

NETWORK TOWER PERFORMANCE ANALYSIS USING PYSPARK

UNDER THE GUIDANCE OF :

Prof. Huma Mam
Prof. Mayur Sir



1.Introduction

2.Raw Data Details

3>Loading Data

4.Data Cleaning

5.Transformations

6.Aggregates Prepared

7.Data Visualization

8.Conclusion

1. Introduction

The project ' Network Tower Performance Analysis using PySpark', focuses on analyzing large-scale telecom data to evaluate the performance of different network towers. Using the distributed computing power of PySpark, the project aims to identify performance bottlenecks, monitor signal quality, and optimize network efficiency for better connectivity and user experience.



2. Raw Data Details

- **Dataset Name:** Network Tower Performance Data
- **Data Source:** Collected from telecom network monitoring system (CSV format)
- **Total Records:** 1000
- **Total Columns:** 12



Tower_ID

Signal_Strength

Latency_ms

Power_status

Location

Network_Type

Uplink_Utilization

Temperature_C

Timestamp

Bandwidth_Mbps

Downlink_Utilization

Alarm_Status

3. Loading Data

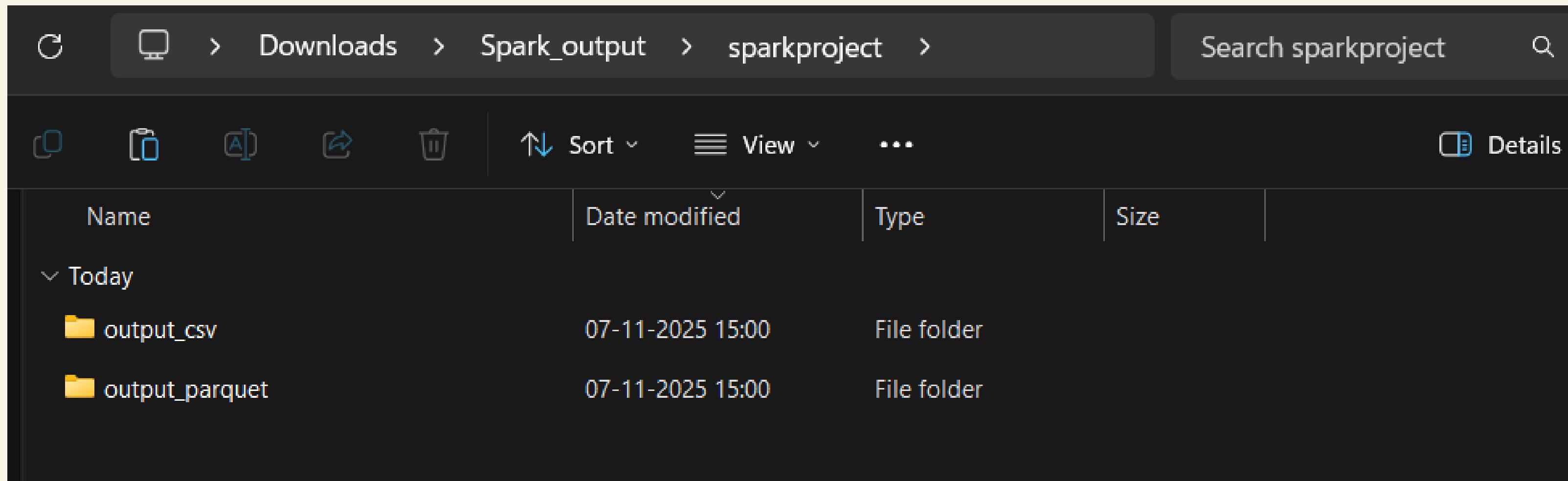
4.Data Cleaning

*Removing Duplicate Values
Managing Null Values*

```
df.createOrReplaceTempView("network_tower")
clean_df = spark.sql("SELECT DISTINCT * FROM network_tower")
clean_df.createOrReplaceTempView("clean_network_tower")
clean_df.show()
clean_df.printSchema()
cleaned_final = spark.sql("""SELECT Tower_ID, Location, COALESCE(Signal_Strength_dBm, (SELECT AVG(Signal_Strength_dBm) FROM clean_network_tower)) AS Signal_Strength_dBm, Network_Type,
COALESCE(Bandwidth_Mbps, (SELECT AVG(Bandwidth_Mbps) FROM clean_network_tower)) AS Bandwidth_Mbps, Latency_ms,
COALESCE(Uplink_Utilization, 0) AS Uplink_Utilization,
COALESCE(Downlink_Utilization, 0) AS Downlink_Utilization,
COALESCE(Power_Status, 'Unknown') AS Power_Status,
COALESCE(Temperature_C, (SELECT AVG(Temperature_C) FROM clean_network_tower)) AS Temperature_C,
Alarm_Status FROM clean_network_tower""")
cleaned_final.createOrReplaceTempView("cleaned_tower")
cleaned_final.show()
```

5. Transformation

```
# Save as Parquet  
cleaned_final.write.mode("overwrite").parquet(r"C:\Users\ASUS\Downloads\Spark_output\sparkproject\output_parquet")  
#Save as CSV (with header)  
cleaned_final.write.mode("overwrite").option( key: "header", value: True).csv(r"C:\Users\ASUS\Downloads\Spark_output\sparkproject\output_csv")
```



6. Aggregates Prepared

Loading Cleaned Data

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("df").getOrCreate()
df= spark.read.format("csv").option( key: "header", value: "true").option( key: "inferSchema", value: "true").load(r"C:\Users\ASUS\Downloads\Spark_output\sparkproject\output_csv")
df.show()
df.createOrReplaceTempView("aggregate")
```

Tower_ID	Location	Signal_Strength_dBm	Network_Type	Bandwidth_Mbps	Latency_ms	Uplink_Utilization	Downlink_Utilization	Power_Status	Temperature_C	Alarm_Status
TWR002	Nagpur	-78.0	5G	160.6	109	43	49	Battery	42.0	Normal
TWR002	Mumbai	-69.0	5G	122.1	34	61	83	ON	40.0	Minor Alarm
TWR010	Aurangabad	-84.0	3G	153.5	89	65	97	OFF	46.0	Normal
TWR002	Nashik	-76.0	3G	174.6	21	80	89	OFF	45.0	Normal
TWR007	Pune	-84.0	3G	8.5	86	79	62	Battery	48.0	Normal
TWR018	Aurangabad	-93.0	4G	244.6	80	70	90	OFF	33.0	Normal
TWR013	Nagpur	-93.0	5G	206.1	46	93	82	OFF	45.0	Normal
TWR013	Aurangabad	-87.0	5G	29.2	21	90	46	ON	34.0	Normal
TWR008	Aurangabad	-62.0	3G	85.7	96	51	65	Battery	30.0	Normal
TWR019	Aurangabad	-70.0	4G	90.3	93	78	70	OFF	46.0	Normal
TWR018	Nashik	-74.0	5G	198.4	48	94	57	OFF	43.0	Normal
TWR006	Aurangabad	-66.0	5G	97.4	92	51	89	Battery	33.0	Normal
TWR001	Pune	-59.0	2G	67.6	34	89	81	OFF	30.0	Normal
TWR005	Nagpur	-80.0	3G	135.3	62	75	73	ON	30.0	Minor Alarm
TWR004	Nashik	-85.0	4G	102.5	72	85	94	OFF	33.0	Normal
TWR013	Nashik	-73.0	3G	55.2	31	57	62	OFF	34.0	Normal
TWR006	Nagpur	-67.0	4G	238.9	42	86	50	OFF	40.0	Normal
TWR002	Aurangabad	-85.0	3G	196.8	70	72	52	ON	37.0	Normal
TWR020	Aurangabad	-95.0	2G	64.9	87	64	64	OFF	41.0	Normal
TWR007	Aurangabad	-93.0	4G	42.0	76	76	49	OFF	32.0	Normal

Max Function

```
df_max=spark.sql("select Location,max(Bandwidth_Mbps) from aggregate group by location")
df_max.show()
df_max.write.mode("overwrite").parquet(r"C:\Users\ASUS\Downloads\output_of_Parquet\df_max_output")
df_max.write.mode("overwrite").option( key: "header", value: True).csv(r"C:\Users\ASUS\Downloads\output_of_csv\df_max.csv")
```

Location	max(Bandwidth_Mbps)
Aurangabad	249.5
Mumbai	248.1
Pune	249.3
Nagpur	249.7
Nashik	247.8

Min Function

```
df_min=spark.sql("select Location,min(Signal_Strength_dBm) from aggregate group by location")
df_min.show()
df_min.write.mode("overwrite").parquet(r"C:\Users\ASUS\Downloads\output_of_Parquet\df_min.parquet")
df_min.write.mode("overwrite").option( key: "header", value: True).csv(r"C:\Users\ASUS\Downloads\output_of_csv\df_min.csv")
```

Location	min(Signal_Strength_dBm)
Aurangabad	-95.0
Mumbai	-95.0
Pune	-95.0
Nagpur	-95.0
Nashik	-95.0

Avg Function

```
df_avg=spark.sql("select Location,avg(Latency_ms) from aggregate group by location")
df_avg.write.mode("overwrite").parquet(r"C:\Users\ASUS\Downloads\output_of_Parquet\df_avg.parquet")
df_avg.write.mode("overwrite").option( key: "header", value: True).csv(r"C:\Users\ASUS\Downloads\output_of_csv\df_avg.csv")
```

Location	Avg_Latency
Aurangabad	65.22335025380711
Mumbai	62.932367149758456
Pune	65.1076923076923
Nagpur	66.678391959799
Nashik	64.42574257425743

Sum Function

```
df_sum=spark.sql("select Location,sum(Uplink_Utilization) from aggregate group by location")
df_sum.show()
df_sum.write.mode("overwrite").parquet(r"C:\Users\ASUS\Downloads\output_of_Parquet\df_sum.parquet")
df_sum.write.mode("overwrite").option( key: "header", value: True).csv(r"C:\Users\ASUS\Downloads\output_of_csv\df_sum.csv")
```

Location	sum(Uplink_Utilization)
Aurangabad	13297
Mumbai	14155
Pune	13516
Nagpur	13552
Nashik	14024

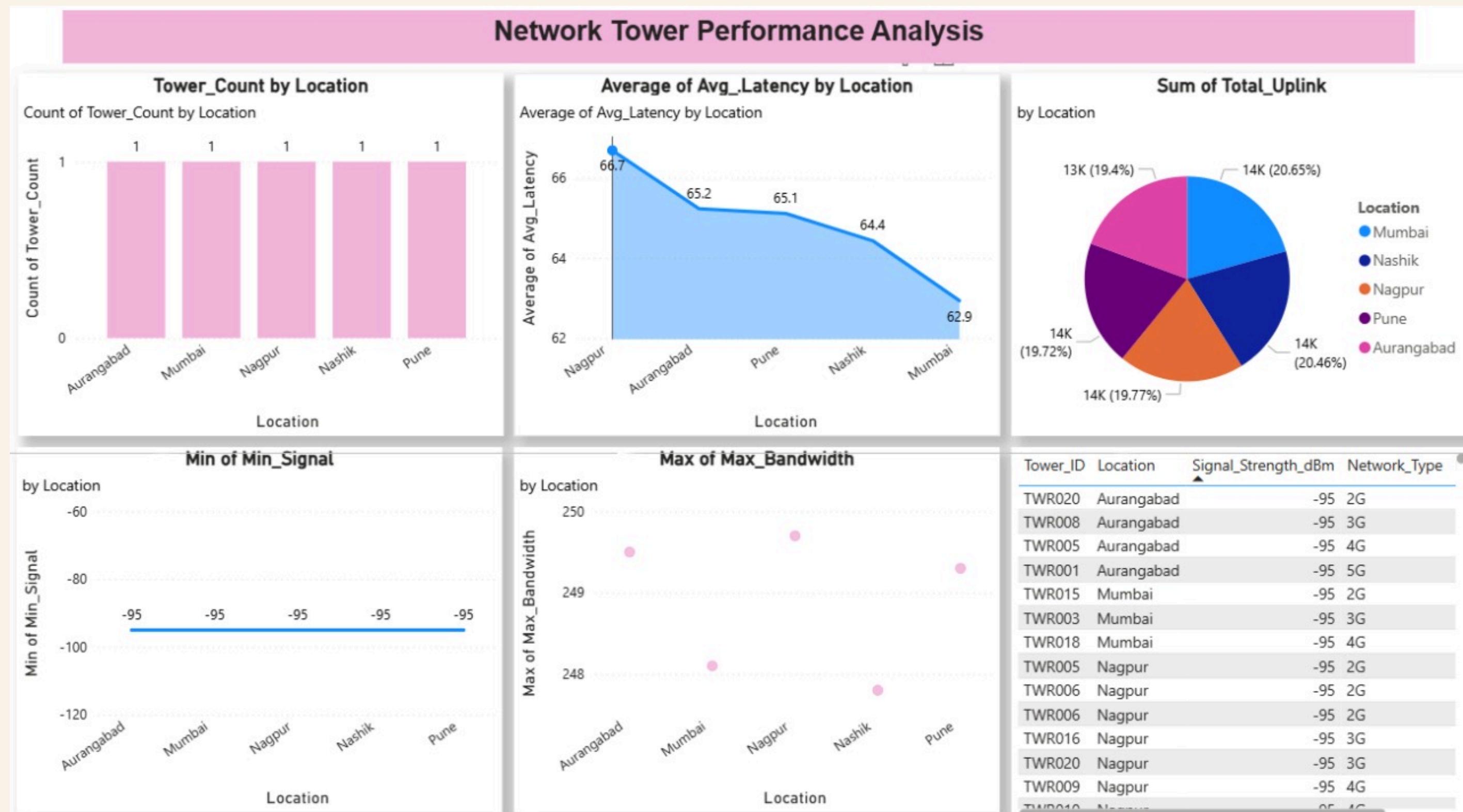
Count Function

```
df_count=spark.sql("select Location,count(Tower_ID) from aggregate group by location")
df_count.show()
df_count.write.mode("overwrite").parquet(r"C:\Users\ASUS\Downloads\output_of_Parquet\df_count.parquet")
df_count.write.mode("overwrite").option( key: "header", value: True).csv(r"C:\Users\ASUS\Downloads\output_of_csv\df_count.csv")
```

Location	count(Tower_ID)
Aurangabad	197
Mumbai	207
Pune	195
Nagpur	199
Nashik	202

7. Data Visualization

PowerBI



Databricks

The screenshot shows the Databricks interface with the Catalog Explorer open. The left sidebar includes links for New, Workspace, Recents, Catalog (selected), Jobs & Pipelines, Compute, Marketplace, SQL (SQL Editor, Queries, Dashboards, Genie, Alerts, Query History, SQL Warehouses), Data Engineering (Job Runs, Data Ingestion), AI/ML (Playground, Experiments), and a workspace dropdown.

The Catalog Explorer view displays the following details:

- Catalog:** Serverless Starter Warehouse, Serverless, 2XS
- Search Bar:** Search data, notebooks, recents, and more... (CTRL + P)
- Breadcrumbs:** Catalog Explorer > mini_project_spark > default >
- Table:** avg_csv_table (selected)
- Actions:** Open in a dashboard, Share, Create
- Tabs:** Overview, Sample Data (selected), Details, Permissions, Policies, History, Lineage, Insights, Quality
- Sample Data:** Ask your question about the sample data...
 - What is the total count of records by Location?
 - Which Location has the highest average Latency?
 - Are there any negative Latency values?
- Sample Table:** Shows a table with columns Location and avg(Latency_ms). The data is:

	Location	avg(Latency_ms)
1	Aurangabad	65.22335025380711
2	Mumbai	62.932367149758456
3	Pune	65.1076923076923
4	Nagpur	66.678391959799
5	Nashik	64.42574257425743

04. Conclusion

- The raw dataset was efficiently imported into PySpark using a Spark session.
- Schema detection and header options ensured accurate data loading.
- This step laid the foundation for data cleaning, transformation, and Power BI dashboard creation

Thank you