Table of Contents

CASE STUDY 7

Title: Telecom Network Quality and Customer Experience Analytics using PySpark

Spark Session Initialization

```
# Initialize Spark Session
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("TelecomNetworkQuality") \
    .getOrCreate()
```

Required Imports

```
from pyspark.sql.functions import (
    col, trim, upper, lower, when, regexp_replace,
    avg, count, sum as _sum,
    try_to_timestamp, coalesce,
    row_number, lag,
    round, expr
)
from pyspark.sql.window import Window
from pyspark.sql.types import StructType, StructField, StringType, IntegerType,
DoubleType
```

PHASE 1: INGESTION

1. Define Schema and Read CSV

```
# 1. Read network_logs.csv as all StringType
schema_string = StructType([
    StructField("event_id", StringType(), True),
    StructField("subscriber_id", StringType(), True),
    StructField("tower_id", StringType(), True),
    StructField("city", StringType(), True),
    StructField("network_type", StringType(), True),
    StructField("signal_strength", StringType(), True),
    StructField("download_speed_mbps", StringType(), True),
    StructField("upload_speed_mbps", StringType(), True),
    StructField("latency_ms", StringType(), True),
    StructField("call_drop", StringType(), True),
    StructField("event_time", StringType(), True),
    StructField("device_type", StringType(), True)
])

df_raw = spark.read.csv("/content/network_logs.csv", header=True,
schema=schema_string)
```

2. Print Schema and Row Count

```
# 2. Print schema and row count
df_raw.printSchema()
print(f"Raw Row Count: {df_raw.count()}")
```

```
root
 |-- event_id: string (nullable = true)
 |-- subscriber_id: string (nullable = true)
 |-- tower_id: string (nullable = true)
 |-- city: string (nullable = true)
 |-- network_type: string (nullable = true)
 |-- signal_strength: string (nullable = true)
 |-- download_speed_mbps: string (nullable = true)
 |-- upload_speed_mbps: string (nullable = true)
 |-- latency_ms: string (nullable = true)
 |-- call_drop: string (nullable = true)
 |-- event_time: string (nullable = true)
 |-- device_type: string (nullable = true)

Raw Row Count: 180000
```

3. Show Sample Rows

```
# 3. Show sample rows
df_raw.show(5)
```

```
+--------+-------------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+-------------------+------------+
|event_id|subscriber_id|tower_id|
city|network_type|signal_strength|download_speed_mbps|upload_speed_mbps|latency_ms|call_dr
op|         event_time| device_type|
+--------+-------------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+-------------------+------------+
| E100000|        S5975|    T837|Bangalore|          3G|        invalid|
NULL|             NULL|   invalid|      YES|01/01/2026 00:00:00|     Android|
| E100001|        S3537|    T283|Hyderabad|          5G|            -83|
124.07|            41.26|       114|       NO|2026-01-01 00:00:03|FeaturePhone|
| E100002|        S1629|    T877|     Pune|          4G|            -72|
41.01|             3.36|       221|       NO|2026-01-01 00:00:06|FeaturePhone|
| E100003|        S9422|    T431|    Delhi|          3G|            -97|
46.98|            13.36|       148|       NO|2026-01-01 00:00:09|     Android|
| E100004|        S1776|    T432|Hyderabad|          3G|            -83|
15.3|             31.1|       251|       NO|2026-01-01 00:00:12|FeaturePhone|
+--------+-------------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+-------------------+------------+
```

PHASE 2: CLEANING

1. Trim String Columns

```
# 1. Trim string columns
df_trimmed = df_raw.select([trim(col(c)).alias(c) for c in df_raw.columns])

df_trimmed.show(5)
```

```
+--------+-------------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+-------------------+------------+
|event_id|subscriber_id|tower_id|
city|network_type|signal_strength|download_speed_mbps|upload_speed_mbps|latency_ms|call_dr
op|         event_time| device_type|
+--------+-------------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+-------------------+------------+
| E100000|        S5975|    T837|Bangalore|          3G|        invalid|
NULL|             NULL|   invalid|      YES|01/01/2026 00:00:00|     Android|
| E100001|        S3537|    T283|Hyderabad|          5G|            -83|
124.07|            41.26|       114|       NO|2026-01-01 00:00:03|FeaturePhone|
| E100002|        S1629|    T877|     Pune|          4G|            -72|
41.01|             3.36|       221|       NO|2026-01-01 00:00:06|FeaturePhone|
| E100003|        S9422|    T431|    Delhi|          3G|            -97|
46.98|            13.36|       148|       NO|2026-01-01 00:00:09|     Android|
| E100004|        S1776|    T432|Hyderabad|          3G|            -83|
15.3|             31.1|       251|       NO|2026-01-01 00:00:12|FeaturePhone|
+--------+-------------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+-------------------+------------+
```

2. Normalize String Fields

```
# 2. Normalize string fields (city, network_type, device_type, call_drop)
df_normalized = df_trimmed.withColumn("city", upper(col("city"))) \
       .withColumn("network_type", upper(col("network_type"))) \
       .withColumn("device_type", upper(col("device_type"))) \
       .withColumn("call_drop", upper(col("call_drop")))

df_normalized.show(5)
```

```
+--------+-------------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+-------------------+------------+
|event_id|subscriber_id|tower_id|
city|network_type|signal_strength|download_speed_mbps|upload_speed_mbps|latency_ms|call_dr
op|         event_time| device_type|
+--------+-------------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+-------------------+------------+
| E100000|        S5975|    T837|BANGALORE|          3G|        invalid|
NULL|             NULL|   invalid|      YES|01/01/2026 00:00:00|     ANDROID|
| E100001|        S3537|    T283|HYDERABAD|          5G|            -83|
124.07|            41.26|       114|       NO|2026-01-01 00:00:03|FEATUREPHONE|
| E100002|        S1629|    T877|     PUNE|          4G|            -72|
41.01|             3.36|       221|       NO|2026-01-01 00:00:06|FEATUREPHONE|
| E100003|        S9422|    T431|    DELHI|          3G|            -97|
46.98|            13.36|       148|       NO|2026-01-01 00:00:09|     ANDROID|
| E100004|        S1776|    T432|HYDERABAD|          3G|            -83|
15.3|             31.1|       251|       NO|2026-01-01 00:00:12|FEATUREPHONE|
+--------+-------------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+-------------------+------------+
```

3. Clean Numeric Fields Safely

```
# 3. Clean numeric fields safely
df_cleaned = (
    df_normalized
    .withColumn(
        "signal_strength",
        when(
            regexp_replace(col("signal_strength"), "[^0-9-]", "") == "",
            None
        ).otherwise(
            regexp_replace(col("signal_strength"), "[^0-9-]",
"").cast(IntegerType())
        )
    )
    .withColumn(
        "download_speed_mbps",
        when(
            regexp_replace(col("download_speed_mbps"), "[^0-9.]", "") == "",
            None
        ).otherwise(
            regexp_replace(col("download_speed_mbps"), "[^0-9.]",
"").cast(DoubleType())
        )
    )
    .withColumn(
        "upload_speed_mbps",
        when(
            regexp_replace(col("upload_speed_mbps"), "[^0-9.]", "") == "",
            None
        ).otherwise(
            regexp_replace(col("upload_speed_mbps"), "[^0-9.]",
"").cast(DoubleType())
        )
    )
    .withColumn(
        "latency_ms",
        when(
            regexp_replace(col("latency_ms"), "[^0-9]", "") == "",
            None
        ).otherwise(
            regexp_replace(col("latency_ms"), "[^0-9]", "").cast(IntegerType())
        )
    )
)

df_cleaned.show(5)
```

```
+--------+-------------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+-------------------+------------+
|event_id|subscriber_id|tower_id|
city|network_type|signal_strength|download_speed_mbps|upload_speed_mbps|latency_ms|call_dr
op|         event_time| device_type|
+--------+-------------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+-------------------+------------+
| E100000|        S5975|    T837|BANGALORE|          3G|           NULL|
NULL|             NULL|      NULL|      YES|01/01/2026 00:00:00|     ANDROID|
| E100001|        S3537|    T283|HYDERABAD|          5G|            -83|
124.07|            41.26|       114|       NO|2026-01-01 00:00:03|FEATUREPHONE|
| E100002|        S1629|    T877|     PUNE|          4G|            -72|
41.01|             3.36|       221|       NO|2026-01-01 00:00:06|FEATUREPHONE|
| E100003|        S9422|    T431|    DELHI|          3G|            -97|
46.98|            13.36|       148|       NO|2026-01-01 00:00:09|     ANDROID|
| E100004|        S1776|    T432|HYDERABAD|          3G|            -83|
15.3|             31.1|       251|       NO|2026-01-01 00:00:12|FEATUREPHONE|
+--------+-------------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+-------------------+------------+
```

4. Parse Event Time with Multiple Formats

```python
# 4. Parse event_time with multiple formats
df = df_cleaned.withColumn(
    "event_time_clean",
    expr("""
        coalesce(
            try_to_timestamp(event_time, 'yyyy-MM-dd HH:mm:ss'),
            try_to_timestamp(event_time, 'dd/MM/yyyy HH:mm:ss'),
            try_to_timestamp(event_time, 'yyyy/MM/dd HH:mm:ss')
        )
    """)
)

df.show(5)
```

```
+--------+-------------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+-------------------+------------+----------------
---+
|event_id|subscriber_id|tower_id|
city|network_type|signal_strength|download_speed_mbps|upload_speed_mbps|latency_ms|call_dr
op|         event_time| device_type|   event_time_clean|
+--------+-------------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+-------------------+------------+----------------
---+
| E100000|        S5975|    T837|BANGALORE|          3G|           NULL|
NULL|             NULL|      NULL|      YES|01/01/2026 00:00:00|     ANDROID|2026-01-01
00:00:00|
| E100001|        S3537|    T283|HYDERABAD|          5G|            -83|
124.07|            41.26|       114|       NO|2026-01-01 00:00:03|FEATUREPHONE|2026-01-01
00:00:03|
| E100002|        S1629|    T877|     PUNE|          4G|            -72|
41.01|             3.36|       221|       NO|2026-01-01 00:00:06|FEATUREPHONE|2026-01-01
00:00:06|
| E100003|        S9422|    T431|    DELHI|          3G|            -97|
46.98|            13.36|       148|       NO|2026-01-01 00:00:09|     ANDROID|2026-01-01
00:00:09|
| E100004|        S1776|    T432|HYDERABAD|          3G|            -83|
15.3|             31.1|       251|       NO|2026-01-01 00:00:12|FEATUREPHONE|2026-01-01
00:00:12|
+--------+-------------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+-------------------+------------+----------------
---+
```

PHASE 3: VALIDATION

1. Count Invalid Values for Each Numeric Field

```
# 1. Count invalid values for each numeric field
invalid_counts = df.select(

_sum(col("signal_strength").isNull().cast("int")).alias("invalid_signal_strength"
),

_sum(col("download_speed_mbps").isNull().cast("int")).alias("invalid_download_spe
ed"),

_sum(col("upload_speed_mbps").isNull().cast("int")).alias("invalid_upload_speed")
,
    _sum(col("latency_ms").isNull().cast("int")).alias("invalid_latency")
)
print("Invalid Value Counts:")
invalid_counts.show()
```

```
Invalid Value Counts:
+-----------------------+---------------------+--------------------+---------------+
|invalid_signal_strength|invalid_download_speed|invalid_upload_speed|invalid_latency|
+-----------------------+---------------------+--------------------+---------------+
|                   9474|                13764|                5807|           4865|
+-----------------------+---------------------+--------------------+---------------+
```

2. Count Invalid Timestamps

```
# 2. Count invalid timestamps
print("Invalid timestamps:")
df.filter(col("event_time_clean").isNull()).count()
```

```
Invalid timestamps:
1605
```

3. Remove Duplicate Logs

```
# 3. Remove duplicate logs
df = df.dropDuplicates(["event_id"])
print(f"Count after removal: {df.count()}")
```

```
Count after removal: 180000
```

PHASE 4: NETWORK KPIS

1. Average Download Speed per City

```
# 1. Average download speed per city
avg_download_city = df.groupBy("city") \
    .agg(round(avg("download_speed_mbps"), 2).alias("avg_download_speed"))

avg_download_city.show()
```

```
+---------+------------------+
|     city|avg_download_speed|
+---------+------------------+
|  KOLKATA|             75.83|
|    DELHI|             75.79|
|BANGALORE|             75.49|
|HYDERABAD|              75.4|
|  CHENNAI|             75.51|
|     PUNE|             75.33|
|   MUMBAI|             75.53|
+---------+------------------+
```

2. Average Latency per City

```
# 2. Average latency per city
avg_latency_city = df.groupBy("city") \
    .agg(round(avg("latency_ms"), 2).alias("avg_latency"))

avg_latency_city.show()
```

```
+---------+-----------+
|     city|avg_latency|
+---------+-----------+
|  KOLKATA|     154.95|
|    DELHI|     155.44|
|BANGALORE|      156.1|
|HYDERABAD|     155.13|
|  CHENNAI|     154.65|
|     PUNE|     154.94|
|   MUMBAI|     154.44|
+---------+-----------+
```

3. Call Drop Rate per City

```
# 3. Call drop rate per city
call_drop_city = df.groupBy("city") \
    .agg(
        round(
            _sum(when(col("call_drop") == "YES", 1).otherwise(0)) / count("*"),
            3
        ).alias("call_drop_rate")
    )

call_drop_city.show()
```

```
+---------+--------------+
|     city|call_drop_rate|
+---------+--------------+
|  KOLKATA|          0.05|
|    DELHI|         0.051|
|BANGALORE|          0.05|
|HYDERABAD|         0.051|
|  CHENNAI|         0.049|
|     PUNE|         0.049|
|   MUMBAI|         0.051|
+---------+--------------+
```

4. Call Drop Rate per Tower

```python
# 4. Call drop rate per tower
call_drop_tower = df.groupBy("tower_id") \
    .agg(
        round(
            _sum(when(col("call_drop") == "YES", 1).otherwise(0)) / count("*"),
            3
        ).alias("call_drop_rate")
    )

call_drop_tower.show(5)
```

```
+--------+--------------+
|tower_id|call_drop_rate|
+--------+--------------+
|    T352|         0.051|
|    T929|         0.027|
|    T947|         0.038|
|    T590|         0.063|
|    T855|         0.057|
+--------+--------------+
only showing top 5 rows
```

5. Top 10 Worst Towers

```python
# 5. Top 10 worst towers
# Criteria: High Drop Rate, High Latency, Low Download Speed
worst_towers = df.groupBy("tower_id") \
    .agg(
        round(avg("latency_ms"), 2).alias("avg_latency"),
        round(avg("download_speed_mbps"), 2).alias("avg_download_speed"),
        round(
            _sum(when(col("call_drop") == "YES", 1).otherwise(0)) / count("*"),
            3
        ).alias("call_drop_rate")
    ) \
    .orderBy(col("call_drop_rate").desc(),
             col("avg_latency").desc(),
             col("avg_download_speed").asc()) \
    .limit(10)

worst_towers.show()
```

```
+--------+-----------+------------------+--------------+
|tower_id|avg_latency|avg_download_speed|call_drop_rate|
+--------+-----------+------------------+--------------+
|    T358|     162.98|             75.27|         0.104|
|    T275|     149.97|             75.14|           0.1|
|    T241|     152.49|             72.87|         0.097|
|    T455|     156.39|             77.05|         0.096|
|    T697|     150.16|             80.92|         0.092|
|    T538|     160.16|             72.71|         0.089|
|    T257|     155.76|             79.97|         0.089|
|    T653|     151.88|             78.33|         0.088|
|    T157|     155.75|             77.22|         0.087|
|    T659|      153.3|             81.18|         0.087|
+--------+-----------+------------------+--------------+
```

PHASE 5: CUSTOMER EXPERIENCE

Compute Metrics for Each Subscriber

```
# Compute metrics for each subscriber_id
subscriber_metrics = df.groupBy("subscriber_id") \
    .agg(
        count("*").alias("event_count"),
        round(avg("download_speed_mbps"), 2).alias("avg_download_speed"),
        round(avg("latency_ms"), 2).alias("avg_latency"),
        _sum(when(col("call_drop") == "YES",
1).otherwise(0)).alias("call_drop_count")
    )

subscriber_metrics.show(5)
```

```
+-------------+-----------+------------------+-----------+---------------+
|subscriber_id|event_count|avg_download_speed|avg_latency|call_drop_count|
+-------------+-----------+------------------+-----------+---------------+
|        S2422|         22|              76.5|     156.36|              0|
|        S1828|         26|             61.47|     120.62|              0|
|        S2414|         14|             73.25|     142.62|              0|
|        S7616|         17|             95.41|     158.35|              0|
|        S6467|         18|              80.4|     174.33|              1|
+-------------+-----------+------------------+-----------+---------------+
only showing top 5 rows
```

Poor Experience Users

```
# Poor experience logic
# Logic defined: High drops (>=3), Low speed (<5 Mbps), High latency (>200ms)
poor_experience_users = subscriber_metrics.filter(
    (col("call_drop_count") >= 3) |
    (col("avg_download_speed") < 5) |
    (col("avg_latency") > 200)
)

poor_experience_users.show(5)
```

```
+-------------+-----------+------------------+-----------+---------------+
|subscriber_id|event_count|avg_download_speed|avg_latency|call_drop_count|
+-------------+-----------+------------------+-----------+---------------+
|        S7972|         21|             74.89|     150.95|              5|
|        S7123|         20|              59.2|      145.6|              3|
|        S2497|         18|             92.98|     138.06|              3|
|        S4911|         27|             71.12|     121.85|              3|
|        S2181|         23|             77.06|     132.18|              3|
+-------------+-----------+------------------+-----------+---------------+
only showing top 5 rows
```

PHASE 6: WINDOW FUNCTIONS

1. Rank Towers Within Each City by Call Drop Rate

```
# 1. Rank towers within each city by call drop rate
tower_city_window =
Window.partitionBy("city").orderBy(col("call_drop_rate").desc())

tower_ranked = call_drop_city.withColumn(
    "tower_rank",
    row_number().over(tower_city_window)
)
tower_ranked.show(5)
```

```
+---------+--------------+----------+
|     city|call_drop_rate|tower_rank|
+---------+--------------+----------+
|BANGALORE|          0.05|         1|
|  CHENNAI|         0.049|         1|
|    DELHI|         0.051|         1|
|HYDERABAD|         0.051|         1|
|  KOLKATA|          0.05|         1|
+---------+--------------+----------+
only showing top 5 rows
```

2. Rank Subscribers Within Each City by Worst Experience

```
# 2. Rank subscribers within each city by worst experience
subscriber_city_df = df.join(subscriber_metrics, "subscriber_id")

subscriber_window = Window.partitionBy("city").orderBy(
    col("call_drop_count").desc(),
    col("avg_latency").desc(),
    col("avg_download_speed").asc()
)

subscriber_ranked = subscriber_city_df.withColumn(
    "experience_rank",
    row_number().over(subscriber_window)
)
subscriber_ranked.show()
```

```
+-------------+--------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+
|subscriber_id|event_id|tower_id|
city|network_type|signal_strength|download_speed_mbps|upload_speed_mbps|latency_ms|call_dr
op|
+-------------+--------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+
|        S8721| E100973|    T662|BANGALORE|          3G|            -77|
78.83|             5.34|       101|       NO|
|        S8721| E105340|    T581|BANGALORE|          5G|            -89|
86.53|            48.66|       226|      YES|
|        S8721| E108460|    T131|BANGALORE|          4G|            -82|
116.09|            29.66|       286|      YES|

[Output truncated for brevity]
```

3. Signal Deterioration Detection Using Lag

```
# 3. Signal deterioration detection using lag
signal_window = Window.partitionBy("tower_id").orderBy("event_time_clean")

df_signal_lag = df.withColumn(
    "prev_signal_strength",
    lag("signal_strength").over(signal_window)
).withColumn(
    "signal_drop",
    col("prev_signal_strength") - col("signal_strength")
)

df_signal_lag.show()
```

```
+--------+-------------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+
|event_id|subscriber_id|tower_id|
city|network_type|signal_strength|download_speed_mbps|upload_speed_mbps|latency_ms|call_dr
op|
+--------+-------------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+
| E100028|        S9371|    T102|BANGALORE|          5G|            -62|
52.78|            45.88|       285|       NO|
| E101314|        S5759|    T102|   MUMBAI|          3G|            -81|
38.74|            17.97|       278|       NO|

[Output truncated for brevity]
```

PHASE 7: ANOMALY DETECTION

Detect Towers with Anomalies

```
# Detect towers where: Latency spikes, Download speed drops, Call drops spikes
rolling_window = Window.partitionBy("tower_id") \
    .orderBy("event_time_clean") \
    .rowsBetween(-3, -1)

df_anomaly = df.withColumn(
    "rolling_avg_latency",
    avg("latency_ms").over(rolling_window)
).withColumn(
    "rolling_avg_download",
    avg("download_speed_mbps").over(rolling_window)
).withColumn(
    "latency_spike",
    col("latency_ms") > col("rolling_avg_latency") * 1.5
).withColumn(
    "download_drop",
    col("download_speed_mbps") < col("rolling_avg_download") * 0.5
).withColumn(
    "call_drop_spike",
    col("call_drop") == "YES"
)

anomalous_events = df_anomaly.filter(
    col("latency_spike") | col("download_drop") | col("call_drop_spike")
)

anomalous_events.show()
```

```
+--------+-------------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+
|event_id|subscriber_id|tower_id|
city|network_type|signal_strength|download_speed_mbps|upload_speed_mbps|latency_ms|call_dr
op|
+--------+-------------+--------+---------+------------+---------------+------------------
-+-----------------+----------+---------+
| E103037|        S7132|    T102|HYDERABAD|          5G|            -92|
27.06|             4.23|       257|       NO|
| E103866|        S2572|    T102|  KOLKATA|          3G|            -81|
31.06|            44.33|       168|       NO|
| E104673|        S3570|    T102|HYDERABAD|          5G|            -72|
18.78|            42.25|       118|       NO|

[Output truncated for brevity]
```

End of Document

Telecom Network Quality and Customer Experience Analytics using PySpark