USED CAR DATA ANALYSIS

A PROJECT REPORT

Submitted by

BL.EN.U4AIE19007 APOORVA.M

BL.EN.U4AIE19041 TANUJ.M

BL.EN.U4AIE19068 AISHWARYA.V

19AIE304 BIG DATA AND DATABASE MANAGEMENT

AMRITA SCHOOL OF ENGINEERING, BANGALORE

AMRITA VISHWA VIDYAPEETHAM

BANGALORE 560 035

TABLE OF CONTENTS

ABSTRACT

The auto industry is changing rapidly and car prices are only going up. So, to speak, new cars are getting costlier each year, making them a very high value purchase for the common man. And quite ironically, the average life span of a car is going down despite the steady rise in prices, which brings in good news for potential used car buyers.

A used car in fact makes more sense for first time buyers upgrading from two-wheelers or public transportation, or for that matter, someone looking to buy a second set of wheels in the family. This Project is focused on filtering out cars for people from different walks of life depending on their requirements, be it for personal use or professional use. The project is implemented using Spark SQL data frame, structed streaming and Visualizations have been performed using spark.

Key Words: Pyspark, Spark Streaming, Spark SQL, DataFrame, Visualization, Big Data, Database

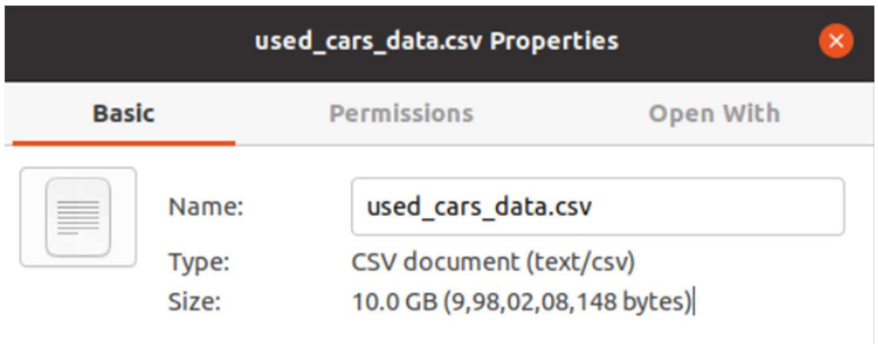
INTRODUCTION

ABOUT THE DATASET

Context

Kaggle Dataset**:** https://www.kaggle.com/ananaymital/us-used-cars-dataset

Dataset size: 9.98 GB



The dataset contains 3 million real world used cars details.

This data was obtained by running a self-made crawler on CarGurus inventory in September 2020.

Acknowledgements

This data is for academic, research and individual experimentation only and is not intended for commercial purposes.

MOTIVATION

Imagine a web app that can estimate the listing price of a vehicle. What features of the vehicle should be used to build a price prediction regression model?

A used car in fact makes more sense for first time buyers upgrading from two-wheelers or public transportation, or for that matter, someone looking to buy a second set of wheels in the family. This Project is focused on filtering out cars

for people from different walks of life depending on their requirements, be it for personal use or professional use. The project is implemented using Spark SQL data frame, structed streaming and Visualizations have been performed using spark.

Pros And Cons of Buying A Certified Pre Owned Car | CarTrade Blog
Read an auto guide on Pros and Cons of Buying a Certified Pre-Owned Car on CarTrade Blog.

DATASET ATTRIBUTES:

1. vin: Type String. Vehicle Identification Number is a unique encoded string for every vehicle.
2. back_legroom: Type String. Legroom in the rear seat.
3. bed: Type String. Category of bed size (open cargo area) in pickup truck. Null usually means the vehicle isn't a pickup truck
4. bed_height: Type String. Height of bed in inches
5. bed_length: Type String. Length of bed in inches
6. body_type: Type String. Body Type of the vehicle. Like Convertible, Hatchback, Sedan, etc.
7. cabin: Type String. Category of cabin size (open cargo area) in pickup truck. Eg: Crew Cab, Extended Cab, etc.
8. city: Type String. city where the car is listed. Eg: Houston, San Antonio, etc.
9. city*fuel*economy: Type Float. Fuel economy in city traffic in km per litre
10. combine*fuel*economy: Type Float. Combined fuel economy is a weighted average of City and Highway fuel economy in km per litre
11. daysonmarket: Type Integer. Days since the vehicle was first listed on the website.
12. dealer_zip: Type Integer. Zipcode of the dealer
13. description: Type String. Vehicle description on the vehicle's listing page
14. engine_cylinders: Type String. The engine configuration. Eg: I4, V6, etc.

15. engine*displacement: Type Float. engine*displacement is the measure of the cylinder volume swept by all of the pistons of a piston engine, excluding the combustion chambers.
16. engine_type: Type String. The engine configuration. Eg: I4, V6, etc.
17. exterior_color: Type String. Exterior color of the vehicle, usually a fancy one same as the brochure.
18. fleet: Type Boolean. Whether the vehicle was previously part of a fleet.
19. frame_damaged: Type Boolean. Whether the vehicle has a damaged frame.
20. franchise_dealer: Type Boolean. Whether the dealer is a franchise dealer.
21. franchise_make: Type String. The company that owns the franchise.
22. front_legroom: Type String. The legroom in inches for the passenger seat
23. fuel*tank*volume: Type String. Fuel tank's filling capacity in gallons
24. fuel_type: Type String. Dominant type of fuel ingested by the vehicle.
25. has_accidents: Type Boolean. Whether the vin has any accidents registered.
26. height: Type String. Height of the vehicle in inches
27. highway*fuel*economy: Type Float. Fuel economy in highway traffic in km per litre
28. horsepower: Type Float. Horsepower is the power produced by an engine.
29. interior_color: Type String. Interior color of the vehicle, usually a fancy one same as the brochure.
30. isCab: Type Boolean. Whether the vehicle was previously taxi/cab.
31. is_certified: Type Boolean. Whether the vehicle is certified. Certified cars are covered through warranty period
32. is_cpo: Type Boolean. Pre-owned cars certified by the dealer. Certified vehicles come with a manufacturer warranty for free repairs for a certain time period.

33. is_new: Type Boolean. If True means the vehicle was launched less than 2 years ago.
34. is_oemcpo: Type Boolean. Pre-owned cars certified by the manufacturer.
35. latitude: Type Float. Latitude from the geolocation of the dealership.
36. length: Type String. Length of the vehicle in inches
37. listed*date: Type String. The date the vehicle was listed on the website. Does not make days*on*market obsolete. The prices is days*on_market days after the listed date.
38. listing_color: Type String. Dominant color group from the exterior color.
39. listing_id: Unique Type Integer. Listing id from the website
40. longitude: Type Float. Longitude from the geolocation of the dealership.
41. main*picture*url: Type String.

Spark SQL Architecture

The following illustration explains the architecture of Spark SQL −

This architecture contains three layers namely, Language API, Schema RDD, and Data Sources.

- **Language API** − Spark is compatible with different languages and Spark SQL. It is also, supported by these languages- API (python, scala, java, HiveQL).
- **Schema RDD** − Spark Core is designed with special data structure called RDD. Generally, Spark SQL works on schemas, tables, and records. Therefore, we can use the Schema RDD as temporary table. We can call this Schema RDD as Data Frame.
- **Data Sources** − Usually the Data source for spark-core is a text file, Avro file, etc. However, the Data Sources for Spark SQL is different. Those are Parquet file, JSON document, HIVE tables, and Cassandra database.

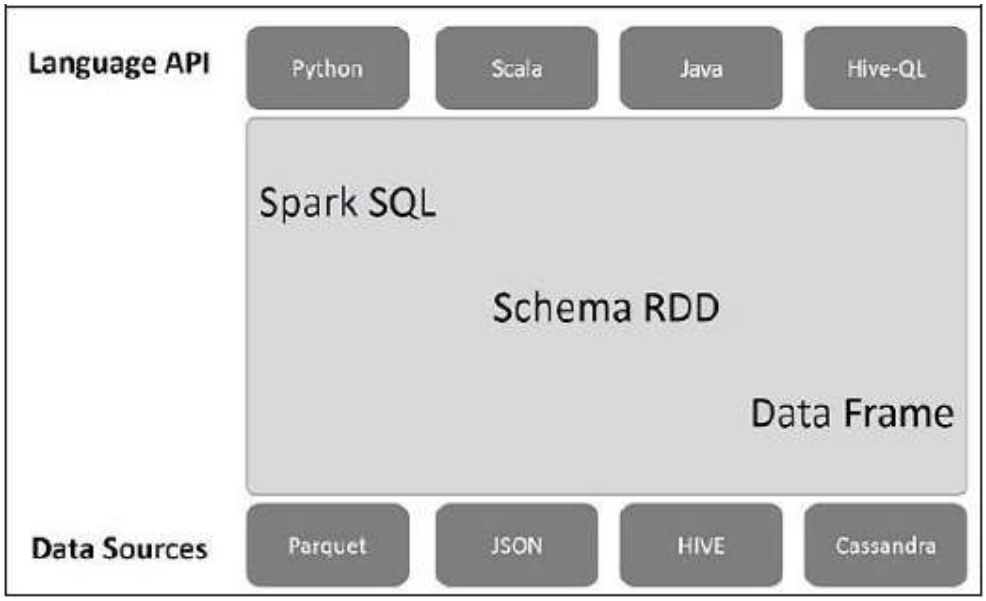


Fig: Spark SQL Architecture

Python Spark (pySpark)

We are using the Python programming interface to Spark (pySpark) .It provides an easy-to-use programming abstraction and parallel runtime. RDDs are the key concept.

PySpark is an interface for Apache Spark in Python. It not only allows you to write Spark applications using Python APIs, but also provides the PySpark shell for interactively analyzing your data in a distributed environment. PySpark supports most of Spark's features such as Spark SQL, DataFrame, Streaming, MLlib (Machine Learning) and Spark Core

Spark Driver and Workers:

A Spark program is two programs: A driver program and a workers program Worker programs run on cluster nodes or in local threads.RDDs are distributed across workers.

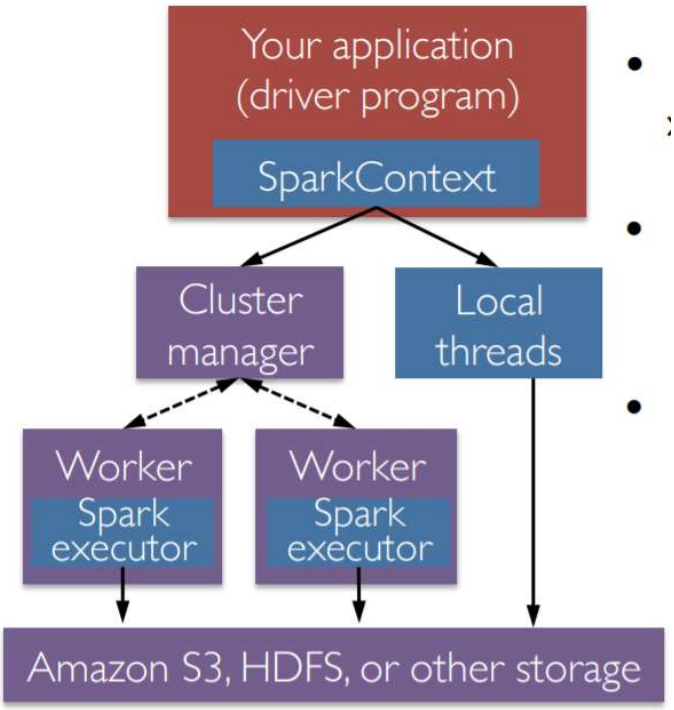


Fig: Spark Context

Spark Context

A Spark program first creates a SparkContext object

- » Tells Spark how and where to access a cluster
- » pySpark shell and Databricks Cloud automatically create the sc variable
- » iPython and programs must use a constructor to create a new SparkContext

• Use SparkContext to create RDDs

Spark Essentials: Master

The master parameter for a SparkContext determines which type and size of cluster to use.

Master Parameter	Description
`local`	run Spark locally with one worker thread (no parallelism)
`local[K]`	run Spark locally with K worker threads (ideally set to number of cores)
`spark://HOST:PORT`	connect to a Spark standalone cluster; PORT depends on config (7077 by default)
`mesos://HOST:PORT`	connect to a Mesos cluster; PORT depends on config (5050 by default)

Fig: Table

STRUCTURED STREAMING

Structured Streaming is a high-level API for stream processing that became production-ready in Spark 2.2. Structured Streaming allows you to take the same operations that you perform in batch mode using Spark's structured APIs, and run them in a streaming fashion. This can reduce latency and allow for incremental processing. The best thing about Structured Streaming is that it allows you to get value out rapidly and quickly of streaming systems with virtually no code changes. It also makes it easy to reason about because you can write your batch job to prototype it and then you can convert it to a streaming job. The way all this works is by incrementally processing that data.

Structured Streaming is a scalable and fault-tolerant stream processing engine built on the Spark SQL engine. You can express your streaming computation the same way you would express a batch computation on static data. The Spark SQL engine will take care of running it incrementally and continuously and updating the final result as streaming data continues to arrive. You can use the Dataset/DataFrame API in Scala, Java, Python or R to express streaming aggregations, event-time windows, stream-to-batch joins, etc. The computation is executed on the same optimized Spark SQL engine. Finally, the system

ensures end-to-end exactly-once fault-tolerance guarantees through checkpointing and Write-Ahead Logs. In short, Structured Streaming provides fast, scalable, fault-tolerant, end-to-end exactly-once stream processing without the user having to reason about streaming.

Internally, by default, Structured Streaming queries are processed using a micro-batch processing engine, which processes data streams as a series of small batch jobs thereby achieving end-to-end latencies as low as 100 milliseconds and exactly-once fault-tolerance guarantees. However, since Spark 2.3, we have introduced a new low-latency processing mode called Continuous Processing, which can achieve end-to-end latencies as low as 1 millisecond with at-least-once guarantees. Without changing the Dataset/DataFrame operations in your queries, you will be able to choose the mode based on your application requirements.

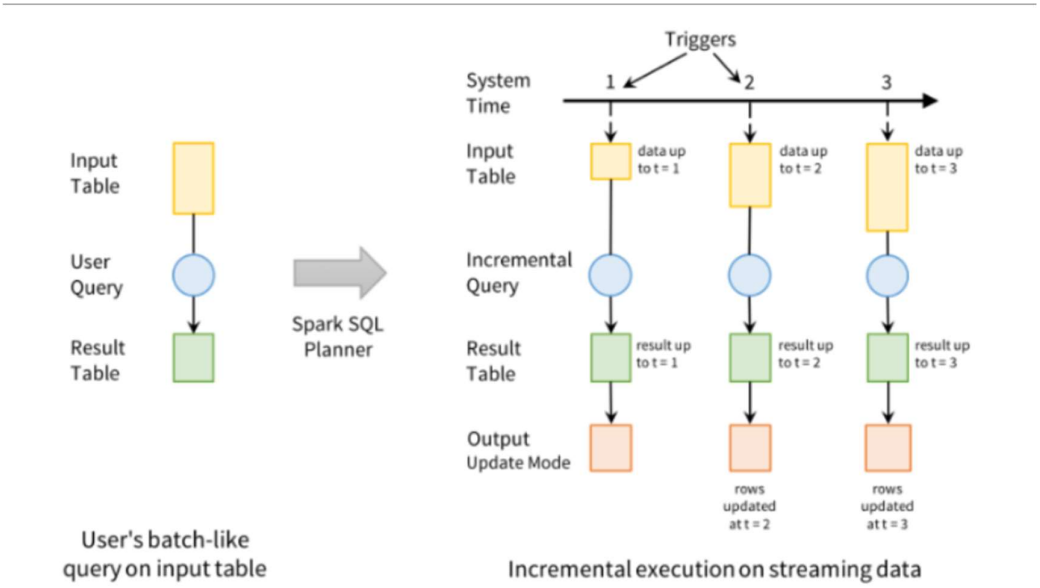


Fig: Structured Streaming Processing Model

PROPOSED SYSTEM ARCHITECTURE

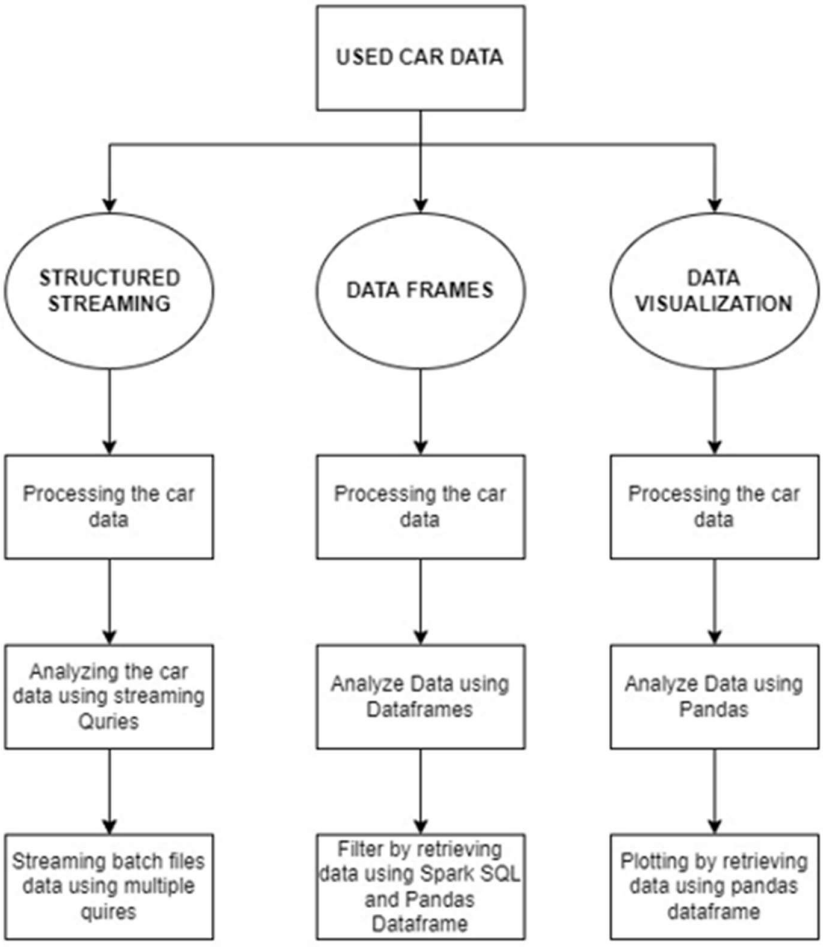


Fig: Flowchart describing our proposed system architecture

RESULTS

SPARK STREAMING:

SPLITTING THE DATA

The UsedCar.csv data set file size 9.4GB. As this Size is large, it is split into 61 batches and stored in a Folder S2 . Out of which Four of the batches are moved to another folder S1. A python program has been written to move the remaining batches in S2 to S1 in a time interval of 5 milli seconds.

STREAMING

Spark shell is started along with running the python file. Required attributes for Streaming are initialized in spark. files moving from Folder S2 to S1 can be observed. Required Queries are executed and batches of data being printed with top 20 rows will be observed.

Fig: Python code for moving files from folder S2 to S1

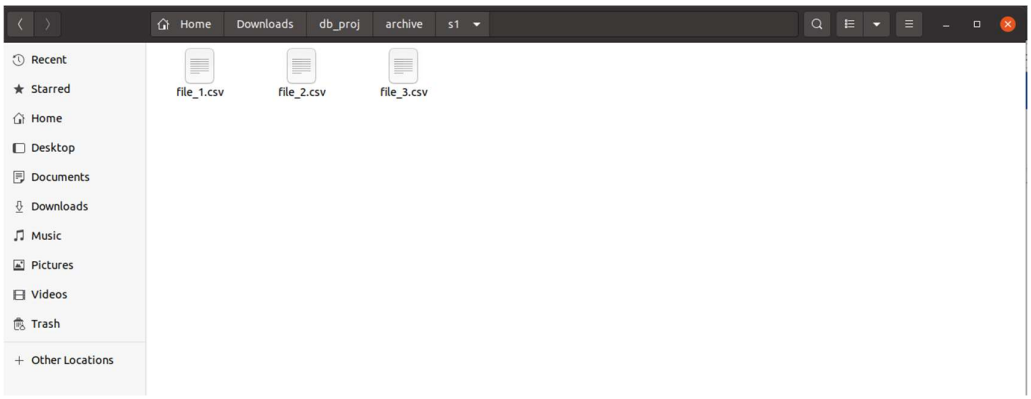


Fig: Folder S1

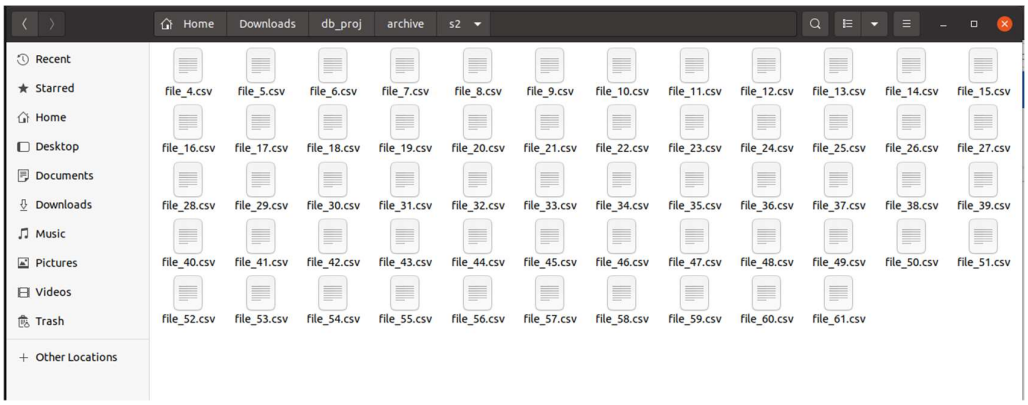


Fig: Folder S2

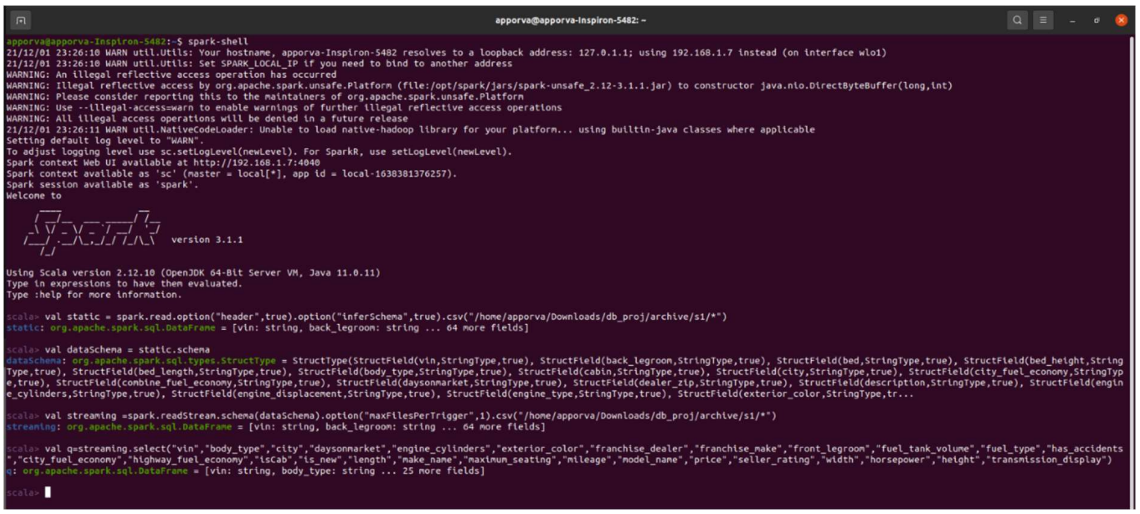


Fig: Initializing spark session and the required variables for streaming the dataset

PARALLELISM

Here the figures below display all the required components for executing Streaming. Fig (1) is a snippet of the components before streaming is commenced and Fig (2) is a snippet of the components taken during streaming.
It can be observed from Fig (I) that the CPU cores are not running at a high percentage whereas when Fig (II) is observed it is notable that all the 8 cores are being used in high percentage.
This shows that load has been equally distributed among all the 8 cores. From this it can be inferred that parallelism is observed.

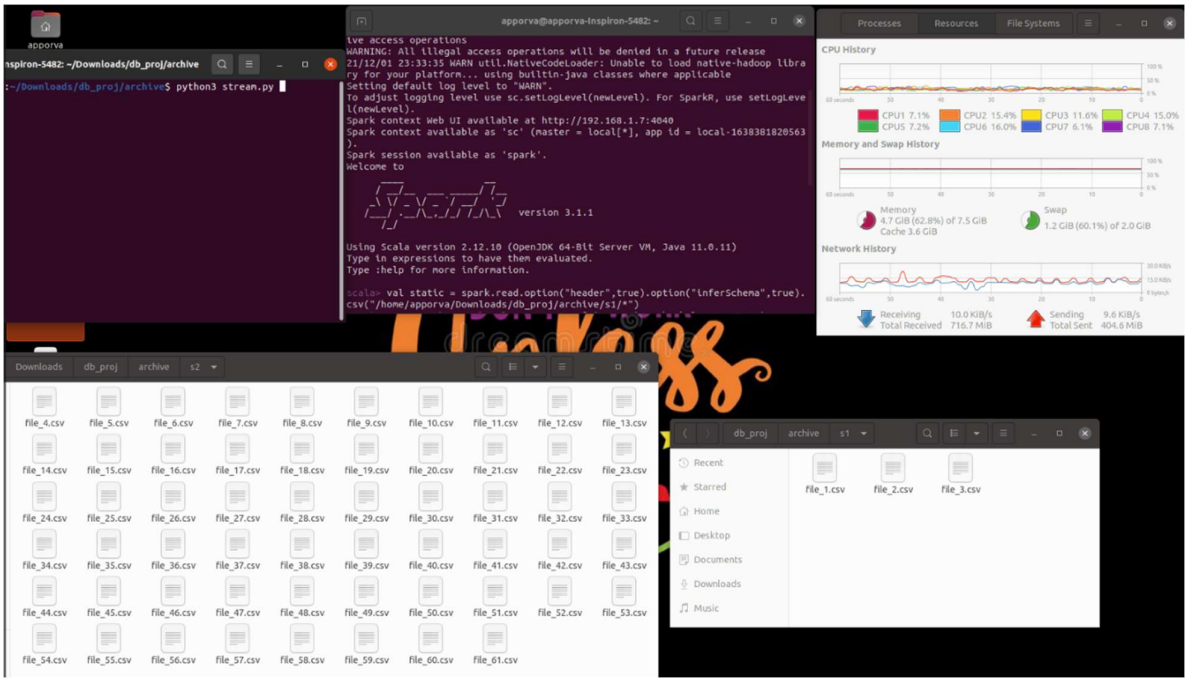


Fig: Snippet of the components before streaming is commenced

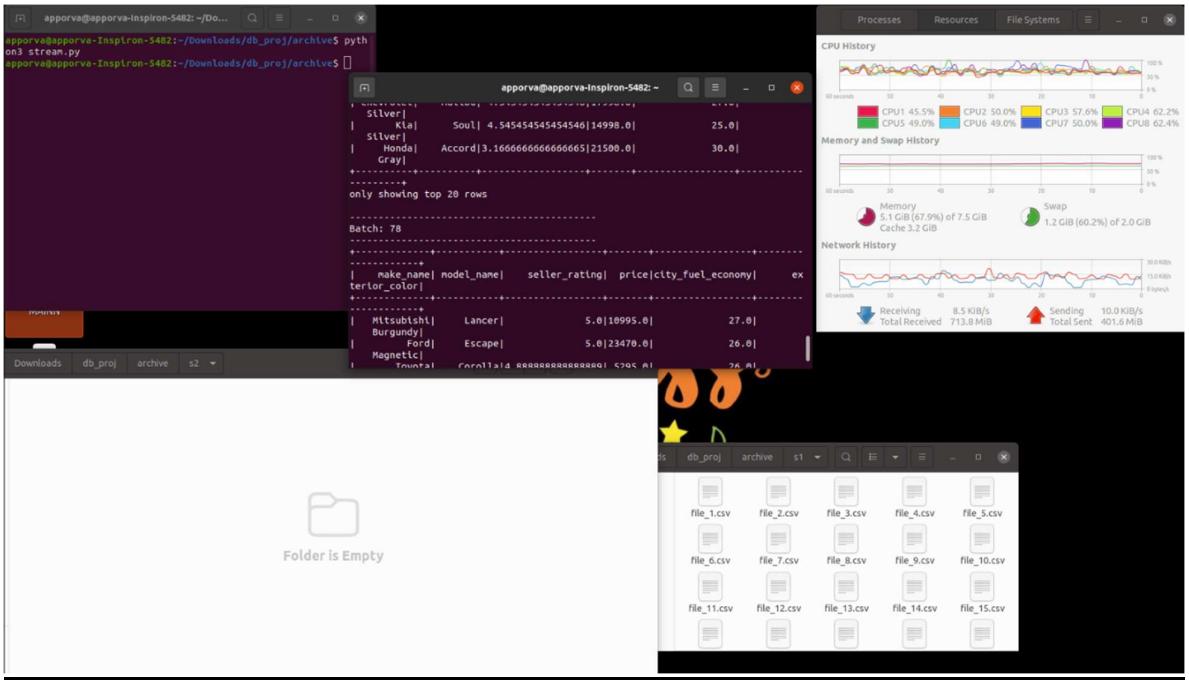


Fig: Snippet of the components taken during streaming spark session

QUERIES:

1) Query: Basic

This query is based on analyzing all the basic attributes that an average working person would consider for buying a second-hand car for daily use. The attributes that have been taken into consideration are price, fuel type and city fuel economy.

```
scala> val a1=q.filter($"price"<=30000).filter($"fuel_type"==="Gasoline").filter($"city_fuel_economy">20).select("make_name","model_name","seller_rating","price","city_fuel_economy","exterior_color")
a1: org.apache.spark.sql.DataFrame = [make_name: string, model_name: string ... 4 more fields]

scala> val r1=a1.writeStream.queryName("Basic").format("console").outputMode("append").start()
21/12/01 23:38:49 WARN streaming.StreamingQueryManager: Temporary checkpoint location created which is deleted normally when the query didn't fail: /tmp/temporary-893164f9-3b17-4665-bf76-2bf7ce3935af. If
 it's required to delete it under any circumstances, please set spark.sql.streaming.forceDeleteTempCheckpointLocation to true. Important to know deleting temp checkpoint folder is best effort.
r1: org.apache.spark.sql.streaming.StreamingQuery = org.apache.spark.sql.execution.streaming.StreamingQueryWrapper@2866f0d3

-------------------------------------------
Batch: 0
-------------------------------------------
+---------+----------+------------------+-------+----------------+--------------------+
|make_name|model_name|     seller_rating|  price|city_fuel_economy|      exterior_color|
+---------+----------+------------------+-------+----------------+--------------------+
|Chevrolet|    Malibu|3.447761194029851|14639.0|            27.0| Silver Ice Metallic|
|     Jeep|  Renegade|3.447761194029851|14849.0|            24.0|         Mojave Sand|
|     Jeep|  Renegade|3.447761194029851|12924.0|            24.0|               Black|
|Chevrolet|    Malibu|3.447761194029851|14359.0|            27.0| Silver Ice Metallic|
|   Subaru|   Outback|3.447761194029851|22159.0|            25.0| Crystal White Pearl|
|  Hyundai|    Tucson|3.447761194029851|15939.0|            21.0|          Molten Silver|
|Chevrolet|      Trax|3.447761194029851|15724.0|            25.0| Silver Ice Metallic|
|Chevrolet|     Cruze|3.447761194029851|14824.0|            28.0| Pepperdust Metallic|
|Chevrolet|    Malibu|3.447761194029851|15324.0|            27.0|  Cajun Red Tintcoat|
|Chevrolet|     Cruze|3.447761194029851|15439.0|            29.0|Mosaic Black Meta...|
|    Honda|     Civic|3.447761194029851|13623.0|            31.0|     Dyno Blue Pearl|
|      Kia|   Sorento|3.447761194029851|16969.0|            21.0|         Ebony Black|
|      BMW|  5 Series| 2.963636363636364|12495.0|            22.0|Azurite Black Met...|
|      BMW|  5 Series| 2.963636363636364|11195.0|            22.0|Azurite Black Met...|
|Chevrolet|     Cruze|3.447761194029851|14224.0|            30.0|   Tungsten Metallic|
|     Ford|Transit Connect| 4.666666666666667| 4999.0|            21.0|               white|
|      BMW|        X3| 2.963636363636364| 9995.0|            21.0|           Jet Black|
|    Mazda|      CX-5|3.447761194029851|15439.0|            24.0|Crystal White Pea...|
|   Nissan|Rogue Sport|3.447761194029851|16624.0|            24.0|       Glacier White|
|  Hyundai|    Sonata|3.447761194029851|14359.0|            25.0|  Quartz White Pearl|
+---------+----------+------------------+-------+----------------+--------------------+
only showing top 20 rows
```

```
-------------------------------------------
Batch: 60
-------------------------------------------
+----------+--------------------+------------------+-------+----------------+--------------------+
| make_name|          model_name|     seller_rating|  price|city_fuel_economy|      exterior_color|
+----------+--------------------+------------------+-------+----------------+--------------------+
|      Ford|              Escape|  4.510638297872341|27195.0|            26.0|Agate Black Metallic|
|      Ford|     Transit Connect|  4.510638297872341|29945.0|            24.0|        Frozen White|
|     Nissan|               Rogue| 3.914285714285714|16888.0|            25.0|         Gun Metallic|
|      Ford|     Transit Connect|  4.510638297872341|29945.0|            24.0|        Frozen White|
|      Ford|              Escape|  4.510638297872341|27227.0|            26.0|Agate Black Metallic|
|      Ford|              Fusion|3.714285714285714|15990.0|            23.0|                Gray|
|      Ford|     Transit Connect|  4.510638297872341|27590.0|            24.0|        Frozen White|
|      Ford|     Transit Connect|  4.510638297872341|27590.0|            24.0|        Frozen White|
|      Ford|     Transit Connect|  4.510638297872341|29945.0|            24.0|        Frozen White|
|      Ford|              Escape|  4.510638297872341|27797.0|            27.0|Star White Metall...|
|Volkswagen|                Jetta|             4.9375|17587.0|            30.0|             Platnum|
|     Honda|       Civic Hatchback|             4.9375|24587.0|            31.0|    Sonic Gray Pearl|
|      Ford|     Transit Connect|  4.510638297872341|27805.0|            24.0|        Frozen White|
|   Lincoln|                 MKC|3.1666666666666665|19914.0|            21.0|       Ingot Silver|
|     Nissan|              Altima|3.714285714285714|14990.0|            22.0|               Brown|
|       BMW|3 Series Gran Tur...| 3.914285714285714|22995.0|            22.0|Glacier Silver Me...|
|       GMC|             Terrain|3.1666666666666665|18995.0|            22.0|          Onyx Black|
|     Honda|               Civic|             4.9375|17587.0|            31.0| Crystal Black Pearl|
|   Hyundai|Sonata Hybrid Plu...|3.714285714285714|17990.0|            38.0|               White|
|    Toyota|                RAV4|             4.9375|22987.0|            22.0|       Magnetic Gray|
+----------+--------------------+------------------+-------+----------------+--------------------+
only showing top 20 rows
```

2) **Query: Comfort**

This query is designed for the customers who would buy cars with comfort as priority. Keeping Comfort in mind the attributes considered here are front leg room, back leg room and maximum seating.

```
scala> val a1=q.filter($"price"<=30000).filter($"fuel_type"==="Gasoline").filter($"city_fuel_economy">20).select("make_name","model_name","seller_rating","price","city_fuel_economy","exterior_color")
a1: org.apache.spark.sql.DataFrame = [make_name: string, model_name: string ... 4 more fields]

scala> val r1=a1.writeStream.queryName("Basic").format("console").outputMode("append").start()
21/12/01 23:38:49 WARN streaming.StreamingQueryManager: Temporary checkpoint location created which is deleted normally when the query didn't fail: /tmp/temporary-893164f9-3b17-4665-bf76-2bf7ce3935af. If
 it's required to delete it under any circumstances, please set spark.sql.streaming.forceDeleteTempCheckpointLocation to true. Important to know deleting temp checkpoint folder is best effort.
r1: org.apache.spark.sql.streaming.StreamingQuery = org.apache.spark.sql.execution.streaming.StreamingQueryWrapper@2866f0d3

-------------------------------------------
Batch: 0
-------------------------------------------
+---------+-------------+------------------+-------+------------------+--------------------+
|make_name|   model_name|     seller_rating|  price|city_fuel_economy|      exterior_color|
+---------+-------------+------------------+-------+------------------+--------------------+
|Chevrolet|      Malibu|3.447761194029851|14639.0|              27.0| Silver Ice Metallic|
|     Jeep|     Renegade|3.447761194029851|14849.0|              24.0|         Mojave Sand|
|     Jeep|     Renegade|3.447761194029851|12924.0|              24.0|               Black|
|Chevrolet|      Malibu|3.447761194029851|14359.0|              27.0| Silver Ice Metallic|
|   Subaru|      Outback|3.447761194029851|22159.0|              25.0| Crystal White Pearl|
|  Hyundai|       Tucson|3.447761194029851|15939.0|              21.0|         Molten Silver|
|Chevrolet|         Trax|3.447761194029851|15724.0|              25.0| Silver Ice Metallic|
|Chevrolet|        Cruze|3.447761194029851|14824.0|              28.0| Pepperdust Metallic|
|Chevrolet|      Malibu|3.447761194029851|15324.0|              27.0|  Cajun Red Tintcoat|
|Chevrolet|        Cruze|3.447761194029851|15439.0|              29.0|Mosaic Black Meta...|
|    Honda|        Civic|3.447761194029851|13623.0|              31.0|     Dyno Blue Pearl|
|      Kia|      Sorento|3.447761194029851|16969.0|              21.0|         Ebony Black|
|      BMW|     5 Series| 2.963636363636364|12495.0|              22.0|Azurite Black Met...|
|      BMW|     5 Series| 2.963636363636364|11195.0|              22.0|Azurite Black Met...|
|Chevrolet|        Cruze|3.447761194029851|14224.0|              30.0|   Tungsten Metallic|
|     Ford|Transit Connect| 4.666666666666667| 4999.0|              21.0|               White|
|      BMW|           X3| 2.963636363636364| 9995.0|              21.0|          Jet Black|
|    Mazda|         CX-5|3.447761194029851|15439.0|              24.0|Crystal White Pea...|
|   Nissan|    Rogue Sport|3.447761194029851|16624.0|              24.0|       Glacier White|
|  Hyundai|       Sonata|3.447761194029851|14359.0|              25.0|  Quartz White Pearl|
+---------+-------------+------------------+-------+------------------+--------------------+
only showing top 20 rows
```

```
-------------------------------------------
Batch: 60
-------------------------------------------
+---------+-------------+------------------+-------+------------------+--------------------+
|make_name|   model_name|     seller_rating|  price|city_fuel_economy|      exterior_color|
+---------+-------------+------------------+-------+------------------+--------------------+
|Chevrolet|      Malibu|3.447761194029851|14639.0|              27.0| Silver Ice Metallic|
|     Jeep|     Renegade|3.447761194029851|14849.0|              24.0|         Mojave Sand|
|     Jeep|     Renegade|3.447761194029851|12924.0|              24.0|               Black|
|Chevrolet|      Malibu|3.447761194029851|14359.0|              27.0| Silver Ice Metallic|
|   Subaru|      Outback|3.447761194029851|22159.0|              25.0| Crystal White Pearl|
|  Hyundai|       Tucson|3.447761194029851|15939.0|              21.0|         Molten Silver|
|Chevrolet|         Trax|3.447761194029851|15724.0|              25.0| Silver Ice Metallic|
|Chevrolet|        Cruze|3.447761194029851|14824.0|              28.0| Pepperdust Metallic|
|Chevrolet|      Malibu|3.447761194029851|15324.0|              27.0|  Cajun Red Tintcoat|
|Chevrolet|        Cruze|3.447761194029851|15439.0|              29.0|Mosaic Black Meta...|
|    Honda|        Civic|3.447761194029851|13623.0|              31.0|     Dyno Blue Pearl|
|      Kia|      Sorento|3.447761194029851|16969.0|              21.0|         Ebony Black|
|      BMW|     5 Series| 2.963636363636364|12495.0|              22.0|Azurite Black Met...|
|      BMW|     5 Series| 2.963636363636364|11195.0|              22.0|Azurite Black Met...|
|Chevrolet|        Cruze|3.447761194029851|14224.0|              30.0|   Tungsten Metallic|
|     Ford|Transit Connect| 4.666666666666667| 4999.0|              21.0|               White|
|      BMW|           X3| 2.963636363636364| 9995.0|              21.0|          Jet Black|
|    Mazda|         CX-5|3.447761194029851|15439.0|              24.0|Crystal White Pea...|
|   Nissan|    Rogue Sport|3.447761194029851|16624.0|              24.0|       Glacier White|
|  Hyundai|       Sonata|3.447761194029851|14359.0|              25.0|  Quartz White Pearl|
+---------+-------------+------------------+-------+------------------+--------------------+
only showing top 20 rows
```

3) Query: Traveler

This query filters out cars that are apt for Travelers who are always on the go. The attributes that are considered here are high way fuel economy, fuel type and fuel tank volume.

```
scala> val d1=q.filter($"highway_fuel_economy">15).filter($"fuel_type"==="Gasoline").filter($"fuel_tank_volume"==="26 gal").select("seller_rating","price","highway_fuel_economy","is_new")
d1: org.apache.spark.sql.DataFrame = [seller_rating: string, price: string ... 2 more fields]

scala> val r4=d1.writeStream.queryName("Traveller").format("console").outputMode("append").start()
21/12/01 23:48:39 WARN streaming.StreamingQueryManager: Temporary checkpoint location created which is deleted normally when the query didn't fail: /tmp/temporary-6fcaed80-8dd6-41f2-b208-e45d5f39279a. If
 it's required to delete it under any circumstances, please set spark.sql.streaming.forceDeleteTempCheckpointLocation to true. Important to know deleting temp checkpoint folder is best effort.
r4: org.apache.spark.sql.streaming.StreamingQuery = org.apache.spark.sql.execution.streaming.StreamingQueryWrapper@7eaca92e

scala> -------------------------------------------
Batch: 0
-------------------------------------------
+------------------+------+--------------------+------+
|     seller_rating|  price|highway_fuel_economy|is_new|
+------------------+------+--------------------+------+
|3.4477611940298507|34224.0|                22.0| False|
|3.4477611940298507|31841.0|                22.0| False|
|3.4477611940298507|43859.0|                21.0| False|
|3.4477611940298507|28924.0|                22.0| False|
|3.4477611940298507|32439.0|                22.0| False|
|3.4477611940298507|32439.0|                22.0| False|
|3.4477611940298507|32479.0|                22.0| False|
|3.4477611940298507|30924.0|                22.0| False|
|3.4477611940298507|29724.0|                22.0| False|
|3.4477611940298507|32924.0|                22.0| False|
|3.4477611940298507|34439.0|                22.0| False|
|3.4477611940298507|32439.0|                22.0| False|
|3.4477611940298507|33259.0|                22.0| False|
| 4.666666666666667| 5499.0|                18.0| False|
|3.4477611940298507|36871.0|                22.0| False|
|3.4477611940298507|32739.0|                22.0| False|
|3.4477611940298507|33439.0|                22.0| False|
|3.4477611940298507|34521.0|                22.0| False|
|3.4477611940298507|23439.0|                21.0| False|
|3.4477611940298507|33759.0|                22.0| False|
+------------------+------+--------------------+------+
only showing top 20 rows
```

```
-------------------------------------------
Batch: 60
-------------------------------------------
+------------------+------+--------------------+------+
|     seller_rating|  price|highway_fuel_economy|is_new|
+------------------+------+--------------------+------+
| 4.117647058823529|43588.0|                20.0| False|
|               4.5| 9990.0|                17.0| False|
|3.9545454545454546|41100.0|                26.0|  True|
|3.9545454545454546|61475.0|                23.0|  True|
|3.9545454545454546|65440.0|                22.0|  True|
|3.9545454545454546|51035.0|                26.0|  True|
|3.9545454545454546|51125.0|                23.0|  True|
|3.9545454545454546|53530.0|                22.0|  True|
| 4.538461538461538|64900.0|                23.0|  True|
| 4.538461538461538|84115.0|                21.0|  True|
| 4.538461538461538|84115.0|                21.0|  True|
| 4.538461538461538|84115.0|                21.0|  True|
| 4.615384615384615|56965.0|                21.0|  True|
|             3.625|29494.0|                19.0| False|
|              4.65|42510.0|                26.0|  True|
|              4.65|66775.0|                22.0|  True|
|              3.75|64485.0|                21.0|  True|
|              3.75|64565.0|                21.0|  True|
|              3.75|65115.0|                21.0|  True|
| 4.363636363636363|57897.0|                21.0| False|
+------------------+------+--------------------+------+
only showing top 20 rows
```

4)Query: Cab Driver

This is created for helping cab driver find suitable cars for their work. The attributes considered here are city fuel economy, fuel type and fuel tank volume.

```
scala> val e1=q.filter($"city_fuel_economy">10).filter($"fuel_type"==="Gasoline").filter($"fuel_tank_volume"==="26 gal").filter($"is_new"==="False").filter($"has_accidents"==="False").select("seller_rati
ng","price","city_fuel_economy")
e1: org.apache.spark.sql.DataFrame = [seller_rating: string, price: string ... 1 more field]

scala> val r5=e1.writeStream.queryName("Cab_driver").format("console").outputMode("append").start()
21/12/01 23:54:32 WARN streaming.StreamingQueryManager: Temporary checkpoint location created which is deleted normally when the query didn't fail: /tmp/temporary-bebd4584-e31c-42ac-8809-4115f939f01c. If
 it's required to delete it under any circumstances, please set spark.sql.streaming.forceDeleteTempCheckpointLocation to true. Important to know deleting temp checkpoint folder is best effort.
r5: org.apache.spark.sql.streaming.StreamingQuery = org.apache.spark.sql.execution.streaming.StreamingQueryWrapper@ff9be70

-------------------------------------------
Batch: 0
-------------------------------------------
+------------------+-------+------------------+
|     seller_rating|  price|city_fuel_economy|
+------------------+-------+------------------+
|3.4477611940298507|34224.0|              16.0|
|3.4477611940298507|31841.0|              16.0|
|3.4477611940298507|43859.0|              15.0|
|3.4477611940298507|28924.0|              16.0|
|3.4477611940298507|32439.0|              16.0|
|3.4477611940298507|32439.0|              16.0|
|3.4477611940298507|32479.0|              16.0|
|3.4477611940298507|29724.0|              16.0|
|3.4477611940298507|32924.0|              16.0|
|3.4477611940298507|34439.0|              16.0|
|3.4477611940298507|32439.0|              16.0|
| 4.666666666666667| 5499.0|              13.0|
|3.4477611940298507|36871.0|              16.0|
|3.4477611940298507|32739.0|              16.0|
|3.4477611940298507|33439.0|              16.0|
|3.4477611940298507|34521.0|              16.0|
|3.4477611940298507|23439.0|              15.0|
|3.4477611940298507|33759.0|              16.0|
|3.4477611940298507|34539.0|              16.0|
|3.4477611940298507|34569.0|              15.0|
+------------------+-------+------------------+
only showing top 20 rows
```

```
-------------------------------------------
Batch: 60
-------------------------------------------
+------------------+-------+------------------+
|     seller_rating|  price|city_fuel_economy|
+------------------+-------+------------------+
| 4.1176470588235294|43588.0|              14.0|
|               4.5| 9990.0|              14.0|
|             3.625|29494.0|              14.0|
| 4.3636363636363635|57097.0|              15.0|
|              3.75|59995.0|              15.0|
|              3.75|56995.0|              15.0|
|              3.75|56995.0|              15.0|
| 4.3636363636363635|57097.0|              15.0|
|3.6666666666666665|27877.0|              14.0|
| 4.235294117647059|39978.0|              13.0|
| 4.171428571428572|53995.0|              15.0|
| 4.215909090909091|44998.0|              15.0|
| 4.215909090909091|36998.0|              14.0|
| 4.215909090909091|50998.0|              14.0|
| 4.215909090909091|32998.0|              13.0|
| 4.583333333333333|27995.0|              14.0|
|              4.25| 9995.0|              12.0|
|               4.0|30995.0|              16.0|
| 4.333333333333333|31998.0|              13.0|
| 4.333333333333333|46998.0|              15.0|
+------------------+-------+------------------+
only showing top 20 rows
```

5)Query: Packers and Movers

This is for Packing agencies who want to get big Vehicles. The attribute that is considered here are length and city fuel economy.

Importing Dataset

```python
vehicle_listings = spark.read.format("csv").option("header", "true").load("/content/drive/MyDrive/unzipped_dbms/used_cars_data.csv")
type(vehicle_listings)
```

```
pyspark.sql.dataframe.DataFrame
```

```python
vehicle_listings.show()
```

vin	back_legroom	bed	bed_height	bed_length	body_type	cabin	city	city_fuel_economy	combine_fuel_economy	daysonmarket	dealer_zip	description	engine_cyl
ZACNJABB5KPJ92081	35.1 in	null	null	null	SUV / Crossover	null	Bayamon	null	null	522	00960	[!@@Additional In...	
SALCJ2FX1LH858117	38.1 in	null	null	null	SUV / Crossover	null	San Juan	null	null	207	00922	[!@@Additional In...	
JF1VA2M67G9829723	35.4 in	null	null	null	Sedan	null	Guaynabo	17.0	null	1233	00969	null	
SALRR2RV0L2433391	37.6 in	null	null	null	SUV / Crossover	null	San Juan	null	null	196	00922	[!@@Additional In...	
SALCJ2FXXLH862327	38.1 in	null	null	null	SUV / Crossover	null	San Juan	null	null	137	00922	[!@@Additional In...	
SALYK2EX1LA261711	37.1 in	null	null	null	SUV / Crossover	null	San Juan	null	null	242	00922	[!@@Additional In...	
3MZBPABL6KM107908	35.1 in	null	null	null	Sedan	null	Bayamon	null	null	447	00960	[!@@Additional In...	
SALYK2EX5LA275434	37.1 in	null	null	null	SUV / Crossover	null	San Juan	null	null	70	00922	[!@@Additional In...	
SALCJ2FX6LH858128	38.1 in	null	null	null	SUV / Crossover	null	San Juan	null	null	196	00922	[!@@Additional In...	
SALZL2GX4LH007593	33.8 in	null	null	null	SUV / Crossover	null	San Juan	null	null	510	00922	[!@@Additional In...	
ZARBAAC41FM129303	--	null	null	null	Coupe	null	Guaynabo	null	null	1252	00969	[!@@Additional In...	
SALZJ2FX0LH081763	33.8 in	null	null	null	SUV / Crossover	null	San Juan	null	null	179	00922	[!@@Additional In...	
WBA8B7C53GK368522	35.1 in	null	null	null	Sedan	null	Guaynabo	22.0	null	1233	00969	[!@@Additional In...	
SALYK2EX8LA268316	37.1 in	null	null	null	SUV / Crossover	null	San Juan	null	null	201	00922	[!@@Additional In...	
3MZBPABL1KM108237	35.1 in	null	null	null	Sedan	null	Bayamon	null	null	447	00960	[!@@Additional In...	
3MZBPABL4KM107969	35.1 in	null	null	null	Sedan	null	Bayamon	null	null	447	00960	[!@@Additional In...	
SALCP2FX9LH857747	38.1 in	null	null	null	SUV / Crossover	null	San Juan	null	null	187	00922	[!@@Additional In...	
SALYK2EX8LA284533	37.1 in	null	null	null	SUV / Crossover	null	San Juan	null	null	20	00922	[!@@Additional In...	
SALYK2EX0LA284669	37.1 in	null	null	null	SUV / Crossover	null	San Juan	null	null	13	00922	[!@@Additional In...	
SALCP2FX8LH854709	38.1 in	null	null	null	SUV / Crossover	null	San Juan	null	null	230	00922	[!@@Additional In...	

only showing top 20 rows

Selecting required columns and filling null values

```python
print(vehicle_listings.columns)
```

```
['vin', 'back_legroom', 'bed', 'bed_height', 'bed_length', 'body_type', 'cabin', 'city', 'city_fuel_economy', 'combine_fuel_economy', 'daysonmarket', 'dealer_zip', 'description', 'engi
```

```python
print(vehicle_listings.count(),len(vehicle_listings.columns))
```

```
3000507 66
```

```python
vehicle_listings_clean=vehicle_listings.drop_duplicates(['vin', 'back_legroom', 'bed', 'bed_height', 'bed_length', 'body_type', 'cabin', 'city', 'city_fuel_economy', 'combine_fuel_ecc
```

```python
df=vehicle_listings_clean.select("vin","body_type","city","daysonmarket","engine_cylinders","exterior_color","franchise_dealer","franchise_make",
"front_legroom","fuel_tank_volume","fuel_type","has_accidents","city_fuel_economy","highway_fuel_economy","isCab","is_new","length","make_name",
"maximum_seating","mileage","model_name","price","seller_rating","width","horsepower","height","transmission_display")
```

```python
cars=df.na.fill({'body_type':'Sedan','city':'all','daysonmarket':220,'engine_cylinders':'unknown','exterior_color':'unknown','franchise_dealer':False,'franchise_make':'unknown','front
```

```python
cars.show(5)
```

vin	body_type	city	daysonmarket	engine_cylinders	exterior_color	franchise_dealer	franchise_make	front_legroom	fuel_tank_volume
3C4NJDDB9KT752727	SUV / Crossover	Bohemia	4	Premium Cloth/Lea...	GPS Antenna Input	4-Wheel Disc Brakes	6 Speakers	Air Conditioning	Electronic Stabil...
2GNAXUEV1L6249535	SUV / Crossover	Bay Shore	70	Front Bucket Seats	Driver Convenienc...	Radio: Chevrolet ...	Remote Start	8-Way Power Drive...	Front Passenger 4...
WBAJB1C52KB375534	Sedan	Great Neck	50	I4	White	False	unknown	41.4 in	12.1 gal
5UXXW3C51F0M86752	SUV / Crossover	North Plainfield	26	I4	White	False	unknown	40.4 in	17.7 gal
5UXWX9C56H0T00993	SUV / Crossover	Detroit	34	I4	Orange CHESTNUT	False	unknown	39.9 in	17.7 gal

only showing top 5 rows

Query1

```
basic=["price","make_name","model_name","city","seller_rating"]
```

```
q1=cars.where("price<=100000 and city='Howell'").select(basic).orderBy(desc("seller_rating")).show(10)
```

```
+-------+---------+------------------+------+-------------+
|  price|make_name|        model_name|  city|seller_rating|
+-------+---------+------------------+------+-------------+
|13995.0|     Ford|            Fusion|Howell|          5.0|
|54900.0|     Ford|  F-350 Super Duty|Howell|          5.0|
|25900.0| Chrysler|          Pacifica|Howell|          5.0|
|20900.0|     Jeep|Wrangler Unlimited|Howell|          5.0|
|26900.0|     Ford|          Explorer|Howell|          5.0|
|69900.0|    Dodge|           Charger|Howell|          5.0|
|15900.0|   Nissan|            Maxima|Howell|          5.0|
|20900.0|      GMC|           Terrain|Howell|          5.0|
|20988.0|     Ford|     Fusion Hybrid|Howell|          5.0|
|30900.0|     Jeep|Wrangler Unlimited|Howell|          5.0|
+-------+---------+------------------+------+-------------+
only showing top 10 rows
```

Query2

```
q2=cars.where("make_name='BMW'or make_name='Ford'").select("make_name","model_name","body_type","is_new","price","seller_rating").show(20)
```

```
+---------+---------------+---------------+------+-------+------------------+
|make_name|     model_name|      body_type|is_new|  price|     seller_rating|
+---------+---------------+---------------+------+-------+------------------+
|      BMW|       5 Series|          Sedan| False|38995.0|3.3461538461538463|
|      BMW|             X4|SUV / Crossover| False|23900.0|4.4186046511627906|
|      BMW|             X3|SUV / Crossover| False|24995.0|             4.125|
|     Ford|         Fusion|          Sedan| False|13995.0|               5.0|
|     Ford|F-350 Super Duty|   Pickup Truck| False|54900.0|               5.0|
|     Ford|         Fusion|          Sedan|  True|25295.0|               4.5|
|      BMW|       3 Series|          Sedan| False|44250.0| 4.161290322580645|
|      BMW|       3 Series|          Sedan| False|25795.0| 4.161290322580645|
|     Ford| Focus Electric|      Hatchback| False|12995.0| 4.237333333333333|
|     Ford|       Explorer|SUV / Crossover| False|26990.0| 4.717948717948718|
|     Ford|         Fusion|          Sedan|  True|24859.0| 4.769230769230769|
|      BMW|             i8|          Coupe| False|63995.0| 4.571428571428571|
|     Ford|         Escape|SUV / Crossover| False| 3399.0|3.6470588235294117|
|     Ford|          F-150|   Pickup Truck|  True|49527.0| 4.769230769230769|
|      BMW|       5 Series|          Sedan|  True|65550.0|3.9565217391304346|
|     Ford|          F-150|   Pickup Truck|  True|47314.0| 4.769230769230769|
|      BMW|             X3|SUV / Crossover|  True|48535.0| 4.225806451612903|
|      BMW|             X3|SUV / Crossover|  True|50535.0| 4.225806451612903|
|     Ford|         Fusion|          Sedan|  True|23476.0| 4.571428571428571|
|     Ford|         Fusion|          Sedan|  True|27768.0| 3.933333333333333|
+---------+---------------+---------------+------+-------+------------------+
only showing top 20 rows
```

Query3:

```
q3=cars.select("body_type").distinct().show(5)
```

```
+-------------+
|    body_type|
+-------------+
|    Hatchback|
|  Convertible|
|        Sedan|
| Pickup Truck|
|        Wagon|
+-------------+
only showing top 5 rows
```

```
1 q3=cars.filter("make_name='Ford'").select("model_name").distinct().show(10)
```

```
+--------------------+
|          model_name|
+--------------------+
|            E-Series|
|     Transit Connect|
|                Flex|
|               Focus|
|       Fusion Energi|
|               F-150|
|F-350 Super Duty ...|
|              Fusion|
|Mustang Shelby GT350|
|     Explorer Hybrid|
+--------------------+
only showing top 10 rows
```

Query4

```
q4=cars.filter("transmission_display='Automatic' and seller_rating=5.0")
.select("make_name","model_name","body_type","transmission_display","horsepower","price","seller_rating").sort(desc("seller_rating")).show()
```

```
+------------+-------------+--------------+--------------------+----------+-------+-------------+
|   make_name|   model_name|     body_type|transmission_display|horsepower|  price|seller_rating|
+------------+-------------+--------------+--------------------+----------+-------+-------------+
|        Jeep|      Compass|SUV / Crossover|           Automatic|     180.0|22900.0|          5.0|
|       Volvo|         XC60|SUV / Crossover|           Automatic|     316.0|53210.0|          5.0|
|        Ford|       Escape|SUV / Crossover|           Automatic|     180.0|29934.0|          5.0|
|       Volvo|         XC60|SUV / Crossover|           Automatic|     250.0|55310.0|          5.0|
|    INFINITI|         QX60|SUV / Crossover|           Automatic|     295.0|54845.0|          5.0|
|    INFINITI|         QX50|SUV / Crossover|           Automatic|     325.0|26995.0|          5.0|
|        Jeep|      Compass|SUV / Crossover|           Automatic|     180.0|21288.0|          5.0|
|       Volvo|          V90|          Wagon|           Automatic|     250.0|55740.0|          5.0|
|        Ford|       Fusion|          Sedan|           Automatic|     175.0|25181.0|          5.0|
|       Volvo|         XC40|SUV / Crossover|           Automatic|     248.0|43035.0|          5.0|
|Mercedes-Benz|      C-Class|          Sedan|           Automatic|     241.0|22488.0|          5.0|
|        Audi|           Q7|SUV / Crossover|           Automatic|     248.0|49988.0|          5.0|
|    INFINITI|          Q50|          Sedan|           Automatic|     208.0|21988.0|          5.0|
|   Chevrolet|Silverado 1500|  Pickup Truck|           Automatic|     420.0|55715.0|          5.0|
|        Ford|       Fusion|          Sedan|           Automatic|     181.0|31002.0|          5.0|
|       Volvo|         XC60|SUV / Crossover|           Automatic|     250.0|49710.0|          5.0|
|         BMW|           X5|SUV / Crossover|           Automatic|     300.0|26995.0|          5.0|
|       Volvo|         XC60|SUV / Crossover|           Automatic|     316.0|53510.0|          5.0|
|        Ford|   Expedition|SUV / Crossover|           Automatic|     375.0|64425.0|          5.0|
|    INFINITI|          Q60|         Coupe|           Automatic|     300.0|34888.0|          5.0|
+------------+-------------+--------------+--------------------+----------+-------+-------------+
only showing top 20 rows
```

Query5

```
q5=cars.filter("make_name='BMW' and model_name='X3'").select(basic).orderBy(desc("seller_rating")).show(10)
```

```
+-------+---------+----------+-----------+-------------+
|  price|make_name|model_name|       city|seller_rating|
+-------+---------+----------+-----------+-------------+
|23900.0|      BMW|        X3|   Brockton|          5.0|
|52745.0|      BMW|        X3| South Bend|          5.0|
|11900.0|      BMW|        X3|  Salisbury|          5.0|
|12900.0|      BMW|        X3|New Windsor|          5.0|
|11900.0|      BMW|        X3|Londonderry|          5.0|
|25900.0|      BMW|        X3|   Hooksett|          5.0|
|50235.0|      BMW|        X3| South Bend|          5.0|
|11995.0|      BMW|        X3| Bloomfield|          5.0|
|14500.0|      BMW|        X3|    Methuen|          5.0|
| 6950.0|      BMW|        X3|   Coventry|          5.0|
+-------+---------+----------+-----------+-------------+
only showing top 10 rows
```

Query6

```
q6=cars.filter("make_name='Ford' and model_name='Fusion'").select
("price","make_name","model_name","city","seller_rating","daysonmarket").orderBy(asc("daysonmarket")).show(10)
```

```
+-------+---------+----------+----------------+------------------+------------+
|  price|make_name|model_name|            city|     seller_rating|daysonmarket|
+-------+---------+----------+----------------+------------------+------------+
|30935.0|     Ford|    Fusion|       Amsterdam|              3.75|           0|
|17995.0|     Ford|    Fusion|   Wall Township| 4.666666666666667|           0|
|22999.0|     Ford|    Fusion|      Portsmouth| 4.428571428571429|           0|
| 6900.0|     Ford|    Fusion|North Chelmsford|               5.0|           0|
|12999.0|     Ford|    Fusion| North Attleboro|                 2|           0|
|16990.0|     Ford|    Fusion|        Freeport|2.8771929824561404|           0|
|11795.0|     Ford|    Fusion|           Alton|               4.6|           0|
|12650.0|     Ford|    Fusion|             Bow| 4.611111111111111|           0|
| 7495.0|     Ford|    Fusion|         Sheldon| 4.666666666666667|           0|
|27669.0|     Ford|    Fusion|     Fowlerville|               3.9|           0|
+-------+---------+----------+----------------+------------------+------------+
only showing top 10 rows
```

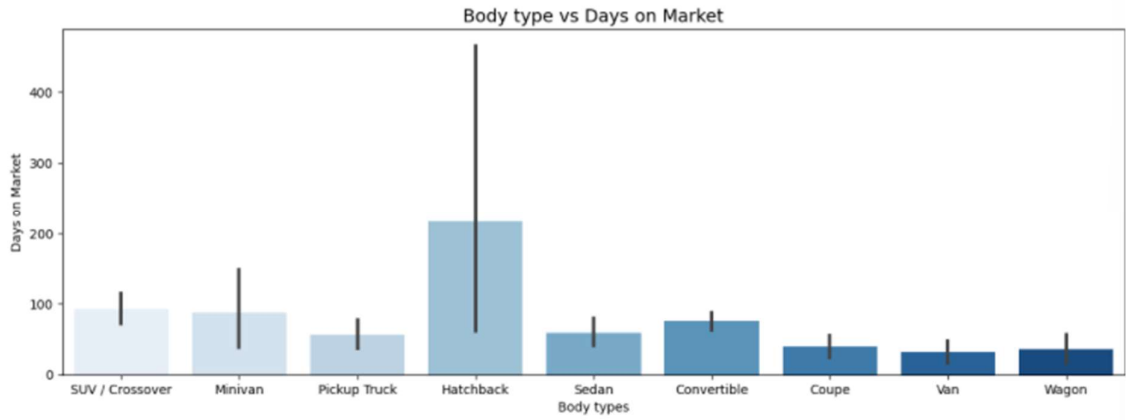
Data Visualization

```
df_heat=cars.sample(False,0.0001,63).toPandas()
```

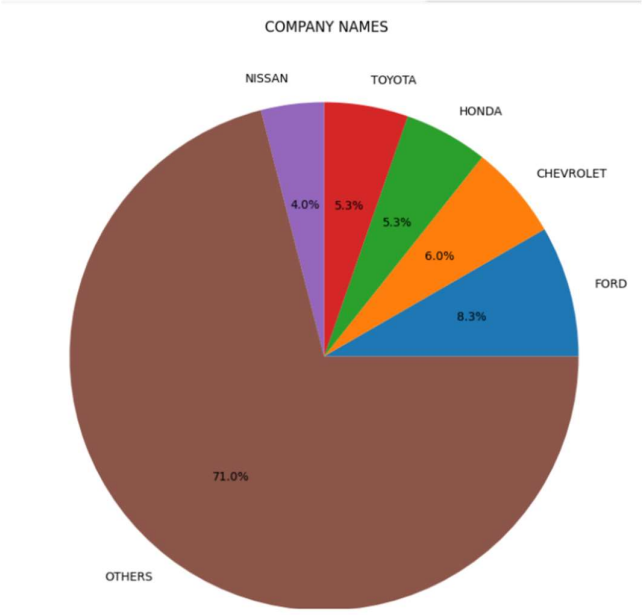
```
df_heat.head(2)
```

	vin	body_type	city	daysonmarket	engine_cylinders	exterior_color	franchise_dealer	franchise_make	front_legroom	fuel_tank_volume	fuel_type	has_accidents	c
0	1FM5K8D83JGA45727	SUV / Crossover	Little Rock	16	V6	Ingot Silver Metallic	True	Nissan	42.9 in	18.6 gal	Gasoline	True	
1	JTEBU5JR3J5583039	SUV / Crossover	Show Low	1	V6	Nautical Blue Metallic	True	Chevrolet	41.7 in	23 gal	Gasoline	False	

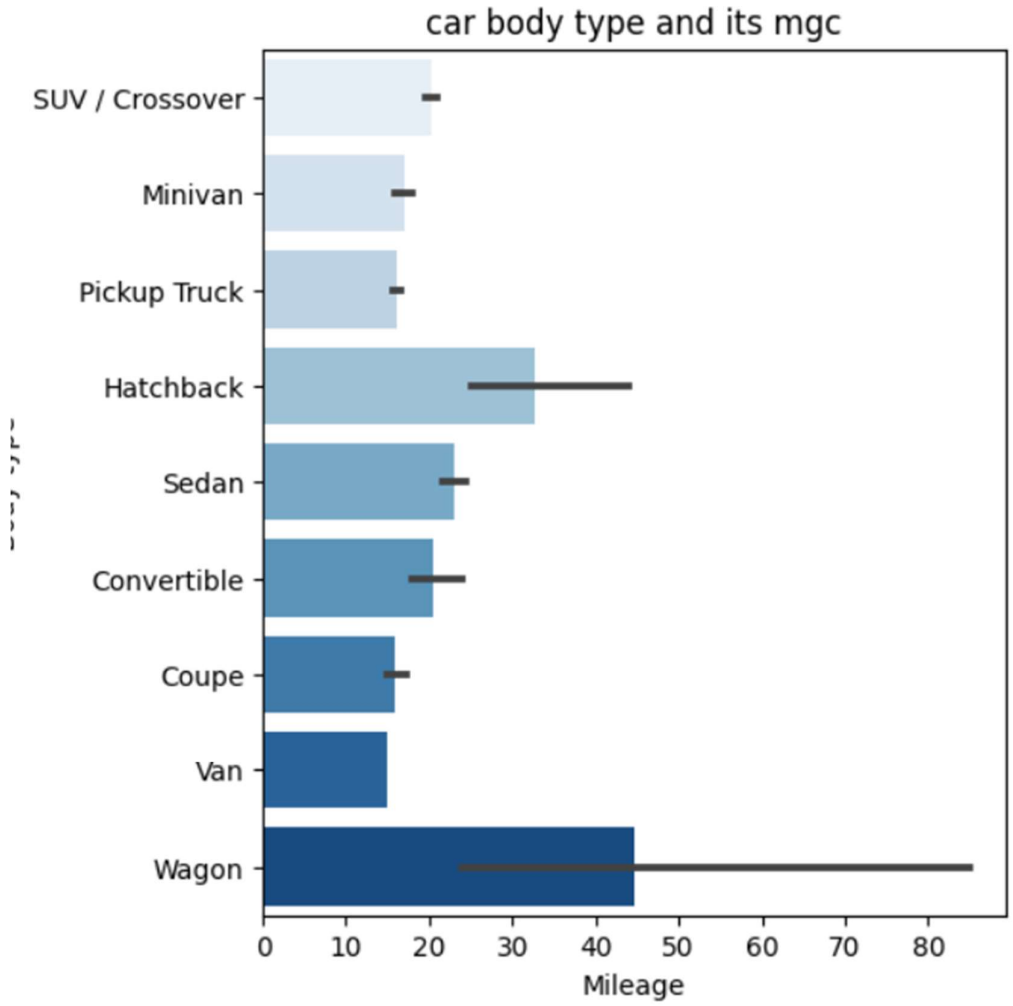
Bar plot for car model and the number of days it's available on market



Pie chart for car company names and their available car count



Bar plot for car body type and it’s milage given in city.



CONCLUSION

Opting for used cars these days has become a more wise option keeping in mind maintenance, the ever rising prices and their life span. This project has focused on filtering out cars for different kinds of people according to their uses and requirements. Various aspects have been considered to accomplish maximum satisaction of the buyers. Enquiries for both personal and professional vehicle needs have been taken care off. Using Structured Streaming, Sql Dataframes and Visualizations satisfactory results were obtained.

REFERENCES

1. *https://www.researchgate.net/publication/319306871_Predicting_the_Price_of_Used_Cars_using_Machine_Learning_Techniques*
2. *https://www.irjet.net/archives/V8/i4/IRJET-V8I4278.pdf*
3. *https://www.temjournal.com/content/81/TEMJournalFebruary2019_113_118.pdf*
4. *http://ripublication.com/irph/ijict_spl/ijictv4n7spl_17.pdf*

ANNEXURE

Structured Streaming:

spark-shell

```
val static =
spark.read.option("header",true).option("inferSchema",true).csv("/home/apporva/Downloads/db_proj/archive/s1/*")

val dataSchema = static.schema

val streaming
=spark.readStream.schema(dataSchema).option("maxFilesPerTrigger",1).csv("/home/apporva/Downloads/db_proj/archive/s1/*")

val
q=streaming.select("vin","body_type","city","daysonmarket","engine_cylinders","exterior_color","franchise_dealer","franchise_make","front_legroom","fuel_tank_volume","fuel_type","has_accidents","city_fuel_economy","highway_fuel_economy","isCab","is_new","length","make_name","maximum_seating","mileage","model_name","price","seller_rating","width","horsepower","height","transmission_display")



#Q1

val
a1=q.filter($"price"<=30000).filter($"fuel_type"==="Gasoline").filter($"city_fuel_economy">20).select("make_name","model_name","seller_rating","price","city_fuel_economy","exterior_color")

val
r1=a1.writeStream.queryName("Basic").format("console").outputMode("append").start()
```

#Q2

```
val c1=q.filter($"maximum_seating"==="7
seats").filter($"back_legroom"==="38.3
in").select("body_type","seller_rating","price")

val
r3=c1.writeStream.queryName("Comfort").format("console").outputMode("app
end").start()
```

#Q3

```
val
d1=q.filter($"highway_fuel_economy">15).filter($"fuel_type"==="Gasoline").f
ilter($"fuel_tank_volume"==="26
gal").select("seller_rating","price","highway_fuel_economy","is_new")

val
r4=d1.writeStream.queryName("Traveller").format("console").outputMode("ap
pend").start()
```

#Q4

```
val
e1=q.filter($"city_fuel_economy">10).filter($"fuel_type"==="Gasoline").filter(
$"fuel_tank_volume"==="26
gal").filter($"is_new"==="False").filter($"has_accidents"==="False").select("se
ller_rating","price","city_fuel_economy")

val
r5=e1.writeStream.queryName("Cab_driver").format("console").outputMode("a
ppend").start()
```

#Q5

```
val bq=q.filter($"length"==="231.9
in").filter($"city_fuel_economy">10).select("make_name","seller_rating","price
","has_accidents")

val rq=bq.writeStream.queryName("Packers and
Movers").format("console").outputMode("append").start()
```

Pyspark DataFrame Queries:

```
import traceback
import findspark
import os
import sys
from pyspark.sql import SparkSession
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.pylab as pylab
import seaborn as sns
import subprocess
from pyspark.sql.functions import *
from functools import reduce
from pyspark.ml.feature import StringIndexer
from pyspark.sql.types import DoubleType
from pyspark.ml import Pipeline
spark = SparkSession \
    .builder \
    .master("local[*]") \
    .appName("PySpark Craigslist") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()

sc = spark.sparkContext

vehicle_listings = spark.read.format("csv").option("header", "true").lo
ad("/content/drive/MyDrive/unzipped_dbms/used_cars_data.csv")
type(vehicle_listings)

vehicle_listings.show()

print(vehicle_listings.columns)

print(vehicle_listings.count(),len(vehicle_listings.columns))

vehicle_listings_clean=vehicle_listings.drop_duplicates(['vin', 'back_l
egroom', 'bed', 'bed_height', 'bed_length', 'body_type', 'cabin', 'city
', 'city_fuel_economy', 'combine_fuel_economy', 'daysonmarket', 'dealer
_zip', 'description', 'engine_cylinders', 'engine_displacement', 'engin
e_type', 'exterior_color', 'fleet', 'frame_damaged', 'franchise_dealer'
, 'franchise_make', 'front_legroom', 'fuel_tank_volume', 'fuel_type', '
has_accidents', 'height', 'highway_fuel_economy', 'horsepower', 'interi
or_color', 'isCab', 'is_certified', 'is_cpo', 'is_new', 'is_oemcpo', 'l
atitude', 'length', 'listed_date', 'listing_color', 'listing_id', 'long
```

```
itude', 'main_picture_url', 'major_options', 'make_name', 'maximum_seat
ing', 'mileage', 'model_name', 'owner_count', 'power', 'price', 'salvag
e', 'savings_amount', 'seller_rating', 'sp_id', 'sp_name', 'theft_title
', 'torque', 'transmission', 'transmission_display', 'trimId', 'trim_na
me', 'vehicle_damage_category', 'wheel_system', 'wheel_system_display',
 'wheelbase', 'width', 'year'])

cars=df.na.fill({'body_type':'Sedan','city':'all','daysonmarket':220,'e
ngine_cylinders':'unknown','exterior_color':'unknown','franchise_dealer
':False,'franchise_make':'unknown','front_legroom':39,'fuel_tank_volume
': 13.2,'fuel_type':'unknown','has_accidents':False,'city_fuel_economy'
:15,'highway_fuel_economy':12,'isCab':False,'is_new':False,'length':230
,'make_name':'unknown','maximum_seating':5,'mileage':0,'model_name':'un
known','price':'negotiable','seller_rating':2,'width':73.2,'horsepower'
:145.0,'height':61})

cars.show(5)

basic=["price","make_name","model_name","city","seller_rating"]

q1=cars.where("price<=100000 and city='Howell'").select(basic).orderBy(
desc("seller_rating")).show(10)

q2=cars.where("make_name='BMW'or make_name='Ford'").select("make_name",
"model_name","body_type","is_new","price","seller_rating").show(20)

q3=cars.select("body_type").distinct().show(5)
q3=cars.filter("make_name='Ford'").select("model_name").distinct().show(10)

q4=cars.filter("transmission_display='Automatic' and seller_rating=5.0"
).select("make_name","model_name","body_type","transmission_display","h
orsepower","price","seller_rating").sort(desc("seller_rating")).show()

q5=cars.filter("make_name='BMW' and model_name='X3'").select(basic).orderBy(desc("seller_rati
ng")).show(10)

q6=cars.filter("make_name='Ford' and model_name='Fusion'").select("price","make_name","mode
l_name","city","seller_rating","daysonmarket").orderBy(asc("daysonmarket")).show(10)
```

Pyspark Data Visualization code:

```
df_heat=cars.sample(False,0.0001,63).toPandas()
df_heat.head(2)
df_heat['daysonmarket']= pd.to_numeric(df_heat['daysonmarket'])
```

```
import seaborn as sns
plt.figure(figsize=(15,5), dpi=100)
sns.barplot(x = 'body_type',y = 'daysonmarket',data = df_heat,palette = "Blues")
plt.title('Body type vs Days on Market', fontsize = 14)
plt.xlabel('Body types')
plt.ylabel('Days on Market')
plt.show()
```

```
import seaborn as sns
cars = ['FORD', 'CHEVROLET', 'HONDA', 'TOYOTA', 'NISSAN', 'OTHERS']
data = [25, 18, 16, 16, 12, 213]
plt.figure(figsize=(15,10), dpi=100)
plt.pie(data, labels = cars,autopct='%1.1f%%')
plt.title(' COMPANY NAMES')
plt.show()
```

```
import seaborn as sns
plt.figure(figsize=(5,6), dpi=100)
sns.barplot(x ='city_fuel_economy',y='body_type',data = df_heat,palette = "Blues")
plt.title('car body type and its mgc')
plt.xlabel('Mileage')
plt.ylabel('Body type')
plt.show()
```