In [2]:
```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("FinancialHealthAnalysis") \
    .getOrCreate()

print("✅ Spark Session Created Successfully!")
```

```
✅ Spark Session Created Successfully!
```

In [86]:
```
file_path = "AMC D49-FinancialHealth.csv"  # Update path if needed

df = spark.read.csv(file_path, header=True, inferSchema=True)
print("✅ Dataset Loaded Successfully!")
```

```
✅ Dataset Loaded Successfully!
```

In [6]:
```
df.printSchema()
df.show(5)
```

```
root
 |-- City Name: string (nullable = true)
 |-- Year: string (nullable = true)
 |-- Municipal's Revenue: double (nullable = true)
 |-- Municipal's Expenditure: double (nullable = true)
 |-- Total Profit / Loss: double (nullable = true)
 |-- _c5: string (nullable = true)
 |-- _c6: string (nullable = true)
 |-- _c7: string (nullable = true)
 |-- _c8: string (nullable = true)
 |-- _c9: string (nullable = true)
 |-- _c10: string (nullable = true)
 |-- _c11: string (nullable = true)
 |-- _c12: string (nullable = true)
 |-- _c13: string (nullable = true)
 |-- _c14: string (nullable = true)
 |-- _c15: string (nullable = true)
 |-- _c16: string (nullable = true)
 |-- _c17: string (nullable = true)
 |-- _c18: string (nullable = true)
 |-- _c19: string (nullable = true)
 |-- _c20: string (nullable = true)
 |-- _c21: string (nullable = true)

+---------+-------+------------------+----------------------
+-------------------+----+----+----+----+----+----+----+----+----+----+----
+----+----+----+----+----+----+
|City Name|   Year|Municipal's Revenue|Municipal's Expenditure|Total Profit / L
oss| _c5| _c6| _c7| _c8| _c9|_c10|_c11|_c12|_c13|_c14|_c15|_c16|_c17|_c18|_c1
9|_c20|_c21|
+---------+-------+------------------+----------------------
+-------------------+----+----+----+----+----+----+----+----+----+----+----
+----+----+----+----+----+----+
|    Ajmer|2017-18|           17352.86|               15691.13|            166
1.73|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NUL
L|NULL|NULL|
|    Ajmer|2016-17|           18833.88|               13165.58|             566
8.3|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NUL
L|NULL|NULL|
|    Ajmer|2015-16|           12585.55|                9521.61|            306
3.94|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NUL
L|NULL|NULL|
|    Ajmer|2014-15|           10323.43|                8601.01|            172
2.42|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NUL
L|NULL|NULL|
|     NULL|   NULL|               NULL|                   NULL|               N
ULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NUL
L|NULL|NULL|
+---------+-------+------------------+----------------------
+-------------------+----+----+----+----+----+----+----+----+----+----+----
+----+----+----+----+----+----+
only showing top 5 rows
```

In [8]:
```
# Number of rows and columns
```

```
print("Number of rows:", df.count())
print("Number of columns:", len(df.columns))

# Column names
print("Columns:", df.columns)
```

```
Number of rows: 959
Number of columns: 22
Columns: ['City Name', 'Year', "Municipal's Revenue", "Municipal's Expenditur
e", 'Total Profit / Loss', '_c5', '_c6', '_c7', '_c8', '_c9', '_c10', '_c11',
'_c12', '_c13', '_c14', '_c15', '_c16', '_c17', '_c18', '_c19', '_c20', '_c21']
```

In [10]:
```
df.describe().show()
```

```
+-------+---------+-------+------------------+----------------------
+-------------------+----+----+----+----+----+----+----+----+----+----+----
+----+----+----+----+----+----+
|summary|City Name|   Year|Municipal's Revenue|Municipal's Expenditure|Total Pr
ofit / Loss| _c5| _c6| _c7| _c8| _c9|_c10|_c11|_c12|_c13|_c14|_c15|_c16|_c17|_c
18|_c19|_c20|_c21|
+-------+---------+-------+------------------+----------------------
+-------------------+----+----+----+----+----+----+----+----+----+----+----
+----+----+----+----+----+----+
|  count|        4|      4|                 4|                     4|
4|   0|   0|   0|   0|   0|   0|   0|   0|   0|   0|   0|   0|   0|   0|
0|   0|
|   mean|     NULL|   NULL| 14773.930000000002|             11744.8325| 3029.09
75000000003|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NU
LL|NULL|NULL|NULL|
| stddev|     NULL|   NULL|  3988.735439618261|     3287.2544579164633|  1874.7
18094796744|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NU
LL|NULL|NULL|NULL|
|    min|    Ajmer|2014-15|           10323.43|                8601.01|
1661.73|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|N
ULL|NULL|NULL|
|    max|    Ajmer|2017-18|           18833.88|               15691.13|
5668.3|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NU
LL|NULL|NULL|
+-------+---------+-------+------------------+----------------------
+-------------------+----+----+----+----+----+----+----+----+----+----+----
+----+----+----+----+----+----+
```

In [12]:
```
from pyspark.sql.functions import col, when, count, isnan

df.select([count(when(col(c).isNull(), c)).alias(c) for c in df.columns]).show
```

```
+---------+----+------------------+----------------------+------------------
+---+---+---+---+---+----+----+----+----+----+----+----+----+----+----+----
+----+
|City Name|Year|Municipal's Revenue|Municipal's Expenditure|Total Profit / Los
s|_c5|_c6|_c7|_c8|_c9|_c10|_c11|_c12|_c13|_c14|_c15|_c16|_c17|_c18|_c19|_c20|_c
21|
+---------+----+------------------+----------------------+------------------
+---+---+---+---+---+----+----+----+----+----+----+----+----+----+----+----
+----+
|      955| 955|               955|                   955|               95
5|959|959|959|959|959| 959| 959| 959| 959| 959| 959| 959| 959| 959| 959| 9
59|
+---------+----+------------------+----------------------+------------------
+---+---+---+---+---+----+----+----+----+----+----+----+----+----+----+----
+----+
```

In [14]:
```
df_clean = df.dropna()
print("✅ Null values dropped. Clean dataset ready.")
```

```
✅ Null values dropped. Clean dataset ready.
```

In [18]:
```
df.select("City Name", "Municipal's Revenue", "Total Profit / Loss").show(5)
```

```
+---------+------------------+------------------+
|City Name|Municipal's Revenue|Total Profit / Loss|
+---------+------------------+------------------+
|    Ajmer|          17352.86|           1661.73|
|    Ajmer|          18833.88|            5668.3|
|    Ajmer|          12585.55|           3063.94|
|    Ajmer|          10323.43|           1722.42|
|     NULL|              NULL|              NULL|
+---------+------------------+------------------+
only showing top 5 rows
```

In [20]:
```
df.filter(col("Total Profit / Loss") > 1000).show(5)
```

```
+---------+-------+------------------+----------------------
+-------------------+----+----+----+----+----+----+----+----+----+----+----
+----+----+----+----+----+----+
|City Name|   Year|Municipal's Revenue|Municipal's Expenditure|Total Profit / L
oss| _c5| _c6| _c7| _c8| _c9|_c10|_c11|_c12|_c13|_c14|_c15|_c16|_c17|_c18|_c1
9|_c20|_c21|
+---------+-------+------------------+----------------------
+-------------------+----+----+----+----+----+----+----+----+----+----+----
+----+----+----+----+----+----+
|    Ajmer|2017-18|           17352.86|               15691.13|            166
1.73|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NUL
L|NULL|NULL|
|    Ajmer|2016-17|           18833.88|               13165.58|             566
8.3|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NUL
L|NULL|NULL|
|    Ajmer|2015-16|           12585.55|                9521.61|            306
3.94|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NUL
L|NULL|NULL|
|    Ajmer|2014-15|           10323.43|                8601.01|            172
2.42|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NULL|NUL
L|NULL|NULL|
+---------+-------+------------------+----------------------
+-------------------+----+----+----+----+----+----+----+----+----+----+----
+----+----+----+----+----+----+
```

In [22]:
```
from pyspark.sql.functions import col

df = df.withColumn("Profit_Margin", (col("Total Profit / Loss") / col("Municip
df.select("City Name", "Municipal's Revenue", "Total Profit / Loss", "Profit_M
```

```
+---------+-------------------+-------------------+------------------+
|City Name|Municipal's Revenue|Total Profit / Loss|     Profit_Margin|
+---------+-------------------+-------------------+------------------+
|    Ajmer|           17352.86|            1661.73| 9.576115983186632|
|    Ajmer|           18833.88|             5668.3|30.096294550034298|
|    Ajmer|           12585.55|            3063.94| 24.34490348057892|
|    Ajmer|           10323.43|            1722.42| 16.68457092264877|
|     NULL|               NULL|               NULL|              NULL|
+---------+-------------------+-------------------+------------------+
only showing top 5 rows
```

In [24]:
```
df.groupBy("City Name").agg({"Total Profit / Loss": "sum"}).orderBy("sum(Total
```

```
+---------+------------------------+
|City Name|sum(Total Profit / Loss)|
+---------+------------------------+
|    Ajmer|      12116.390000000001|
|     NULL|                    NULL|
+---------+------------------------+
```

In [26]:
```
df.groupBy("Year").agg({"Municipal's Revenue": "avg"}).orderBy("Year").show()
```

```
+-------+------------------------+
|   Year|avg(Municipal's Revenue)|
+-------+------------------------+
|   NULL|                    NULL|
|2014-15|                10323.43|
|2015-16|                12585.55|
|2016-17|                18833.88|
|2017-18|                17352.86|
+-------+------------------------+
```

In [64]:
```
df.groupBy("Year") \
  .agg({"Municipal's Expenditure": "sum"}) \
  .orderBy("Year") \
  .show()
```

```
+-------+----------------------------+
|   Year|sum(Municipal's Expenditure)|
+-------+----------------------------+
|   NULL|                        NULL|
|2014-15|                     8601.01|
|2015-16|                     9521.61|
|2016-17|                    13165.58|
|2017-18|                    15691.13|
+-------+----------------------------+
```

In [66]:
```
df.groupBy("Year") \
  .agg({"Total Profit / Loss": "sum"}) \
  .orderBy("Year") \
  .show()
```

```
+-------+------------------------+
|   Year|sum(Total Profit / Loss)|
+-------+------------------------+
|   NULL|                    NULL|
|2014-15|                 1722.42|
|2015-16|                 3063.94|
|2016-17|                  5668.3|
|2017-18|                 1661.73|
+-------+------------------------+
```

In [76]:
```
from pyspark.sql import functions as F
from pyspark.sql.window import Window

# Rename column to remove special character
df = df.withColumnRenamed("Municipal's Revenue", "Municipal_Revenue")

# Compute total revenue per year
revenue_yearly = df.groupBy("Year").agg(F.sum("Municipal_Revenue").alias("Tota

# Define window for previous year
windowSpec = Window.orderBy("Year")
```

```
# Add previous year revenue
revenue_growth = revenue_yearly.withColumn("Prev_Revenue", F.lag("Total_Revenu

# Calculate growth percentage
revenue_growth = revenue_growth.withColumn(
    "Revenue_Growth(%)",
    ((F.col("Total_Revenue") - F.col("Prev_Revenue")) / F.col("Prev_Revenue"))
)

revenue_growth.show()
```

```
+-------+-------------+------------+------------------+
|   Year|Total_Revenue|Prev_Revenue| Revenue_Growth(%)|
+-------+-------------+------------+------------------+
|   NULL|         NULL|        NULL|              NULL|
|2014-15|     10323.43|        NULL|              NULL|
|2015-16|     12585.55|    10323.43|21.912484513383625|
|2016-17|     18833.88|    12585.55| 49.64685691129909|
|2017-18|     17352.86|    18833.88|-7.863594755833637|
+-------+-------------+------------+------------------+
```

In [78]:
```
df_city_profit = df.groupBy("City Name").agg(F.avg("Profit_Margin").alias("Avg
df_city_profit.orderBy(desc("Avg_Profit_Margin")).show(10)
```

```
+---------+------------------+
|City Name| Avg_Profit_Margin|
+---------+------------------+
|    Ajmer|20.175471234112155|
|     NULL|              NULL|
+---------+------------------+
```

In [80]:
```
df = df.withColumn(
    "Performance_Category",
    F.when(col("Profit_Margin") > 20, "Excellent")
     .when((col("Profit_Margin") <= 20) & (col("Profit_Margin") > 10), "Good")
     .when((col("Profit_Margin") <= 10) & (col("Profit_Margin") > 0), "Average
     .otherwise("Loss")
)
df.select("City Name", "Year", "Profit_Margin", "Performance_Category").show(1
```

```
+---------+-------+------------------+--------------------+
|City Name|   Year|     Profit_Margin|Performance_Category|
+---------+-------+------------------+--------------------+
|    Ajmer|2017-18| 9.576115983186632|             Average|
|    Ajmer|2016-17|30.096294550034298|           Excellent|
|    Ajmer|2015-16| 24.34490348057892|           Excellent|
|    Ajmer|2014-15| 16.68457092264877|                Good|
|     NULL|   NULL|              NULL|                Loss|
|     NULL|   NULL|              NULL|                Loss|
|     NULL|   NULL|              NULL|                Loss|
|     NULL|   NULL|              NULL|                Loss|
|     NULL|   NULL|              NULL|                Loss|
|     NULL|   NULL|              NULL|                Loss|
+---------+-------+------------------+--------------------+
only showing top 10 rows
```

In [84]:
```
df = df.withColumn(
    "Performance_Category",
    F.when(col("Profit_Margin") > 20, "Excellent")
     .when((col("Profit_Margin") <= 20) & (col("Profit_Margin") > 10), "Good")
     .when((col("Profit_Margin") <= 10) & (col("Profit_Margin") > 0), "Average
     .otherwise("Loss")
)
df.select("City Name", "Year", "Profit_Margin", "Performance_Category").show(1
```

```
+---------+-------+------------------+--------------------+
|City Name|   Year|     Profit_Margin|Performance_Category|
+---------+-------+------------------+--------------------+
|    Ajmer|2017-18| 9.576115983186632|             Average|
|    Ajmer|2016-17|30.096294550034298|           Excellent|
|    Ajmer|2015-16| 24.34490348057892|           Excellent|
|    Ajmer|2014-15| 16.68457092264877|                Good|
|     NULL|   NULL|              NULL|                Loss|
|     NULL|   NULL|              NULL|                Loss|
|     NULL|   NULL|              NULL|                Loss|
|     NULL|   NULL|              NULL|                Loss|
|     NULL|   NULL|              NULL|                Loss|
|     NULL|   NULL|              NULL|                Loss|
+---------+-------+------------------+--------------------+
only showing top 10 rows
```

In [28]:
```
pdf = df.toPandas()
print("✅ Converted to Pandas for Visualization.")
```

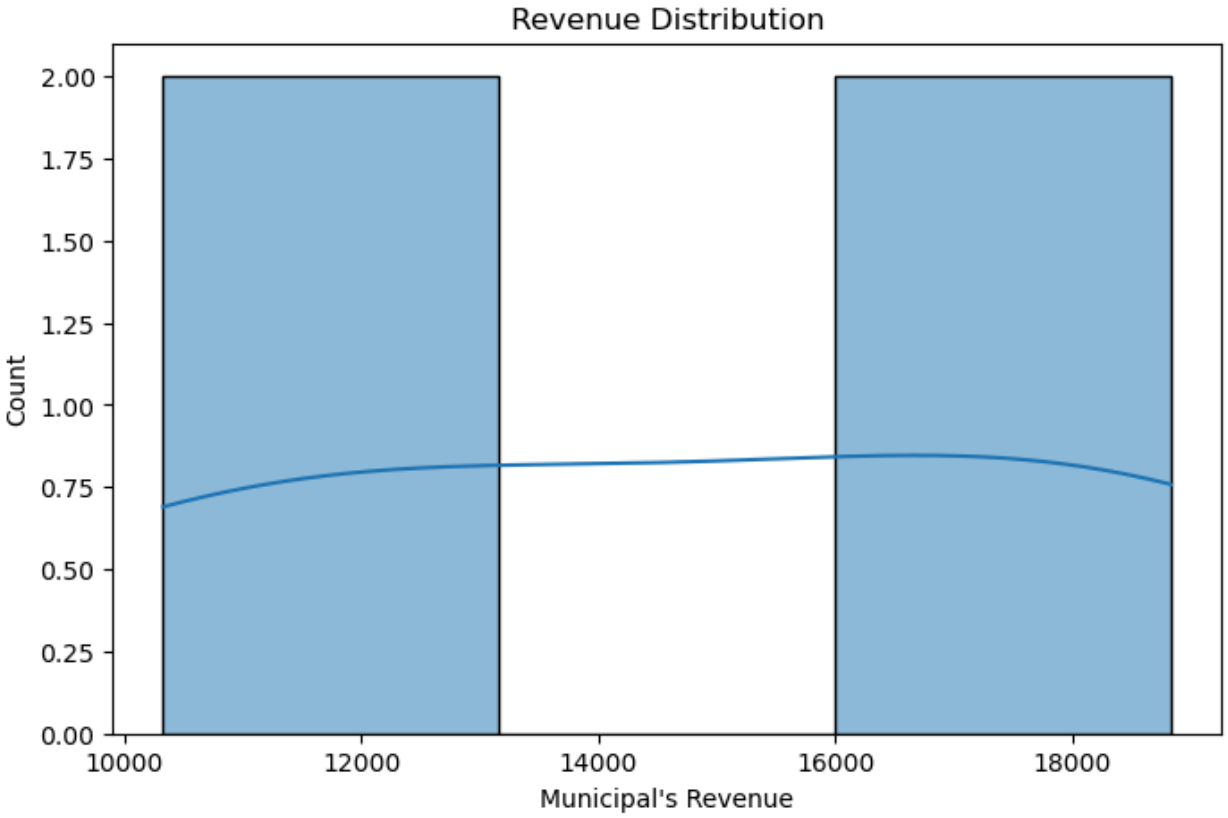
```
✅ Converted to Pandas for Visualization.
```

In [30]:
```
!pip install matplotlib seaborn
```

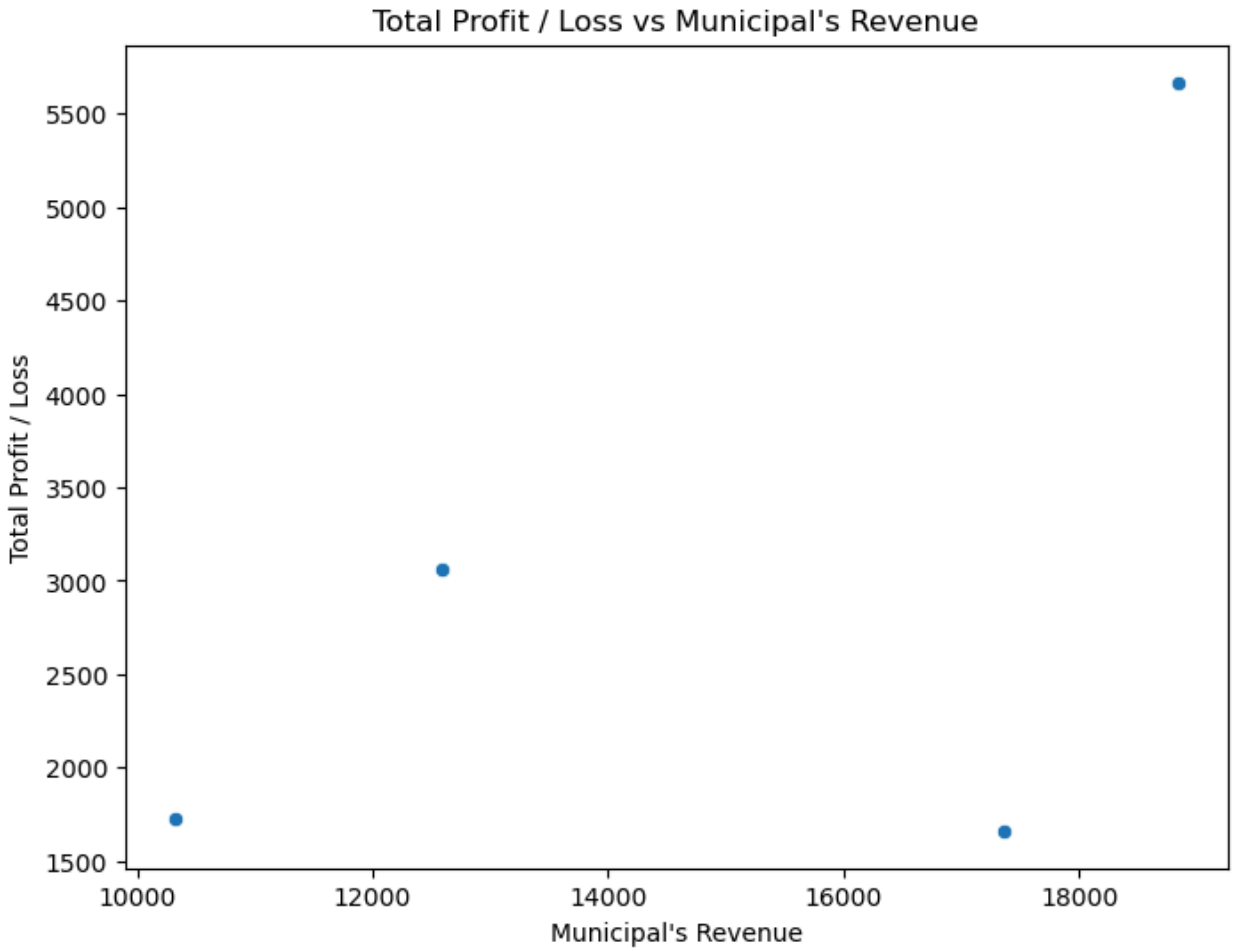
```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: matplotlib in c:\program files\anaconda3\lib\sit
e-packages (3.9.2)
Requirement already satisfied: seaborn in c:\program files\anaconda3\lib\site-p
ackages (0.13.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\program files\anaconda3\l
ib\site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\program files\anaconda3\lib\s
ite-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\program files\anaconda3\
lib\site-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\program files\anaconda3\
lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: numpy>=1.23 in c:\program files\anaconda3\lib\si
te-packages (from matplotlib) (1.26.4)
Requirement already satisfied: packaging>=20.0 in c:\program files\anaconda3\li
b\site-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=8 in c:\program files\anaconda3\lib\sit
e-packages (from matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\program files\anaconda3\l
ib\site-packages (from matplotlib) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\program files\anacond
a3\lib\site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: pandas>=1.2 in c:\program files\anaconda3\lib\si
te-packages (from seaborn) (2.2.2)
Requirement already satisfied: pytz>=2020.1 in c:\program files\anaconda3\lib\s
ite-packages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\program files\anaconda3\li
b\site-packages (from pandas>=1.2->seaborn) (2023.3)
Requirement already satisfied: six>=1.5 in c:\program files\anaconda3\lib\site-
packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

In [36]:

```python
import matplotlib.pyplot as plt
import seaborn as sns

if "Municipal's Revenue" in pdf.columns:
    plt.figure(figsize=(8,5))
    sns.histplot(pdf["Municipal's Revenue"], kde=True)
    plt.title("Revenue Distribution")
    plt.show()
```

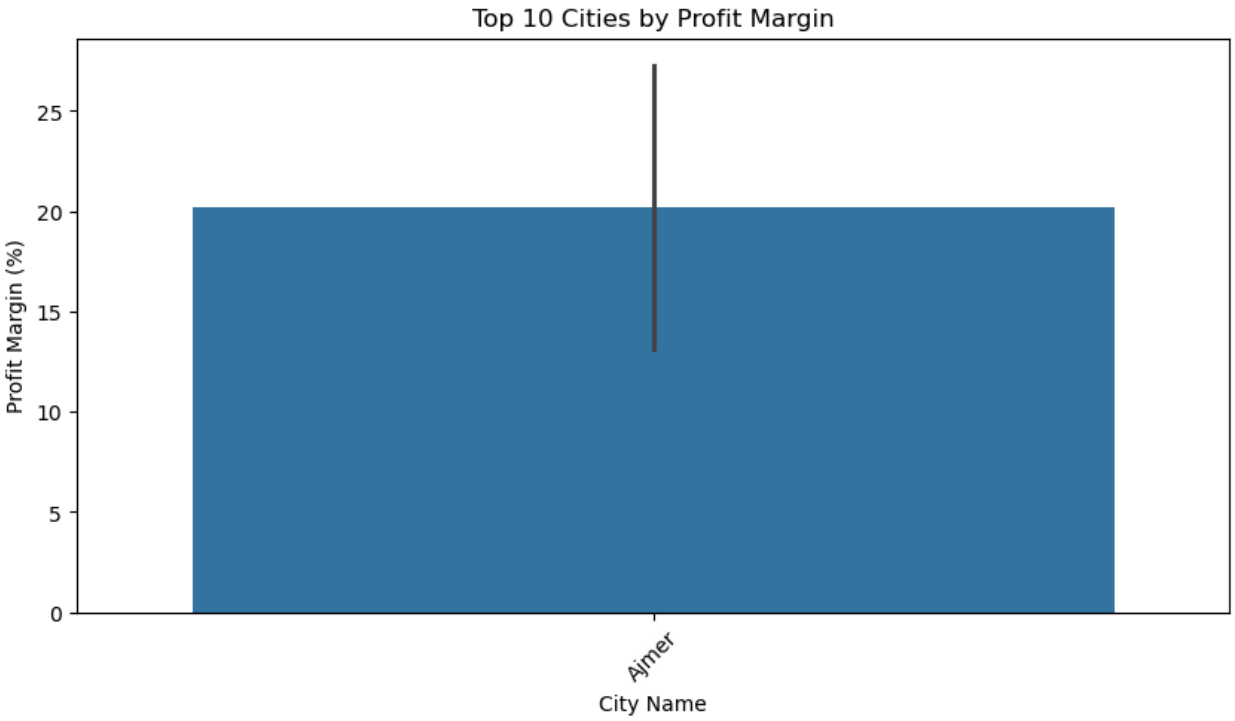


In [44]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Ensure plots show inside Jupyter
%matplotlib inline

# Use correct column names
if {"Total Profit / Loss", "Municipal's Revenue"}.issubset(pdf.columns):
    plt.figure(figsize=(8,6))
    sns.scatterplot(x="Municipal's Revenue", y="Total Profit / Loss", data=pdf
    plt.title("Total Profit / Loss vs Municipal's Revenue")
    plt.xlabel("Municipal's Revenue")
    plt.ylabel("Total Profit / Loss")
    plt.show()
```



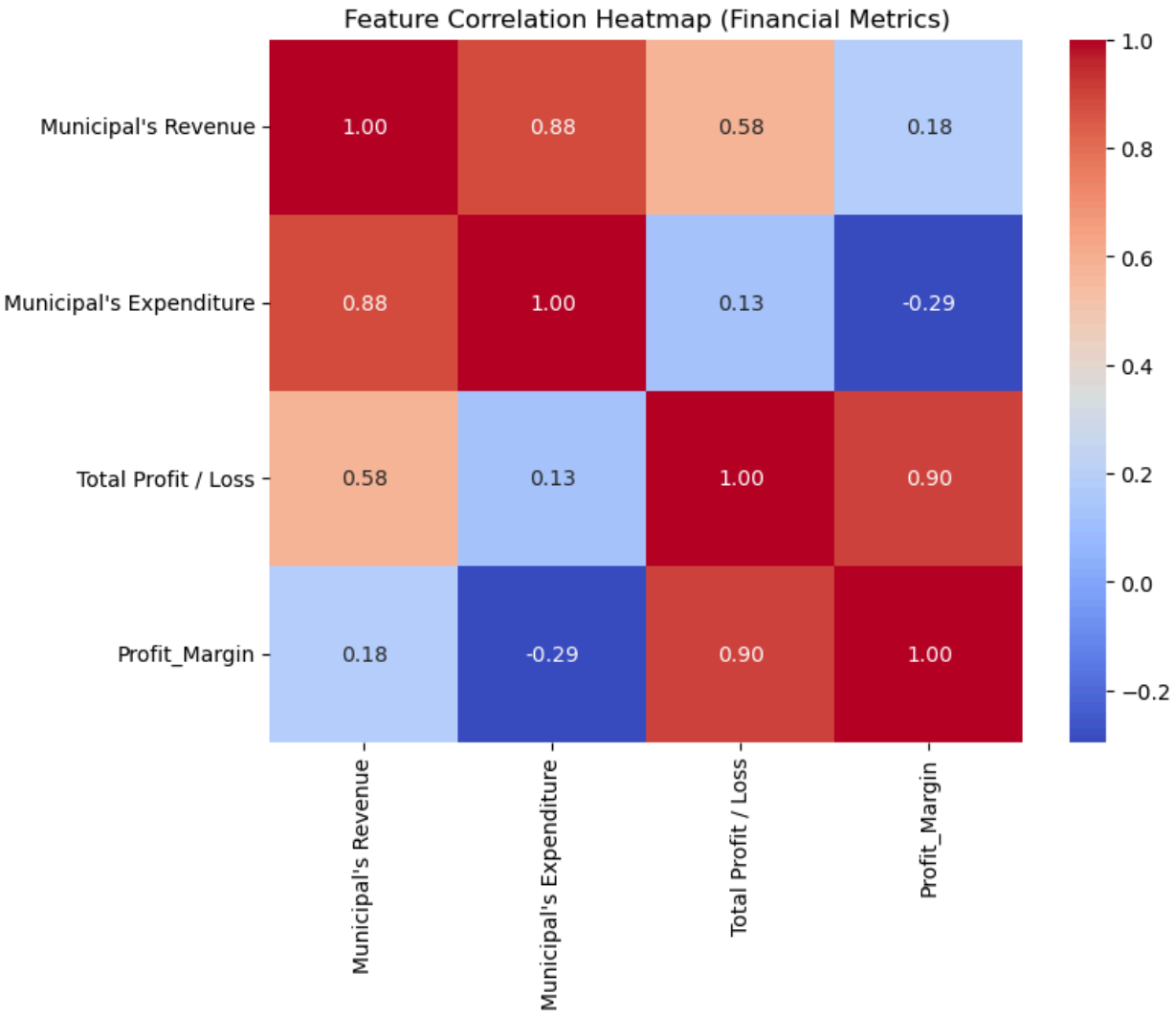
In [48]:

```python
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Ensure the Profit_Margin column exists (create it if needed)
# if "Profit_Margin" not in pdf.columns and \
#    {"Municipal's Revenue", "Total Profit / Loss"}.issubset(pdf.columns):
#     pdf["Municipal's Revenue"] = pd.to_numeric(pdf["Municipal's Revenue"], e
#     pdf["Total Profit / Loss"] = pd.to_numeric(pdf["Total Profit / Loss"], e
#     pdf["Profit_Margin"] = (pdf["Total Profit / Loss"] / pdf["Municipal's Re

# Plot the Top 10 Cities by Profit Margin
if {"City Name", "Profit_Margin"}.issubset(pdf.columns):
    top10 = pdf.sort_values("Profit_Margin", ascending=False).head(10)
    plt.figure(figsize=(10,5))
    sns.barplot(x="City Name", y="Profit_Margin", data=top10)
    plt.title("Top 10 Cities by Profit Margin")
    plt.xlabel("City Name")
    plt.ylabel("Profit Margin (%)")
    plt.xticks(rotation=45)
    plt.show()
else:
    print("❌ Required columns not found. Available columns:", pdf.columns.tol
```



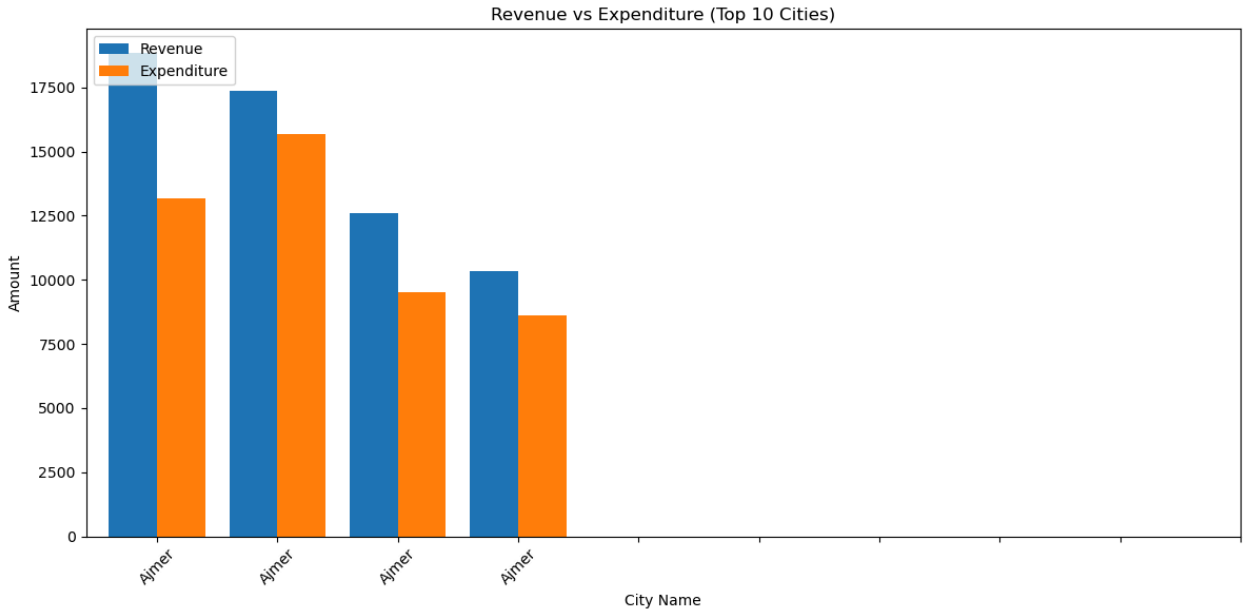
In [50]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Convert financial columns to numeric (if not already)
for col in ["Municipal's Revenue", "Municipal's Expenditure", "Total Profit /
    if col in pdf.columns:
        pdf[col] = pd.to_numeric(pdf[col], errors='coerce')

# Compute correlation only for numeric columns
numeric_df = pdf.select_dtypes(include=['float64', 'int64'])

# Check if there are numeric columns before plotting
if not numeric_df.empty:
    plt.figure(figsize=(8,6))
    sns.heatmap(numeric_df.corr(), annot=True, cmap="coolwarm", fmt=".2f")
    plt.title("Feature Correlation Heatmap (Financial Metrics)")
    plt.show()
else:
    print("❌ No numeric columns found for correlation heatmap.")
```

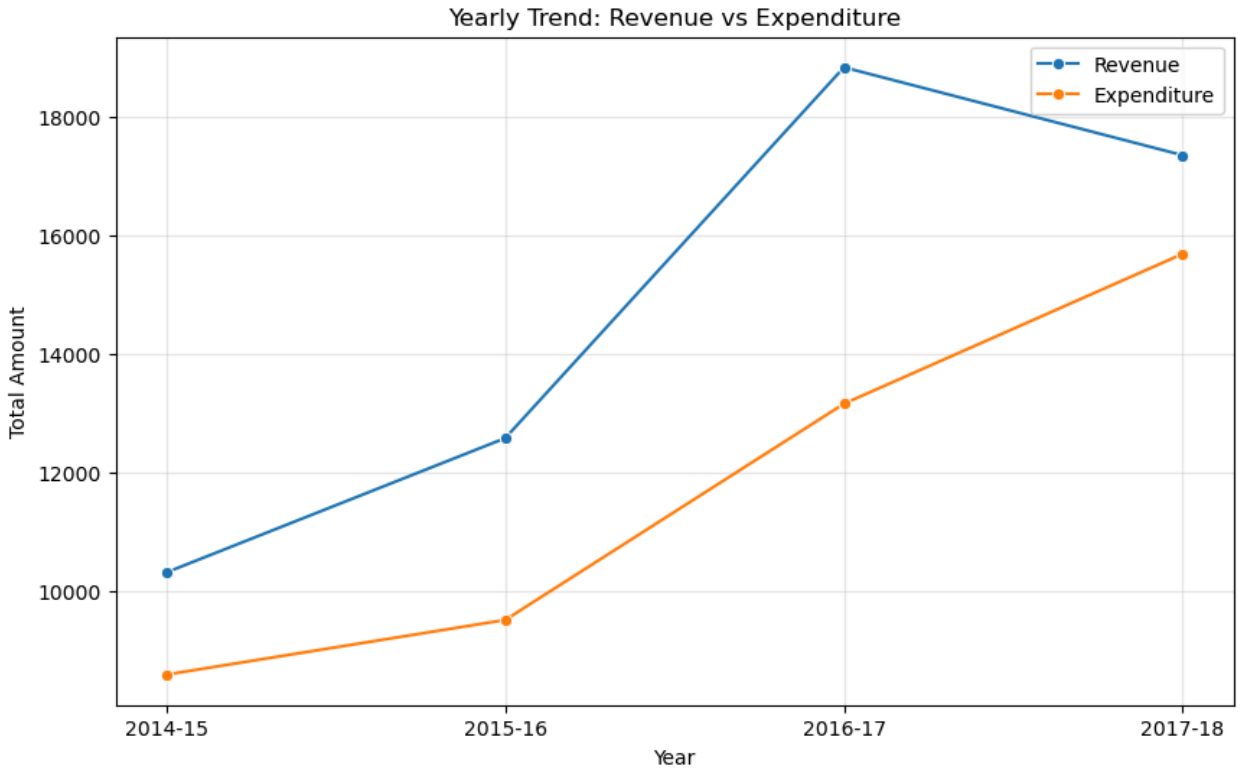


In [52]:

```
import matplotlib.pyplot as plt
import seaborn as sns

top_cities = pdf.sort_values("Municipal's Revenue", ascending=False).head(10)

plt.figure(figsize=(12,6))
bar_width = 0.4
x = range(len(top_cities))

plt.bar(x, top_cities["Municipal's Revenue"], width=bar_width, label="Revenue"
plt.bar([i + bar_width for i in x], top_cities["Municipal's Expenditure"], wid

plt.xticks([i + bar_width/2 for i in x], top_cities["City Name"], rotation=45)
plt.xlabel("City Name")
plt.ylabel("Amount")
plt.title("Revenue vs Expenditure (Top 10 Cities)")
plt.legend()
plt.tight_layout()
plt.show()
```

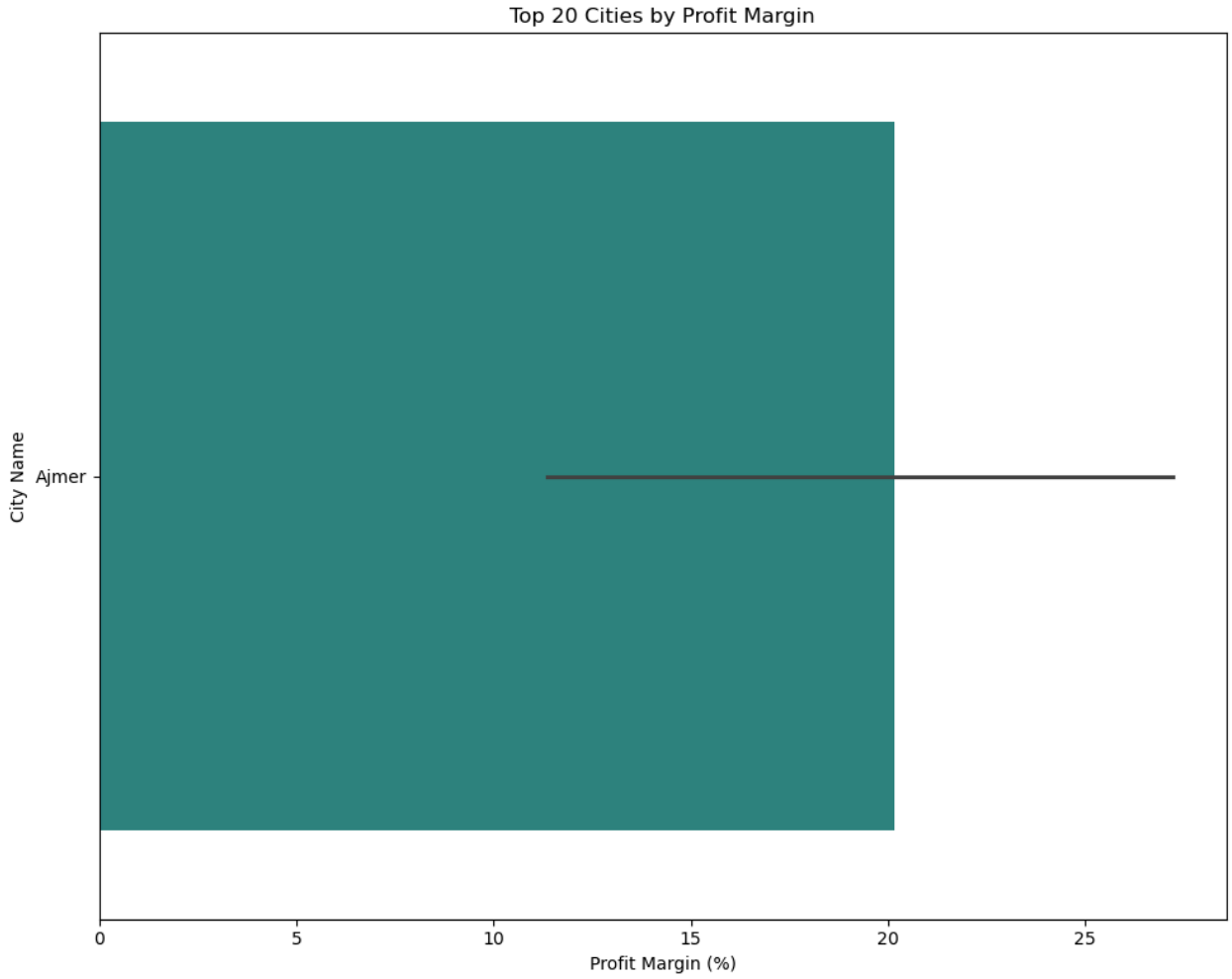


In [54]:
```
if "Year" in pdf.columns:
    yearly = pdf.groupby("Year")[["Municipal's Revenue", "Municipal's Expendit

    plt.figure(figsize=(10,6))
    sns.lineplot(x="Year", y="Municipal's Revenue", data=yearly, marker='o', l
    sns.lineplot(x="Year", y="Municipal's Expenditure", data=yearly, marker='o
    plt.title("Yearly Trend: Revenue vs Expenditure")
    plt.xlabel("Year")
    plt.ylabel("Total Amount")
    plt.legend()
    plt.grid(True, alpha=0.3)
    plt.show()
```

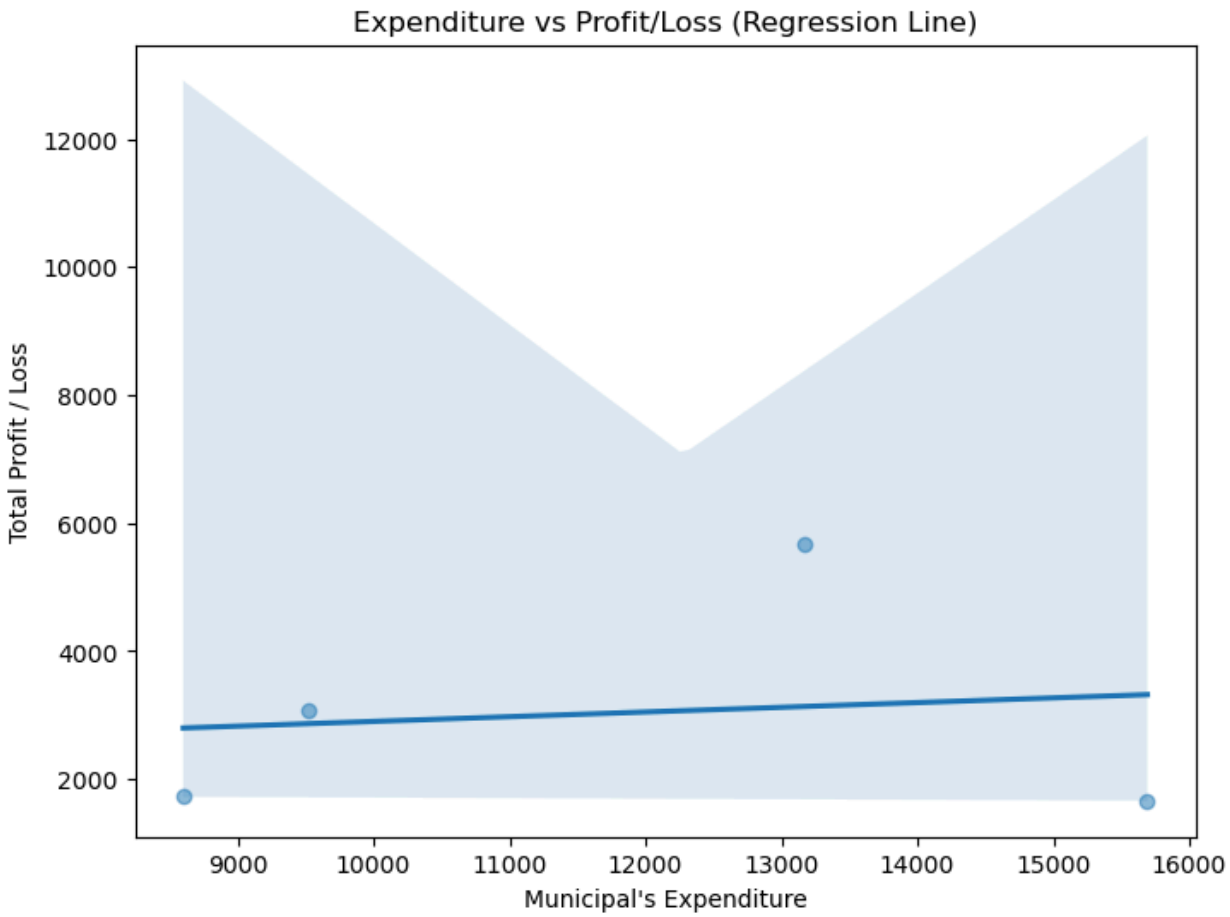


In [56]:
```
top20 = pdf.sort_values("Profit_Margin", ascending=False).head(20)

plt.figure(figsize=(10,8))
sns.barplot(y="City Name", x="Profit_Margin", data=top20, palette="viridis")
plt.title("Top 20 Cities by Profit Margin")
plt.xlabel("Profit Margin (%)")
plt.ylabel("City Name")
plt.tight_layout()
plt.show()
```

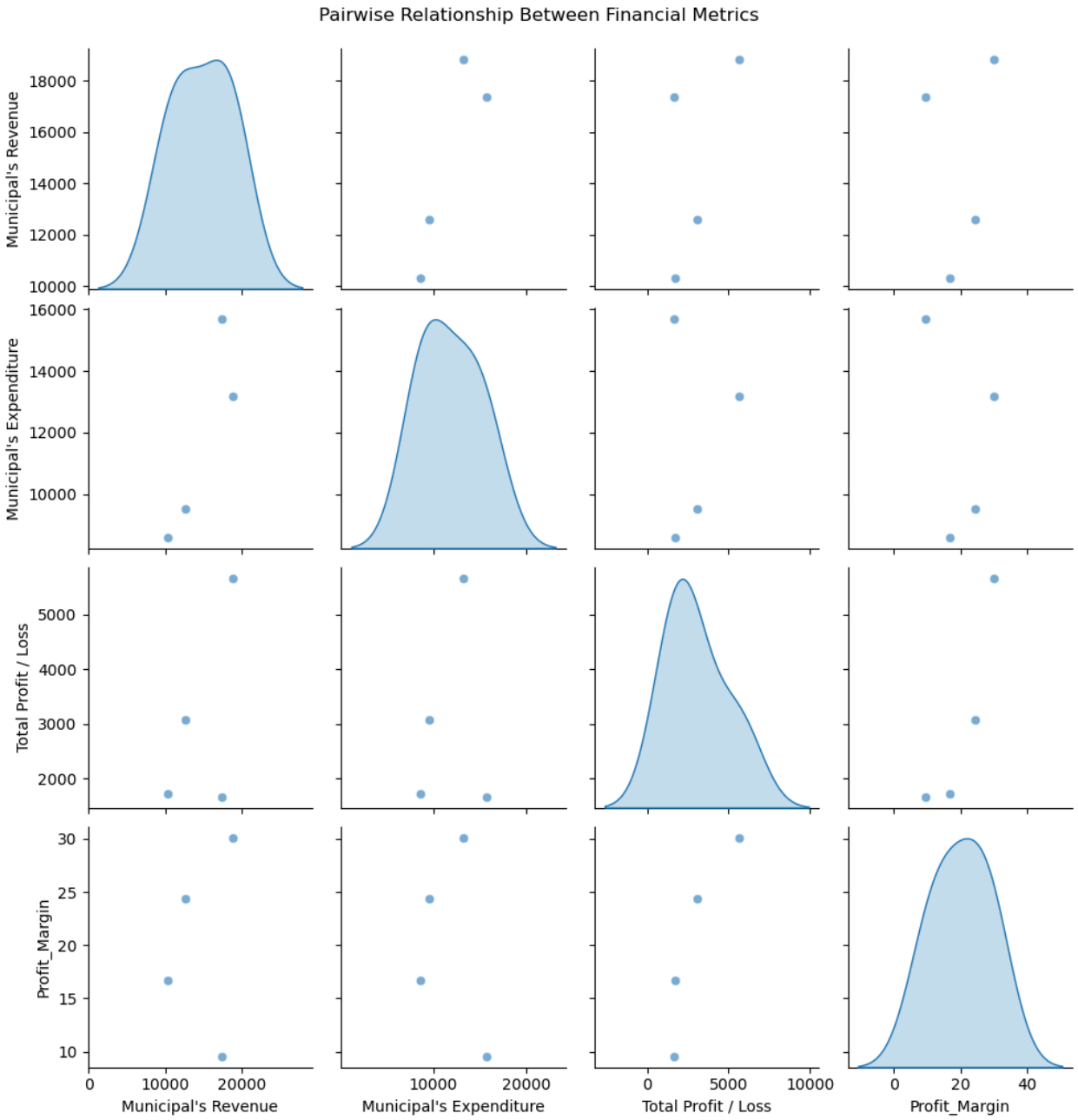
```
C:\Users\HOME\AppData\Local\Temp\ipykernel_37004\2197121833.py:4: FutureWarnin
g:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same e
ffect.

  sns.barplot(y="City Name", x="Profit_Margin", data=top20, palette="viridis")
```



In [58]:
```
plt.figure(figsize=(8,6))
sns.regplot(x="Municipal's Expenditure", y="Total Profit / Loss", data=pdf, sc
plt.title("Expenditure vs Profit/Loss (Regression Line)")
plt.xlabel("Municipal's Expenditure")
plt.ylabel("Total Profit / Loss")
plt.show()
```

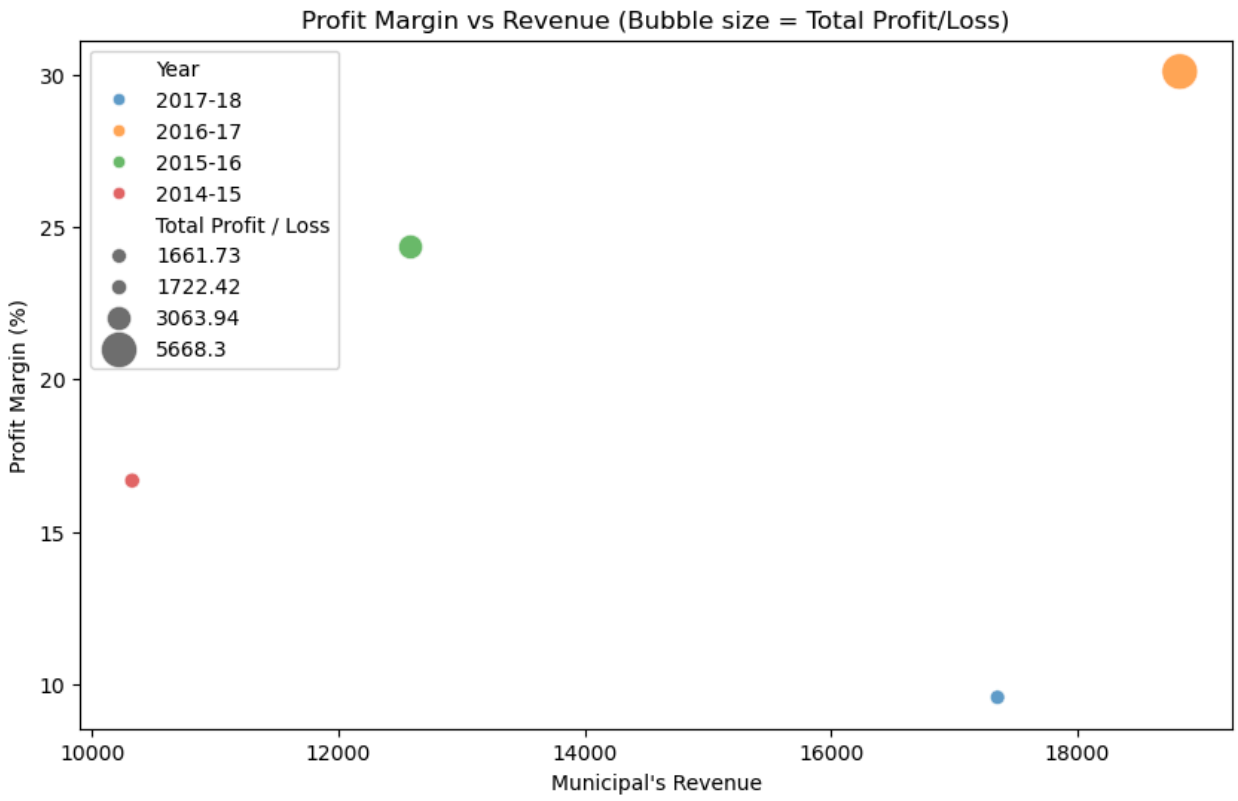


In [60]:
```
numeric_cols = ["Municipal's Revenue", "Municipal's Expenditure", "Total Profi

sns.pairplot(pdf[numeric_cols].dropna(), diag_kind="kde", plot_kws={'alpha':0.
plt.suptitle("Pairwise Relationship Between Financial Metrics", y=1.02)
plt.show()
```



In [62]:

```
plt.figure(figsize=(10,6))
sns.scatterplot(
    x="Municipal's Revenue",
    y="Profit_Margin",
    size="Total Profit / Loss",
    hue="Year" if "Year" in pdf.columns else None,
    data=pdf,
    alpha=0.7,
    sizes=(50, 300)
)
plt.title("Profit Margin vs Revenue (Bubble size = Total Profit/Loss)")
plt.xlabel("Municipal's Revenue")
plt.ylabel("Profit Margin (%)")
plt.show()
```


Profit Margin vs Revenue (Bubble size = Total Profit/Loss)
Year
2017-18
2016-17
2015-16
2014-15
Total Profit / Loss
1661.73
1722.42
3063.94
5668.3
Profit Margin (%)
30
25
20
15
10
10000
12000
14000
16000
18000
Municipal's Revenue

In [ ]: