# databricksAir_Quality_5

```
%scala
import org.apache.spark.eventhubs._
import com.microsoft.azure.eventhubs._
import org.apache.spark.sql.functions.{ explode, split }



// Build connection string with the above information
val namespaceName = "telemetrynamespace"
val eventHubName = "airqualityeventhub"
val sasKeyName = "iothubroutes_AirQualityHub3196"
val sasKey = "lT6Aq9Rht9X7dnzKDD5GRyYB9ZKU3zmDdE/CtxECi/M="
val connectionString = new
com.microsoft.azure.eventhubs.ConnectionStringBuilder()
            .setNamespaceName(namespaceName)
            .setEventHubName(eventHubName)
            .setSasKeyName(sasKeyName)
            .setSasKey(sasKey)

val eventHubsConf = EventHubsConf(connectionString.toString())
  .setMaxEventsPerTrigger(5)


val eventhubs = spark.readStream
  .format("eventhubs")
  .options(eventHubsConf.toMap)
  .load()


//val messages = eventhubs.selectExpr("cast (body as string) AS body")
//
messages.writeStream.outputMode("append").format("console").option("truncate",
false).start().awaitTermination()



import org.apache.spark.eventhubs._
import com.microsoft.azure.eventhubs._
import org.apache.spark.sql.functions.{explode, split}
namespaceName: String = telemetrynamespace
eventHubName: String = airqualityeventhub
sasKeyName: String = iothubroutes_AirQualityHub3196
sasKey: String = lT6Aq9Rht9X7dnzKDD5GRyYB9ZKU3zmDdE/CtxECi/M=
connectionString: com.microsoft.azure.eventhubs.ConnectionStringBuilder = Endp
```

```
oint=sb://telemetrynamespace.servicebus.windows.net;EntityPath=airqualityevent
hub;SharedAccessKeyName=iothubroutes_AirQualityHub3196;SharedAccessKey=lT6Aq9R
ht9X7dnzKDD5GRyYB9ZKU3zmDdE/CtxECi/M=
eventHubsConf: org.apache.spark.eventhubs.EventHubsConf = org.apache.spark.eve
nthubs.EventHubsConf@2a2faa71
eventhubs: org.apache.spark.sql.DataFrame = [body: binary, partition: string
... 7 more fields]
```

```
import org.apache.spark.sql.functions._

// The IoT data comes inside data fiels which is in body column. To access
data, we need to cast body as string (it is imported as binary?) and extract
data as string. On a second transformation, we extract all vairables from data
column. It took me a weekend to figure it out.

var streamingSelectDF =
  eventhubs
    .select(
      ($"enqueuedTime").as("Enqueued_Time"),
      ($"systemProperties.iothub-connection-device-id").as("Device_ID"),
      get_json_object(($"body").cast("string"), "$.data").alias("data")
    )
```

```
import org.apache.spark.sql.functions._
streamingSelectDF: org.apache.spark.sql.DataFrame = [Enqueued_Time: timestamp,
Device_ID: string ... 1 more field]
```

```
// display(streamingSelectDF)
```

```
//this df has time stamp values, id value and the data from IoT. On schema, we
define the datatype of each attriute

import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._
import org.apache.spark.sql.types._

val schema = (new StructType)
      .add("temp", DoubleType)
      .add("humidity", DoubleType)
      .add("pressure", DoubleType)
      .add("air-quality", StringType)
      .add("dust-lpo", DoubleType)
      .add("dust-ratio", DoubleType)
      .add("dust-concentration", DoubleType)


val df_temp = streamingSelectDF
          .select(
            $"Enqueued_Time",
            $"Device_ID",
            (from_json($"data".cast("string"), schema).as("telemetry_json")))
           .select("Enqueued_Time","Device_ID", "telemetry_json.*")
```

```
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._
import org.apache.spark.sql.types._
schema: org.apache.spark.sql.types.StructType = StructType(StructField(temp,Do
ubleType,true), StructField(humidity,DoubleType,true), StructField(pressure,Do
ubleType,true), StructField(air-quality,StringType,true), StructField(dust-lp
o,DoubleType,true), StructField(dust-ratio,DoubleType,true), StructField(dust-
concentration,DoubleType,true))
df_temp: org.apache.spark.sql.DataFrame = [Enqueued_Time: timestamp, Device_I
D: string ... 7 more fields]
```

```
val df = df_temp
            .select(
              $"Enqueued_Time",
              $"Device_ID",
              $"temp",
              $"humidity",
              $"pressure",
              ($"air-quality").as("air_quality"),
              ($"dust-lpo").as("dust_lpo"),
              ($"dust-ratio").as("dust_ratio"),
              ($"dust-concentration").as("dust_concentration")
            )


//display(df)
```

```
df: org.apache.spark.sql.DataFrame = [Enqueued_Time: timestamp, Device_ID: str
ing ... 7 more fields]
```

```
df.createOrReplaceTempView("device_telemetry_data")
```

```
val finalDF = spark.sql("SELECT Date(Enqueued_Time) Date_Enqueued,
Hour(Enqueued_Time) Hour_Enqueued, Enqueued_Time, Device_ID, temp, humidity,
pressure, air_quality, dust_lpo, dust_ratio, dust_concentration FROM
device_telemetry_data")
```

```
finalDF: org.apache.spark.sql.DataFrame = [Date_Enqueued: date, Hour_Enqueued:
int ... 9 more fields]
```

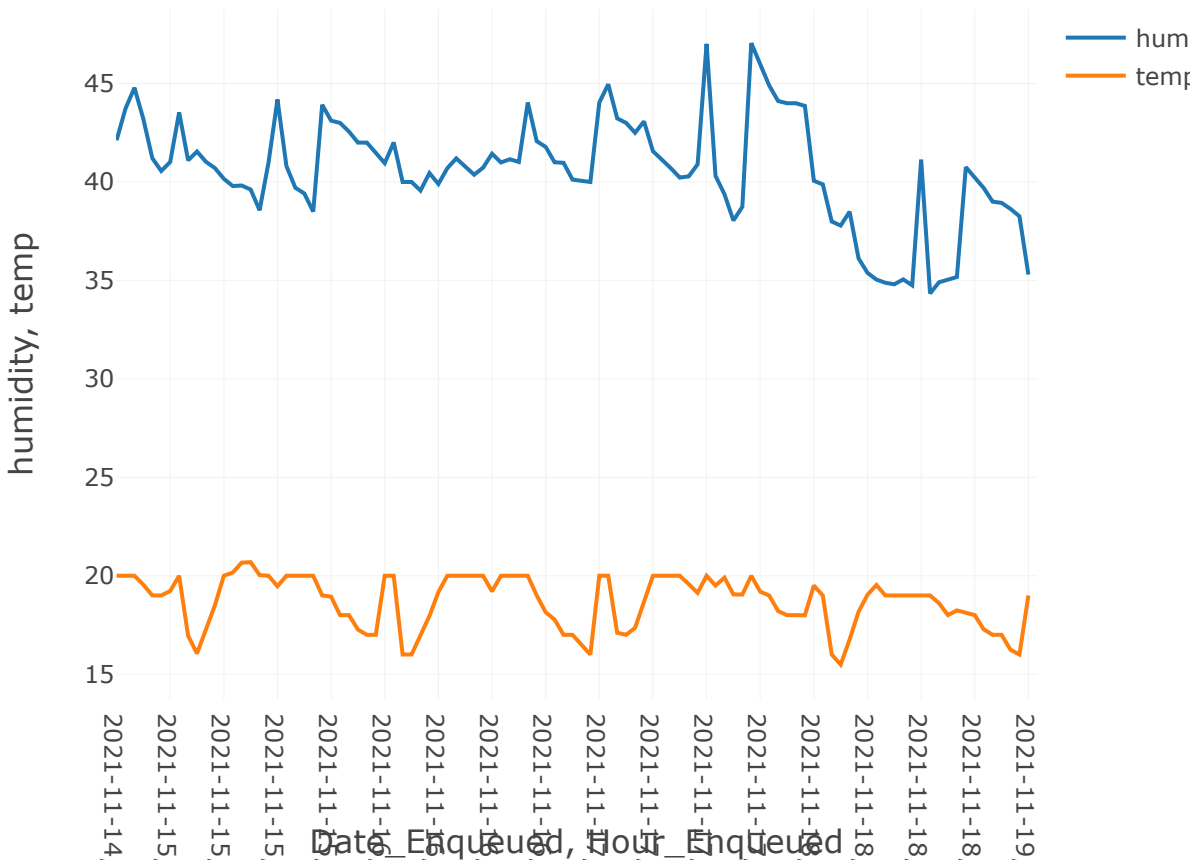
```
finalDF.writeStream
  .outputMode("append")
  .option("mergeSchema", "true")
  .option("checkpointLocation", "/delta/events/_checkpoints/etl-from-json")
  .format("delta")
  .partitionBy("Date_Enqueued", "Hour_Enqueued")
  .table("telemetry_data")
```

95f24036-93f9-4327-9d90-c8db4aaaece5 *Last updated: 288 days ago*

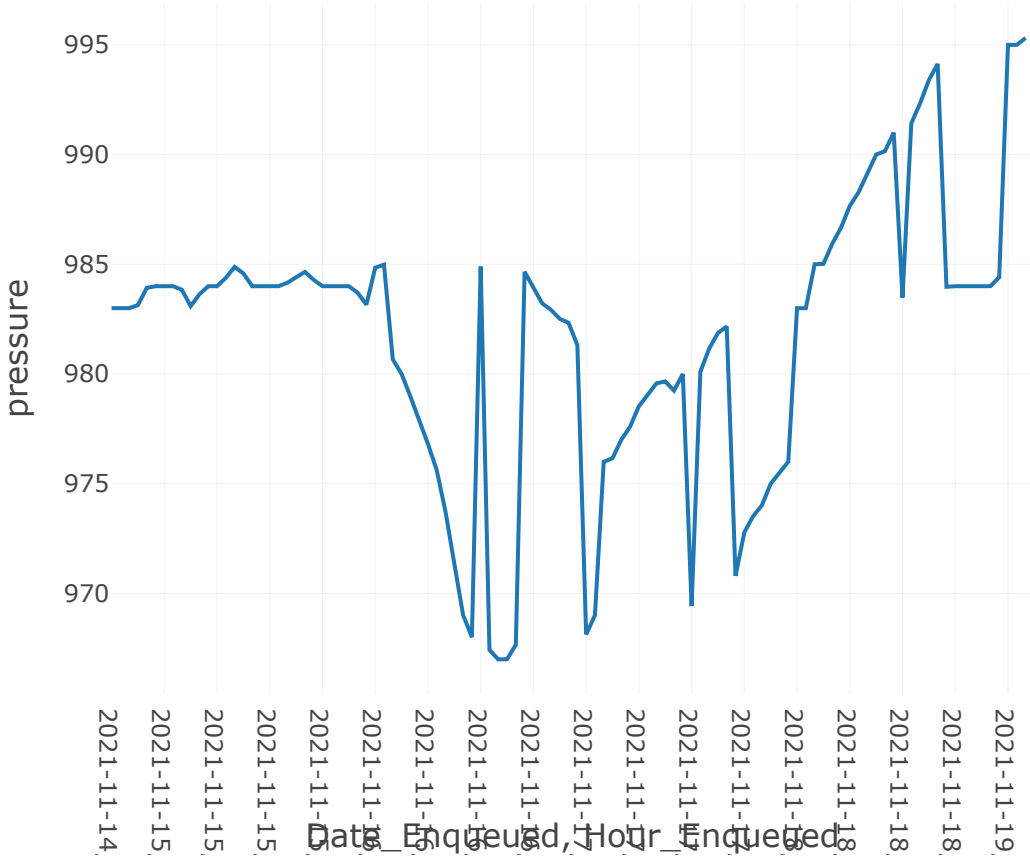
```
Lost connection to cluster. The notebook may have been detached or the clus
ter may have been terminated due to an error in the driver such as an OutOf
MemoryError.
```

```
res7: org.apache.spark.sql.streaming.StreamingQuery = org.apache.spark.sql.exe
cution.streaming.StreamingQueryWrapper@6f7a334e
```

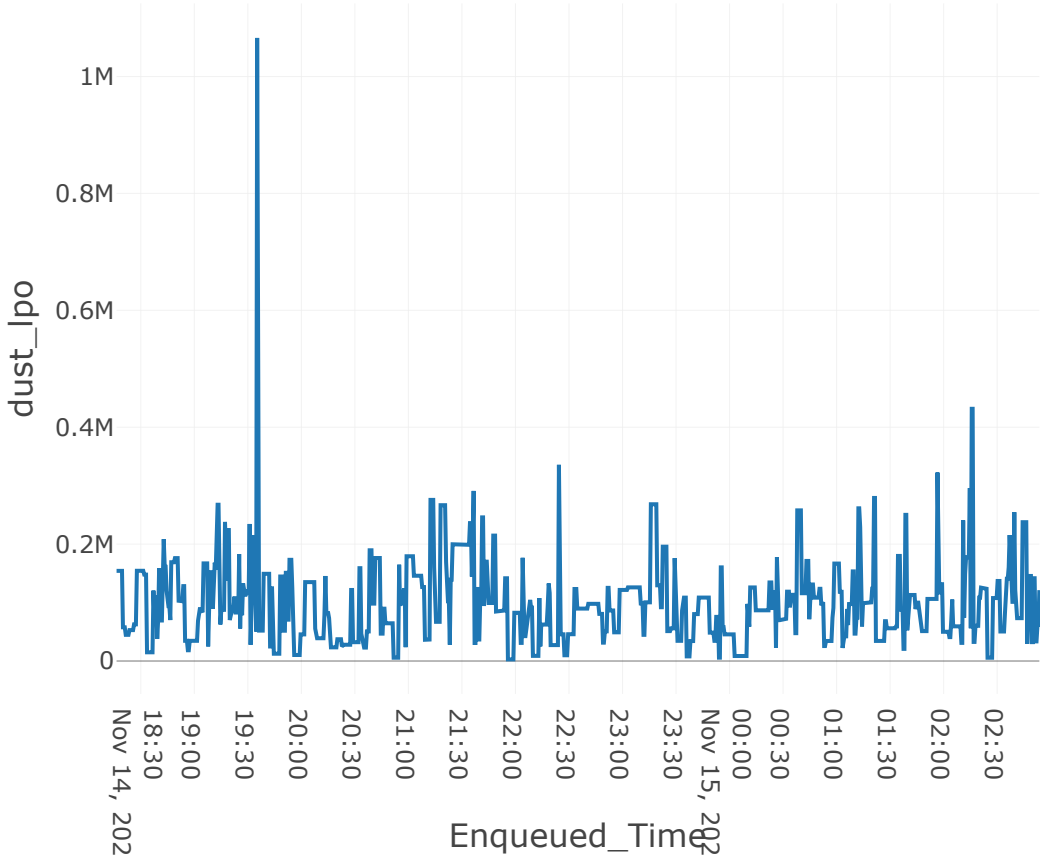
```
%sql
SELECT * FROM telemetry_data ORDER BY enqueued_time ASC
```




```
%sql
SELECT * FROM telemetry_data ORDER BY enqueued_time ASC
```




```
%sql
SELECT * FROM telemetry_data ORDER BY enqueued_time ASC
```




```
%sql
SELECT date_enqueued, hour_enqueued, air_quality, COUNT(air_quality) FROM
telemetry_data GROUP BY date_enqueued, hour_enqueued, air_quality ORDER BY
date_enqueued, hour_enqueued, air_quality ASC
```

| | date_enqueued | hour_enqueued | air_quality | count(air_quality) |
|---|---|---|---|---|
| 1 | 2021-11-14 | 18 | Fresh Air | 84 |
| 2 | 2021-11-14 | 18 | Low Pollution | 1 |
| 3 | 2021-11-14 | 19 | Fresh Air | 114 |
| 4 | 2021-11-14 | 20 | Fresh Air | 117 |
| 5 | 2021-11-14 | 21 | Fresh Air | 116 |
| 6 | 2021-11-14 | 22 | Fresh Air | 116 |
| 7 | 2021-11-14 | 23 | Fresh Air | 116 |

Showing all 124 rows.

| | Date_Enqueued | Hour_Enqueued | Enqueued_Time | Device_ |
|---|---|---|---|---|
| 1 | 2021-11-14 | 18 | 2021-11-14T18:16:24.821+0000 | "e00fce6 |

| | | | | |
|---|---|---|---|---|
| 2 | 2021-11-14 | 18 | 2021-11-14T18:16:55.297+0000 | "e00fce6 |
| 3 | 2021-11-14 | 18 | 2021-11-14T18:17:25.770+0000 | "e00fce6 |
| 4 | 2021-11-14 | 18 | 2021-11-14T18:17:56.446+0000 | "e00fce6 |
| 5 | 2021-11-14 | 18 | 2021-11-14T18:18:27.498+0000 | "e00fce6 |
| 6 | 2021-11-14 | 18 | 2021-11-14T18:18:57.675+0000 | "e00fce6 |
| 7 | 2021-11-14 | 18 | 2021-11-14T18:19:28.226+0000 | "e00fce6 |

Truncated results, showing first 1000 rows.