```
!pip install -q pyspark
```

```
import os
import json
import time
import random
from datetime import datetime, timezone
output_dir = "/content/beatblast_events_raw"
os.makedirs(output_dir, exist_ok=True)
event_types = ['songPlay', 'songSkip', 'songLike', 'appOpen']
platforms = ['android', 'ios', 'web']
countries = ['US', 'CA', 'GB', 'IN']
songs = ['song_001', 'song_002', 'song_003', 'song_004', 'song_005']
users = ['user_1', 'user_2', 'user_3']
sessions = ['sess_1', 'sess_2', 'sess_3']
for i in range(30):
    event = {
        "eventType": random.choice(event_types),
        "eventTimestamp": datetime.now(timezone.utc).isoformat(),
        "songId": random.choice(songs),
        "sessionId": random.choice(sessions),
        "userId": random.choice(users),
        "platform": random.choice(platforms),
        "country": random.choice(countries)
    }


    filename = f"event_{int(time.time() * 1000)}.json"
    with open(os.path.join(output_dir, filename), 'w') as f:
        json.dump(event, f)

    print(f"Generated: {filename}")
    time.sleep(1)
```

```
Generated: event_1752878997055.json
Generated: event_1752878998056.json
Generated: event_1752878999056.json
Generated: event_1752879000057.json
Generated: event_1752879001057.json
Generated: event_1752879002058.json
Generated: event_1752879003059.json
Generated: event_1752879004059.json
Generated: event_1752879005060.json
Generated: event_1752879006060.json
Generated: event_1752879007061.json
Generated: event_1752879008061.json
Generated: event_1752879009062.json
Generated: event_1752879010063.json
Generated: event_1752879011063.json
Generated: event_1752879012064.json
Generated: event_1752879013064.json
Generated: event_1752879014065.json
Generated: event_1752879015065.json
Generated: event_1752879016066.json
Generated: event_1752879017066.json
Generated: event_1752879018067.json
Generated: event_1752879019068.json
Generated: event_1752879020068.json
Generated: event_1752879021069.json
Generated: event_1752879022070.json
Generated: event_1752879023070.json
Generated: event_1752879024071.json
Generated: event_1752879025072.json
Generated: event_1752879026072.json
```

```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
    .appName("BeatBlastStructuredStreaming") \
    .config("spark.sql.shuffle.partitions", "2") \
    .getOrCreate()
spark.sparkContext.setLogLevel("ERROR")
```

```
from pyspark.sql.types import StructType, StringType, TimestampType
from pyspark.sql.functions import col, current_timestamp, to_timestamp
event_schema = StructType() \
    .add("eventType", StringType()) \
    .add("eventTimestamp", StringType()) \
    .add("songId", StringType()) \
    .add("sessionId", StringType()) \
    .add("userId", StringType()) \
    .add("platform", StringType()) \
    .add("country", StringType())
stream_df = spark.readStream \
    .schema(event_schema) \
    .json("/content/beatblast_events_raw")
stream_df = stream_df \
    .withColumn("eventTimestamp", to_timestamp("eventTimestamp")) \
    .withColumn("processingTimestamp", current_timestamp())



from pyspark.sql.functions import window, count
popular_songs = stream_df \
    .filter(col("eventType") == "songPlay") \
    .withWatermark("eventTimestamp", "10 minutes") \
    .groupBy(
        window(col("eventTimestamp"), "5 minutes"),
        col("songId")
    ) \
    .agg(count("*").alias("play_count")) \
    .select(
        col("window.start").alias("window_start"),
        col("window.end").alias("window_end"),
        "songId", "play_count"
    )

query1 = popular_songs.writeStream \
    .outputMode("update") \
    .format("console") \
    .option("truncate", False) \
    .start()



from pyspark.sql.functions import approx_count_distinct
active_sessions = stream_df \
    .filter(col("eventType") == "appOpen") \
    .withWatermark("eventTimestamp", "10 minutes") \
    .groupBy(
        window(col("eventTimestamp"), "10 minutes", "5 minutes"),
        col("platform")
    ) \
    .agg(approx_count_distinct("sessionId").alias("distinct_session_count")) \
    .select(
        col("window.start").alias("window_start"),
        col("window.end").alias("window_end"),
        "platform", "distinct_session_count"
    )

query2 = active_sessions.writeStream \
    .outputMode("complete") \
    .format("console") \
    .option("truncate", False) \
    .start()



from pyspark.sql.functions import year, month, dayofmonth
song_plays = stream_df \
    .filter(col("eventType") == "songPlay") \
    .withColumn("year", year("eventTimestamp")) \
    .withColumn("month", month("eventTimestamp")) \
    .withColumn("day", dayofmonth("eventTimestamp"))
query3 = song_plays.writeStream \
    .partitionBy("year", "month", "day", "country") \
    .format("parquet") \
    .option("path", "/content/beatblast_datalake/song_plays/") \
    .option("checkpointLocation", "/content/beatblast_datalake/checkpoints/") \
    .trigger(processingTime="1 minute") \
    .outputMode("append") \
```

```
    .start()


query1.awaitTermination()
query2.awaitTermination()
query3.awaitTermination()
```

```
ERROR:root:KeyboardInterrupt while sending command.
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/py4j/java_gateway.py", line 1038, in send_command
    response = connection.send_command(command)
               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/py4j/clientserver.py", line 511, in send_command
    answer = smart_decode(self.stream.readline()[:-1])
                          ^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3.11/socket.py", line 718, in readinto
    return self._sock.recv_into(b)
           ^^^^^^^^^^^^^^^^^^^^^^^
KeyboardInterrupt
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/tmp/ipython-input-12-282753091.py in <cell line: 0>()
----> 1 query1.awaitTermination()
      2 query2.awaitTermination()
      3 query3.awaitTermination()

 ──────────────────────────────── 4 frames ────────────────────────────────
/usr/lib/python3.11/socket.py in readinto(self, b)
    716         while True:
    717             try:
--> 718                 return self._sock.recv_into(b)
    719             except timeout:
    720                 self._timeout_occurred = True

KeyboardInterrupt:
```

```
!pip install -q pyspark
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StringType, TimestampType
from pyspark.sql.functions import col, current_timestamp, to_timestamp, window, count, approx_count_distinct, year, month, dayofmonth
import os, json, time, random
from datetime import datetime, timezone

spark = SparkSession.builder \
    .appName("BeatBlastQuickTest") \
    .config("spark.sql.shuffle.partitions", "2") \
    .getOrCreate()
spark.sparkContext.setLogLevel("ERROR")


input_dir = "/content/beatblast_events_raw"
os.makedirs(input_dir, exist_ok=True)

event_types = ['songPlay', 'songSkip', 'songLike', 'appOpen']
platforms = ['android', 'ios', 'web']
countries = ['US', 'CA', 'GB', 'IN']
songs = ['song_001', 'song_002', 'song_003']
users = ['user_1', 'user_2', 'user_3']
sessions = ['sess_1', 'sess_2', 'sess_3']

for i in range(100):
    event = {
        "eventType": random.choice(event_types),
        "eventTimestamp": datetime.now(timezone.utc).isoformat(),
        "songId": random.choice(songs),
        "sessionId": random.choice(sessions),
        "userId": random.choice(users),
        "platform": random.choice(platforms),
        "country": random.choice(countries)
    }
    with open(os.path.join(input_dir, f"event_{int(time.time()*1000)}.json"), 'w') as f:
        json.dump(event, f)
    time.sleep(0.1)

schema = StructType() \
```

```
        .add("eventType", StringType()) \
        .add("eventTimestamp", StringType()) \
        .add("songId", StringType()) \
        .add("sessionId", StringType()) \
        .add("userId", StringType()) \
        .add("platform", StringType()) \
        .add("country", StringType())

    df = spark.readStream.schema(schema).json(input_dir)
    df = df.withColumn("eventTimestamp", to_timestamp("eventTimestamp")) \
           .withColumn("processingTimestamp", current_timestamp())

    popular_songs = df.filter(col("eventType") == "songPlay") \
        .withWatermark("eventTimestamp", "10 minutes") \
        .groupBy(window(col("eventTimestamp"), "5 minutes"), col("songId")) \
        .agg(count("*").alias("play_count")) \
        .select("window.start", "window.end", "songId", "play_count")

    active_sessions = df.filter(col("eventType") == "appOpen") \
        .withWatermark("eventTimestamp", "10 minutes") \
        .groupBy(window(col("eventTimestamp"), "10 minutes", "5 minutes"), col("platform")) \
        .agg(approx_count_distinct("sessionId").alias("distinct_session_count")) \
        .select("window.start", "window.end", "platform", "distinct_session_count")


    song_plays = df.filter(col("eventType") == "songPlay") \
        .withColumn("year", year("eventTimestamp")) \
        .withColumn("month", month("eventTimestamp")) \
        .withColumn("day", dayofmonth("eventTimestamp"))

    query1 = popular_songs.writeStream.outputMode("update").format("console").option("truncate", False).start()
    query2 = active_sessions.writeStream.outputMode("complete").format("console").option("truncate", False).start()
    query3 = song_plays.writeStream \
        .partitionBy("year", "month", "day", "country") \
        .format("parquet") \
        .option("path", "/content/beatblast_datalake/song_plays/") \
        .option("checkpointLocation", "/content/beatblast_datalake/checkpoints/") \
        .trigger(processingTime="10 seconds") \
        .outputMode("append") \
        .start()


    time.sleep(60)
    query1.stop()
    query2.stop()
    query3.stop()



    !ls -R /content/beatblast_datalake/song_plays/
```

```
/content/beatblast_datalake/song_plays/:
 _spark_metadata  'year=2025'

/content/beatblast_datalake/song_plays/_spark_metadata:
0  1

'/content/beatblast_datalake/song_plays/year=2025':
'month=7'

'/content/beatblast_datalake/song_plays/year=2025/month=7':
'day=18'

'/content/beatblast_datalake/song_plays/year=2025/month=7/day=18':
'country=CA'  'country=GB'  'country=IN'  'country=US'

'/content/beatblast_datalake/song_plays/year=2025/month=7/day=18/country=CA':
part-00000-76c584e6-abfb-40a9-8936-84b1f746f786.c000.snappy.parquet
part-00000-9ff281ec-749e-4313-8913-7ce3b558a27f.c000.snappy.parquet
part-00001-c183357c-d47a-4bc5-b407-f83a36d9707f.c000.snappy.parquet
part-00001-f4126170-f008-43c1-8162-2e9546bdacd3.c000.snappy.parquet
part-00002-852b5d6b-8609-4c11-a3e5-94343dbcace3.c000.snappy.parquet

'/content/beatblast_datalake/song_plays/year=2025/month=7/day=18/country=GB':
part-00000-f8133993-625d-4255-bbe9-dad2d0f58239.c000.snappy.parquet
part-00001-865a0fc2-26f6-4ec0-8a49-a4f0e132ef6a.c000.snappy.parquet
part-00002-db0298c9-abd7-4662-ac1a-444b0a9e3267.c000.snappy.parquet

'/content/beatblast_datalake/song_plays/year=2025/month=7/day=18/country=IN':
```

```
part-00000-b2e23e6f-d779-4484-a3fa-b3b9014d516d.c000.snappy.parquet
part-00000-d31247c1-095a-44aa-877d-802ab360beec.c000.snappy.parquet
part-00001-2ee0ba5e-1c64-4941-8945-3cd02b15a150.c000.snappy.parquet
part-00002-5783613a-d2df-4d27-a50a-f1c89c1db8f3.c000.snappy.parquet

'/content/beatblast_datalake/song_plays/year=2025/month=7/day=18/country=US':
part-00000-87b15282-7478-4518-bfe9-1e38dbbc9b0c.c000.snappy.parquet
part-00001-18d38ed5-3006-4ffc-8966-a0bd6a19c22a.c000.snappy.parquet
part-00001-370dc11a-4549-4f67-9a95-4b3af7a45f2a.c000.snappy.parquet
```

```
df = spark.read.parquet("/content/beatblast_datalake/song_plays/")
df.show(truncate=False)
```

```
+---------+-------------------------+--------+---------+------+--------+----------------------+----+-----+---+-------+
|eventType|eventTimestamp           |songId  |sessionId|userId|platform|processingTimestamp   |year|month|day|country|
+---------+-------------------------+--------+---------+------+--------+----------------------+----+-----+---+-------+
|songPlay |2025-07-18 22:56:19.640731|song_001|sess_3   |user_2|android |2025-07-18 22:56:32.197|2025|7    |18 |GB     |
|songPlay |2025-07-18 22:56:23.776646|song_002|sess_3   |user_3|android |2025-07-18 22:56:32.197|2025|7    |18 |GB     |
|songPlay |2025-07-18 22:56:27.30528 |song_003|sess_3   |user_2|android |2025-07-18 22:56:32.197|2025|7    |18 |GB     |
|songPlay |2025-07-18 22:56:29.420627|song_003|sess_1   |user_1|android |2025-07-18 22:56:32.197|2025|7    |18 |GB     |
|songPlay |2025-07-18 22:56:20.64601 |song_001|sess_2   |user_3|web     |2025-07-18 22:56:32.197|2025|7    |18 |IN     |
|songPlay |2025-07-18 22:56:22.559096|song_003|sess_1   |user_3|web     |2025-07-18 22:56:32.197|2025|7    |18 |IN     |
|songPlay |2025-07-18 22:56:24.284934|song_003|sess_3   |user_2|web     |2025-07-18 22:56:32.197|2025|7    |18 |IN     |
|songPlay |2025-07-18 22:56:24.78907 |song_001|sess_2   |user_2|ios     |2025-07-18 22:56:32.197|2025|7    |18 |IN     |
|songPlay |2025-07-18 22:56:24.889543|song_001|sess_3   |user_3|ios     |2025-07-18 22:56:32.197|2025|7    |18 |IN     |
|songPlay |2025-07-18 22:56:25.594065|song_001|sess_3   |user_3|ios     |2025-07-18 22:56:32.197|2025|7    |18 |CA     |
|songPlay |2025-07-18 22:56:27.405744|song_001|sess_2   |user_1|web     |2025-07-18 22:56:32.197|2025|7    |18 |CA     |
|songPlay |2025-07-18 22:56:27.606718|song_003|sess_3   |user_2|web     |2025-07-18 22:56:32.197|2025|7    |18 |CA     |
|songPlay |2025-07-18 22:56:28.914399|song_002|sess_3   |user_2|web     |2025-07-18 22:56:32.197|2025|7    |18 |CA     |
|songPlay |2025-07-18 22:56:20.94778 |song_002|sess_2   |user_3|web     |2025-07-18 22:56:32.197|2025|7    |18 |CA     |
|songPlay |2025-07-18 22:56:21.048247|song_002|sess_1   |user_3|web     |2025-07-18 22:56:32.197|2025|7    |18 |CA     |
|songPlay |2025-07-18 22:56:21.1493  |song_001|sess_2   |user_1|web     |2025-07-18 22:56:32.197|2025|7    |18 |CA     |
|songPlay |2025-07-18 22:56:24.184431|song_001|sess_1   |user_2|ios     |2025-07-18 22:56:32.197|2025|7    |18 |CA     |
|songPlay |2025-07-18 22:50:04.059625|song_002|sess_2   |user_1|ios     |2025-07-18 22:53:36.067|2025|7    |18 |US     |
|songPlay |2025-07-18 22:50:12.064185|song_001|sess_1   |user_1|ios     |2025-07-18 22:53:36.067|2025|7    |18 |US     |
|songPlay |2025-07-18 22:50:13.064697|song_002|sess_2   |user_1|web     |2025-07-18 22:53:36.067|2025|7    |18 |US     |
+---------+-------------------------+--------+---------+------+--------+----------------------+----+-----+---+-------+
only showing top 20 rows
```

```
Start coding or generate with AI.
```

```
!ls -R /content/beatblast_datalake/song_plays/
```

```
/content/beatblast_datalake/song_plays/:
 _spark_metadata  'year=2025'

/content/beatblast_datalake/song_plays/_spark_metadata:
0  1

'/content/beatblast_datalake/song_plays/year=2025':
'month=7'

'/content/beatblast_datalake/song_plays/year=2025/month=7':
'day=18'

'/content/beatblast_datalake/song_plays/year=2025/month=7/day=18':
'country=CA'  'country=GB'  'country=IN'  'country=US'

'/content/beatblast_datalake/song_plays/year=2025/month=7/day=18/country=CA':
part-00000-76c584e6-abfb-40a9-8936-84b1f746f786.c000.snappy.parquet
part-00000-9ff281ec-749e-4313-8913-7ce3b558a27f.c000.snappy.parquet
part-00001-c183357c-d47a-4bc5-b407-f83a36d9707f.c000.snappy.parquet
part-00001-f4126170-f008-43c1-8162-2e9546bdacd3.c000.snappy.parquet
part-00002-852b5d6b-8609-4c11-a3e5-94343dbcace3.c000.snappy.parquet

'/content/beatblast_datalake/song_plays/year=2025/month=7/day=18/country=GB':
part-00000-f8133993-625d-4255-bbe9-dad2d0f58239.c000.snappy.parquet
part-00001-865a0fc2-26f6-4ec0-8a49-a4f0e132ef6a.c000.snappy.parquet
part-00002-db0298c9-abd7-4662-ac1a-444b0a9e3267.c000.snappy.parquet

'/content/beatblast_datalake/song_plays/year=2025/month=7/day=18/country=IN':
part-00000-b2e23e6f-d779-4484-a3fa-b3b9014d516d.c000.snappy.parquet
part-00000-d31247c1-095a-44aa-877d-802ab360beec.c000.snappy.parquet
part-00001-2ee0ba5e-1c64-4941-8945-3cd02b15a150.c000.snappy.parquet
part-00002-5783613a-d2df-4d27-a50a-f1c89c1db8f3.c000.snappy.parquet

'/content/beatblast_datalake/song_plays/year=2025/month=7/day=18/country=US':
part-00000-87b15282-7478-4518-bfe9-1e38dbbc9b0c.c000.snappy.parquet
```

```
part-00001-18d38ed5-3006-4ffc-8966-a0bd6a19c22a.c000.snappy.parquet
part-00001-370dc11a-4549-4f67-9a95-4b3af7a45f2a.c000.snappy.parquet
```

```
from pyspark.sql.functions import window
agg_song_count = df.groupBy(
    window("eventTimestamp", "10 minutes"),
    "songId"
).count().withColumnRenamed("count", "play_count")

agg_song_count.show(truncate=False)
```

```
+------------------------------------------+--------+----------+
|window                                    |songId  |play_count|
+------------------------------------------+--------+----------+
|{2025-07-18 22:50:00, 2025-07-18 23:00:00}|song_001|12        |
|{2025-07-18 22:50:00, 2025-07-18 23:00:00}|song_003|9         |
|{2025-07-18 22:50:00, 2025-07-18 23:00:00}|song_002|13        |
|{2025-07-18 22:50:00, 2025-07-18 23:00:00}|song_005|1         |
|{2025-07-18 22:40:00, 2025-07-18 22:50:00}|song_001|1         |
|{2025-07-18 22:50:00, 2025-07-18 23:00:00}|song_004|2         |
+------------------------------------------+--------+----------+
```

```
from pyspark.sql.functions import dense_rank
from pyspark.sql.window import Window as W

agg_session_activity = df.groupBy(
    window("eventTimestamp", "10 minutes"),
    "sessionId"
).count().withColumnRenamed("count", "activity")


window_spec = W.partitionBy("window").orderBy(agg_session_activity["activity"].desc())

top_sessions = agg_session_activity.withColumn(
    "rank", dense_rank().over(window_spec)
).filter("rank = 1").drop("rank")

top_sessions.show(truncate=False)
```

```
+------------------------------------------+---------+--------+
|window                                    |sessionId|activity|
+------------------------------------------+---------+--------+
|{2025-07-18 22:40:00, 2025-07-18 22:50:00}|sess_1   |1       |
|{2025-07-18 22:50:00, 2025-07-18 23:00:00}|sess_3   |15      |
+------------------------------------------+---------+--------+
```

```
from pyspark.sql.functions import dense_rank
from pyspark.sql.window import Window as W

agg_session_activity = df.groupBy(
    window("eventTimestamp", "10 minutes"),
    "sessionId"
).count().withColumnRenamed("count", "activity")

window_spec = W.partitionBy("window").orderBy(agg_session_activity["activity"].desc())

top_sessions = agg_session_activity.withColumn(
    "rank", dense_rank().over(window_spec)
).filter("rank = 1").drop("rank")

top_sessions.show(truncate=False)
```

```
+------------------------------------------+---------+--------+
|window                                    |sessionId|activity|
+------------------------------------------+---------+--------+
|{2025-07-18 22:40:00, 2025-07-18 22:50:00}|sess_1   |1       |
|{2025-07-18 22:50:00, 2025-07-18 23:00:00}|sess_3   |15      |
+------------------------------------------+---------+--------+
```

```
import json, random, time, os
from datetime import datetime
```

```
def simulate_events(output_dir="/content/stream_input", num_events=30, sleep_time=0.2):
    os.makedirs(output_dir, exist_ok=True)
    for i in range(num_events):
        event = {
            "eventType": "play",
            "eventTimestamp": datetime.utcnow().isoformat(),
            "songId": f"song_00{random.randint(1,5)}",
            "sessionId": f"sess_{random.randint(1,3)}",
            "userId": f"user_{random.randint(1,10)}",
            "platform": random.choice(["android", "ios", "web"]),
            "country": random.choice(["US", "IN", "CA", "GB"])
        }
        with open(f"{output_dir}/event_{int(time.time() * 1000)}.json", "w") as f:
            json.dump(event, f)
        time.sleep(sleep_time)

simulate_events()
```