's or lower")  
 .when(col("EDUCATION\_LEVELS\_NAME").rlike("|".join([f"(?i){deg}" for deg in higher\_deg])), "Master's or PhD")  
)  
  
df = df.withColumn("MAX\_YEARS\_EXPERIENCE", col("MAX\_YEARS\_EXPERIENCE").cast("float"))  
df = df.withColumn("Average\_Salary", col("Average\_Salary").cast("float"))  
  
df = df.filter(  
 col("MAX\_YEARS\_EXPERIENCE").isNotNull() &  
 col("Average\_Salary").isNotNull() &  
 (col("MAX\_YEARS\_EXPERIENCE") > 0) &  
 (col("Average\_Salary") > 0)  
)  
  
df\_filtered = df.filter(col("EDU\_GROUP").isin("Bachelor's or lower", "Master's or PhD"))  
  
df\_pd = df\_filtered.toPandas()

[Stage 39:> (0 + 1) / 1]

import numpy as np  
  
# Add jitter to MAX\_YEARS\_EXPERIENCE  
jitter\_strength = 0.2  
np.random.seed(42)  
df\_pd["MAX\_YEARS\_EXPERIENCE\_JITTER"] = df\_pd["MAX\_YEARS\_EXPERIENCE"] + np.random.uniform(-jitter\_strength, jitter\_strength, size=len(df\_pd))  
  
fig1 = px.scatter(  
 df\_pd,  
 x="MAX\_YEARS\_EXPERIENCE\_JITTER",  
 y="Average\_Salary",  
 color="EDU\_GROUP",  
 hover\_data=["LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME"],  
 title="<b>Experience vs Salary by Education Level</b>",  
 opacity=0.7,  
 width=800  
)  
  
fig1.update\_traces(marker=dict(size=7, line=dict(width=1)))  
  
fig1.update\_layout(  
 template="nike",  
 font=dict(size=14),  
 title\_font=dict(size=22),  
 xaxis\_title="Years of Experience",  
 yaxis\_title="Average Salary (USD)",  
 legend\_title="Education Group",  
 hoverlabel=dict(font\_size=13),  
 margin=dict(t=70, b=60, l=60, r=60,),  
 xaxis=dict(  
 tickmode='linear',  
 dtick=1)  
)  
  
fig1.show()  
  
fig.write\_html("output/Q3.html")  
fig.write\_image("output/Q3.svg", width=850, height=500, scale=1)



This output shows there are more opportunities for those with a bachelor’s degree or lower especially once the experience required is greater than 8 years. The output shows that an advanced degree doesn’t necessarily lead to a higher salary and that roles seeking more experienced candidates value advanced degrees even less.

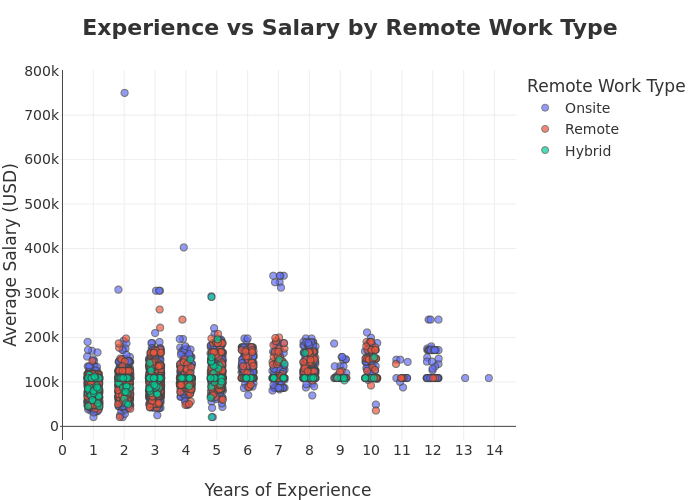
# 6. Question 4: Salary by Remote Work Type

Split into three groups based on REMOTE\_TYPE\_NAME: Remote Hybrid Onsite (includes [None] and blank) Plot scatter plots for each group using, MAX\_YEARS\_EXPERIENCE (with jitter), Average\_Salary, LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME Also, create salary histograms for all three groups. After each graph, briefly describe any patterns or comparisons.

#Create 3 Groups  
Remote = ["Remote"]  
Hybrid = ["Hybrid Remote"]  
Onsite = ["Not Remote", "\\[None\\]"] # Escape the brackets since they're regex special characters  
  
# Add REMOTE\_Group column with corrected regex patterns  
df = df.withColumn(  
 "REMOTE\_GROUP",  
 when(col("REMOTE\_TYPE\_NAME").rlike("(?i)^Remote$"), "Remote")  
 .when(col("REMOTE\_TYPE\_NAME").rlike("(?i)^Hybrid Remote$"), "Hybrid")  
 .when(col("REMOTE\_TYPE\_NAME").rlike("(?i)^(Not Remote|\\[None\\])$"), "Onsite")  
 .otherwise(None) # This will help you see unmatched values  
)  
  
# Rest of your code remains the same  
df = df.withColumn("MAX\_YEARS\_EXPERIENCE", col("MAX\_YEARS\_EXPERIENCE").cast("float"))  
df = df.withColumn("Average\_Salary", col("Average\_Salary").cast("float"))  
  
df = df.filter(  
 col("MAX\_YEARS\_EXPERIENCE").isNotNull() &  
 col("Average\_Salary").isNotNull() &  
 (col("MAX\_YEARS\_EXPERIENCE") > 0) &  
 (col("Average\_Salary") > 0)  
)  
  
df\_filtered = df.filter(col("REMOTE\_GROUP").isin("Remote", "Hybrid", "Onsite"))  
df\_remote = df\_filtered.toPandas()

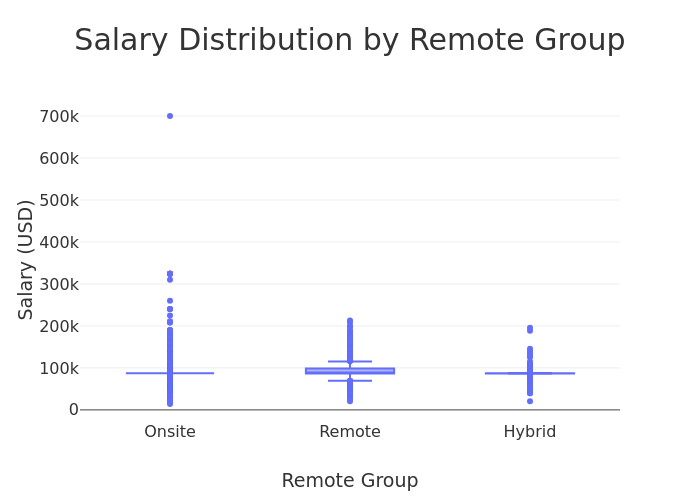
[Stage 40:> (0 + 1) / 1]

import numpy as np  
  
jitter\_strength = 0.2  
np.random.seed(42)  
df\_remote["MAX\_YEARS\_EXPERIENCE\_JITTER"] = df\_remote["MAX\_YEARS\_EXPERIENCE"] + np.random.uniform(-jitter\_strength, jitter\_strength, size=len(df\_remote))  
  
fig3 = px.scatter(  
 df\_remote,  
 x="MAX\_YEARS\_EXPERIENCE\_JITTER",  
 y="Average\_Salary",  
 color="REMOTE\_GROUP",  
 hover\_data=["LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME"],  
 title="<b>Experience vs Salary by Remote Work Type</b>",  
 opacity=0.7,  
 width=800  
)  
  
fig3.update\_traces(marker=dict(size=7, line=dict(width=1)))  
  
fig3.update\_layout(  
 template="nike",  
 font=dict(size=14),  
 title\_font=dict(size=22),  
 xaxis\_title="Years of Experience",  
 yaxis\_title="Average Salary (USD)",  
 legend\_title="Remote Work Type",  
 hoverlabel=dict(font\_size=13),  
 margin=dict(t=70, b=60, l=60, r=60,),  
 xaxis=dict(  
 tickmode='linear',  
 dtick=1)  
)  
  
fig3.show()  
  
fig3.write\_html("output/Q4part1.html")  
fig3.write\_image("output/Q4part1.svg", width=850, height=500, scale=1)



This output shows that there are limited hybrid employment opportunities once the experience required is greater than 5 years and no hybrid opportunities for positions requiring greater than 10 years of experience. This suggests organizations are open to offering hybrid opportunities to attract more junior workers but senior level or leadership positions are likely to be either fully in office or fully remote depending on company culture.

fig4 = px.box(df\_remote, x="REMOTE\_GROUP", y="SALARY\_FROM", title="Salary Distribution by Remote Group")  
fig4.update\_layout(  
 template="nike",  
 xaxis\_title="Remote Group",  
 yaxis\_title="Salary (USD)",  
 )  
  
fig4.show()  
  
fig4.write\_html("output/Q4part2.html")  
fig4.write\_image("output/Q4part2.svg", width=850, height=500, scale=1)



This output shows hybrid positions have slightly lower range of salary outputs. Onsite and remote positions have a similar range of consistent salary levels suggesting both position types are viewed equally however onsite positions have higher outliers.