CIS5200 Term Project Tutorial

Authors: Vijay J Muthupillai; Priya Ramdas;

Savita Yadav; Heta Parekh;

Sonam Suryawanshi

Instructor: Jongwook Woo

Date: 12/9/2020

Cargurus Used Cars Data Analysis Using Hadoop

Term Project(Group 2)

Vijay J Muthupillai (vmuthup@calstatela.edu), Priya Ramdas (pramdas2@calstatela.edu),

Savita Yadav (syadav5@calstatela.edu), Heta Parekh (hparekh2@calstatela.edu),

Sonam Suryawanshi (ssuryaw@calstatela.edu)

Objectives

In this hands-on lab, you will learn how to:

- Download data from amazon S3 and upload the data to the Hadoop cluster
- Create Hive tables in HDFS using HiveQL.
- Create HiveQL queries to manipulate and analyze the data
- Visualize the result in Excel, Tableau, and Power BI.

Introduction

Due to the increased cost of new cars and the consumer's desire to save income, used car sales are on a worldwide increase. The dataset used in this project is from the Kaggle website. It contains the most relevant information that CarGurus provides for used cars. Analysis of this data can help to infer the inventory of used cars based on make and model, price, transmission, mileage, city, body types, and each region-wise, etc.

Platform Spec

- Cluster Version – Oracle Big Data Compute Edition
- Number of Nodes – 3
- Memory size – 180 GB
- CPU – 12 OCPUs
- Storage – 957 GB

Prerequisites

Everything you need to go through the scripts and queries is already provisioned with the cluster. To export the analyzed data to Microsoft Excel, you must meet the following requirements:

- You must have a username and ipaddress to connect to an Oracle Cloud account.
- You must have **Microsoft Excel 2010**, **2013 or 2016** installed.
- You must have Tableau and Power BI installed on your machine for visualizations.
- You must have an account with the Power BI.

1. Connect to Oracle Cloud: Big Data Compute

You need to remotely access your Oracle Big Data that you executed in your Oracle Cloud account using ssh. Your CalStateLA username(**syadav5**) should be a username/password to connect to the Hadoop cluster at BDCE as follows:

NOTE: Do not forget to change **syadav5** with your username.

$ ssh syadav5@129.150.69.91

To enter password, type in your user name and press enter.

```
msavi@LAPTOP-3O6GU84C MINGW64 ~
$ ssh syadav5@129.150.69.91
-- WARNING -- This system is for the use of authorized users only. Individuals
using this computer system without authority or in excess of their authority
are subject to having all their activities on this system monitored and
recorded by system personnel. Anyone using this system expressly consents to
such monitoring and is advised that if such monitoring reveals possible
evidence of criminal activity system personnel may provide the evidence of such
monitoring to law enforcement officials.

syadav5@129.150.69.91's password:
Last login: Thu Nov 12 07:47:26 2020 from 24.6.79.203
-bash-4.1$ |
```

You are successfully connected to Oracle cloud.

2. Used Car data Loaded into Oracle Big Data

Loading the project datasets in your local directory will end up crashing due to space constraints. Try to load the data in any of the available directories.
For instance, point out the current location to **the "/dev/shm"** directory

Note: Make sure to delete the files from the **/dev/shm** home directory once you are done uploading the file in the **HDFS** directory.

Use the following command to change the directory.

$ cd dev/shm/

Below is the location of the CarGurus usedcar data that is used for this project. You can download the data file from amazon S3 as follows:

$ wget -O CarData.zip http://usedcardataset.s3.amazonaws.com/archive.zip

You should get something like this:

```
-bash-4.1$ wget -O CarData.zip http://usedcardataset.s3.amazonaws.com/archive.zip
--2020-11-12 00:15:18--  http://usedcardataset.s3.amazonaws.com/archive.zip
Resolving usedcardataset.s3.amazonaws.com... 52.216.104.59
Connecting to usedcardataset.s3.amazonaws.com|52.216.104.59|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2287140843 (2.1G) [application/zip]
Saving to: "CarData.zip"

100%[===========================================================================================================================>] 2,287,140,843 33.0M/s   in 89s

2020-11-12 00:16:48 (24.5 MB/s) - "CarData.zip" saved [2287140843/2287140843]

-bash-4.1$ 
```

Then, you need to unzip the files.

$ unzip CarData.zip

List current directory to check if all the files are available in your home directory:

```
-bash-4.1$ ls
CarData.zip  used_cars_data.csv
-bash-4.1$ 
```

3. Create directories in HDFS

Run the following commands for creating directories:

```
$ hdfs dfs -mkdir GP2TermProject
$ hdfs dfs -mkdir GP2TermProject/tables
$ hdfs dfs -mkdir GP2TermProject/tables/cgdata
```

To view the files and directories created at HDFS use the following command:

```
$ hdfs dfs -ls GP2TermProject/tables
```

```
-bash-4.1$
-bash-4.1$ hdfs dfs -ls GP2TermProject/tables
Found 1 items
drwxr-xr-x   - syadav5 hdfs          0 2020-11-12 22:00 GP2TermProject/tables/cgdata
-bash-4.1$
```

4. Put files in HDFS directories

Run the following commands to put dictionaries into respective folders:

```
$ hdfs dfs -put /dev/shm/used_cars_data.csv GP2TermProject/tables/cgdata/
```

You can run the following commands to check the files are there:

```
$ hdfs dfs -ls GP2TermProject/tables/cgdata/
```

```
-bash-4.1$
-bash-4.1$ hdfs dfs -ls GP2TermProject/tables/cgdata/
Found 1 items
-rw-r--r--   2 syadav5 hdfs 9980208148 2020-11-12 22:06 GP2TermProject/tables/cgdata/used_cars_data.csv
-bash-4.1$ |
```

Run the following HDFS command to provide permission to the files under the project folder.
NOTE: To make your beeline command work properly, use the following command.

```
$ hdfs dfs -chmod -R o+w GP2TermProject/
```

5. Creating Hive tables to query data

The following Hive statement creates an external table that allows Hive to query data stored in HDFS. External tables preserve the data in the original file format while allowing the Hive to perform queries against the data within the file.
The Hive statements below creates a new table named cargurus_usedcars, by describing the fields within the file, the delimiter (Comma) between fields.
Now open another terminal window and login into your account using ssh command.

Open **beeline** Command Line Interface using the following command to run hive queries.
Beeline is for multiple user's access to Hive Server 2 of a Hadoop cluster. You have to copy and paste the "!connect …" command given by the instructor at beeline and press enter without any password when it asks for a password.

```
$ beeline
```

-bash-4.1$ beeline
WARNING: Use "yarn jar" to launch YARN applications.
Beeline version 1.2.1000.2.4.2.0-258 by Apache Hive
beeline>

NOTE: Type the **connect** command as follows, and it should be given by the instructor:

beeline> **!connect jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigdai-nov-bdcsce-2:2181,bigdai-nov-bdcsce-3:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2?tez.queue.name=interactive bdcsce_admin**
Connecting to
jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigdai-nov-bdcsce-2:2181,bigdai-nov-bdcsce-3:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2?tez.queue.name=interactive
Enter password for
jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigdai-nov-bdcsce-2:2181,bigdai-nov-bdcsce-3:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2?tez.queue.name=interactive:
Connected to: Apache Hive (version 1.2.1000.2.4.2.0-258)
Driver: Hive JDBC (version 1.2.1000.2.4.2.0-258)

Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigd>

NOTE: If you see "**CLOSED**" in the above beeline shell prompt, it is not connected to Hive Server2.
Now you have to create your database with your username to separate your tables with other users. For example, the user **syadav5** should run the following:

NOTE: You have to use your username.

0: jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigd> create database if not exists syadav5;

No rows affected (0.188 seconds)

0: jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigd> show databases;

```
+----------------+--+
| database_name  |
+----------------+--+
| default        |
| jwoo5          |
| kyayosh        |
| malam          |
| mcampo61       |
| pramdas2       |
| syadav5        |
+----------------+--+
```
7 rows selected (0.167 seconds)

0: jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigd> use syadav5;

No rows affected (0.17 seconds)

In the beeline shell CLI, you need to copy and paste the following HiveQL code to create an external table "cargurus_usedcars".

NOTE: Don't forget to replace **syadav5** to your account name in the following HQL code.

```
DROP TABLE IF EXISTS cargurus_usedcars;


CREATE EXTERNAL TABLE IF NOT EXISTS cargurus_usedcars(vin string, back_legroom string, bed string,
bed_height  string, bed_length string, body_type string, cabin string, city string, city_fuel_economy
float, combine_fuel_economy int, daysonmarket int, dealer_zip string, description string,
engine_cylinders string, engine_displacement float, engine_type string, exterior_color string, fleet
string, frame_damaged string, franchise_dealer string, franchise_make string, front_legroom string,
fuel_tank_volume string, fuel_type string, has_accidents string, height string, highway_fuel_economy
float, horsepower int, interior_color string, iscab string, is_certified string, is_cpo string, is_new
boolean, is_oemcpo string, latitude double, length string, listed_date date, listing_color string,
listing_id bigint, longitude double, main_picture_url string, major_options string, make_name string,
maximum_seating string, mileage float, model_name string, owner_count int, power string, price float,
salvage string, savings_amount int, seller_rating float, sp_id bigint, sp_name string, theft_title string,
torque string, transmission string, transmission_display string, trimid string, trim_name string,
vehicle_damage_category string, wheel_system  string, wheel_system_display string, wheelbase
string, width string, year int) ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES ("separatorChar" = "," , "quoteChar"  = "\"")  STORED AS TEXTFILE LOCATION
'/user/syadav5/GP2TermProject/tables/cgdata/' TBLPROPERTIES ('skip.header.line.count' = '1');
```

Then, in the beeline shell, you need to check if the table **"cargurus_usedcars"** is shown:

```
show tables;
```

```
0: jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigd> show tables;

+--------------------+--+
|     tab_name       |
+--------------------+--+
| cargurus_usedcars  |
+--------------------+--+
1 row selected (0.173 seconds)
```

Next you can query the content of the cargurus_usedcars table:

select vin, body_type, city, latitude, longitude, make_name, maximum_seating, mileage, model_name, price, seller_rating, transmission, trim_name, year from cargurus_usedcars limit 20;

```
0: jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigd> select vin, body_type, city, latitude, longitude, make_name, maximum_seating, mileage, model_name, price, seller_rating, transmission, trim_name, year from carguru
s_usedcars limit 20;
+----------------+-----------------+----------+----------+-----------+------------+-----------------+---------+------------------+---------+---------------+--------------+----------------------+
|      vin       |    body_type    |   city   | latitude | longitude |  make_name | maximum_seating | mileage |    model_name    |  price  | seller_rating | transmission |      trim_name       |
|      year      |
+----------------+-----------------+----------+----------+-----------+------------+-----------------+---------+------------------+---------+---------------+--------------+----------------------+
| ZACNJABB5KPJ92081 | SUV / Crossover | Bayamon  | 18.3988  | -66.1582  | Jeep       | 5 seats         | 7.0     | Renegade         | 23141.0 | 2.8           | A            | Latitude FWD         |
|      2019      |
| SALCJ2FX1LH858117 | SUV / Crossover | San Juan | 18.4439  | -66.0785  | Land Rover | 7 seats         | 8.0     | Discovery Sport  | 46500.0 | 3.0           | A            | S AWD                |
|      2020      |
| JF1VA2M67G9829723 | Sedan           | Guaynabo | 18.3467  | -66.1098  | Subaru     | 5 seats         |         | WRX STI          | 46995.0 |               | M            | Base                 |
|      2016      |
| SALRR2RV0L2433391 | SUV / Crossover | San Juan | 18.4439  | -66.0785  | Land Rover | 7 seats         | 11.0    | Discovery        | 67430.0 | 3.0           | A            | V6 HSE AWD           |
|      2020      |
| SALCJ2FXXLH862327 | SUV / Crossover | San Juan | 18.4439  | -66.0785  | Land Rover | 7 seats         | 7.0     | Discovery Sport  | 48880.0 | 3.0           | A            | S AWD                |
|      2020      |
| SALYK2EX1LA261711 | SUV / Crossover | San Juan | 18.4439  | -66.0785  | Land Rover | 5 seats         | 12.0    | Range Rover Velar| 66903.0 | 3.0           | A            | P250 R-Dynamic S AWD |
|      2020      |
| 3MZBPABL6KM107908 | Sedan           | Bayamon  | 18.3988  | -66.1582  | Mazda      | 5 seats         | 14.0    | MAZDA3           | 23695.0 | 2.8           | A            | Sedan FWD            |
|      2019      |
| SALYK2EX5LA275434 | SUV / Crossover | San Juan | 18.4439  | -66.0785  | Land Rover | 5 seats         | 11.0    | Range Rover Velar| 68520.0 | 3.0           | A            | P250 R-Dynamic S AWD |
|      2020      |
| SALCJ2FX6LH858128 | SUV / Crossover | San Juan | 18.4439  | -66.0785  | Land Rover | 7 seats         | 8.0     | Discovery Sport  | 51245.0 | 3.0           | A            | S AWD                |
|      2020      |
| SALZL2GX4LH007593 | SUV / Crossover | San Juan | 18.4439  | -66.0785  | Land Rover | 5 seats         | 254.0   | Range Rover Evoque| 84399.0| 3.0           | A            | P300 R-Dynamic SE AWD|
|      2020      |
| ZARBAAC41FM129303 | Coupe           | Guaynabo | 18.3467  | -66.1098  | Alfa Romeo | 2 seats         | 301.0   | 4C               | 97579.0 |               | A            | Launch Edition Coupe |
RWD | 2015   |
| SALZJ2FX0LH081763 | SUV / Crossover | San Juan | 18.4439  | -66.0785  | Land Rover | 5 seats         | 22.0    | Range Rover Evoque| 51885.0| 3.0           | A            | P250 S AWD           |
|      2020      |
| WBA8B7C53GK368522 | Sedan           | Guaynabo | 18.3467  | -66.1098  | BMW        | 5 seats         | 6903.0  | 3 Series         | 58995.0 |               | A            | 340i xDrive Sedan AWD|
|      2016      |
| SALYK2EX8LA268316 | SUV / Crossover | San Juan | 18.4439  | -66.0785  | Land Rover | 5 seats         | 20.0    | Range Rover Velar| 68725.0 | 3.0           | A            | P250 R-Dynamic S AWD |
|      2020      |
| 3MZBPABL1KM108237 | Sedan           | Bayamon  | 18.3988  | -66.1582  | Mazda      | 5 seats         | 204.0   | MAZDA3           | 23695.0 | 2.8           | A            | Sedan FWD            |
|      2019      |
| 3MZBPABL4KM107969 | Sedan           | Bayamon  | 18.3988  | -66.1582  | Mazda      | 5 seats         | 61.0    | MAZDA3           | 23695.0 | 2.8           | A            | Sedan FWD            |
|      2019      |
| SALCP2FX9LH857747 | SUV / Crossover | San Juan | 18.4439  | -66.0785  | Land Rover | 7 seats         | 6.0     | Discovery Sport  | 52275.0 | 3.0           | A            | SE AWD               |
|      2020      |
| SALYK2EX8LA284533 | SUV / Crossover | San Juan | 18.4439  | -66.0785  | Land Rover | 5 seats         | 7.0     | Range Rover Velar| 68760.0 | 3.0           | A            | P250 R-Dynamic S AWD |
|      2020      |
| SALYK2EX0LA284669 | SUV / Crossover | San Juan | 18.4439  | -66.0785  | Land Rover | 5 seats         | 5.0     | Range Rover Velar| 68760.0 | 3.0           | A            | P250 R-Dynamic S AWD |
|      2020      |
| SALCP2FX8LH854709 | SUV / Crossover | San Juan | 18.4439  | -66.0785  | Land Rover | 7 seats         | 8.0     | Discovery Sport  | 53770.0 | 3.0           | A            | SE AWD               |
|      2020      |
+----------------+-----------------+----------+----------+-----------+------------+-----------------+---------+------------------+---------+---------------+--------------+----------------------+
20 rows selected (0.25 seconds)
```

6. Creating Hive Queries to Analyze Data

You will create the following tables which will have all the data from the main table. All the subsequent queries will be based on the main table "cargurus_usedcars".

1. **The below command will create a table based on the make and model of the car which has more seller ratings in the past 5 years.**

NOTE: Don't forget to replace **hparekh2** to your account name in the following HQL code.

Copy and paste the following Hive code to Beeline shell to create a table **car_ratings**:

```
CREATE EXTERNAL TABLE IF NOT EXISTS car_rating (
make_name string, model_name string,year int, seller_rating string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE LOCATION '/user/hparekh2/GP2TermProject/tables/car_rating/';


INSERT OVERWRITE TABLE car_rating
Select make_name, model_name, year, AVG(seller_rating)
FROM cargurus_usedcars
WHERE year >= 2015 and year <= 2020
GROUP BY make_name, model_name, year;
```

Then, in the beeline shell, you need to check if the tables **"car_rating"** is shown:

```
show tables;
```

```
describe car_rating
```

Now you can query the content of the table:

```
select * from car_rating LIMIT 5;
```

It will display the result as follows:

```
0: jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigd> Select * from car_rating LIMIT 5;
+-------------------+---------------------+-----------------+---------------------------+--+
| car_rating.make_name | car_rating.model_name | car_rating.year | car_rating.seller_rating |
+-------------------+---------------------+-----------------+---------------------------+--+
| Acura             | NSX                 | 2020            | 4.38957153630683          |
| Acura             | RLX Hybrid Sport    | 2019            | 4.397438423645321         |
| Acura             | TLX                 | 2017            | 4.220877661354583         |
| Alfa Romeo        | 4C                  | 2015            | 4.1595165697161995        |
| Alfa Romeo        | 4C                  | 2018            | 4.45261250524917          |
+-------------------+---------------------+-----------------+---------------------------+--+
5 rows selected (0.204 seconds)
```

2. The below command provides N-gram sentiment analysis of top seller rating cars.

NOTE: Don't forget to replace **hparekh2** to your account name in the following HQL code.

Create table bigram_analysis(new_ar array<struct<ngram:array<string>, estfrequency:double>>);

INSERT OVERWRITE TABLE bigram_analysis
Select context_ngrams(sentences(lower(description)), array(null,null),100)as bigram
FROM cargurus_usedcars
WHERE make_name IN ('Lotus','Bentley', 'Ferrari', 'Karma', 'Lamborghini', 'Rolls-Royce', 'Porsche', 'McLaren', 'Aston Martin', 'smart', 'MINI', 'Jaguar', 'Tesla', 'Lexus', 'Audi', 'Land Rover', 'Mercedes')
AND year >= 2015;

Create table frequency_bigram (ngram string, estfrequency double)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE LOCATION '/user/**hparekh2**/GP2TermProject/tables/ngramAnalysis/';

Insert overwrite table frequency_bigram
Select concat(X.ngram[0], ' ',X.ngram[1]) as bigram, X.estfrequency
From bigram_analysis LATERAL VIEW explode(new_ar) Z as X;

Now you can query the content of the table:

select * from frequency_bigram limit 10;

It will display the result as follows:

```
+------------------------------+----------------------------------+
| frequency_bigram.ngram       | frequency_bigram.estfrequency    |
+------------------------------+----------------------------------+
| steering wheel               | 435699.0                         |
| air bag                      | 183303.0                         |
| vanity mirror                | 169875.0                         |
| power door                   | 140672.0                         |
| rear seat                    | 132615.0                         |
| keyless entry                | 124803.0                         |
| seat power                   | 124400.0                         |
| door mirrors                 | 119999.0                         |
| a c                          | 119706.0                         |
| driver seat                  | 119248.0                         |
+------------------------------+----------------------------------+
```

3. The below query will create a table based on the inventory of cars based on body type and price range for the past 5 years.

NOTE: Don't forget to replace **pramdas2** to your account name in the following HQL code.

Copy and paste the following Hive code to Beeline shell to create a table carsPriceRange :

This table will analyse data based on the price range.

```
CREATE TABLE carsPriceRange AS select body_type, price , case when price > 0 and price < 10000 then
'0 - 10000' when price > 10000 and price < 20000 then '10000 - 20000' when price > 20000 and price <
30000 then '20000 - 30000' when price > 30000 and price < 40000 then '30000 - 40000' when price >
40000 and price < 50000 then '40000 - 50000' when price > 50000 and price < 60000 then '50000 -
60000' when price > 60000 and price < 70000 then '60000 - 70000' when price > 70000 and price <
80000 then '70000 - 80000' when price > 80000 and price < 90000 then '80000 - 90000' else '90000 -
above' end as price_range from cargurus_usedcars where  body_type != '' and year >= 2015 and year
<= 2020;
```

Now you can query the content of the table:

```
select * from carsPriceRange limit 10;
```

It will display the result as follows:

```
[0: jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigd> select * from carsPriceRange limit 10;
+--------------------------+----------------------+----------------------------+--+
| carspricerange.body_type | carspricerange.price | carspricerange.price_range |
+--------------------------+----------------------+----------------------------+--+
| SUV / Crossover          | 23141.0              | 20000 - 30000              |
| SUV / Crossover          | 46500.0              | 40000 - 50000              |
| Sedan                    | 46995.0              | 40000 - 50000              |
| SUV / Crossover          | 67430.0              | 60000 - 70000              |
| SUV / Crossover          | 48880.0              | 40000 - 50000              |
| SUV / Crossover          | 66903.0              | 60000 - 70000              |
| Sedan                    | 23695.0              | 20000 - 30000              |
| SUV / Crossover          | 68520.0              | 60000 - 70000              |
| SUV / Crossover          | 51245.0              | 50000 - 60000              |
| SUV / Crossover          | 84399.0              | 80000 - 90000              |
+--------------------------+----------------------+----------------------------+--+
10 rows selected (0.119 seconds)                      _
```

Once the carsPriceRange table has been created we will group the data by body type and price range to get the final result.

CREATE EXTERNAL TABLE IF NOT EXISTS CarsPriceRangeByBodyType(price_range string, body_type string,Car_count string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE LOCATION '/user/**pramdas2**/GP2TermProject/tables/CarsPriceRangeByBodyType/';

INSERT OVERWRITE TABLE CarsPriceRangeByBodyType select price_range,body_type,count(price_range) from carsPriceRange group by price_range,body_type order by price_range;

Now you can query the content of the table:

---

select * from CarsPriceRangeByBodyType limit 10;

It will display the result something like follows:

```
[0: jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigd> select * from CarsPriceRangeByBodyType limit 10;
+-----------------------------------+-----------------------------------+-----------------------------------+--
+
| carspricerangebybodytype.price_range | carspricerangebybodytype.body_type | carspricerangebybodytype.car_count |
+-----------------------------------+-----------------------------------+-----------------------------------+--
+
| 0 - 10000                         | Minivan                           | 555                               |
| 0 - 10000                         | SUV / Crossover                   | 2142                              |
| 0 - 10000                         | Coupe                             | 145                               |
| 0 - 10000                         | Convertible                       | 20                                |
| 0 - 10000                         | Hatchback                         | 4462                              |
| 0 - 10000                         | Pickup Truck                      | 31                                |
| 0 - 10000                         | Sedan                             | 13951                             |
| 0 - 10000                         | Van                               | 168                               |
| 0 - 10000                         | Wagon                             | 928                               |
| 10000 - 20000                     | Coupe                             | 4535                              |
+-----------------------------------+-----------------------------------+-----------------------------------+--
```

4. The below command will create a table based on the make and model of the car to check which will take longer time to sell.

NOTE: Don't forget to replace **ssuryaw** to your account name in the following HQL code.

CREATE TABLE daysonmarketbydays AS select make_name,model_name, case when daysonmarket > 0 and daysonmarket < 91 then '0 - 91' when daysonmarket > 91 and daysonmarket < 182 then '91 - 182' when daysonmarket > 182 and daysonmarket < 365 then '182 - 365' else '365 - above' end as days_range from cargurus_usedcars_1 where
body_type != '' and year >= 2015 and year <= 2020;

Now you can query the content of the table:

select * from daysonmarketbydays limit 10;

It will display the result as follows:

```
0: jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigd> select * from daysonmarketbydays LIMIT
10 ;
+-----------------------------+-----------------------------------+-------------------
-----------+--+
| daysonmarketbydays.make_name | daysonmarketbydays.model_name |
daysonmarketbydays.days_range |
+-----------------------------+-----------------------------------+-------------------
-----------+--+
| Jeep                        | Renegade                          | 365 - above
| Land Rover                  | Discovery Sport                   | 182 - 365
| Subaru                      | WRX STI                           | 365 - above
| Land Rover                  | Discovery                         | 182 - 365
| Land Rover                  | Discovery Sport                   | 91 - 182
| Land Rover                  | Range Rover Velar                 | 182 - 365
| Mazda                       | MAZDA3                            | 365 - above
| Land Rover                  | Range Rover Velar                 | 0 - 91
| Land Rover                  | Discovery Sport                   | 182 - 365
| Land Rover                  | Range Rover Evoque                | 365 - above
+-----------------------------+-----------------------------------+-------------------
-----------+--+
```

CREATE EXTERNAL TABLE IF NOT EXISTS car_market_days_range(make_name string,model_name string,days_range string ,Car_count string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE LOCATION '/user/**ssuryaw**/GP2TermProject/tables/CarsdaysRangeByMake/';

INSERT OVERWRITE TABLE car_market_days_range select make_name,model_name,days_range,count(days_range) from daysonmarketbydays group by make_name,model_name,days_range order by days_range;

Now you can query the content of the table:

select * from car_market_days_range limit 10;

It will display the result as follows:

```
0: jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigd> select * from car_market_days_range LIMIT 10;
+---------------------------------+---------------------------------+---------------------------------+---------------------------------+--+
| car_market_days_range.make_name | car_market_days_range.model_name | car_market_days_range.days_range | car_market_days_range.car_count |
+---------------------------------+---------------------------------+---------------------------------+---------------------------------+--+
| Acura                           | ILX                             | 0 - 91                          | 2076                            |
| Acura                           | MDX                             | 0 - 91                          | 8135                            |
| Acura                           | MDX Hybrid Sport                | 0 - 91                          | 187                             |
| Acura                           | NSX                             | 0 - 91                          | 27                              |
| Acura                           | RDX                             | 0 - 91                          | 3395                            |
| Acura                           | RLX                             | 0 - 91                          | 130                             |
| Acura                           | RLX Hybrid Sport                | 0 - 91                          | 53                              |
| Acura                           | TLX                             | 0 - 91                          | 3847                            |
| Alfa Romeo                      | 4C                              | 0 - 91                          | 14                              |
| Alfa Romeo                      | Giulia                          | 0 - 91                          | 1456                            |
+---------------------------------+---------------------------------+---------------------------------+---------------------------------+--+
10 rows selected (0.108 seconds)
```

5. The below command will create a table based on the inventory of cars having more accidents.

CREATE EXTERNAL TABLE IF NOT EXISTS car_has_accidents_model (
make_name string, model_name string,year int, car_has_accidents string, has_accidents_count string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE LOCATION '/user/**ssuryaw**/GP2TermProject/tables/CarsHasAccidentByMake/' ;

INSERT OVERWRITE TABLE car_has_accidents_model
Select make_name, model_name, has_accidents, count(has_accidents)
FROM cargurus_usedcars_1
WHERE year >= 2005 and year <= 2020
GROUP BY make_name, model_name, year , has_accidents;

Now you can query the content of the table:

select * from car_has_accidents_model limit 10;

It will display the result as follows:

```
0: jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigd> select * from car_has_accidents_model LIMIT 10;
+-------------------------------+-------------------------------+-------------------------------+------------------------------------------+-----------------------------------------+--+
| car_has_accidents_model.make_name | car_has_accidents_model.model_name | car_has_accidents_model.year | car_has_accidents_model.car_has_accidents | car_has_accidents_model.has_accidents_count |
+-------------------------------+-------------------------------+-------------------------------+------------------------------------------+-----------------------------------------+--+
| Acura                         | ILX                           | 2013                          | False                                    | 56                                      |
| Acura                         | ILX                           | 2017                          | True                                     | 102                                     |
| Acura                         | MDX                           | 2008                          | True                                     | 64                                      |
| Acura                         | MDX                           | 2012                          | False                                    | 184                                     |
| Acura                         | MDX                           | 2014                          |                                          | 1                                       |
| Acura                         | MDX                           | 2017                          | False                                    | 1747                                    |
| Acura                         | MDX Hybrid Sport              | 2019                          | False                                    | 13                                      |
| Acura                         | MDX Hybrid Sport              | 2020                          | False                                    | 7                                       |
| Acura                         | NSX                           | 2019                          | True                                     | 1                                       |
| Acura                         | RDX                           | 2018                          |                                          | 2                                       |
+-------------------------------+-------------------------------+-------------------------------+------------------------------------------+-----------------------------------------+--+
10 rows selected (0.248 seconds)
```

6. The below command will create a table based on the inventory of used cars by body type.

NOTE: Don't forget to replace **vmuthup** to your account name in the following HQL code.

CREATE TABLE IF NOT EXISTS count_by_bodytype ROW FORMAT DELIMITED FIELDS TERMINATED BY ":" STORED AS TEXTFILE LOCATION "/user/**vmuthup**/GP2TermProject/bodytype" AS select body_type as BodyType, count(vin) as total from cargurus_usedcars group by body_type having count(vin) > 100 and trim(body_type) != '' and body_type is NOT NULL order by body_type;

```
0: jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigd> select * from count_by_bodytype;
+--------------------------+------------------------+--+
| count_by_bodytype.body_type | count_by_bodytype.total |
+--------------------------+------------------------+--+
| Convertible              | 26010                  |
| Coupe                    | 71607                  |
| Hatchback                | 88374                  |
| Minivan                  | 79802                  |
| Pickup Truck             | 474595                 |
| SUV / Crossover          | 1416402                |
| Sedan                    | 742036                 |
| Van                      | 47166                  |
| Wagon                    | 40505                  |
+--------------------------+------------------------+--+
9 rows selected (0.078 seconds)
```

7. The below command will create a table showing inventory of used cars by year.

NOTE: Don't forget to replace **hparekh2** to your account name in the following HQL code.

CREATE EXTERNAL TABLE IF NOT EXISTS inventory_car (year int, no_of_cars int) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE LOCATION '/user/**hparekh2**/GP2TermProject/tables/inventory_car';

INSERT OVERWRITE TABLE inventory_car SELECT year, Count(*) FROM cargurus_usedcars WHERE year > 2005 GROUP BY year;

Now you can query the content of the table:

select * from inventory_car limit 10;

It will display the result as follows:

```
0: jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigd> Select * from inventory_car LIMIT 10;
+----------------------+------------------------------+--+
| inventory_car.year   | inventory_car.no_of_cars     |
+----------------------+------------------------------+--+
| 2009                 | 22567                        |
| 2007                 | 22866                        |
| 2008                 | 27638                        |
| 2006                 | 16580                        |
| 2013                 | 67903                        |
| 2020                 | 1349815                      |
| 2016                 | 122201                       |
| 2010                 | 31470                        |
| 2015                 | 92554                        |
| 2019                 | 284650                       |
+----------------------+------------------------------+--+
10 rows selected (0.106 seconds)
```

8. The below command will create a table showing inventory of used cars by body styles and geographical location of the past ten years.

NOTE: Don't forget to replace **syadav5** to your account name in the following HQL code.

CREATE EXTERNAL TABLE IF NOT EXISTS bodystyle_region(body_type string, city string, latitude string, longitude string, make_name string, year string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE LOCATION '/user/**syadav5**/GP2TermProject/tables/bodystyle_region/';

INSERT OVERWRITE TABLE bodystyle_region
SELECT body_type, city, latitude, longitude, make_name, year from cargurus_usedcars
where body_type != ' ' and year >= 2010 and year <= 2020 and year != ' ';

```
0: jdbc:hive2://bigdai-nov-bdcsce-1:2181,bigd> select * from bodystyle_region limit 10;
+------------------------+----------------------+--------------------------+---------------------------+---------------------------+-------------------------+
| bodystyle_region.body_type | bodystyle_region.city | bodystyle_region.latitude | bodystyle_region.longitude | bodystyle_region.make_name | bodystyle_region.year |
+------------------------+----------------------+--------------------------+---------------------------+---------------------------+-------------------------+
| SUV / Crossover        | Bayamon              | 18.3988                  | -66.1582                  | Jeep                      | 2019                    |
| SUV / Crossover        | San Juan             | 18.4439                  | -66.0785                  | Land Rover                | 2020                    |
| Sedan                  | Guaynabo             | 18.3467                  | -66.1098                  | Subaru                    | 2016                    |
| SUV / Crossover        | San Juan             | 18.4439                  | -66.0785                  | Land Rover                | 2020                    |
| SUV / Crossover        | San Juan             | 18.4439                  | -66.0785                  | Land Rover                | 2020                    |
| SUV / Crossover        | San Juan             | 18.4439                  | -66.0785                  | Land Rover                | 2020                    |
| Sedan                  | Bayamon              | 18.3988                  | -66.1582                  | Mazda                     | 2019                    |
| SUV / Crossover        | San Juan             | 18.4439                  | -66.0785                  | Land Rover                | 2020                    |
| SUV / Crossover        | San Juan             | 18.4439                  | -66.0785                  | Land Rover                | 2020                    |
| SUV / Crossover        | San Juan             | 18.4439                  | -66.0785                  | Land Rover                | 2020                    |
+------------------------+----------------------+--------------------------+---------------------------+---------------------------+-------------------------+
10 rows selected (0.076 seconds)
```

7. Downloading data into your PC

After the Hive tables are created, you can download it to your lab (or personal PC/Laptop) as follows.

1. Switch on to the first terminal connected to the Oracle cloud to download the output file at the HDFS path.

$ ssh syadav5@ipaddress
syadav5@ipaddress's password:

Run the following command to check if files are present:

```
-bash-4.1$ hdfs dfs -ls GP2TermProject/tables/car_rating
-bash-4.1$ hdfs dfs -ls GP2TermProject/tables/ngramAnalysis
-bash-4.1$ hdfs dfs -ls GP2TermProject/tables/CarsPriceRangeByBodyType
-bash-4.1$ hdfs dfs -ls GP2TermProject/tables/CarsdaysRangeByMake
-bash-4.1$ hdfs dfs -ls GP2TermProject/tables/CarsHasAccidentByMake
-bash-4.1$ hdfs dfs -ls GP2TermProject/tables/bodytype
-bash-4.1$ hdfs dfs -ls GP2TermProject/tables/bodystyle_region
-bash-4.1$ hdfs dfs -ls GP2TermProject/tables/inventory_car
```

```
-bash-4.1$ hdfs dfs -ls GP2TermProject/tables/bodystyle_region
Found 1 items
-rwxr-xrwx   2 bdcsce_admin hdfs  135289281 2020-11-20 01:06 GP2TermProject/tables/bodystyle_region/000000_0
-bash-4.1$
```

You will see only one file named 000000_0 is present in all the following folders:

```
$ hdfs dfs -ls GP2TermProject/tables/CarsPriceRangeByBodyType/000000_0
$ hdfs dfs -ls GP2TermProject/tables/bodytype/000000_0
$ hdfs dfs -ls GP2TermProject/tables/bodystyle_region/000000_0
```

Except at the following locations. The latter locations have multiple files and will need a merge.

```
GP2TermProject/tables/car_rating
GP2TermProject/tables/ngramAnalysis
GP2TermProject/tables/CarsdaysRangeByMake
GP2TermProject/tables/CarsHasAccidentByMake
GP2TermProject/tables/inventory_car
```

2. Download the output files to the local file systems and rename it

Since all the folders have the same name file 000000_0, after downloading into local file system you will rename the file

```
-bash-4.1$ hdfs dfs -get GP2TermProject/tables/bodystyle_region/000000_0

-bash-4.1$ ls
 000000_0
```

Copy the "000000_0" file content to region.csv and verify if the file exists.

```
-bash-4.1$ cat 000000_0 > region.csv
-bash-4.1$ ls
000000_0  region.csv
-bash-4.1$
```

Similarly, do it for the other files.

```
-bash-4.1$ hdfs dfs -get GP2TermProject/tables/CarsPriceRangeByBodyType/000000_0
cat 000000_0 > carsPriceRangeByBodyType.csv

-bash-4.1$ hdfs dfs -get GP2TermProject/tables/bodytype/000000_0
cat 000000_0 > bodytype_result.csv
```

3. Now, you have to merge multiples files (for the directories car_rating, ngramAnalysis, CarsdaysRangeByMake, CarsHasAccidentByMake, inventory_car) into a single output file using the following commands:

-bash-4.1$ hdfs dfs -cat GP2TermProject/tables/car_rating/00*_0 | hdfs dfs -put - GP2TermProject/tables/car_rating/seller_rating.csv

-bash-4.1$ hdfs dfs -cat GP2TermProject/tables/ngramAnalysis/00*_0 | hdfs dfs -put - GP2TermProject/tables/ngramAnalysis/bigram.csv

-bash-4.1$ hdfs dfs -cat GP2TermProject/tables/CarsdaysRangeByMake/00*_0 | hdfs dfs -put -

GP2TermProject/tables/CarsdaysRangeByMake/days_range_data.csv

-bash-4.1$ hdfs dfs -cat GP2TermProject/tables/CarsHasAccidentByMake/00*_0 | hdfs dfs -put -

GP2TermProject/tables/CarsHasAccidentByMake/hasaccidentsBymake.csv

-bash-4.1$ hdfs dfs -cat GP2TermProject/tables/inventory_car/00*_0 | hdfs dfs -put -

GP2TermProject/tables/inventory_car/inventory_car.csv
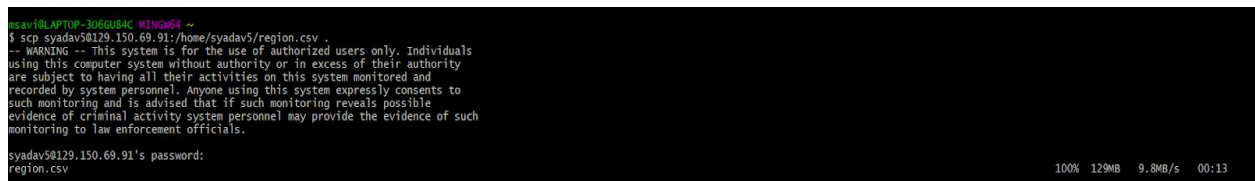
Download the output file using the following command:

-bash-4.1$ hdfs dfs -get GP2TermProject/tables/car_rating/seller_rating.csv

Likewise, do it for other files.

4. Open another terminal with git bash in order to import the output file using your lab
   computer (or your PC/Laptop) - you have to download the file to your lab computer
   (or your PC/Laptop). For example, your output file at the oracle cloud server is located
   at /home/**syadav5**/region.csv and remotely download the file.

$ scp syadav5@ipaddress:/home/**syadav5**/region.csv .

Similarly, do it for other files.

8. Visualizing Data

1. Seller rating of car brand and its models.

a) Open the seller_rating.csv file in Microsoft Excel. For the first row of the file, you need to insert the header to each column as follows:

Make Name   **Model Name**   **Year**   **Seller Rating**

	A	B		C	D	E
A1			f_x	Make Name		
	A	B		C	D	E
1	Make Name	Model Name		Year	Seller Rating	
2	Acura	NSX		2020	4.389571536	
3	Acura	RLX Hybrid Sport		2019	4.397438424	
4	Acura	TLX		2017	4.220877661	
5	Alfa Romeo	4C		2015	4.15951657	
6	Alfa Romeo	4C		2018	4.452612505	
7	Alfa Romeo	Stelvio		2019	4.320064587	
8	Audi	A4		2015	4.251184476	
9	Audi	A6		2017	4.296658396	
10	Audi	A8 Hybrid Plug-In		2020	4.426322816	
11	Audi	Q7		2018	4.361817625	
12	Audi	S5		2020	4.511433723	
13	Audi	TT		2018	4.558065667	
14	Audi	e-tron		2019	4.487198209	
15	BMW	2 Series		2015	4.262106866	
16	BMW	2 Series		2019	4.387646715	
17	BMW	3 Series		2019	4.425792265	
18	BMW	Z4		2015	4.384837057	
19	Buick	Encore		2017	4.33780166	
20	Buick	Regal TourX		2020	4.22979798	
21	Cadillac	ATS-V Coupe		2017	4.309177336	
22	Cadillac	Escalade		2015	4.298842067	
23	Chevrolet	Bolt EV		2018	4.27412219	

seller_rating

b) Open Tableau. Import the excel sheet.

c) Select graph Horizontal bars. Drag Make-Name in columns and Seller rating in rows. Change the seller rating aggregation to AVG and choose for Make-Name in color tab.

d) Select treemap. Drag Model-Name in columns and AVG (seller rating) in rows. Filter Make Name set and set 2.

e) Click on the dashboard and drag both the above worksheets to see the below results.
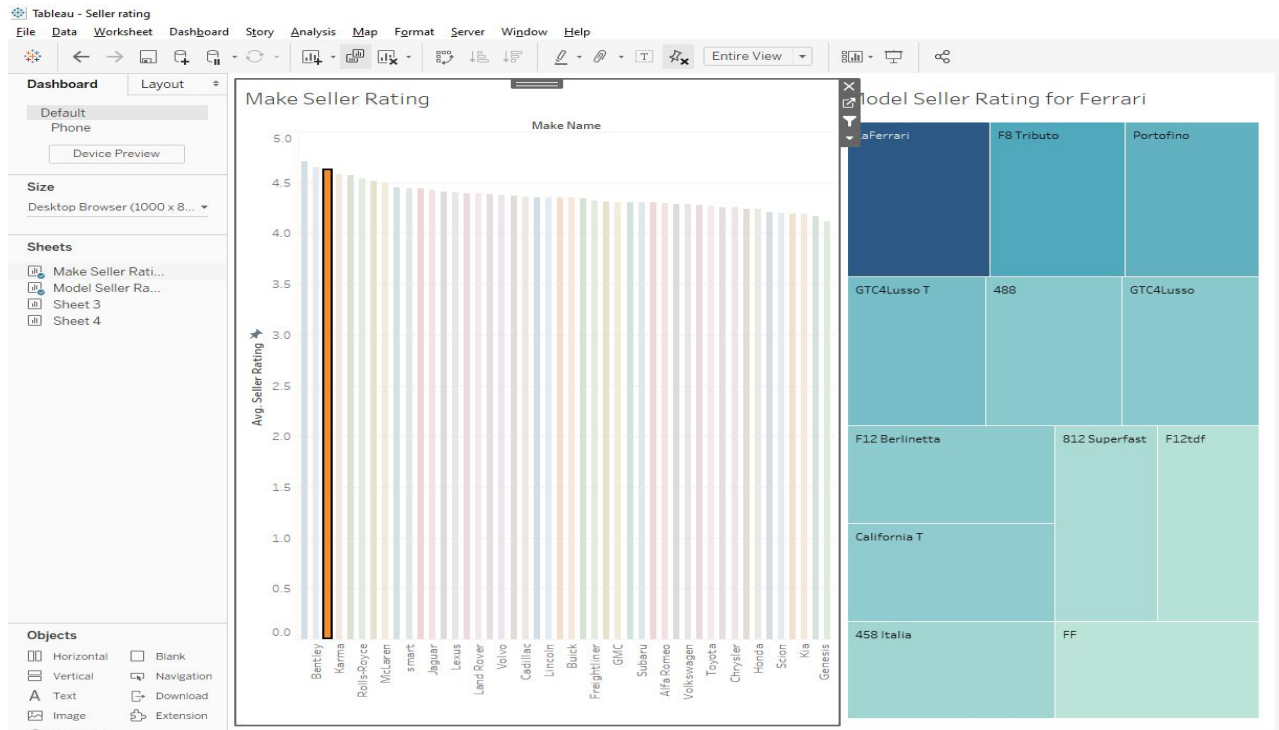
2. N-gram sentiment analysis of top seller rating cars.

a. Open the bigram.csv file in Microsoft Excel. For the first row of the file, you need to insert the header to each column as follows:

word   Frequency

	A	B	C
1	Word	Frequency	
2	air bag	183303	
3	keyless entry	124803	
4	camera	132653	
5	intermittent wipers	117791	
6	trip computer	115141	
7	traction control	112051	
8	stability control	106799	
9	air conditioning	103819	
10	center armrest	101312	
11	tire pressure	100167	
12	power steering	95276	
13	reading lights	94844	
14	cruise control	94718	
15	impact airbags	94082	
16	anti-roll bar	92587	
17	disc brakes	84462	
18	navigation system	76844	
19	leather steering	74156	
20	mirrors power	73513	
21	passenger vanity	72484	
22	driver vanity	72223	
23	rain sensing	70831	

b. Using SAP predictive Expert Analytics

SAP BusinessObjects Predictive Analytics®

File   Help

Data Manager

Automated Analytics

    Modeler

    Social

    Recommendation

Expert Analytics

Toolkit

Welcome to
SAP BusinessObjects Predictive Analytics®

Expert Analytics

c. Importing the excel sheet and drawing a tag cloud
    Filtering frequency as measure and words as dimension in order to get the required results.
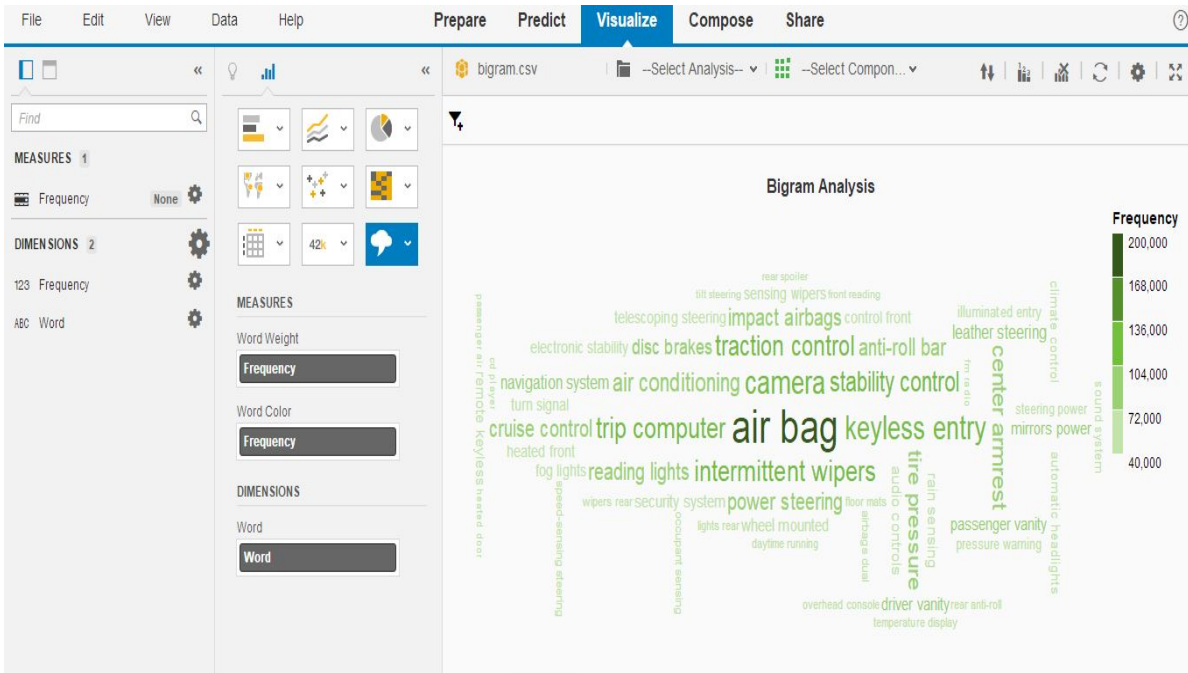
3. Inventory of cars by body type and price range

a. Open the carsPriceRangeByBodyType.csv file with Microsoft Excel
    and insert the header to each column as follows:

    Price Range   **Body Type**   **Cars Count**

	A	B	C	D
	Price Range	Body Type	Cars Count	
1	Price Range	Body Type	Cars Count	
2	0 - 10000	Minivan	555	
3	0 - 10000	SUV / Crossover	2142	
4	0 - 10000	Coupe	145	
5	0 - 10000	Convertible	20	
6	0 - 10000	Hatchback	4462	
7	0 - 10000	Pickup Truck	31	
8	0 - 10000	Sedan	13951	
9	0 - 10000	Van	168	
10	0 - 10000	Wagon	928	
11	10000 - 20000	Coupe	4535	
12	10000 - 20000	Hatchback	34803	
13	10000 - 20000	Pickup Truck	3621	
14	10000 - 20000	SUV / Crossover	196341	
15	10000 - 20000	Van	4453	
16	10000 - 20000	Convertible	636	
17	10000 - 20000	Minivan	15403	
18	10000 - 20000	Sedan	238366	
19	10000 - 20000	Wagon	11440	
20	20000 - 30000	Sedan	222446	
21	20000 - 30000	Minivan	17652	
22	20000 - 30000	Wagon	9855	
23	20000 - 30000	Coupe	15828	
24	20000 - 30000	Convertible	3802	

b. Upload the updated csv file to Power BI and select the stacked column chart. Then select the fields to generate the graph.
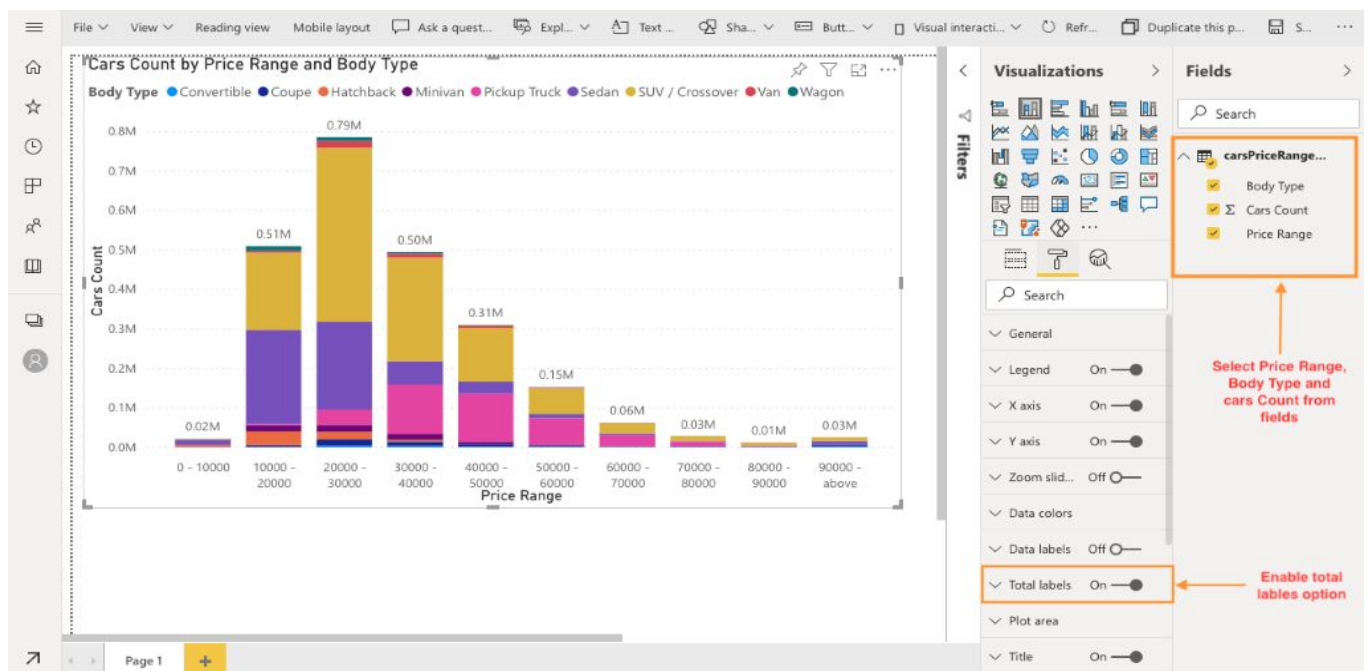
4. Inventory of cars based on the make and model of the car to check which will take longer time to sell.

    a. Open the days_range_data.csv file with Microsoft Excel and insert the header to each column as follows:

       Car_make   Car_model   daysmarket_range Car_count

	A	B	C	D
1	Car_make	Car_model	daysmarket_range	Car_count
2	Acura	ILX	0 - 91	2076
3	Acura	MDX	0 - 91	8135
4	Acura	MDX Hybrid Sport	0 - 91	187
5	Acura	NSX	0 - 91	27
6	Acura	RDX	0 - 91	3395
7	Acura	RLX	0 - 91	130
8	Acura	RLX Hybrid Sport	0 - 91	53
9	Acura	TLX	0 - 91	3847
10	Alfa Romeo	4C	0 - 91	14
11	Alfa Romeo	Giulia	0 - 91	1456
12	Alfa Romeo	Stelvio	0 - 91	1201
13	Aston Martin	DB11	0 - 91	58
14	Aston Martin	DB9	0 - 91	7
15	Aston Martin	DBS	0 - 91	15
16	Aston Martin	Rapide	0 - 91	7
17	Aston Martin	V12 Vantage	0 - 91	9
18	Aston Martin	V8 Vantage	0 - 91	6
19	Aston Martin	Vanquish	0 - 91	15
20	Aston Martin	Vantage	0 - 91	39
21	Audi	A3	0 - 91	2216
22	Audi	A3 Sportback	0 - 91	194
23	Audi	A4	0 - 91	2704
24	Audi	A4 Allroad	0 - 91	185
25	Audi	A5	0 - 91	449
26	Audi	A5 Sportback	0 - 91	809
27	Audi	A6	0 - 91	1835
28	Audi	A6 Allroad	0 - 91	94
29	Audi	A7	0 - 91	338

days_range_data
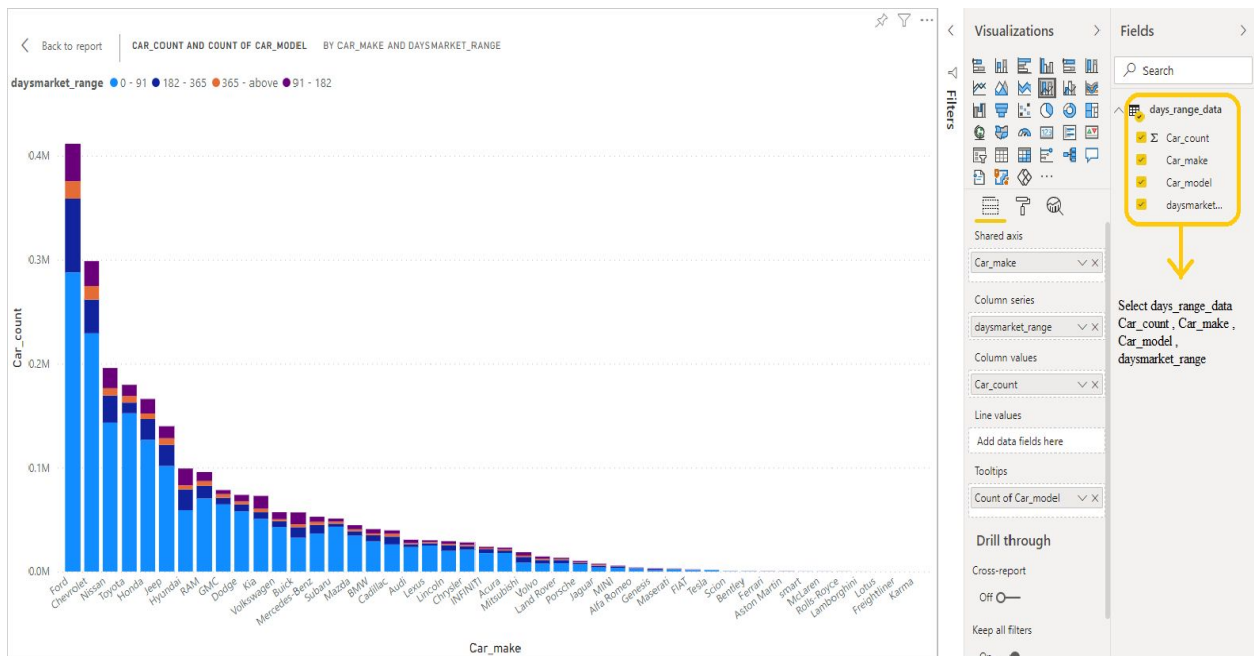
5. Inventory of cars by more accidents

    a.   Open the file hasaccidentsBymake.csv with Microsoft Excel and insert the header to each column as follows:

Car_make   Car_model   year Has_accident   countbyaccident

	A	B	C	D	E
1	Car_make	Car_model	year	Has_accident	countbyaccident
2	Acura	ILX	2013	FALSE	56
3	Acura	ILX	2017	TRUE	102
4	Acura	MDX	2008	TRUE	64
5	Acura	MDX	2012	FALSE	184
6	Acura	MDX	2014		1
7	Acura	MDX	2017	FALSE	1747
8	Acura	MDX Hybrid Sport	2019	FALSE	13
9	Acura	MDX Hybrid Sport	2020	FALSE	7
10	Acura	NSX	2019	TRUE	1
11	Acura	RDX	2018		2
12	Acura	RL	2007	TRUE	2
13	Acura	RSX	2005	FALSE	6
14	Acura	RSX	2005	TRUE	3
15	Acura	TLX	2016	TRUE	50
16	Acura	TLX	2019	FALSE	318
17	Acura	TSX	2014	FALSE	17
18	Alfa Romeo	4C	2018	FALSE	7
19	Alfa Romeo	Giulia	2017	FALSE	396
20	Alfa Romeo	Stelvio	2020	FALSE	16
21	Aston Martin	Rapide	2012	TRUE	1
22	Aston Martin	V8 Vantage	2012	TRUE	1
23	Aston Martin	Vanquish	2015	FALSE	6
24	Audi	A3	2015	TRUE	55
25	Audi	A3	2016	FALSE	230
26	Audi	A3	2019	FALSE	101
27	Audi	A3 Sportback	2018	TRUE	1
28	Audi	A4	2010		1
29	Audi	A4	2017	TRUE	179

hasaccidentsBymake

b. Upload the csv file to Power BI and select the stacked column chart. Then select the fields to generate the graph.
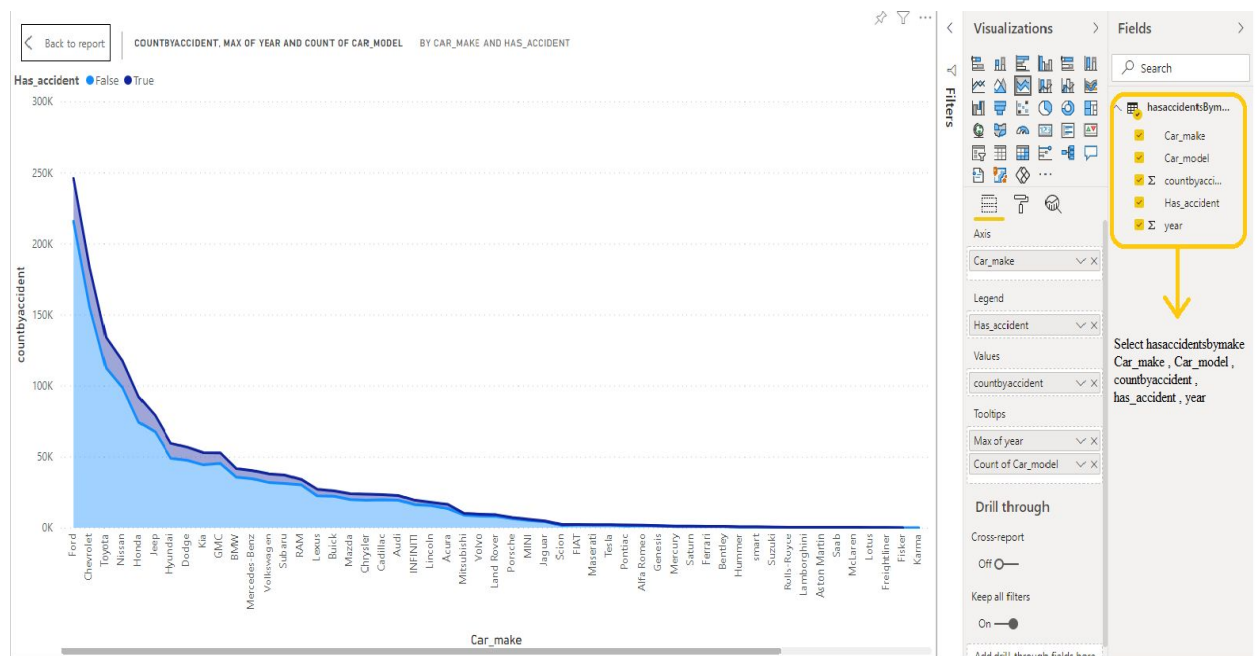
6. Inventory of cars by body type

a. Open the bodytype_result.csv file with Microsoft Excel and insert the header to each column as follows:

BodyType   Count

	A	B
1	BodyType	Count
2	Convertible	26010
3	Coupe	71607
4	Hatchback	88374
5	Minivan	79802
6	Pickup Truck	474595
7	SUV / Crossover	1416402
8	Sedan	742036
9	Van	47166
10	Wagon	40505

b. Go to the "insert" tab to find the pie chart option. You need to click and open "3-D Pie".
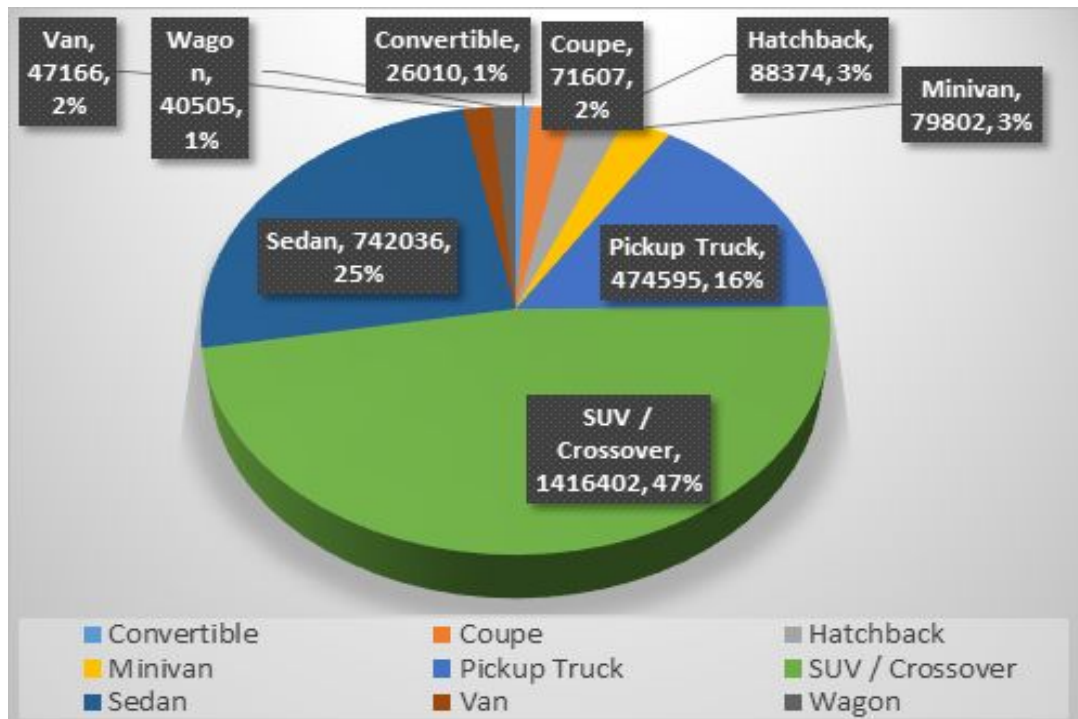
7. Inventory of car by Year

a. Open the inventory_car.csv file in Microsoft Excel. For the first row of the file, you need to insert the header to each column as follows:

year   No. of cars

	A	B
1	Year	No. of cars
2	2009	22567
3	2007	22866
4	2008	27638
5	2006	16580
6	2013	67903
7	2020	1349815
8	2016	122201
9	2010	31470
10	2015	92554
11	2019	284650
12	2011	43582
13	2014	76452
14	2017	348005
15	2018	210375
16	2012	52553

b. Open Tableau and Import car_inventory.csv . Next Drag no.of cars in rows and year in columns. Click on the Show labels.

Sheet 1
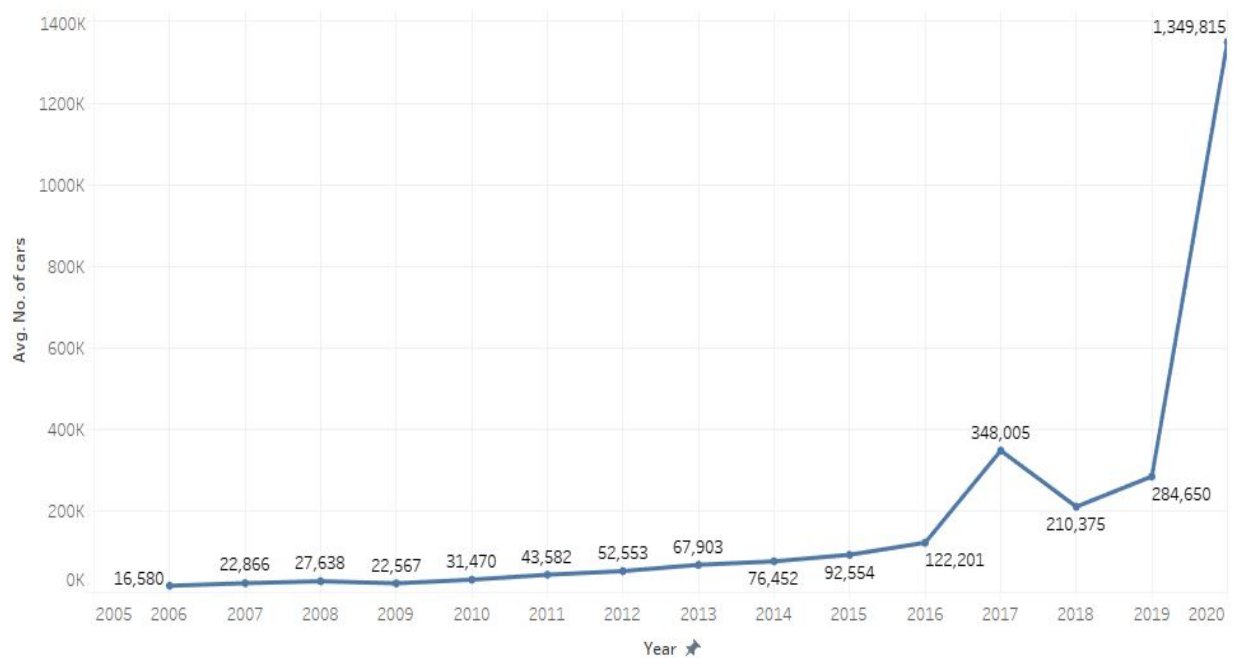
8. Geospatial representation of inventory of used cars by body types and region.

a. Open the region.csv file in Microsoft Excel. For the first row of the file, you need to insert the header to each column as follows:

body_type   city   latitude   longitude   make_name   year

| body_type	city	latitude	longitude	make_name	year
SUV / Crossover	Bayamon	18.3988	-66.1582	Jeep	2019
SUV / Crossover	San Juan	18.4439	-66.0785	Land Rover	2020
Sedan	Guaynabo	18.3467	-66.1098	Subaru	2016
SUV / Crossover	San Juan	18.4439	-66.0785	Land Rover	2020
SUV / Crossover	San Juan	18.4439	-66.0785	Land Rover	2020
SUV / Crossover	San Juan	18.4439	-66.0785	Land Rover	2020
Sedan	Bayamon	18.3988	-66.1582	Mazda	2019
SUV / Crossover	San Juan	18.4439	-66.0785	Land Rover	2020
SUV / Crossover	San Juan	18.4439	-66.0785	Land Rover	2020
SUV / Crossover	San Juan	18.4439	-66.0785	Land Rover	2020
Coupe	Guaynabo	18.3467	-66.1098	Alfa Romeo	2015
SUV / Crossover	San Juan	18.4439	-66.0785	Land Rover	2020
Sedan	Guaynabo	18.3467	-66.1098	BMW	2016
SUV / Crossover	San Juan	18.4439	-66.0785	Land Rover	2020
Sedan	Bayamon	18.3988	-66.1582	Mazda	2019
Sedan	Bayamon	18.3988	-66.1582	Mazda	2019
SUV / Crossover	San Juan	18.4439	-66.0785	Land Rover	2020
SUV / Crossover	San Juan	18.4439	-66.0785	Land Rover	2020
SUV / Crossover	San Juan	18.4439	-66.0785	Land Rover	2020
SUV / Crossover	San Juan	18.4439	-66.0785	Land Rover	2020
Sedan	Bayamon	18.3988	-66.1582	Mazda	2019
Sedan	Bayamon	18.3988	-66.1582	Mazda	2019
SUV / Crossover	San Juan	18.4439	-66.0785	Land Rover	2020
SUV / Crossover	San Juan	18.4439	-66.0785	Land Rover	2020
SUV / Crossover	San Juan	18.4439	-66.0785	Land Rover	2020
SUV / Crossover	San Juan	18.4439	-66.0785	Land Rover	2020
SUV / Crossover	Bayamon	18.3988	-66.1582	Mazda	2019
SUV / Crossover	Bayamon	18.3988	-66.1582	Jeep	2019

Sheet2   Sheet1

b. Save the file as excel format, that is, as region.xlsx.
c. Go to the "insert" tab to find the menu "3D Map" enabled – only excel file in xlsx should enable "3D Map". You need to click and open "3D Map".
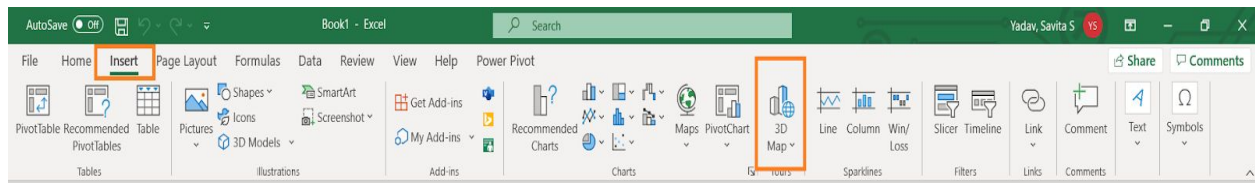
d. If it complaints that 3D Map cannot be open, then you need to make sure if you insert headers into the first row:

e. You will see the 3D map.

f. **NOTE:** If you don't see the layer frame in the right side, you may select all data manually before opening 3D map

g. Add the properties and values in the layer as follows
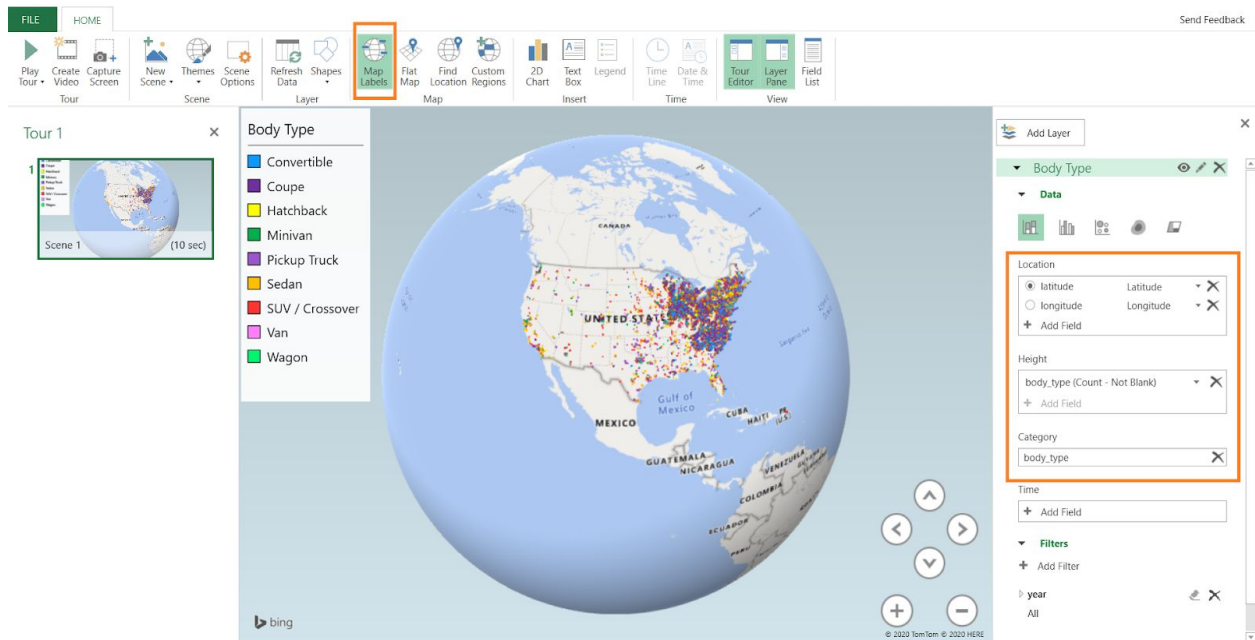
h. Now you can kill the "tour1" frame in the left.

i. You will get a view like below. You can observe sales and popular body styles by region.
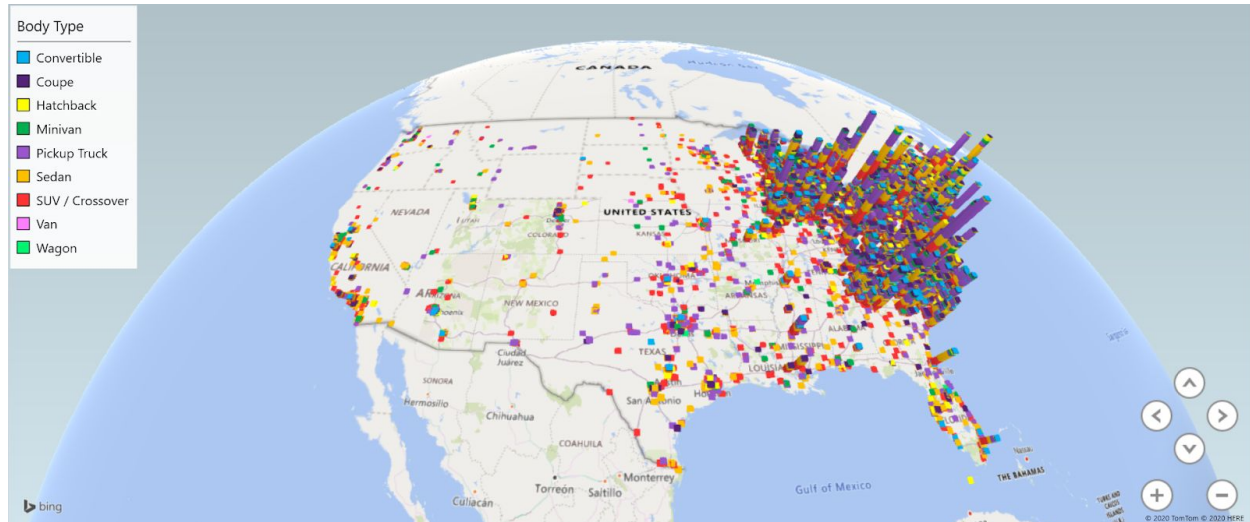
Summary

In this tutorial you learned how Oracle Cloud Big Data can be used to analyze different forms of raw data using Apache Hive. You went through a flow to understand how the raw data is first uploaded to HDFS, and then loaded to Hive tables for performing queries. Finally, you learned how to import the results of Hive queries and to create visualizations using tableau, Microsoft Power BI and 3D Map chart in MS-excel.

References

1. URL of data source: https://www.kaggle.com/ananaymital/us-used-cars-dataset

2. URL of GitHub: https://github.com/MRPriya/UsedCarSet

3. URL to download data source: http://usedcardataset.s3.amazonaws.com/archive.zip