```
import org.apache.spark.sql.functions._
import org.apache.spark.sql.streaming.{Trigger,ProcessingTime}


val retail_data = "/user/bigdata/*.csv"
val staticDataFrame = spark.read.format("csv").option("header", "true").option("inferSchema", "true").load(retail_data)

val staticSchema = staticDataFrame.schema

import spark.implicits._
staticDataFrame.printSchema()
```

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

scala> import org.apache.spark.sql.streaming.{Trigger,ProcessingTime}
import org.apache.spark.sql.streaming.{Trigger, ProcessingTime}

scala>

scala>

scala> val retail_data = "/user/bigdata/*.csv"
retail_data: String = /user/bigdata/*.csv
```

```
scala> val staticDataFrame = spark.read.format("csv").option("header", "true").option("inferSchema", "true").load(retail_data)

staticDataFrame: org.apache.spark.sql.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]


scala>


scala> val staticSchema = staticDataFrame.schema

staticSchema: org.apache.spark.sql.types.StructType = StructType(StructField(InvoiceNo,StringType,true), StructField(StockCode,StringType,true), StructField(Description,StringType,true), StructField(Quantity,IntegerType,true), StructField(InvoiceDate,TimestampType,true), StructField(UnitPrice,DoubleType,true), StructField(CustomerID,DoubleType,true), StructField(Country,StringType,true))


scala>


scala> import spark.implicits._

import spark.implicits._


scala> staticDataFrame.printSchema()

root
 |-- InvoiceNo: string (nullable = true)
 |-- StockCode: string (nullable = true)
 |-- Description: string (nullable = true)
 |-- Quantity: integer (nullable = true)
 |-- InvoiceDate: timestamp (nullable = true)
 |-- UnitPrice: double (nullable = true)
 |-- CustomerID: double (nullable = true)
 |-- Country: string (nullable = true)
```

```
spark.conf.set("spark.sql.shuffle.partitions", 2)

val streamingDataFrame = spark.readStream.schema(staticSchema).option("maxFilesPerTrigger", 10).format("csv").option("header", "true").load(retail_data)

streamingDataFrame.isStreaming

val purchaseQuery = streamingDataFrame

purchaseQuery.createOrReplaceTempView("myTable")
```

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

scala> import org.apache.spark.sql.streaming.{Trigger,ProcessingTime}
import org.apache.spark.sql.streaming.{Trigger, ProcessingTime}

scala>

scala>

scala> val retail_data = "/user/bigdata/*.csv"
retail_data: String = /user/bigdata/*.csv

scala> val staticDataFrame = spark.read.format("csv").option("header", "true").option("inferSchema", "true").load(retail_data)
staticDataFrame: org.apache.spark.sql.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]
```

```
scala>


scala> val staticSchema = staticDataFrame.schema

staticSchema: org.apache.spark.sql.types.StructType =
StructType(StructField(InvoiceNo,StringType,true), StructField(StockCode,StringType,true),
StructField(Description,StringType,true), StructField(Quantity,IntegerType,true),
StructField(InvoiceDate,TimestampType,true), StructField(UnitPrice,DoubleType,true),
StructField(CustomerID,DoubleType,true), StructField(Country,StringType,true))


scala>


scala> import spark.implicits._

import spark.implicits._


scala> staticDataFrame.printSchema()

root
 |-- InvoiceNo: string (nullable = true)
 |-- StockCode: string (nullable = true)
 |-- Description: string (nullable = true)
 |-- Quantity: integer (nullable = true)
 |-- InvoiceDate: timestamp (nullable = true)
 |-- UnitPrice: double (nullable = true)
 |-- CustomerID: double (nullable = true)
 |-- Country: string (nullable = true)
```

```
// Compute average

val average = spark.sql("select StockCode, avg(Quantity) as avgQuantity from myTable where StockCode is not null group by StockCode order by avgQuantity desc")


val query = average.writeStream.format("console").queryName("customer_purchases").outputMode("complete").trigger(ProcessingTime("5 seconds")).start()


scala> val average = spark.sql("select StockCode, avg(Quantity) as avgQuantity from myTable where StockCode is not null group by StockCode order by avgQuantity desc")

average: org.apache.spark.sql.DataFrame = [StockCode: string, avgQuantity: double]


scala>


scala> val query = average.writeStream.format("console").queryName("customer_purchases").outputMode("complete").trigger(ProcessingTime("5 seconds")).start()

query: org.apache.spark.sql.streaming.StreamingQuery = org.apache.spark.sql.execution.streaming.StreamingQueryWrapper@3044dd80


-------------------------------------------

Batch: 0

-------------------------------------------

+---------+-----------------+

|StockCode|      avgQuantity|

+---------+-----------------+

|   17084R|           1440.0|

|    17096|            864.5|

|    17021|            301.0|

|    84950|269.14285714285717|

|    16014|           253.75|
```

| 84077|178.27272727272728|

| 16033| 120.0|

| 22188|118.33333333333333|

| 22492|109.61904761904762|

| 17038| 100.0|

| 71459| 87.42857142857143|

| 22189| 86.76|

| 17003| 75.78571428571429|

| 21137| 71.41176470588235|

| 22275| 69.0|

| 22328| 65.68|

| 51008| 60.2|

| 84826| 60.0|

| 22536| 57.31578947368421|

| 20668| 57.10526315789474|

+---------+------------------+

only showing top 20 rows

Batch: 1

+---------+------------------+

|StockCode| avgQuantity|

+---------+------------------+

| 17096| 577.0|

| 17084R| 552.0|

| 22693| 327.125|

| 16014| 171.5|

| 84950| 164.0|

| 22856| 153.0|

| 84212| 132.25|

| 84077| 129.875|

| 16033| 120.0|

| 17021|114.33333333333333|

| 90057| 104.0|

| 16045| 100.0|

| 22264| 90.5|

| 22492| 84.6875|

| 22188| 83.0|

| 62018| 81.0|

| 85212| 73.0|

| 75178| 72.5|

| 22257| 72.5|

| 22608| 72.4|

+---------+------------------+

only showing top 20 rows

Batch: 2

+---------+------------------+

|StockCode| avgQuantity|

+---------+------------------+

| 47556B| 615.5|

| 17084R| 552.0|

| 17096| 297.5|

| 22957| 240.0|

| 22967| 240.0|

| 17021|183.71428571428572|

| 22693| 180.0625|

| 16014| 171.5|

| 16033| 120.0|

| 84077| 119.7843137254902|

| 40016|112.70588235294117|

| 16045| 100.0|

| 84950| 97.14285714285714|

| 84212| 94.16666666666667|

| 21292| 88.8|

| 17003| 87.95652173913044|

| 22492| 87.2439024390244|

| 21108| 82.07317073170732|

| 85212| 72.66666666666667|

| 75178| 72.5|

+---------+------------------+

only showing top 20 rows

Batch: 3

+---------+------------------+

|StockCode| avgQuantity|

+---------+------------------+

| 47556B| 615.5|

| 17084R| 552.0|

| 17096|214.44444444444446|

| 17021|183.71428571428572|

| 22693| 148.85|

```
|    16014|148.42857142857142|
|    17003|148.06896551724137|
|    16033|             120.0|
|    84077|119.38888888888889|
|    84212|             106.2|
|    16045|             100.0|
|    40016|              99.36|
|    21292|              88.8|
|    84950|              85.75|
|    22492| 78.44680851063829|
|    21108| 75.24444444444444|
|    75178|              72.5|
|    22275|              69.0|
|    62018|              68.5|
|    22856|              66.2|
+---------+------------------+
only showing top 20 rows


-------------------------------------------
Batch: 4
-------------------------------------------
+---------+------------------+
|StockCode|       avgQuantity|
+---------+------------------+
|    37413|            2787.0|
|   79063D|            2560.0|
|   79062D|             960.0|
|   47556B|             615.5|
|   17084R|             360.0|
```

79063C	320.0
17096	197.8
16014	148.42857142857142
17021	147.55555555555554
84077	143.57575757575756
17003	130.58823529411765
16033	120.0
22693	117.48148148148148
44234	110.25
44235	110.25
16045	100.0
84212	91.5
40016	91.35714285714286
21292	88.8
84950	85.75
+---------+------------------+
only showing top 20 rows