Module 04: Lab 01

Visual Reporting and Storytelling

Yuxuan Chen

March 20, 2025

# Objectives

By the end of this lab, you will: 1. Load and analyze the **Lightcast dataset** in **Spark DataFrame**. 2. Create **five easy and three medium-complexity visualizations** using **Plotly**. 3. Explore **salary distributions, employment trends, and job postings**. 4. Analyze **skills in relation to NAICS/SOC/ONET codes and salaries**. 5. Customize **colors, fonts, and styles** in all visualizations (**default themes result in a 2.5-point deduction**). 6. Follow **best practices for reporting on data communication**.

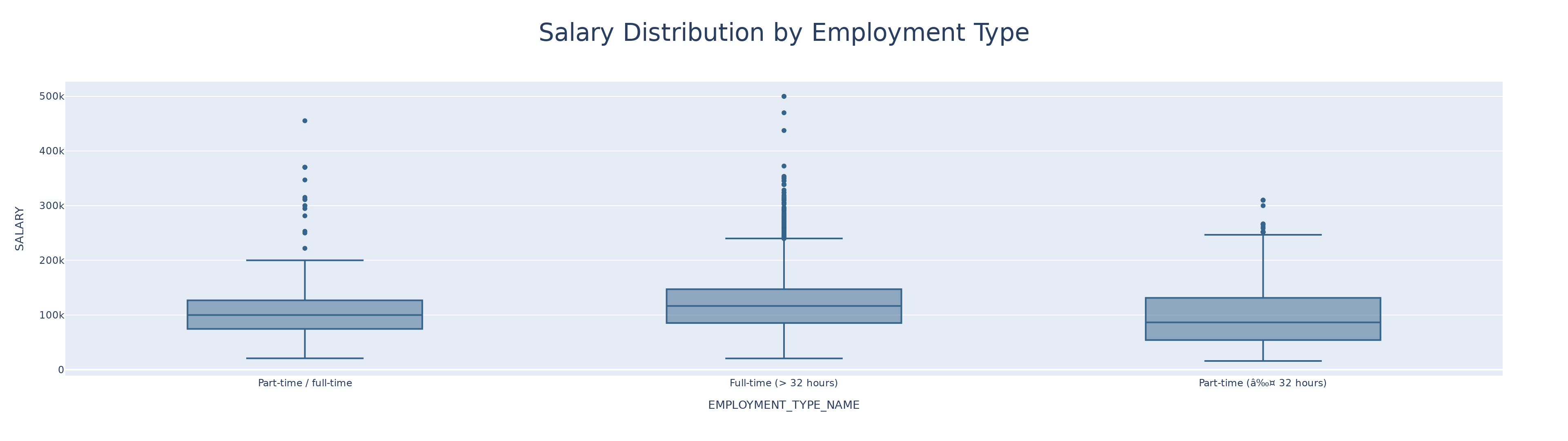
# Step 1: Load the Dataset

import pandas as pd  
import plotly.express as px  
import plotly.io as pio  
pio.renderers.default = "vscode"  
from pyspark.sql import SparkSession  
from pyspark.sql.functions import split, explode, col, regexp\_replace, transform  
  
  
# 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")  
  
# Show Schema and Sample Data  
df.printSchema()  
df.show(5)

# 1. Salary Distribution by Employment Type

* Identify salary trends across different employment types.
* **Filter the dataset**
  + Remove records where **salary is missing or zero**.
* **Aggregate Data**
  + Group by **employment type** and compute salary distribution.
* **Visualize results**
  + Create a **box plot** where:
    - **X-axis** = EMPLOYMENT\_TYPE\_NAME
    - **Y-axis** = SALARY\_FROM
  + Customize **colors, fonts, and styles** to avoid a **2.5-point deduction**.
* **Explanation:** Write two sentences about what the graph reveals.

# Your Code for 1st question here  
df = df.filter((col("SALARY\_FROM").isNotNull()) & (col("SALARY\_FROM") > 0))  
  
pdf = df.select("EMPLOYMENT\_TYPE\_NAME", "SALARY").toPandas()  
pdf.head()  
  
fig = px.box(pdf, x="EMPLOYMENT\_TYPE\_NAME", y="SALARY", title="Salary Distribution by Employment Type", color\_discrete\_sequence=["#36648B"])  
fig.update\_layout(font\_family="Arial", title\_font\_size=30, title\_x=0.5)  
fig.show()  
  
fig.write\_image("output/Q1.png", width=1920, height=540, scale=2)



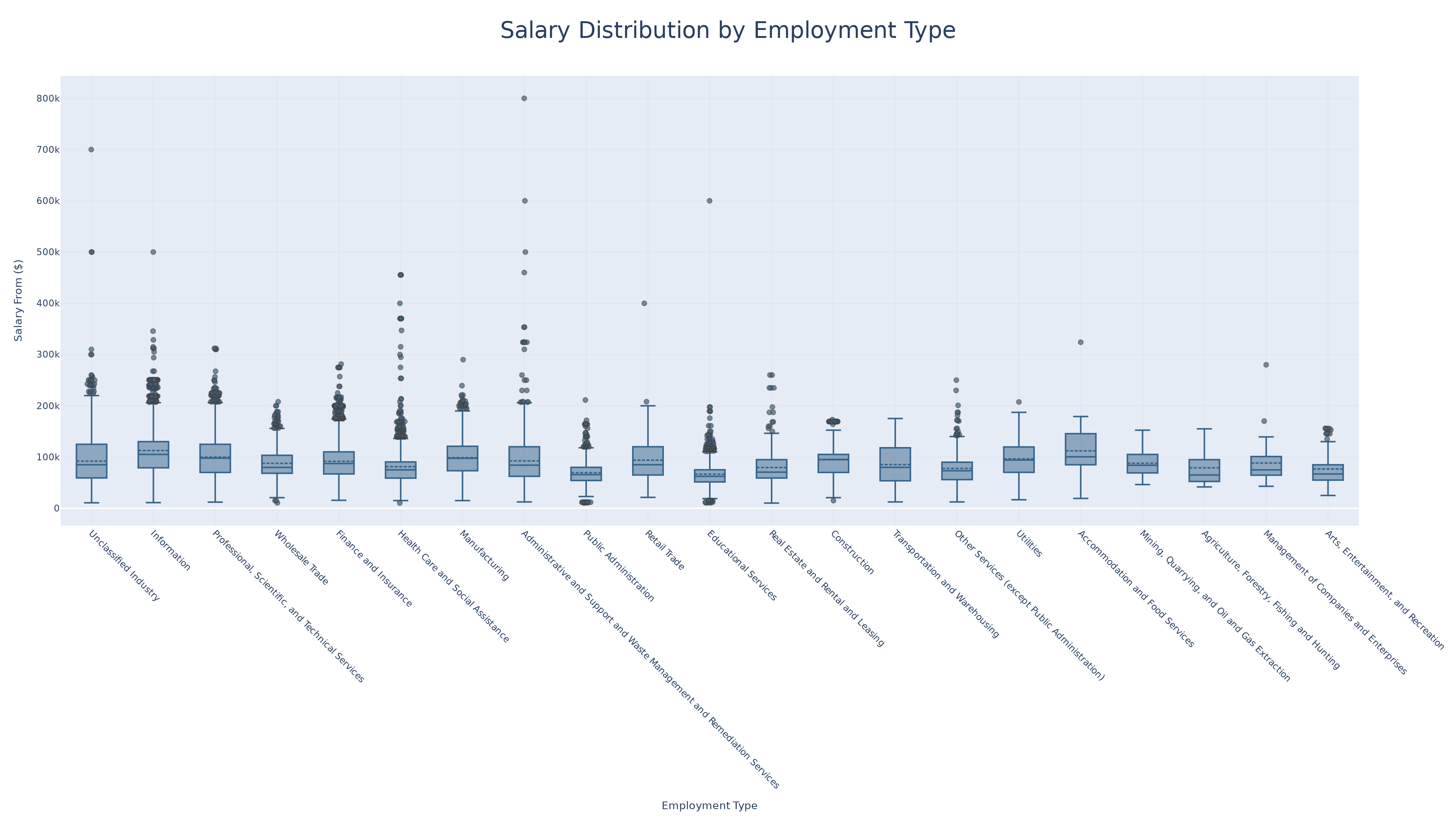
Question 1

This chart shows the salary distribution across different employment types, including part-time/full-time, full-time (> 32 hours), and part-time (around 32 hours). It highlights that full-time workers generally earn higher salaries compared to part-time workers.

# 2. Salary Distribution by Industry

* Compare salary variations across industries.
* **Filter the dataset**
  + Keep records where **salary is greater than zero**.
* **Aggregate Data**
  + Group by **NAICS industry codes**.
* **Visualize results**
  + Create a **box plot** where:
    - **X-axis** = NAICS2\_NAME
    - **Y-axis** = SALARY\_FROM
  + Customize colors, fonts, and styles.
* **Explanation:** Write two sentences about what the graph reveals.

# Your code for 2nd question here  
df = df.filter(col("SALARY\_FROM") > 0)  
  
pdf = df.select("NAICS2\_NAME", "SALARY\_FROM").toPandas()  
pdf.head()  
  
fig = px.box(  
 pdf,   
 x="NAICS2\_NAME",   
 y="SALARY\_FROM",   
 title="Salary Distribution by Employment Type",   
 color\_discrete\_sequence=["#36648B"]  
)  
  
fig.update\_layout(  
 font\_family="Arial",  
 title\_font\_size=30, # 标题字体大小  
 title\_x=0.5, # 标题居中  
 width=1400, # 增大宽度  
 height=800, # 增大高度  
 xaxis\_title="Employment Type", # X轴标签  
 yaxis\_title="Salary From ($)", # Y轴标签  
 xaxis=dict(  
 tickangle=45, # X轴标签旋转角度  
 showgrid=True, # 显示网格线  
 gridcolor="rgba(211, 211, 211, 0.3)" # 网格线颜色  
 ),  
 yaxis=dict(  
 showgrid=True,  
 gridcolor="rgba(211, 211, 211, 0.3)"  
 ),  
 legend=dict(  
 title="Employment Type",  
 font=dict(family="Arial", size=14, color="black"),  
 bgcolor="rgba(255, 255, 255, 0.8)", # 图例背景色  
 bordercolor="rgba(0, 0, 0, 0.1)", # 图例边框  
 borderwidth=1  
 )  
)  
  
# 优化箱线图样式  
fig.update\_traces(  
 marker=dict(opacity=0.7, line=dict(width=1.5)), # 数据点透明度和边框线宽  
 boxmean=True, # 显示均值线  
 jitter=0.3, # 数据点抖动效果  
 line=dict(width=2) # 箱线图边框线宽度  
)  
  
fig.show()  
  
fig.write\_image("output/Q2.png", width=1920, height=1080, scale=2)



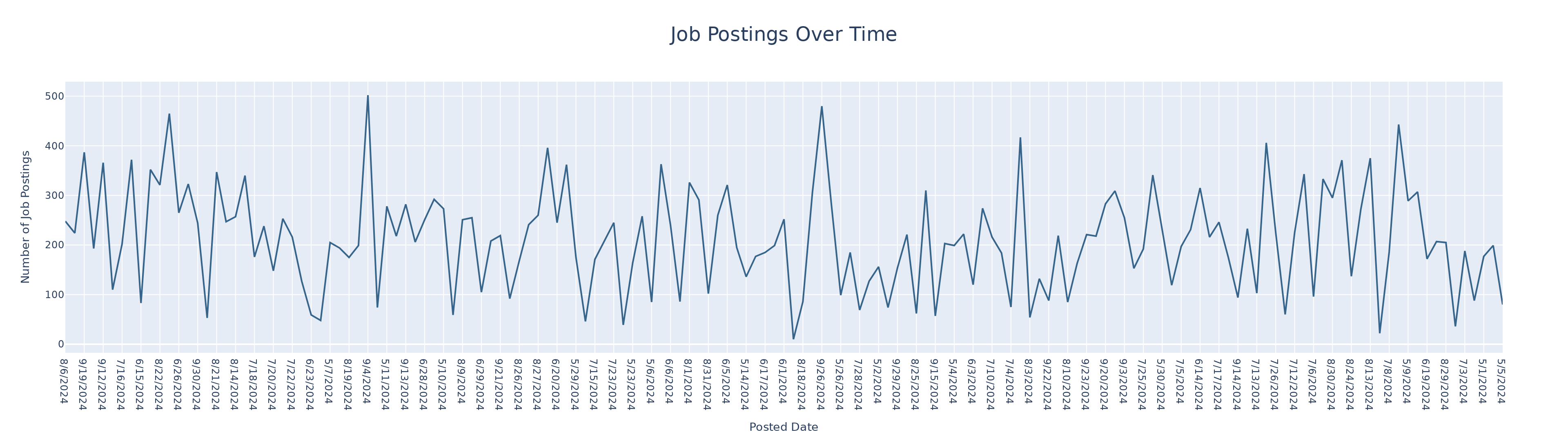
Question 2

This chart illustrates the salary distribution across various industries and employment types. It shows significant differences in median salaries across industries, with sectors like Information having higher median salaries compared to sectors like Educational Services.

# 3. Job Posting Trends Over Time

* Analyze how job postings fluctuate over time.
* **Aggregate Data**
  + Count job postings per **posted date (POSTED)**.
* **Visualize results**
  + Create a **line chart** where:
    - **X-axis** = POSTED
    - **Y-axis** = Number of Job Postings
  + Apply custom colors and font styles.
* **Explanation:** Write two sentences about what the graph reveals.

# Your code for 3rd question here  
df\_aggregated = df.groupBy("POSTED").count()  
pdf = df\_aggregated.toPandas()  
  
fig = px.line(pdf, x="POSTED", y="count", title="Job Postings Over Time", line\_shape="linear",  
 color\_discrete\_sequence=["#36648B"])  
  
fig.update\_layout(  
 font\_family="Arial",  
 title\_font\_size=24,  
 xaxis\_title="Posted Date",  
 yaxis\_title="Number of Job Postings",  
 title\_x=0.5  
)  
  
fig.show()  
  
fig.write\_image("output/Q3.png", width=1920, height=540, scale=2)



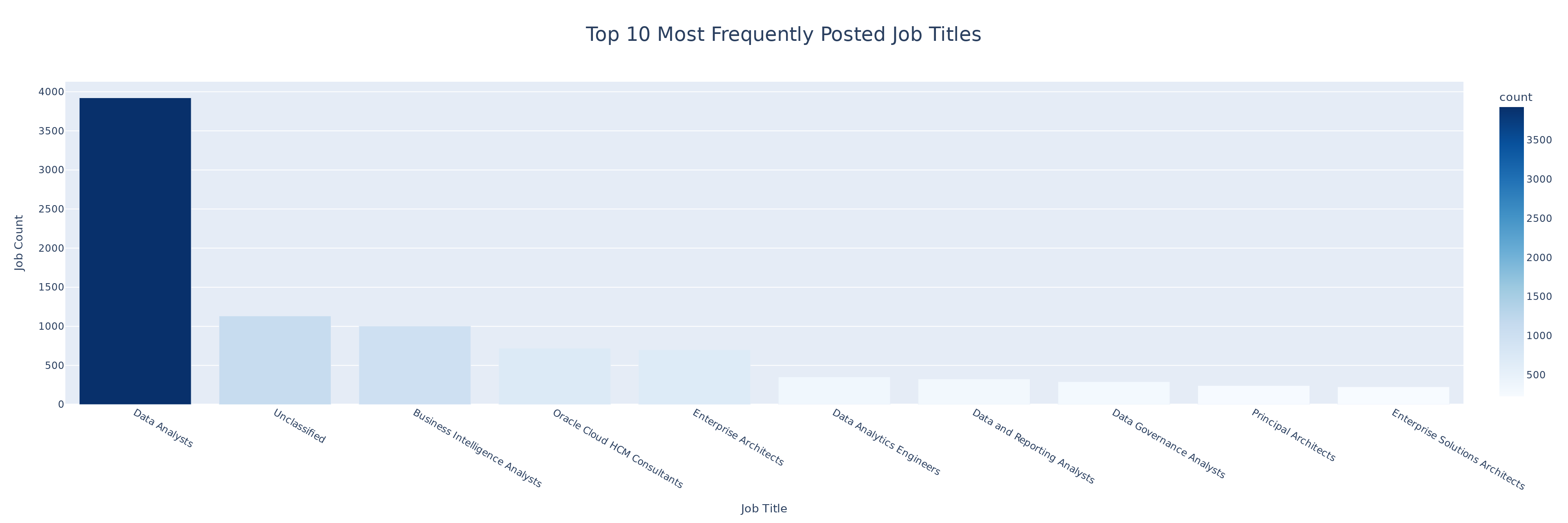
Question 3

This chart illustrates the fluctuating number of job postings over time, with several peaks indicating periods of increased hiring activity. The data reveals a dynamic job market with varying demand for labor, which could be influenced by economic cycles or seasonal factors.

# 4. Top 10 Job Titles by Count

* Identify the most frequently posted job titles.
* **Aggregate Data**
  + Count the occurrences of each **job title (TITLE\_NAME)**.
  + Select the **top 10 most frequent titles**.
* **Visualize results**
  + Create a **bar chart** where:
    - **X-axis** = TITLE\_NAME
    - **Y-axis** = Job Count
  + Apply custom colors and font styles.
* **Explanation:** Write two sentences about what the graph reveals.

# Your code for 4th question here  
df\_aggregated = df.groupBy("TITLE\_NAME").count()  
df\_top\_10 = df\_aggregated.orderBy(col("count").desc()).limit(10)  
pdf = df\_top\_10.toPandas()  
  
fig = px.bar(pdf, x="TITLE\_NAME", y="count", title="Top 10 Most Frequently Posted Job Titles",   
 color="count", color\_continuous\_scale="Blues")  
  
fig.update\_layout(  
 font\_family="Arial",  
 title\_font\_size=24,  
 xaxis\_title="Job Title",  
 yaxis\_title="Job Count",  
 title\_x=0.5  
)  
  
fig.show()  
  
fig.write\_image("output/Q4.png", width=1920, height=640, scale=2)



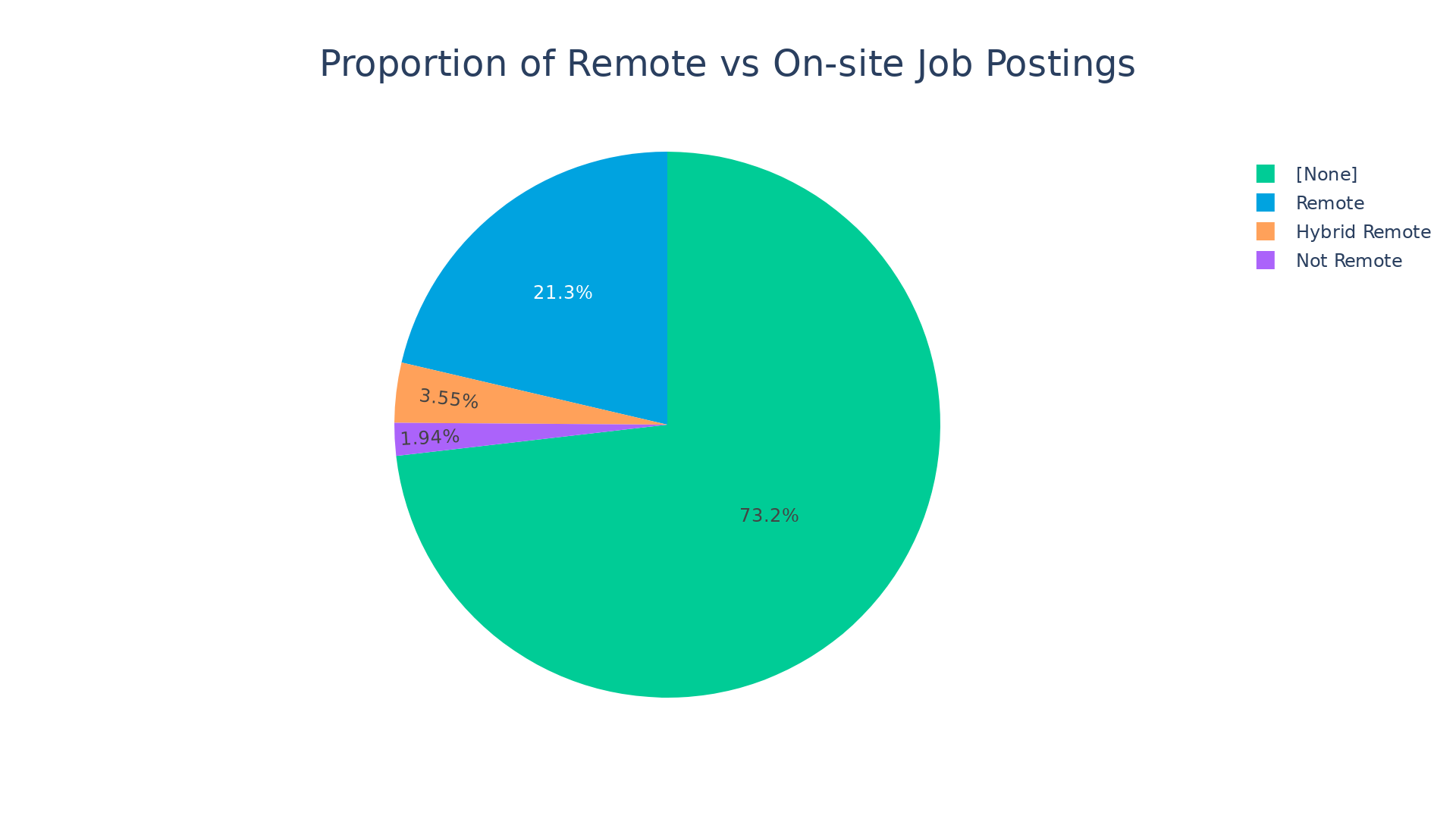
Question 4

This bar chart shows the top 10 most frequently posted job titles. It highlights that “Data Analysts” have the highest number of postings, followed by “Unclassified” and “Business Intelligence Analysts”. The number of jobs posted for ‘Data Analyst’ is significantly higher than other jobs.

# 5. Remote vs On-Site Job Postings

* Compare the proportion of remote and on-site job postings.
* **Aggregate Data**
  + Count job postings by **remote type (REMOTE\_TYPE\_NAME)**.
* **Visualize results**
  + Create a **pie chart** where:
    - **Labels** = REMOTE\_TYPE\_NAME
    - **Values** = Job Count
  + Apply custom colors and font styles.
* **Explanation:** Write two sentences about what the graph reveals.

# Your code for 5th question here  
df\_aggregated = df.groupBy("REMOTE\_TYPE\_NAME").count()  
pdf = df\_aggregated.toPandas()  
  
fig = px.pie(pdf, names="REMOTE\_TYPE\_NAME", values="count", title="Proportion of Remote vs On-site Job Postings",  
 color="REMOTE\_TYPE\_NAME", color\_discrete\_map={"Remote": "#00A3E0", "On-site": "#FF6F61"})  
  
fig.update\_layout(  
 font\_family="Arial",  
 title\_font\_size=24,  
 title\_x=0.5  
)  
  
fig.show()  
  
fig.write\_image("output/Q5.png", width=960, height=540, scale=2)



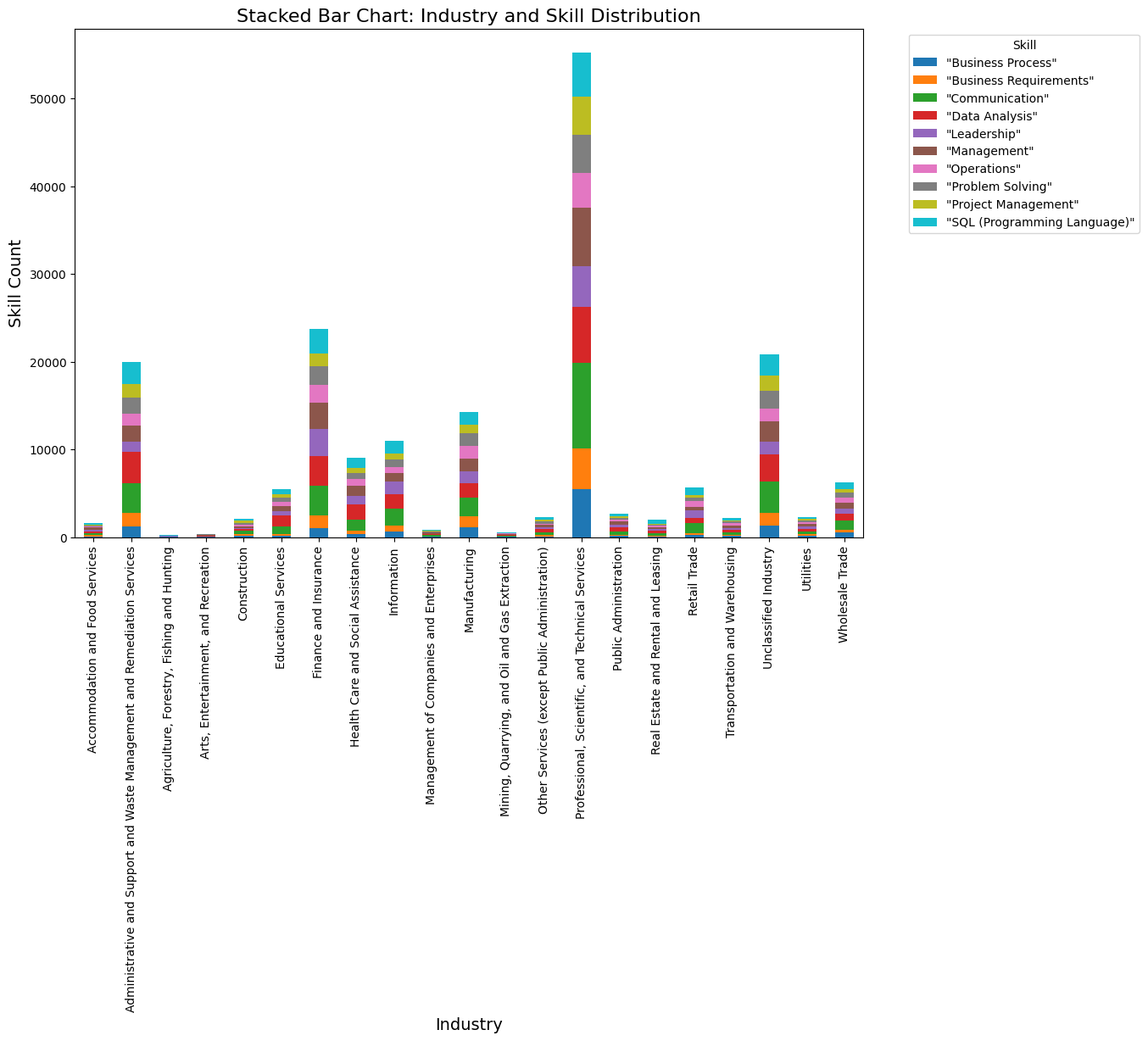
Question 5

This pie chart shows the proportion of remote and on-site job postings. It indicates that a majority of jobs (73.2%) are on-site, while 21.3% are remote, 3.55% are hybrid remote, and 1.94% are not remote, reflecting a growing trend towards remote work.

# 6. Skill Demand Analysis by Industry (Stacked Bar Chart)

* Identify which skills are most in demand in various industries.
* **Aggregate Data**
  + Extract **skills** from job postings.
  + Count occurrences of skills grouped by **NAICS industry codes**.
* **Visualize results**
  + Create a **stacked bar chart** where:
    - **X-axis** = Industry
    - **Y-axis** = Skill Count
    - **Color** = Skill
  + Apply custom colors and font styles.
* **Explanation:** Write two sentences about what the graph reveals.

# Your code for 6th question here  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
# 将 df DataFrame 中的 "SKILLS\_NAME" 列的字符串数据按指定的分隔符 ",\n " 进行分割  
df\_split = df.withColumn("SKILLS\_NAME", split(col("SKILLS\_NAME"), ",\n "))  
  
# 去掉 SKILLS\_NAME 列中每个字符串末尾的换行符  
df\_split = df\_split.withColumn(  
 "SKILLS\_NAME",  
 transform(  
 col("SKILLS\_NAME"),  
 lambda x: regexp\_replace(x, r'\n', '') # 替换每个字符串末尾的换行符  
 )  
)  
  
# 将每个数组元素展开为单独的行存放进skill列  
df\_exploded = df\_split.withColumn("Skill", explode(col("SKILLS\_NAME")))  
  
# 对 df\_exploded 进行分组，并对每组数据进行计数  
df\_aggregated = df\_exploded.groupBy("NAICS2\_NAME", "Skill").count()  
  
# Convert the result to Pandas DataFrame for plotting  
pdf = df\_aggregated.toPandas()  
  
  
  
  
  
# 计算每个技能的总数量  
skill\_count = pdf.groupby('Skill')['count'].sum().sort\_values(ascending=False)  
  
# 选择最多的前十个技能  
top\_skills = skill\_count.head(10).index  
  
# 过滤原始数据，只保留前十个技能  
pdf\_filtered = pdf[pdf['Skill'].isin(top\_skills)]  
  
# 聚合数据，根据NAICS2\_NAME和Skill进行分组，计算每个组合的技能总数  
pdf\_grouped = pdf\_filtered.groupby(['NAICS2\_NAME', 'Skill'])['count'].sum().unstack(fill\_value=0)  
  
# 设置图形大小和分辨率  
plt.figure(figsize=(19.2, 10.8), dpi=100) # figsize 为 1920x1080 像素，dpi 为 100  
  
# 绘制堆积条形图  
plt.figure(figsize=(14, 10))  
pdf\_grouped.plot(kind='bar', stacked=True, figsize=(12, 8))  
  
# 设置标题和标签  
plt.title('Stacked Bar Chart: Industry and Skill Distribution', fontsize=16)  
plt.xlabel('Industry', fontsize=14)  
plt.ylabel('Skill Count', fontsize=14)  
  
# 显示图例  
plt.legend(title='Skill', bbox\_to\_anchor=(1.05, 1), loc='upper left')  
  
# 调整边距，避免标签重叠  
plt.subplots\_adjust(top=0.9, bottom=0.15)  
  
# 保存图像为 PNG 格式  
plt.savefig('output/Q6.png', bbox\_inches='tight')  
  
# 显示图形  
plt.tight\_layout()  
plt.show()



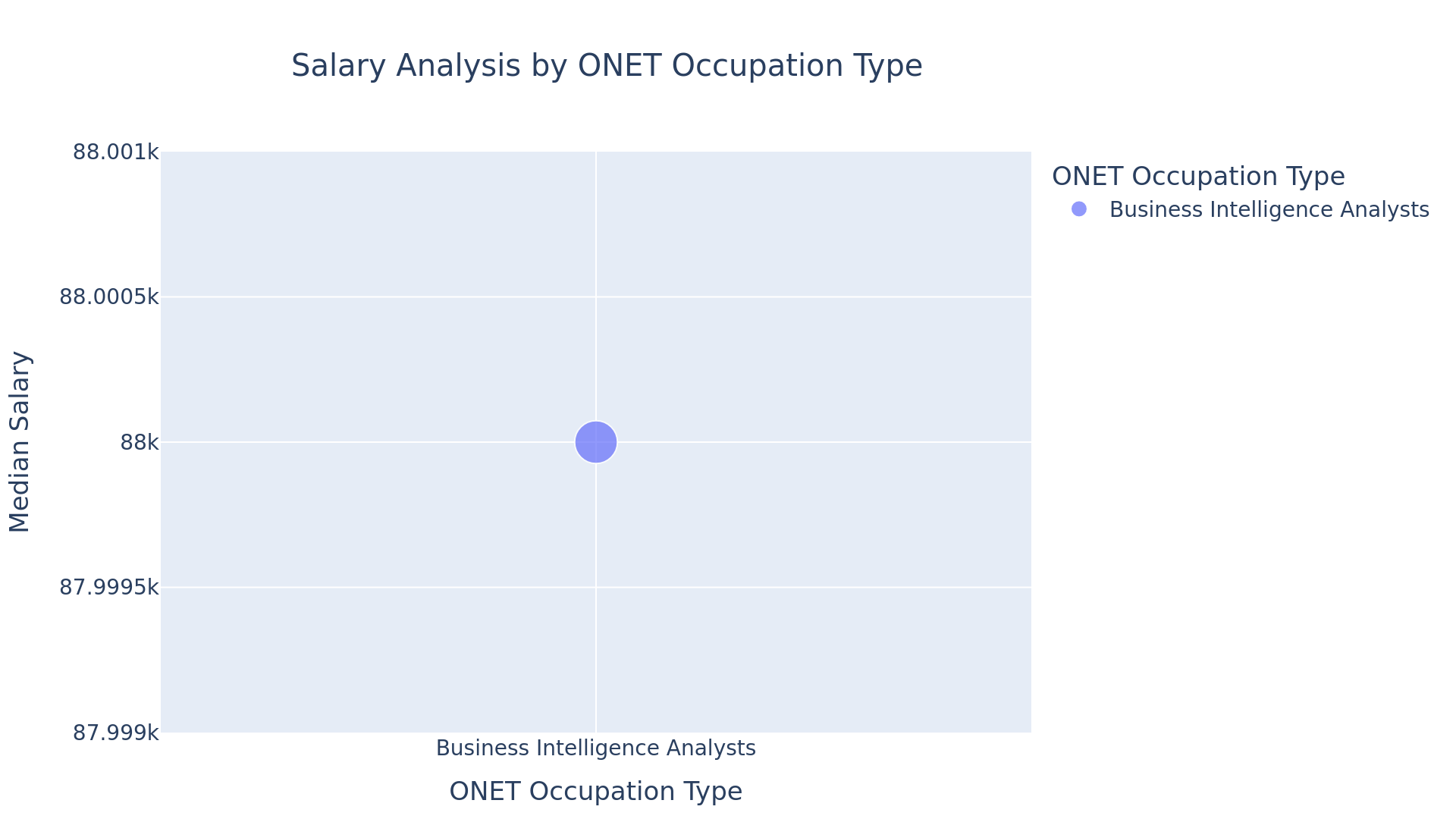
Question 6

Because there are too many types of skills, the images cannot accommodate all of them, so I selected the top ten for drawing. The graph illustrates the distribution of various skills across different industries, highlighting that “Professional, Scientific, and Technical Services” industry has the highest skill count, particularly in “Data Analysis” and “Management”. In contrast, industries like “Arts, Entertainment, and Recreation” and “Utilities” show lower skill counts across all categories.

# 7. 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.

# Your code for 7th question here  
from pyspark.sql import functions as F  
  
df\_aggregated = df.groupBy("ONET\_NAME").agg(  
 F.expr("percentile\_approx(SALARY\_FROM, 0.5)").alias("Median\_Salary"),  
 F.count("ID").alias("Job\_Postings")  
)  
df\_aggregated\_pd = df\_aggregated.toPandas()  
  
  
fig = px.scatter(  
 df\_aggregated\_pd,  
 x="ONET\_NAME",  
 y="Median\_Salary",  
 size="Job\_Postings",  
 color="ONET\_NAME",   
 title="Salary Analysis by ONET Occupation Type",  
 labels={'ONET\_NAME': 'ONET Occupation Type', 'Median\_Salary': 'Median Salary'},  
 color\_continuous\_scale="Viridis",  
)  
  
fig.update\_layout(  
 font=dict(family="Arial, sans-serif", size=14),  
 showlegend=True,  
 xaxis\_title="ONET Occupation Type",  
 yaxis\_title="Median Salary",  
 margin=dict(l=50, r=50, t=100, b=50),  
 title\_font\_size=20,   
 title\_x=0.2  
)  
  
fig.show()  
  
fig.write\_image("output/Q7.png", width=960, height=540, scale=2)



Question 7

There are only two types of data in ONET\_NAME: ‘Business Intelligence Analysts’ and’ null ‘, so this bubble chart only displays data from’ Business Intelligence Analysts’, and it shows that the median salary for Business Intelligence Analysts is $88,000.

# 8. Career Pathway Trends (Sankey Diagram)

* Visualize job transitions between different occupation levels.
* **Aggregate Data**
  + Identify career transitions between **SOC job classifications**.
* **Visualize results**
  + Create a **Sankey diagram** where:
    - **Source** = SOC\_2021\_2\_NAME
    - **Target** = SOC\_2021\_3\_NAME
    - **Value** = Number of transitions
  + Apply custom colors and font styles.
* **Explanation:** Write two sentences about what the graph reveals.

# Your code for 8th question here  
import pandas as pd  
import plotly.graph\_objects as go  
from pyspark.sql import SparkSession  
from pyspark.sql import functions as F  
  
  
# 步骤 1: 汇总数据  
# 统计 SOC\_2021\_2\_NAME 和 SOC\_2021\_3\_NAME 之间的转换次数  
df\_transitions = df.groupBy("SOC\_2021\_2\_NAME", "SOC\_2021\_3\_NAME").count()  
  
# 转换为 Pandas DataFrame  
df\_transitions\_pd = df\_transitions.toPandas()  
#如果数据量过大，桑基图可能会变得复杂。可以过滤掉转换次数较少的记录  
df\_transitions\_pd = df\_transitions\_pd[df\_transitions\_pd["count"] > 10]  
  
# 步骤 2: 创建桑基图  
# 获取唯一的来源和目标节点  
unique\_sources = df\_transitions\_pd["SOC\_2021\_2\_NAME"].unique()  
unique\_targets = df\_transitions\_pd["SOC\_2021\_3\_NAME"].unique()  
  
# 创建节点列表  
nodes = list(unique\_sources) + list(unique\_targets)  
  
# 创建节点索引映射  
node\_indices = {node: i for i, node in enumerate(nodes)}  
  
# 创建桑基图的链接数据  
links = {  
 "source": [node\_indices[source] for source in df\_transitions\_pd["SOC\_2021\_2\_NAME"]],  
 "target": [node\_indices[target] for target in df\_transitions\_pd["SOC\_2021\_3\_NAME"]],  
 "value": df\_transitions\_pd["count"].tolist()  
}  
  
# 步骤 3: 创建桑基图  
fig = go.Figure(go.Sankey(  
 node=dict(  
 pad=15, # 节点之间的间距  
 thickness=20, # 节点厚度  
 line=dict(color="black", width=0.5), # 节点边框样式  
 label=nodes, # 节点标签  
 color="blue" # 节点颜色  
 ),  
 link=dict(  
 source=links["source"], # 来源节点索引  
 target=links["target"], # 目标节点索引  
 value=links["value"], # 转换次数  
 color="rgba(0, 128, 0, 0.4)" # 链接颜色  
 )  
))  
  
# 步骤 4: 自定义布局和样式  
fig.update\_layout(  
 title="Career Pathway Trends (Sankey Diagram)",  
 font=dict(family="Arial", size=12, color="black"),  
 plot\_bgcolor="white", # 背景颜色  
 paper\_bgcolor="white", # 画布颜色  
 xaxis=dict(showgrid=False), # 隐藏 X 轴网格  
 yaxis=dict(showgrid=False), # 隐藏 Y 轴网格  
 title\_font\_size=20,   
 title\_x=0.5  
)  
  
# 显示桑基图  
fig.show()  
  
fig.write\_image("output/Q8.png", width=1920, height=540, scale=2)



Question 8

There is only one type of data in both “SOC\_2021\_2 NAME” and “SOC\_2021\_3 NAME”, so the generated Sankey diagram only has one line connecting Computer and Mathematical Occupations and Mathematical Science Occupations.