Assignment 03

Anu Sharma

September 24, 2025

Table of contents

### 0.1 Repository Link

https://github.com/met-ad-688/assignment-03-anush-09.git

### 0.2 Generative AI use declaration

1. Used generative AI to fix certain environment related errors during plot generation.
2. Used for more understanding on pyspark based query and df generation.

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, lit  
from pyspark.sql import functions as F  
from pyspark.sql.functions import col, monotonically\_increasing\_id  
  
np.random.seed(42)  
  
pio.renderers.default = "notebook"  
  
 # df = pd.read\_csv("./data/lightcast\_job\_postings.csv")  
 # unique\_values = df['ONET\_NAME'].unique()  
 # print(unique\_values)  
  
# 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)  
  
# Section 1.1  
df\_cleaned = df \  
 .withColumn("SALARY", col("SALARY").cast("integer")) \  
 .withColumn("SALARY\_FROM", col("SALARY\_FROM").cast("integer")) \  
 .withColumn("SALARY\_TO", col("SALARY\_TO").cast("integer")) \  
 .withColumn("MIN\_YEARS\_EXPERIENCE", col("MIN\_YEARS\_EXPERIENCE").cast("integer")) \  
 .withColumn("MAX\_YEARS\_EXPERIENCE", col("MAX\_YEARS\_EXPERIENCE").cast("integer"))  
  
# df\_cleaned = df\_cleaned.withColumn("ONET\_NAME", regexp\_replace("ONET\_NAME", "/", "-"))  
# df\_cleaned.show(5)  
  
# Section 1.2  
from pyspark.sql import functions as sf  
median\_from = df\_cleaned.approxQuantile("SALARY\_FROM", [0.5], 0.0)[0]  
median\_to = df\_cleaned.approxQuantile("SALARY\_TO", [0.5], 0.0)[0]  
median\_salary = df\_cleaned.approxQuantile("SALARY", [0.5], 0.0)[0]  
  
print("Median From:", median\_from)  
print("Median To:", median\_to)  
print("Median Salary:", median\_salary)  
  
# Section 1.3  
df\_imputed = df\_cleaned.withColumn(  
 "SALARY",  
 when(col("SALARY").isNull(), lit(median\_salary)).otherwise(col("SALARY"))  
)  
df\_imputed = df\_imputed.withColumn(  
 "SALARY\_FROM",  
 when(col("SALARY\_FROM").isNull(), lit(median\_from)).otherwise(col("SALARY\_FROM"))  
)  
df\_imputed = df\_imputed.withColumn(  
 "SALARY\_TO",  
 when(col("SALARY\_TO").isNull(), lit(median\_to)).otherwise(col("SALARY\_TO"))  
)  
  
df\_imputed = df\_imputed.withColumn(  
 "Average\_Salary",  
 ((col("SALARY\_FROM") + col("SALARY\_TO")) / 2)  
)  
  
# Section 1.4  
from pyspark.sql.functions import regexp\_replace  
df\_final = df\_imputed.withColumn(  
 "EDUCATION\_LEVELS\_NAME",  
 regexp\_replace("EDUCATION\_LEVELS\_NAME", "[\\n\\r]", "")  
)  
  
# Section 1.5  
# df\_cleaned.write.mode("overwrite").option("header", True).csv("data/cleaned\_data.csv")  
df\_final.write.mode("overwrite").parquet("data/cleaned\_data.parquet")  
print(f"Data cleaning complete. Rows retained: {df\_final.count()}")  
df\_final.createOrReplaceTempView("job\_postings")

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

---This is Diagnostic check, No need to print it in the final doc---

[Stage 52:> (0 + 1) / 1] [Stage 53:> (0 + 1) / 1] [Stage 54:> (0 + 1) / 1]

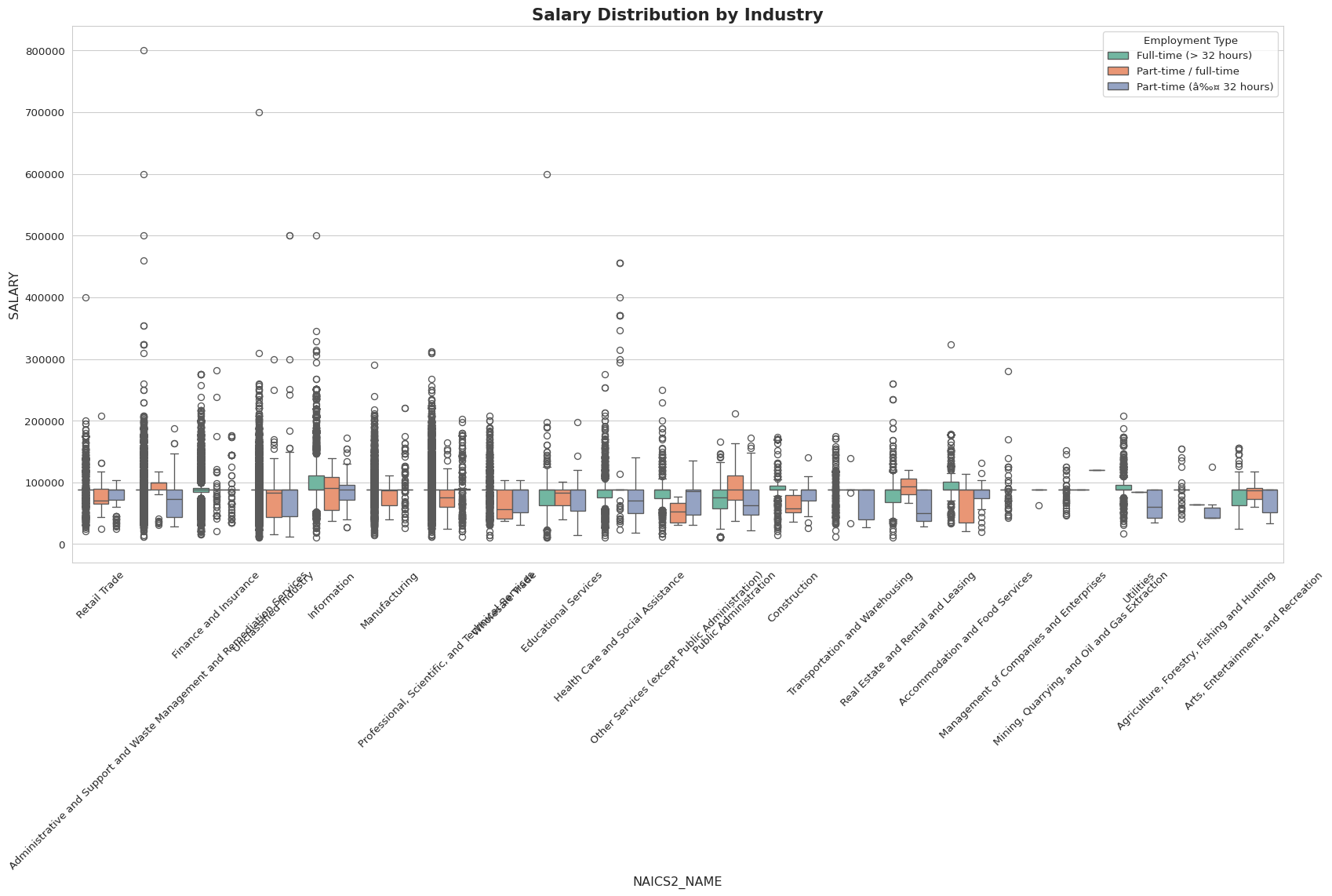
Median From: 88000.0  
Median To: 131040.0  
Median Salary: 116300.0

[Stage 55:> (0 + 1) / 1] [Stage 56:> (0 + 1) / 1]

Data cleaning complete. Rows retained: 72498

# Section 2  
salary\_distribution = spark.sql("""  
 SELECT   
 NAICS2\_NAME,  
 EMPLOYMENT\_TYPE\_NAME,  
 SALARY\_FROM  
 FROM job\_postings  
 WHERE SALARY\_FROM IS NOT NULL AND SALARY\_FROM > 0  
""")  
# salary\_distribution.show()  
df\_pd = salary\_distribution.toPandas()  
  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
plt.figure(figsize=(18, 12))  
sns.set\_style("whitegrid")  
  
sns.boxplot(  
 data=df\_pd,  
 x="NAICS2\_NAME",  
 y="SALARY\_FROM",  
 hue="EMPLOYMENT\_TYPE\_NAME",  
 palette="Set2"  
)  
  
plt.title("Salary Distribution by Industry", fontsize=16, fontweight="bold")  
plt.xlabel("NAICS2\_NAME", fontsize=12)  
plt.ylabel("SALARY", fontsize=12)  
plt.xticks(rotation=45)  
plt.legend(title="Employment Type", fontsize=10)  
plt.tight\_layout()  
plt.show()

[Stage 59:> (0 + 1) / 1]



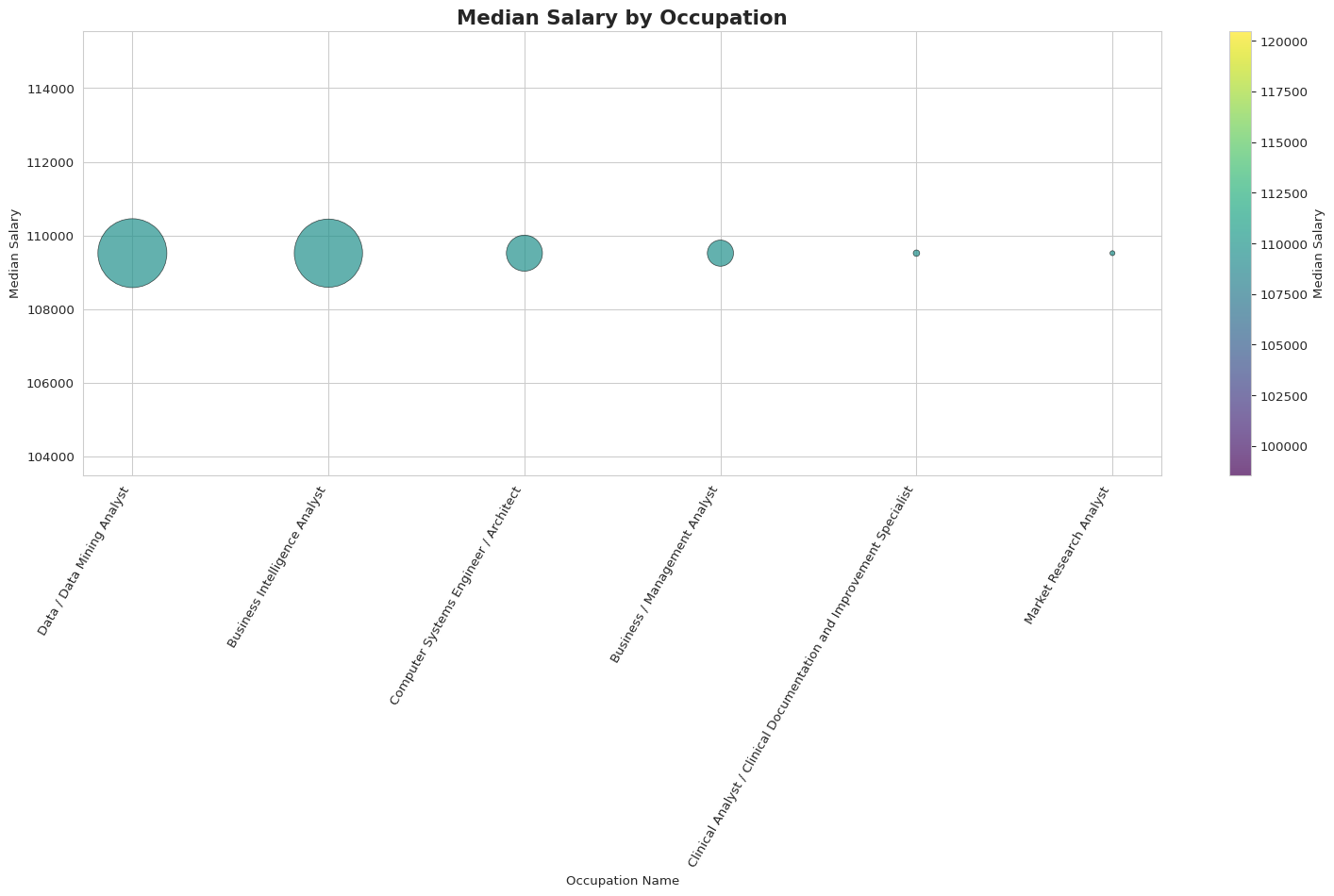
### 0.3 Inference

In most industries, the salary for full-time jobs is often around $88,000. But some sectors like Information, Health Care, and Professional Services show much higher salaries — going above $140,000 in many cases. This means technical and specialised roles are paying better. Also, “Unclassified Industry” appears frequently with varied salaries, which may indicate mixed or unclear job categories. Overall, salary depends a lot on the industry type and job role.

# Section 3  
  
  
# Bubble Chart: Median Salary by LOT\_OCCUPATION\_NAME and Job Count  
df\_bubble = spark.sql("""  
 SELECT  
 LOT\_OCCUPATION\_NAME,  
 percentile\_approx(Average\_Salary, 0.5) AS MEDIAN\_SALARY,  
 COUNT(\*) AS JOB\_COUNT  
 FROM job\_postings  
 WHERE Average\_Salary IS NOT NULL AND Average\_Salary > 0 AND LOT\_OCCUPATION\_NAME IS NOT NULL  
 GROUP BY LOT\_OCCUPATION\_NAME  
 ORDER BY JOB\_COUNT DESC  
""").toPandas()  
  
  
print("Top 5 occupations by job count and median salary:")  
print(df\_bubble.head())  
  
import matplotlib.pyplot as plt  
  
plt.figure(figsize=(16, 10))  
scatter = plt.scatter(  
 x=df\_bubble["LOT\_OCCUPATION\_NAME"],  
 y=df\_bubble["MEDIAN\_SALARY"],  
 s=df\_bubble["JOB\_COUNT"] / 10,   
 c=df\_bubble["MEDIAN\_SALARY"],  
 cmap="viridis",  
 alpha=0.7,  
 edgecolors="black",  
 linewidth=0.5  
)  
  
plt.xticks(rotation=60, ha="right")  
plt.title("Median Salary by Occupation", fontsize=16, fontweight="bold")  
plt.xlabel("Occupation Name")  
plt.ylabel("Median Salary")  
plt.colorbar(scatter, label="Median Salary")  
plt.tight\_layout()  
plt.show()

[Stage 60:> (0 + 1) / 1]

Top 5 occupations by job count and median salary:  
 LOT\_OCCUPATION\_NAME MEDIAN\_SALARY JOB\_COUNT  
0 Data / Data Mining Analyst 109520.0 30057  
1 Business Intelligence Analyst 109520.0 29445  
2 Computer Systems Engineer / Architect 109520.0 8212  
3 Business / Management Analyst 109520.0 4326  
4 Clinical Analyst / Clinical Documentation and ... 109520.0 261

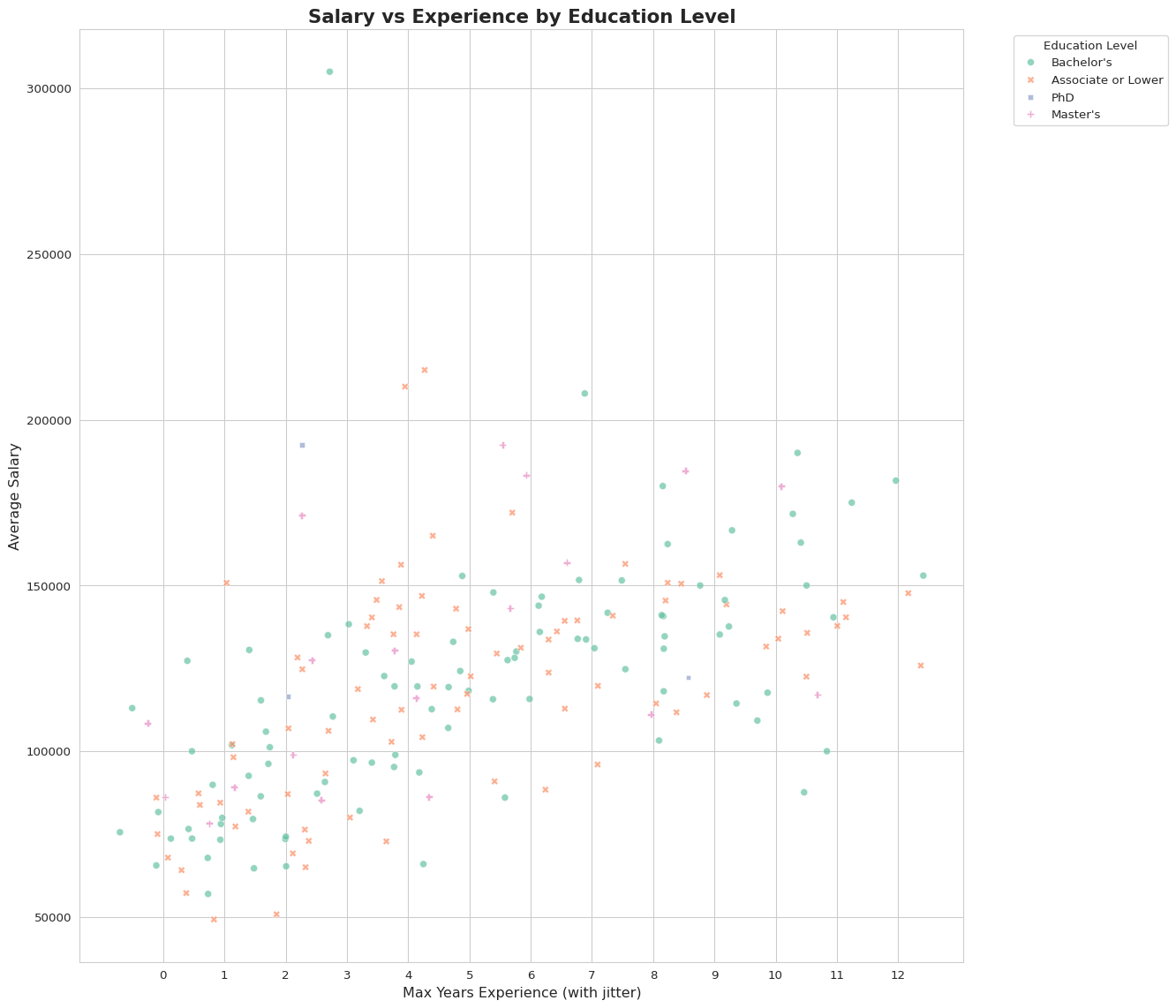


### 0.4 Inference

Most top job roles like Data Analyst, BI Analyst, and Systems Architect show the same median salary of $116,300 as the salary data was cleaned and fixed using median. These jobs also have very high number of openings, which means strong demand. But since all salaries are similar and ONET\_NAME only one category, so I used LOT\_OCCUPATION\_NAME to do this part of the assignment.

from pyspark.sql.functions import col, when  
import seaborn as sns  
import matplotlib.pyplot as plt  
import numpy as np  
  
# Section 4  
df\_grouped = df.withColumn("EDUCATION\_GROUP",  
 when(col("MIN\_EDULEVELS\_NAME").rlike("(?i)ged|associate|no education"), "Associate or Lower")  
 .when(col("MIN\_EDULEVELS\_NAME").rlike("(?i)bachelor"), "Bachelor's")  
 .when(col("MIN\_EDULEVELS\_NAME").rlike("(?i)master"), "Master's")  
 .when(col("MIN\_EDULEVELS\_NAME").rlike("(?i)ph\\.d|doctorate|professional"), "PhD")  
)  
  
  
df\_grouped = df\_grouped.filter(  
 col("EDUCATION\_GROUP").isNotNull() &  
 col("SALARY").isNotNull() &  
 col("MAX\_YEARS\_EXPERIENCE").isNotNull()  
)  
  
  
df\_grouped.createOrReplaceTempView("education\_jobs")  
  
  
scatter\_data = spark.sql("""  
 SELECT   
 EDUCATION\_GROUP,  
 LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME,  
 MAX\_YEARS\_EXPERIENCE,  
 AVG(SALARY) AS AVERAGE\_SALARY  
 FROM education\_jobs  
 GROUP BY EDUCATION\_GROUP, LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME, MAX\_YEARS\_EXPERIENCE  
""")  
  
# Scatter plt  
df\_plot = scatter\_data.toPandas()  
df\_plot["EXP\_WITH\_JITTER"] = df\_plot["MAX\_YEARS\_EXPERIENCE"] + np.random.normal(0, 0.5, size=len(df\_plot))  
  
plt.figure(figsize=(14, 12))  
sns.set\_style("whitegrid")  
  
sns.scatterplot(  
 data=df\_plot,  
 x="EXP\_WITH\_JITTER",  
 y="AVERAGE\_SALARY",  
 hue="EDUCATION\_GROUP",  
 style="EDUCATION\_GROUP",  
 palette="Set2",  
 alpha=0.7  
)  
  
plt.title("Salary vs Experience by Education Level", fontsize=16, fontweight="bold")  
plt.xlabel("Max Years Experience (with jitter)", fontsize=12)  
plt.ylabel("Average Salary", fontsize=12)  
min\_exp = int(df\_plot["EXP\_WITH\_JITTER"].min())  
max\_exp = int(df\_plot["EXP\_WITH\_JITTER"].max()) + 1  
plt.xticks(range(min\_exp, max\_exp, 1))  
plt.legend(title="Education Level", bbox\_to\_anchor=(1.05, 1), loc='upper left')  
plt.tight\_layout()  
plt.show()

[Stage 68:> (0 + 1) / 1]

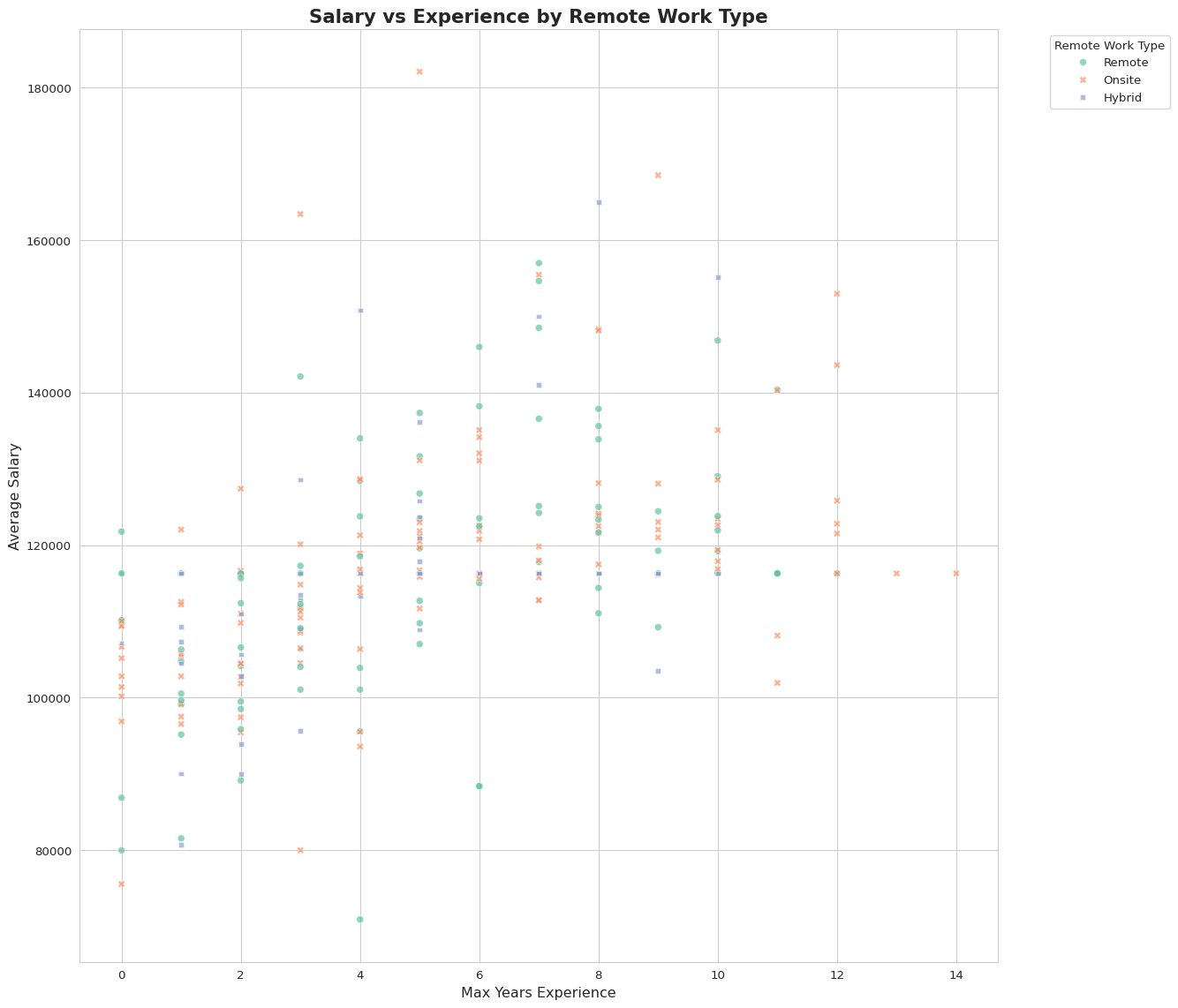


### 0.5 Inference:

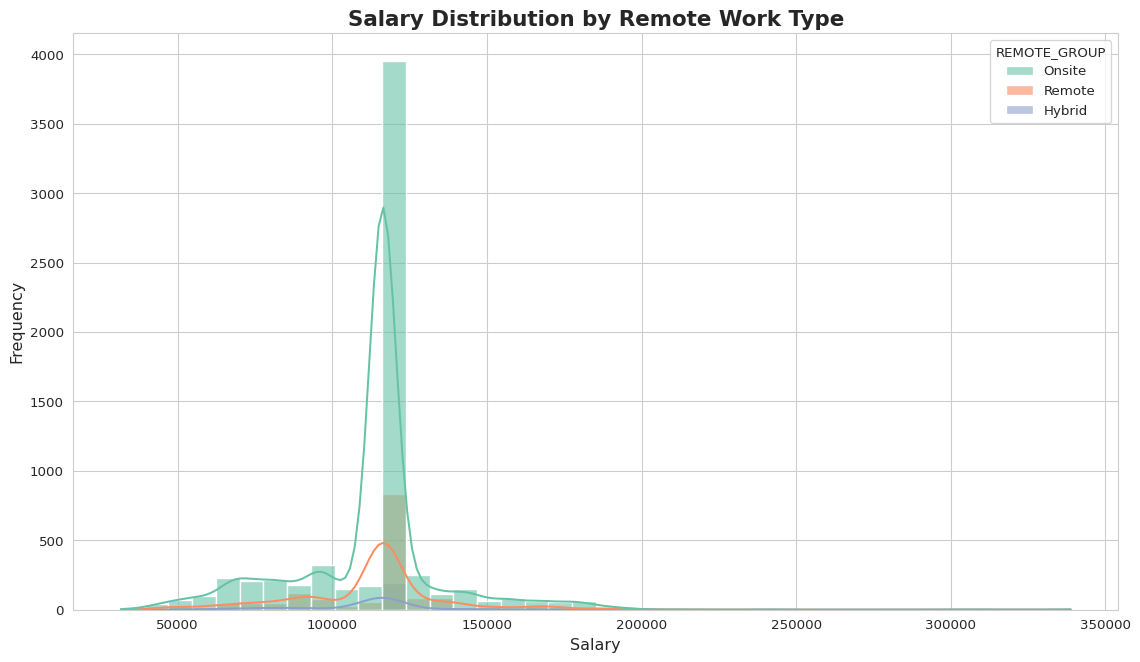
People with Associate or Lower education sometimes getting high salary like $156,000 for Business Intelligence Analyst and $139,000 for ERP Consultant, even they have 4-6 years experience. People with Bachelor degree also earning good, like Enterprise Architect getting more than $200,000 with 6-7 years experience. So, experience and job role looks more important than education level for salary.

from pyspark.sql.functions import when, col, trim, regexp\_replace  
import seaborn as sns  
import matplotlib.pyplot as plt  
import numpy as np  
  
df\_remote = spark.sql("SELECT \* FROM job\_postings")  
# spark.sql("""  
# SELECT DISTINCT REMOTE\_TYPE\_NAME  
# FROM job\_postings  
# WHERE REMOTE\_TYPE\_NAME IS NOT NULL  
# """).show(truncate=False)  
  
  
from pyspark.sql.functions import lower, trim, col, when  
df\_remote = df\_remote.withColumn("REMOTE\_TYPE\_NAME\_CLEAN", trim(lower(col("REMOTE\_TYPE\_NAME"))))  
df\_remote = df\_remote.withColumn("REMOTE\_GROUP",  
 when(col("REMOTE\_TYPE\_NAME\_CLEAN") == "not remote", "Onsite")  
 .when(col("REMOTE\_TYPE\_NAME\_CLEAN").contains("hybrid"), "Hybrid")  
 .when(col("REMOTE\_TYPE\_NAME\_CLEAN").contains("remote"), "Remote")  
 .otherwise("Onsite")  
)  
  
# Filter and create temp view  
df\_remote\_filtered = df\_remote.filter(  
 col("SALARY").isNotNull() &  
 col("MAX\_YEARS\_EXPERIENCE").isNotNull() &  
 col("LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME").isNotNull()  
)  
df\_remote\_filtered.createOrReplaceTempView("remote\_jobs")  
  
scatter\_data = spark.sql("""  
 SELECT   
 REMOTE\_GROUP,  
 LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME,  
 MAX\_YEARS\_EXPERIENCE,  
 AVG(SALARY) AS AVERAGE\_SALARY  
 FROM remote\_jobs  
 GROUP BY REMOTE\_GROUP, LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME, MAX\_YEARS\_EXPERIENCE  
""")  
  
# Scatter plot  
df\_plot = scatter\_data.toPandas()  
# Not using jitter as the image in assigment   
# document does not seem to be using jitter.  
df\_plot["EXP\_WITH\_JITTER"] = df\_plot["MAX\_YEARS\_EXPERIENCE"] #+ np.random.normal(0, 0.5, size=len(df\_plot))  
  
plt.figure(figsize=(14, 12))  
sns.set\_style("whitegrid")  
  
sns.scatterplot(  
 data=df\_plot,  
 x="EXP\_WITH\_JITTER",  
 y="AVERAGE\_SALARY",  
 hue="REMOTE\_GROUP",  
 style="REMOTE\_GROUP",  
 palette="Set2",  
 alpha=0.7  
)  
  
plt.title("Salary vs Experience by Remote Work Type", fontsize=16, fontweight="bold")  
plt.xlabel("Max Years Experience", fontsize=12)  
plt.ylabel("Average Salary", fontsize=12)  
plt.legend(title="Remote Work Type", bbox\_to\_anchor=(1.05, 1), loc='upper left')  
plt.tight\_layout()  
plt.show()  
  
# Histogram  
df\_hist = df\_remote\_filtered.select("SALARY", "REMOTE\_GROUP").toPandas()  
  
plt.figure(figsize=(12, 7))  
sns.histplot(data=df\_hist, x="SALARY", hue="REMOTE\_GROUP", bins=40, kde=True, palette="Set2", alpha=0.6)  
plt.title("Salary Distribution by Remote Work Type", fontsize=16, fontweight="bold")  
plt.xlabel("Salary", fontsize=12)  
plt.ylabel("Frequency", fontsize=12)  
plt.tight\_layout()  
plt.show()

[Stage 71:> (0 + 1) / 1]



[Stage 74:> (0 + 1) / 1]



### 0.6 Inference

Remote jobs giving good salary in many roles like Data Analyst and Enterprise Architect — some above $130,000 with 5–10 years experience. Onsite jobs also paying good, especially for senior roles like SAP Analyst and ERP Consultant. Hybrid jobs between remote and onsite, but some hybrid roles like SAP Analyst and Enterprise Architect getting very high salary too. So, salary mostly depend on job role and experience, not only work type.