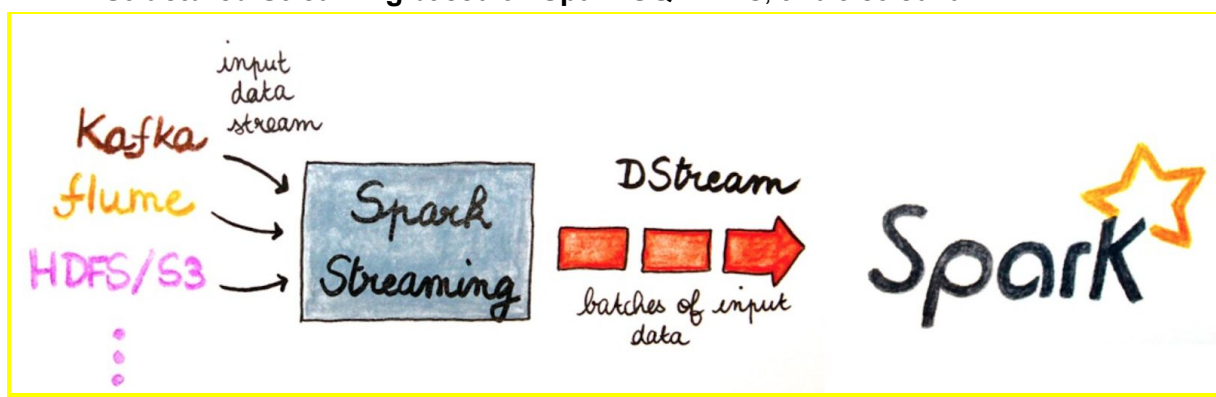


Spark Structured Streaming-Java Application: Read CSV file into Data Frame and Execute multiple queries in stream analytics With Spark Structured Streaming And Java

- I. Introduction
- II. Technologies
- III. Implement some queries using java and spark
- IV. Project Structure
- V. Setup Dependencies on pom.xml
- VI. Configure Log4j file on spark console
- VII. Define Data Model
- VIII. Create a Repository to working with Dataframe(Cars.csv)
- IX. Create a Spark Service
- X. Creating a Menu Driven Program
- XI. Output
- XII. Conclusion

I. Introduction:

In this documentation, we are focused to parse data from a CSV file, perform some queries and output the result in the output using the **Spark Core** and **Spark Structured Streaming** based on **Spark SQL APIs**, and also **Java**.



Note: This Image is about Spark Streaming but in this project we will work with Spark Structured Streaming.

II. Technologies:

- Java 8
- Spark Core 3.0.1

- Spark Structured Streaming 3.0.1
- Spark SQL 3.0.1
- Maven
- IntelliJ IDEA

III. Implement some queries using Java and Spark Structured Streaming:

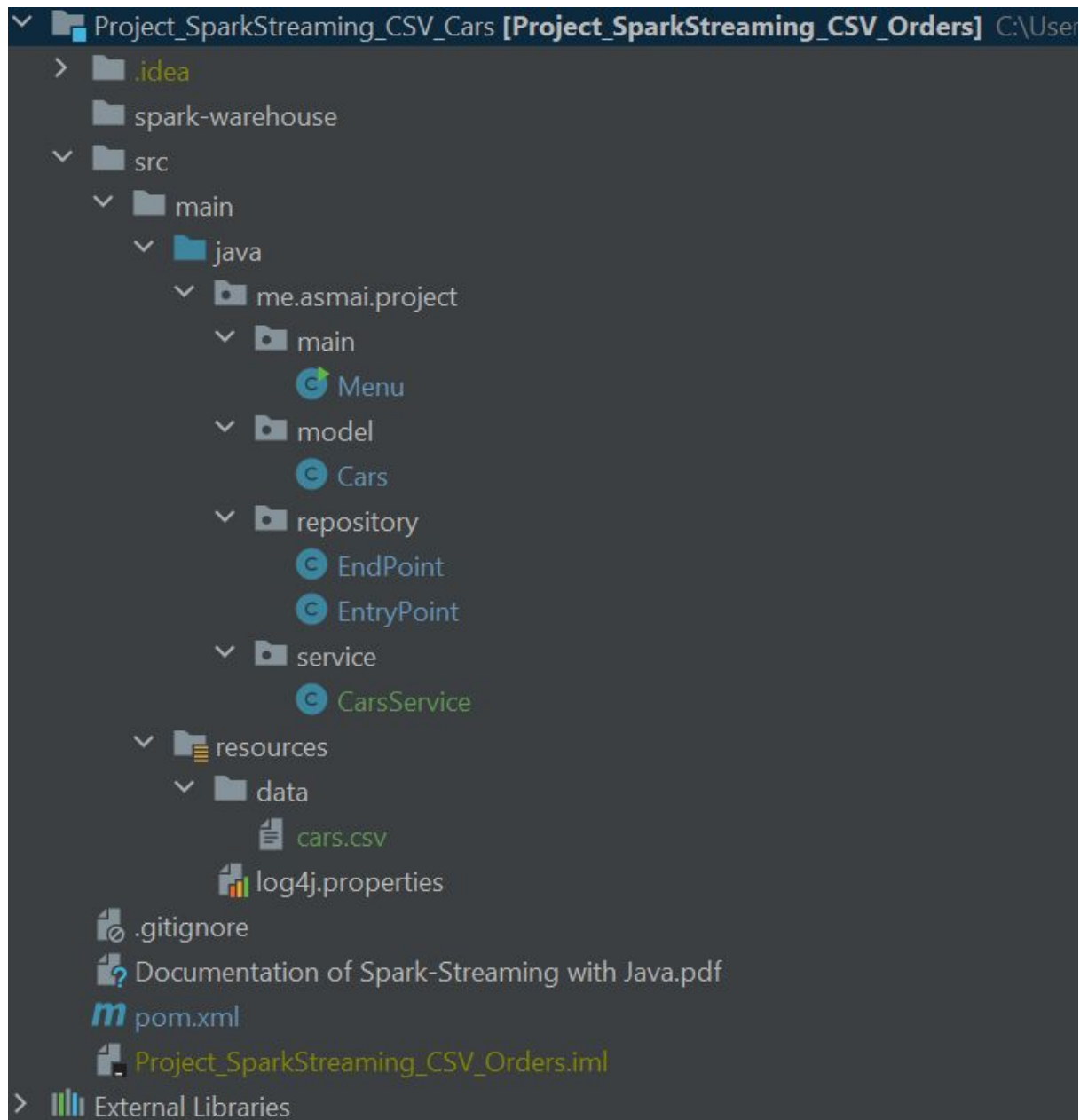
-Let's have a look at the **cars** dataset which we will use for this queries:

Car	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model	Origin
Chevrolet Chevelle	18.0	8	307.0	130.0	3504	12.0	70	US
Buick Skylark	15.0	8	350.0	165.0	3693	11.5	70	US
Plymouth Satellite	18.0	8	318.0	150.0	3436	11.0	70	US
AMC Rebel Sport	16.0	8	304.0	150.0	3433	12.0	70	US
Ford Torino	17.0	8	302.0	140.0	3449	10.5	70	US
Ford Galaxie	15.0	8	429.0	198.0	4341	10.0	70	US
Chevrolet Impala	14.0	8	454.0	220.0	4354	9.0	70	US
Plymouth Fury	14.0	8	440.0	215.0	4312	8.5	70	US
Pontiac Catalina	14.0	8	455.0	225.0	4425	10.0	70	US
AMC Ambassador	15.0	8	390.0	190.0	3850	8.5	70	US
Citroen DS-2	0	4	133.0	115.0	3090	17.5	70	Europe
Chevrolet Chevelle	0	8	350.0	165.0	4142	11.5	70	US
Ford Torino	0	8	351.0	153.0	4034	11.0	70	US
Plymouth Satellite	0	8	383.0	175.0	4166	10.5	70	US
AMC Rebel Sport	0	8	360.0	175.0	3850	11.0	70	US
Dodge Challenger	15.0	8	383.0	170.0	3563	10.0	70	US
Plymouth 'Cuda	14.0	8	340.0	160.0	3609	8.0	70	US
Ford Mustang	0	8	302.0	140.0	3353	8.0	70	US
Chevrolet Malibu	15.0	8	400.0	150.0	3701	8.5	70	US

-These are **queries** to be exported:

- Get all Cars from csv file
- Get Cars By Model
- Get Model of Cars By less HorsePower
- Get Cars sorted by Model and HorsePower.
- Get Cars between two models and Origin and sorted by HorsePower
- Get Cars by Origin and sorted by Model

IV. Project Structure :



V. Setup Dependencies on pom.xml:

After Adding below dependencies on **pom.xml**, It will download all the required packages.

We create a new file **log4j.properties** in order to stop these messages. Here are the contents of **log4j.properties**:

```
#Stop INFO messages displaying on Spark console to get just the result expected
log4j.rootCategory=ERROR, console
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.target=System.err
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{1}: %m%n
```

VII. Define Data Model:

In the **model** package, we define **Cars** class.

model/Cars.class:

```
public class Cars {

    private String car;

    private double MPG;

    private int Cylinders;

    private double Displacement;

    private double Horsepower;

    private double Weight;

    private double Acceleration;

    private double Model;

    private String Origin;

    public Cars(String car, double MPG, int cylinders, double displacement, double horsepower, double weight, double acceleration, double model, String origin) {...}

    public String getCar() { return car; }

    public void setCar(String car) { this.car = car; }

    public double getMPG() { return MPG; }

    public void setMPG(double MPG) { this.MPG = MPG; }
```

```
    public String getCar() { return car; }

    public void setCar(String car) { this.car = car; }

    public double getMPG() { return MPG; }

    public void setMPG(double MPG) { this.MPG = MPG; }

    public double getCylinders() { return Cylinders; }

    public void setCylinders(int cylinders) { Cylinders = cylinders; }

    public double getDisplacement() { return Displacement; }

    public void setDisplacement(double displacement) { Displacement = displacement; }

    public double getHorsepower() { return Horsepower; }

    public void setHorsepower(double horsepower) { Horsepower = horsepower; }

    public double getWeight() { return Weight; }

    public void setWeight(double weight) { Weight = weight; }

    public double getAcceleration() { return Acceleration; }

    public void setAcceleration(double acceleration) { Acceleration = acceleration; }

    public double getModel() { return Model; }

    public void setModel(double model) { Model = model; }
```


VIII. Create a Repository to working with Dataframe(Cars.csv):

Let's create a repository to interact with **Cars** from the csv file.

In the **repository** package, create a class **EntryPoint** which is responsible for reading **CSV file** and loading the data into a **spark dataframe** with a custom schema.

```
import org.apache.spark.sql.types.StructType;

public class EntryPoint {

    public EntryPoint() { }

    public static SparkSession sparkSession(){
        return SparkSession
            .builder()
            .appName(" Application with Spark SQL and Java")
            .master("local[*]")
            .getOrCreate();
    }

    public static Dataset<Cars> getDataset(){
        Encoder<Cars> carsEncoder = Encoders.bean(Cars.class);

        Dataset<Cars> carsDataset = sparkSession().read()
            .option("header", "true")
            .option("treatEmptyValuesAsNulls", "true")
            .option("inferSchema", "true")
            .option("mode", "DROPMALFORMED")
            .option("delimiter", ";")
            .csv( path: "src/main/resources/cars.csv")
            .as(carsEncoder);

        carsDataset.registerTempTable( tableName: "cars");
        return carsDataset;
    }
}
```

However, the second one is **EndPoint** class which is responsible for saving the content of the streaming Dataset out into the console using two output modes ("Append", "Complete"). We are using **Complete** mode in the **last two queries** because the **Append** mode doesn't support **sorting** operations and **aggregation**.

```
public class EndPoint {
```

```

public EndPoint() {}

    public static void displayDatasetWithCars(Dataset<Cars> dataset, int
numberRows) throws StreamingQueryException {
        dataset.writeStream()
            .format("console")
            .outputMode("append")
            .option("numRows", numberOfRows)
            .start()
            .awaitTermination();
    }

    public static void displayDatasetWithRows(Dataset<Row> dataset, int
numberOfRows, String outputMode) throws StreamingQueryException {
        dataset.writeStream()
            .format("console")
            .outputMode(outputMode)
            .option("numRows", numberOfRows)
            .start()
            .awaitTermination();
    }
}

```

IX. Create a Spark Service:

CarsService class uses **Repository/EntryPoint** class for 5 functions:

- **getAllCars(int numberOfRows)**: Get all Cars from csv file
- **getCarsByModel(double model)**: Get Cars By Model
- **getModelOfCarsByLessHorsePower()**: Get only the Model of Cars which have less HorsePower.
- **getCarsSortedByModelAndHorsePower()**: Get Cars sorted by Model and HorsePower.
- **getCarsBetweenTwoModelsOfAnOriginAndSortedByHorsePower(double model1, double model2, String origin)**: Get Cars between two models and Origin and sorted by HorsePower.
- **getCarsByOriginAndSortedByModel(String origin)**: Get Cars by Origin and sorted by Model.

Here is the code of **service/CarsService.java**:

```

package me.asmai.project.service;

import static me.asmai.project.repository.EntryPoint.getDataset;
import static me.asmai.project.repository.EntryPoint.sparkSession;
import me.asmai.project.repository.EndPoint;
import me.asmai.project.repository.EntryPoint;
import me.asmai.project.model.Cars;
import org.apache.spark.api.java.function.FilterFunction;
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Row;
import org.apache.spark.sql.streaming.StreamingQueryException;
import java.util.concurrent.TimeoutException;

public class CarsService {

    public void getAllCars(int numberOfRows) throws StreamingQueryException, TimeoutException {
        EndPoint.displayDatasetWithCars(EntryPoint.getDataset(), numberOfRows, outputMode: "append");
    }

    public void getCarsByModel(double model) throws StreamingQueryException, TimeoutException {

        Dataset<Cars> cars = EntryPoint.getDataset().filter("Model == \"\" + model + "\"");
        EndPoint.displayDatasetWithCars(cars, numberOfRows: 407, outputMode: "append");
    }

    public void getModelOfCarsByLessHorsePower() throws StreamingQueryException, TimeoutException {

        EntryPoint.getDataset().createOrReplaceTempView( viewName: "cars");
        Dataset<Row> cars = sparkSession().sql( sqlText: "SELECT Model,MIN(Horsepower) FROM cars GROUP BY Model");
        EndPoint.displayDatasetWithRows(cars, numberOfRows: 407, outputMode: "complete");
    }
}

```

```

public void getModelOfCarsByLessHorsePower() throws StreamingQueryException, TimeoutException {

    EntryPoint.getDataset().createOrReplaceTempView( viewName: "cars");
    Dataset<Row> cars = sparkSession().sql( sqlText: "SELECT Model,MIN(Horsepower) FROM cars GROUP BY Model");
    EndPoint.displayDatasetWithRows(cars, numberOfRows: 407, outputMode: "complete");
}

public void getCarsSortedByModelAndHorsePower() throws StreamingQueryException, TimeoutException {

    EntryPoint.getDataset().createOrReplaceTempView( viewName: "cars");
    Dataset<Row> cars = getDataset() Dataset<Cars>
        .groupBy( col: "Model", ...cols: "Horsepower","car","HPG","Cylinders","Weight","Acceleration","Displacement") RelationalGroupedDataset
        .count() Dataset<Row>
        .sort( sortCol: "Model", ...sortCols: "Horsepower");
    EndPoint.displayDatasetWithRows(cars, numberOfRows: 407, outputMode: "complete");
}

public void getCarsByBetweenTwoModelsOfAnOriginAndSortedByHorsePower(double model1, double model2, String origin) throws StreamingQueryException, TimeoutException {
    Dataset<Row> cars = getDataset().filter((FilterFunction<Cars>) car -> car.getOrigin().equals(origin))
        .filter("Model >= \"\" + model1 + "\"")
        .filter("Model <= \"\" + model2 + "\"")
        .groupBy( col: "Horsepower")
        .count()
        .sort( sortCol: "Horsepower");

    EndPoint.displayDatasetWithRows(cars, numberOfRows: 407, outputMode: "complete");
}

public void getCarsByOriginAndSortedByModel(String origin) throws StreamingQueryException, TimeoutException {
    Dataset<Row> cars = getDataset().filter((FilterFunction<Cars>) car -> car.getOrigin().equals(origin))
        .groupBy( col: "Model")
        .count()
        .sort( sortCol: "Model");
    EndPoint.displayDatasetWithRows(cars, numberOfRows: 407, outputMode: "append");
}

```


X. Creating a Menu Driven Program :

Let's create a **Menu** class under package **Main** to obtain input from a user by displaying a list of options.

main/Menu.java:

```
package me.asmai.project.main;

import me.asmai.project.service.CarsService;
import org.apache.spark.sql.streaming.StreamingQueryException;

import java.util.Scanner;
import java.util.concurrent.TimeoutException;

public class Menu {

    public static Scanner scanner = new Scanner(System.in);
    public static CarsService carsService = new CarsService();

    public static void main(String[] args) throws TimeoutException, StreamingQueryException {

        int menuOption = 0;
        double model1;
        double model;
        double model2;
        String origin;

        do {
            menuOption = showMenu();

            switch (menuOption) {

                case 1:
                    System.out.println("Enter the rows number of Cars that you want preview : ");
                    int numberRows = scanner.nextInt();
                    carsService.getAllCars(numberRows);
                    break;

                case 2:
                    System.out.println("Enter the model of car : Ex= 70");
```

```
case 1:
    System.out.println("Enter the rows number of Cars that you want preview : ");
    int numberRows = scanner.nextInt();
    carsService.getAllCars(numberRows);
    break;
case 2:
    System.out.println("Enter the model of car : Ex= 70");
    scanner.nextLine();
    model = scanner.nextInt();
    carsService.getCarsByModel(model);
    break;
case 3:
    carsService.getModelOfCarsByLessHorsePower();
    break;
case 4:
    carsService.getCarsSortedByModelAndHorsePower();
    break;
case 5:
    System.out.println("Enter the model of the car : Ex: 80");
    scanner.nextLine();
    model1 = scanner.nextInt();
    System.out.println("Enter the second model of the car : Ex: 81");
    scanner.nextLine();
    model2 = scanner.nextInt();
    System.out.println("Enter the origin of the car : Ex: US");
    scanner.nextLine();
    origin = scanner.nextLine();
    carsService.getCarsByBetweenTwoModelsOfAnOriginAndSortedByHorsePower(model1, model2, origin);
    break;
case 6:
    System.out.println("Enter the origin of the car : Ex: Japan");
    scanner.nextLine();
    origin = scanner.nextLine();
    carsService.getCarsByOriginAndSortedByModel(origin);
```

```

        origin = scanner.nextLine();
        carsService.getCarsByOriginAndSortedByModel(origin);
    case 7:
        System.out.println("Quitting Program...");
        break;
    default:
        System.out.println("Sorry, please enter a valid Option");
    }
} while (menuOption != 7);

// Exiting message when user decides to quit Program
System.out.println("Thanks for using this Program...");
}

public static int showMenu() {
    int option = 0;

    System.out.println("Menu:");
    System.out.println("1. Get All Cars form CSV file ");
    System.out.println("2. Get Cars By Model");
    System.out.println("3. Get Model of Cars By Less HorsePower");
    System.out.println("4. Get Cars Sorted by Model and HorsePower");
    System.out.println("5. Get Cars By Origin and Between Two Models and Sorted by HorsePower");
    System.out.println("6. Get Cars by Origin and Sorted by Model");
    System.out.println("7. Quit Program");

    // Getting query number from above menu
    System.out.println("Enter the number of Query from above...");
    option = scanner.nextInt();

    return option;
}

```

XI. Output:

While executing each query , you will be able to see below its content in the console.

1. Menu:

```

Menu:
1. Get All Cars form CSV file
2. Get Cars By Model
3. Get Model of Cars By Less HorsePower
4. Get Cars Sorted by Model and HorsePower
5. Get Cars By Origin and Between Two Models and Sorted by HorsePower
6. Get Cars by Origin and Sorted by Model
7. Quit Program
Enter the number of Query from above...

```

2. First Query:

Enter the number of Query from above...

1

Enter the rows number of Cars that you want preview :

10

WARNING: An illegal reflective access operation has occurred

WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/C:/Users/alMostapha/

WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform

WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operati

WARNING: All illegal access operations will be denied in a future release

Batch: 0

```
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Car| MPG|Cylinders|Displacement|Horsepower|Weight|Acceleration|Model|Origin|
+-----+-----+-----+-----+-----+-----+-----+-----+
|Chevrolet Chevell...|18.0|      8|      307.0|      130.0|3504.0|      12.0| 70.0|  US|
|  Buick Skylark 320|15.0|      8|      350.0|      165.0|3693.0|      11.5| 70.0|  US|
|Plymouth Satelllite|18.0|      8|      318.0|      150.0|3436.0|      11.0| 70.0|  US|
|      AMC Rebel SST|16.0|      8|      304.0|      150.0|3433.0|      12.0| 70.0|  US|
|      Ford Torino|17.0|      8|      302.0|      140.0|3449.0|      10.5| 70.0|  US|
|  Ford Galaxie 500|15.0|      8|      429.0|      198.0|4341.0|      10.0| 70.0|  US|
|  Chevrolet Impala|14.0|      8|      454.0|      220.0|4354.0|       9.0| 70.0|  US|
|Plymouth Fury iii|14.0|      8|      440.0|      215.0|4312.0|       8.5| 70.0|  US|
|  Pontiac Catalina|14.0|      8|      455.0|      225.0|4425.0|      10.0| 70.0|  US|
| AMC Ambassador DPL|15.0|      8|      390.0|      190.0|3850.0|       8.5| 70.0|  US|
+-----+-----+-----+-----+-----+-----+-----+-----+
```

3. Second Query:

Enter the number of Query from above...

2

Enter the model of car : Ex= 70

72

WARNING: An illegal reflective access operation has occurred

WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/C:/Users/alMostapha/.m2/repository/org/apache/spark/spark-unsafe/2.4.0/spark-unsafe-2.4.0.jar) of method java.lang.ProcessImpl.start0(java.lang.String,java.lang.String[],java.lang.ProcessImpl)

WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform

WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operation

WARNING: All illegal access operations will be denied in a future release

Batch: 0

```
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Car| MPG|Cylinders|Displacement|Horsepower|Weight|Acceleration|Model|Origin|
+-----+-----+-----+-----+-----+-----+-----+-----+
|Toyota Corolla Ha...|24.0|      4|    113.0|     95.0|2278.0|     15.5| 72.0| Japan|
|  Dodge Colt Hardtop|25.0|      4|     97.5|     80.0|2126.0|     17.0| 72.0|   US|
|  Volkswagen Type 3|23.0|      4|     97.0|     54.0|2254.0|     23.5| 72.0|Europe|
|   Chevrolet Vega|20.0|      4|    140.0|     90.0|2408.0|     19.5| 72.0|   US|
| Ford Pinto Runabout|21.0|      4|    122.0|     86.0|2226.0|     16.5| 72.0|   US|
|   Chevrolet Impala|13.0|      8|    350.0|    165.0|4274.0|     12.0| 72.0|   US|
| Pontiac Catalina|14.0|      8|    400.0|    175.0|4385.0|     12.0| 72.0|   US|
| Plymouth Fury III|15.0|      8|    318.0|    150.0|4135.0|     13.5| 72.0|   US|
| Ford Galaxie 500|14.0|      8|    351.0|    153.0|4129.0|     13.0| 72.0|   US|
| AMC Ambassador SST|17.0|      8|    304.0|    150.0|3672.0|     11.5| 72.0|   US|
| Mercury Marquis|11.0|      8|    429.0|    208.0|4633.0|     11.0| 72.0|   US|
|Buick LeSabre Custom|13.0|      8|    350.0|    155.0|4502.0|     13.5| 72.0|   US|
|Oldsmobile Delta ...|12.0|      8|    350.0|    160.0|4456.0|     13.5| 72.0|   US|
|Chrysler Newport ...|13.0|      8|    400.0|    190.0|4422.0|     12.5| 72.0|   US|
| Mazda RX2 Coupe|19.0|      3|     70.0|     97.0|2330.0|     13.5| 72.0| Japan|
| AMC Matador (sw)|15.0|      8|    304.0|    150.0|3892.0|     12.5| 72.0|   US|
|Chevrolet Chevell...|13.0|      8|    307.0|    130.0|4098.0|     14.0| 72.0|   US|
```


4. Third Query:

Enter the number of Query from above...

3

WARNING: An illegal reflective access operation has occurred

WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/C:/Users/a

WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Pl

WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective acce

WARNING: All illegal access operations will be denied in a future release

Batch: 0

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

|Model|min(Horsepower)|

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

| 70.0| 46.0|

| 75.0| 53.0|

| 80.0| 0.0|

| 77.0| 58.0|

| 78.0| 48.0|

| 79.0| 65.0|

| 71.0| 0.0|

| 72.0| 54.0|

| 82.0| 0.0|

| 74.0| 0.0|

| 81.0| 0.0|

| 73.0| 46.0|

| 76.0| 52.0|

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

5. Fourth Query :

Enter the number of Query from above...

4

WARNING: An illegal reflective access operation has occurred

WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/C:/Users/alMostapha/Downloads/spark-2.4.0-bin-hadoop2.7/jars/org.apache.spark.unsafe.Platform-2.4.0.jar) of method java.lang.ProcessImpl.start0(java.lang.String,java.lang.String[],java.lang.ProcessImpl\$NativeStart\$1)

WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform

WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operation

WARNING: All illegal access operations will be denied in a future release

Batch: 0

Model	Horsepower	car	MPG	Cylinders	Weight	Acceleration	Displacement	count
70.0	46.0	Volkswagen 1131 D...	26.0	4	1835.0	20.5	97.0	1
70.0	85.0	Ford Maverick	21.0	6	2587.0	16.0	200.0	1
70.0	87.0	Peugeot 504	25.0	4	2672.0	17.5	110.0	1
70.0	88.0	Datsun PL510	27.0	4	2130.0	14.5	97.0	1
70.0	90.0	AMC Gremlin	21.0	6	2648.0	15.0	199.0	1
70.0	90.0	Audi 100 LS	24.0	4	2430.0	14.5	107.0	1
70.0	95.0	Toyota Corolla Ma...	24.0	4	2372.0	15.0	113.0	1
70.0	95.0	Plymouth Duster	22.0	6	2833.0	15.5	198.0	1
70.0	95.0	Saab 99e	25.0	4	2375.0	17.5	104.0	1
70.0	97.0	AMC Hornet	18.0	6	2774.0	15.5	199.0	1
70.0	113.0	BMW 2002	26.0	4	2234.0	12.5	121.0	1
70.0	115.0	Citroen DS-21 Pallas	0.0	4	3090.0	17.5	133.0	1
70.0	130.0	Chevrolet Chevell...	18.0	8	3504.0	12.0	307.0	1
70.0	140.0	Ford Mustang Boss...	0.0	8	3353.0	8.0	302.0	1
70.0	140.0	Ford Torino	17.0	8	3449.0	10.5	302.0	1
70.0	150.0	AMC Rebel SST	16.0	8	3433.0	12.0	304.0	1
70.0	150.0	Plymouth Satellite	18.0	8	3436.0	11.0	318.0	1
70.0	150.0	Chevrolet Monte C...	15.0	8	3761.0	9.5	400.0	1
70.0	153.0	Ford Torino (sw)	0.0	8	4034.0	11.0	351.0	1
70.0	160.0	Plymouth 'Cuda 340	14.0	8	3609.0	8.0	340.0	1
70.0	165.0	Chevrolet Chevell...	0.0	8	4142.0	11.5	350.0	1
70.0	165.0	Buick Skylark 320	15.0	8	3693.0	11.5	350.0	1
70.0	170.0	Dodge Challenger SE	15.0	8	3563.0	10.0	383.0	1
70.0	175.0	Plymouth Satellit...	0.0	8	4166.0	10.5	383.0	1

6. Fifth Query:

```
Enter the number of Query from above...
5
Enter the model of the car : Ex: 80
81
Enter the second model of the car : Ex: 81
82
Enter the origin of the car : Ex: US
US
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/C:/)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.un
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflecti
WARNING: All illegal access operations will be denied in a future release
-----
Batch: 0
-----
+-----+-----+
|Horsepower|count|
+-----+-----+
|      0.0|    1|
|     63.0|    2|
|     64.0|    1|
|     65.0|    2|
|     70.0|    1|
|     79.0|    1|
|     82.0|    1|
|     84.0|    6|
|     85.0|    3|
|     86.0|    1|
|     88.0|    4|
|     90.0|    2|
|     92.0|    3|
|    105.0|    1|
|    110.0|    3|
|    112.0|    1|
+-----+-----+
```

7. Sixth Query :

```
Enter the number of Query from above...
6
Enter the origin of the car : Ex: Japan
Japan
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.sql.execution.python.PythonUDF$Companion (file:/usr/local/spark/spark-warehouse/scala_2.10/org.apache.spark/spark-warehouse-assembly-2.10.0.jar) of method java.lang.ProcessImpl.start()
WARNING: Please consider reporting this to the maintainers of org.apache.spark
WARNING: Use --illegal-access=warn to enable warnings of this type.
WARNING: All illegal access operations will be denied in future

-----
Batch: 0
-----
+-----+-----+
|Model|count|
+-----+-----+
| 70.0|    2|
| 71.0|    4|
| 72.0|    5|
| 73.0|    4|
| 74.0|    6|
| 75.0|    4|
| 76.0|    4|
| 77.0|    6|
| 78.0|    8|
| 79.0|    2|
| 80.0|   13|
| 81.0|   12|
| 82.0|    9|
+-----+-----+
```

8. Quit Program:

```
Enter the number of Query from above...
7
Quitting Program...
Thanks for using this Program...

Process finished with exit code 0
```

XII. Wrapping Up:

In this project, we have created a spark application using **Spark Core** and **Spark Structured Streaming** with **Java**. Here, we have loaded the CSV file into **Data Frame**. And perform some queries in stream analytics. Finally, We saving the content of the streaming Dataset out into the console.

If you want to test the examples above, you will find my Github code link:

[Read CSV file into Data Frame and Execute multiple queries in stream analytics](#)