Question 1: Write Hive and SparkSQL queries to create a table called used_cars from data and let it infer the schema in SparkSQL and for Hive define the schema yourself.

For Hive:

Firstly, In Hadoop local directory, Cars.zip file is downloaded from the given link on assignment webpage. Then, data is stored in directory of Hadoop HDFS. After that, as per observation of data, schema is created for table in Hive. After that from HDFS, data is transferred in the schema.

Question 1: Part 1/2: Creating Schema and table

```
%sh
wget -q https://github.com/tofighi/BigData/blob/main/datasets/cars/cars.zip?raw=true -O cars.zip
ls -lah cars.zip
```

```
-rw-r--r-- 1 zeppelin zeppelin 89M Feb 20 06:55 cars.zip
```

Took 3 sec. Last updated by anonymous at February 20 2022, 1:55:06 AM. (outdated)

```
%sh
unzip cars.zip && rm cars.zip
hadoop fs -mkdir /user/assignment1hive
hadoop fs -put cars.csv /user/assignment1hive
hadoop fs -ls -h /user/assignment1hive
```

```
replace carsArch.icvsev:?   [cya]ress.,z i[pn
]o, [A]ll, [N]one, [r]ename:  NULL
(EOF or read error, treating as "[N]one" ...)
mkdir: `/user/assignment1hive': File exists
Found 2 items
-rw-r--r--   2 zeppelin hadoop    400.0 M 2022-02-20 06:55 /user/assignment1hive/cars.csv
-rw-r--r--   2 zeppelin hadoop     63.0 K 2022-02-19 15:42 /user/assignment1hive/carstrial.csv
```

Took 10 sec. Last updated by anonymous at February 20 2022, 1:55:20 AM.

```
%Hive
create database if not exists cars_db_hive;
use cars_db_hive;
create table if not exists used_cars (maker string, model string, mileage int, manufacture_year int, engine_displacement int, engine_power int, body_type string, color_slug string, stk_year int,
transmission string, door_count int, seat_count int, fuel_type string, date_created string, date_last_seen string, price_eur float)
row format delimited fields terminated by ',' lines terminated by '\n' stored as TEXTFILE tblproperties("skip.header.line.count"="1");
```

```
Query executed successfully. Affected rows : -1
```

Question 1 (Part 2) loading data in Schema

```
%Hive
load data inpath '/user/assignment1hive/cars.csv' into table cars_db_hive.used_cars ;
```

```
Query executed successfully. Affected rows : -1
```

Took 1 sec. Last updated by anonymous at February 20 2022, 1:55:38 AM.

For Spark:

Cars.csv is downloaded from given link. Firstly, data is stored in directory of Hadoop HDFS. For Spark, we don't need to make schema as it is automatically making table as per data schema.

SPARK Question 1 (Part 1 & Part 2)

```
%sh
wget -q https://github.com/tofighi/BigData/blob/main/datasets/cars/cars.zip?raw=true -O cars.zip
unzip cars.zip && rm cars.zip
hadoop fs -mkdir /user/assignment1spark
hadoop fs -put cars.csv /user/assignment1spark
hadoop fs -ls -h /user/assignment1spark
```

```
Archive:  cars.zip
replace cars.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename:  NULL
(EOF or read error, treating as "[N]one" ...)
mkdir: `/user/assignment1spark': File exists
put: `/user/assignment1spark/cars.csv': File exists
```

Took 8 sec. Last updated by anonymous at February 20 2022, 3:38:48 AM. (outdated)

```
%spark.sql
use cars_db_spark;
Create table if not exists used_cars
USING CSV
OPTIONS (path "/user/assignment1spark/cars.csv", header "true" , inferSchema "true")
```

Took 0 sec. Last updated by anonymous at February 20 2022, 3:42:10 AM.

Question 2: Write Hive and SparkSQL queries to see how many missing values you have in each attribute (column)

For Hive:

Here I considered 'null', blank and 'None' as missing values. In column name: 'stk_year' , almost values are 'None'. Results are as per below.

Question 2: Write Hive and SparkSQL queries to see how many missing values you have in each attribute (column)

```
%Hive
Select
sum(case when maker is null or maker='' or maker='None' then 1 else 0 end) maker,
sum(case when model is null or model='' or model='None'  then 1 else 0 end) model,
sum(case when mileage is null or mileage='' or mileage='None'  then 1 else 0 end) mileage,
sum(case when manufacture_year is null or manufacture_year='' or manufacture_year='None'  then 1 else 0 end) manufacture_year,
sum(case when engine_displacement is null or engine_displacement='' or engine_displacement='None'  then 1 else 0 end) engine_displacement,
sum(case when engine_power is null or engine_power='' or engine_power='None'  then 1 else 0 end) enginer_power,
sum(case when body_type is null or body_type='' or body_type='None'  then 1 else 0 end) body_type,
sum(case when color_slug is null or body_type='' or body_type='None'  then 1 else 0 end) color_slug,
sum(case when stk_year is null or stk_year='' or stk_year='None'  then 1 else 0 end) stk_year,
sum(case when transmission is null or transmission='' or transmission='None'  then 1 else 0 end) transmission,
sum(case when door_count is null or door_count='' or door_count='None'  then 1 else 0 end) door_count,
sum(case when seat_count is null or seat_count='' or seat_count='None'  then 1 else 0 end) seat_count,
sum(case when fuel_type is null or fuel_type='' or fuel_type='None'  then 1 else 0 end) fuel_type,
sum(case when date_created is null or date_created='' or date_created='None'  then 1 else 0 end) date_created,
sum(case when date_last_seen is null or date_last_seen='' or date_last_seen='None'   then 1 else 0 end) date_last_seen,
sum(case when price_eur is null or price_eur='' or price_eur='None'  then 1 else 0 end) price_eur
from cars_db_hive.used_cars
```

```
sum(case when manufacture_year is null or manufacture_year='' or manufacture_year='None'  then 1 else 0 end) manufacture_year,
sum(case when engine_displacement is null or engine_displacement='' or engine_displacement='None'  then 1 else 0 end) engine_displacement,
sum(case when engine_power is null or engine_power='' or engine_power='None'  then 1 else 0 end) enginer_power,
sum(case when body_type is null or body_type='' or body_type='None'  then 1 else 0 end) body_type,
```

settings

maker	model	mileage	manufacture_year	engine_displacement	enginer_power
518915	1133361	362584	370578	743414	554877

For Spark:

SPARK Question 2

```
%spark.sql
Select
sum(case when maker is null or maker='' or maker='None' then 1 else 0 end) maker,
sum(case when model is null or model='' or model='None'  then 1 else 0 end) model,
sum(case when mileage is null or mileage='' or mileage='None'  then 1 else 0 end) mileage,
sum(case when manufacture_year is null or manufacture_year='' or manufacture_year='None'  then 1 else 0 end) manufacture_year,
sum(case when engine_displacement is null or engine_displacement='' or engine_displacement='None'  then 1 else 0 end) engine_displacement,
sum(case when engine_power is null or engine_power='' or engine_power='None'  then 1 else 0 end) enginer_power,
sum(case when body_type is null or body_type='' or body_type='None'  then 1 else 0 end) body_type,
sum(case when color_slug is null or body_type='' or body_type='None'  then 1 else 0 end) color_slug,
sum(case when stk_year is null or stk_year='' or stk_year='None'  then 1 else 0 end) stk_year,
sum(case when transmission is null or transmission='' or transmission='None'  then 1 else 0 end) transmission,
sum(case when door_count is null or door_count='' or door_count='None'  then 1 else 0 end) door_count,
sum(case when seat_count is null or seat_count='' or seat_count='None'  then 1 else 0 end) seat_count,
sum(case when fuel_type is null or fuel_type='' or fuel_type='None'  then 1 else 0 end) fuel_type,
sum(case when date_created is null or date_created='' or date_created='None'  then 1 else 0 end) date_created,
sum(case when date_last_seen is null or date_last_seen='' or date_last_seen='None'   then 1 else 0 end) date_last_seen,
sum(case when price_eur is null or price_eur='' or price_eur='None'  then 1 else 0 end) price_eur
from cars_db_spark.used_cars
```

settings

maker	model	mileage	manufacture_year	engine_displacement	enginer_power
518915	1133361	362584	370578	743414	554877

Final answer from both Hive and Spark:

```
Name of Columns         Total Number rows with Null or none or '' values

maker                   =518915
model                   =1133361
mileage                 =362584
manufacture_year        =370578
engine_displacement     =743414
engine_power            =554877
body_type               =1122914
color_slug              =3343411
stk_year                =3016807
transmission            =741630
door_count              =1090066
seat_count              =1287099
fuel_type               =1847606
date_created            =0
date_last_seen          =0
price_eur               =0
```

Question 3: Write Hive and SparkSQL queries to create a new table called clean_used_cars from used_cars with the following conditions:

- **Drop the columns with more than 50% missing values (NULL or empty) (you do not need to automate this step, you can find them, then exclude them while creating the clean dataset)**
- **The manufacture year between 2000 and 2017 including 2000 and 2017**
- **Both maker and model should exist in the row**
- **The price range is from 3000 to 2000,000 (3000 ≤ price ≤ 2000,000)**

For Hive:

Question 3: Part 1/2 Drop the columns with more than 50% missing values (NULL or empty)

```
%Hive
Select
sum(case when maker is null or maker='' or maker='None' then 1 else 0 end)*100/count(*) maker,
sum(case when model is null or model='' or model='None'  then 1 else 0 end)*100/count(*) model,
sum(case when mileage is null or mileage='' or mileage='None'  then 1 else 0 end)*100/count(*) mileage,
sum(case when manufacture_year is null or manufacture_year='' or manufacture_year='None'  then 1 else 0 end)*100/count(*) manufacture_year,
sum(case when engine_displacement is null or engine_displacement='' or engine_displacement='None'  then 1 else 0 end)*100/count(*) engine_displacement,
sum(case when engine_power is null or engine_power='' or engine_power='None'  then 1 else 0 end)*100/count(*) enginer_power,
sum(case when body_type is null or body_type='' or body_type='None'  then 1 else 0 end)*100/count(*) body_type,
sum(case when color_slug is null or color_slug='' or color_slug='None'  then 1 else 0 end)*100/count(*) color_slug,
sum(case when stk_year is null or stk_year='' or stk_year='None'  then 1 else 0 end)*100/count(*) stk_year,
sum(case when transmission is null or transmission='' or transmission='None'  then 1 else 0 end)*100/count(*) transmission,
sum(case when door_count is null or door_count='' or door_count='None'  then 1 else 0 end)*100/count(*) door_count,
sum(case when seat_count is null or seat_count='' or seat_count='None'  then 1 else 0 end)*100/count(*) seat_count,
sum(case when fuel_type is null or fuel_type='' or fuel_type='None'  then 1 else 0 end)*100/count(*) fuel_type,
sum(case when date_created is null or date_created='' or date_created='None'  then 1 else 0 end)*100/count(*) date_created,
sum(case when date_last_seen is null or date_last_seen='' or date_last_seen='None'  then 1 else 0 end)*100/count(*) date_last_seen,
sum(case when price_eur is null or price_eur='' or price_eur='None'  then 1 else 0 end)*100/count(*) price_eur
from cars_db_hive.used_cars
sum(case when manufacture_year is null or manufacture_year='' or manufacture_year='None'  then 1 else 0 end)*100/count(*) manufacture_year,
sum(case when engine_displacement is null or engine_displacement='' or engine_displacement='None'  then 1 else 0 end)*100/count(*) engine_displacement,
sum(case when engine_power is null or engine_power='' or engine_power='None'  then 1 else 0 end)*100/count(*) enginer_power,
sum(case when body_type is null or body_type='' or body_type='None'  then 1 else 0 end)*100/count(*) body_type,
sum(case when color_slug is null or color_slug='' or color_slug='None'  then 1 else 0 end)*100/count(*) color_slug,
```

settings

maker	model	mileage	manufacture_year	engine_displacement	enginer_power	body_type
14.605343447853478	31.899495399829775	10.205262612752582	10.430261149164403	20.924075800357567	15.617527256515219	31.60545490

Took 26 sec. Last updated by anonymous at February 20 2022, 2:12:54 AM. (outdated)

As per above result, there are total 3 columns which have more than 50% null/none values. Therefore, in part 2, schema table is created without those columns. Then, data is transferred to clean_used_cars table from used_cars table.

<u>Name of the columns which have more than 50% null/none value:</u>

1: "color_slug" =94.10%

2: "stk_year =84.91%

3: "fuel_type" = 52%"

Question 3: Part 2/2 Making clean table with mentioned requirements: The manufacture year between 2000 and 2017 including 2000 and 2017 Both maker and model should exist in the row The price range is from 3000 to 2000,000 (3000 ≤ price ≤ 2000,000)

FINISHED

```
%Hive
create table if not exists cars_db_hive.clean_used_cars
(maker string, model string, mileage int, manufacture_year int, engine_displacement int, engine_power int, body_type string, transmission string, door_count int, seat_count int, date_created string, date_last_seen
    string, price_eur float)
row format delimited fields terminated by ','
lines terminated by '\n'
stored as TEXTFILE tblproperties("skip.header.line.count"="1");
```

Query executed successfully. Affected rows : -1

Took 0 sec. Last updated by anonymous at February 20 2022, 2:48:15 AM.

FINISHED

```
%Hive
from cars_db_hive.used_cars insert into table cars_db_hive.clean_used_cars select maker, model, mileage, manufacture_year, engine_displacement, engine_power, body_type,transmission,
door_count, seat_count, date_created, date_last_seen, price_eur where (price_eur BETWEEN 3000 AND 2000000) AND (manufacture_year BETWEEN 2000 and 2017) AND (maker != '' and model != '');
```

```
INFO  : Completed compiling command(queryId=hive_20220220074820_056f33db-84c5-46bd-a31b-b857a15c9f9f); Time taken: 0.343 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=hive_20220220074820_056f33db-84c5-46bd-a31b-b857a15c9f9f):
from cars_db_hive.used_cars insert into table cars_db_hive.clean_used_cars select maker, model, mileage, manufacture_year, engine_displacement, engine_power, body_type,transmission,
door_count, seat_count, date_created, date_last_seen, price_eur where (price_eur BETWEEN 3000 AND 2000000) AND (manufacture_year BETWEEN 2000 and 2017) AND (maker != '' and model != '')
```

Query executed successfully. Affected rows : -1

Took 27 sec. Last updated by anonymous at February 20 2022, 2:48:47 AM.

For Spark:

Spark Question 3(Part 1/2)

SPARK JOB FINISHED

```
%spark.sql
Select
sum(case when maker is null or maker='' or maker='None' then 1 else 0 end)*100/count(*) maker,
sum(case when model is null or model='' or model='None'  then 1 else 0 end)*100/count(*) model,
sum(case when mileage is null or mileage='' or mileage='None'  then 1 else 0 end)*100/count(*) mileage,
sum(case when manufacture_year is null or manufacture_year='' or manufacture_year='None'  then 1 else 0 end)*100/count(*) manufacture_year,
sum(case when engine_displacement is null or engine_displacement='' or engine_displacement='None'  then 1 else 0 end)*100/count(*) engine_displacement,
sum(case when engine_power is null or engine_power='' or engine_power='None'  then 1 else 0 end)*100/count(*) enginer_power,
sum(case when body_type is null or body_type='' or body_type='None'  then 1 else 0 end)*100/count(*) body_type,
sum(case when color_slug is null or color_slug='' or color_slug='None'  then 1 else 0 end)*100/count(*) color_slug,
sum(case when stk_year is null or stk_year='' or stk_year='None'  then 1 else 0 end)*100/count(*) stk_year,
sum(case when transmission is null or transmission='' or transmission='None'  then 1 else 0 end)*100/count(*) transmission,
sum(case when door_count is null or door_count='' or door_count='None'  then 1 else 0 end)*100/count(*) door_count,
sum(case when seat_count is null or seat_count='' or seat_count='None'  then 1 else 0 end)*100/count(*) seat_count,
sum(case when fuel_type is null or fuel_type='' or fuel_type='None'  then 1 else 0 end)*100/count(*) fuel_type,
sum(case when date_created is null or date_created='' or date_created='None'  then 1 else 0 end)*100/count(*) date_created,
sum(case when date_last_seen is null or date_last_seen='' or date_last_seen='None'  then 1 else 0 end)*100/count(*) date_last_seen,
sum(case when price_eur is null or price_eur='' or price_eur='None'  then 1 else 0 end)*100/count(*) price_eur
from cars_db_spark.used_cars
```

settings

maker	model	mileage	manufacture_year	engine_displacement	enginer_power	body_type	color_slug	stk_year	tran
14.605343447853478	31.899495399829775	10.205262612752582	10.430261149164403	20.924075800357567	15.617527256515219	31.605454905722404	94.10340025308817	84.91082807567426	20.873

Took 6 sec. Last updated by anonymous at February 20 2022, 3:46:21 AM.

As per above result, there are total 3 columns which have more than 50% null/none values. Therefore, in part 2, clean_used_cars table is created by removing those columns from used_cars table.

<u>Name of the columns which have more than 50% null/none value:</u>

1: "color_slug" =94.10%
2: "stk_year =84.91%
3: "fuel_type" = 52%"

SPARK Question 3 (Part 2/2)

FINISHED

```
%spark.sql
Create table if not exists cars_db_spark.clean_used_cars AS
(SELECT maker,model,mileage,manufacture_year,engine_displacement,engine_power,body_type,transmission,door_count,seat_count,date_created,date_last_seen,price_eur
FROM cars_db_spark.used_cars WHERE (price_eur BETWEEN 3000 AND 2000000) AND manufacture_year BETWEEN 2000 AND 2017 AND maker!='' AND model!='')
```

Took 0 sec. Last updated by anonymous at February 20 2022, 3:47:09 AM.

Question 4: Write Hive and SparkSQL queries to find how many records remained clean_used_cars

For Hive and Spark, result is same as per below. There is total 1322853 records in clean_used_cars table.

Question 4: Write Hive and SparkSQL queries to find how many records remained clean_used_cars

```
%Hive
select count(*) from cars_db_hive.clean_used_cars;
```

settings

_c0
1322853

Took 0 sec. Last updated by anonymous at February 20 2022, 2:49:06 AM.

SPARK Question 4

```
%spark.sql
select count(*) from cars_db_spark.clean_used_cars
```

settings

count(1)
1322853

Question 5: Write Hive and SparkSQL queries to find the make and model for the cars with the top 10 highest average price:

Here as per below screenshots, results are same for both Hive and Spark. Avg. of 'price_eur', groupby and orderby (in descending order) functions are used for the result. The range of result (average in EUR) is from 365960.99 to 105946.89.

For Hive:

Question 5: Write Hive and SparkSQL queries to find the make and model for the cars with the top 10 highest average price

```
%Hive
select maker, model, avg(price_eur) as average from cars_db_hive.clean_used_cars group by maker,model order by average desc limit 10;
```

```
es:null)
INFO  : Completed compiling command(queryId=hive_20220220075258_814aa38e-81b9-40c7-90d4-d5a15088c858); Time taken: 0.189 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=hive_20220220075258_814aa38e-81b9-40c7-90d4-d5a15088c858):
select maker, model, avg(price eur) as Average from cars db hive.clean used cars group by maker,model order by average desc limit 10
```

settings

maker	model	average
lamborghini	aventador	365960.994212963
porsche	carrera-gt	302045.21671102336
bmw	z8	245118.60092905405
tesla	roadster	192880.27864583334
tesla	model-x	176418.31510416666
bentley	brooklands	138501.303125
rolls-royce	wraith	137663.46354166666
bentley	continental-gtc	129138.87838541667
bmw	i8	112273.42648466506
bentley	continental-gt	105946.8866799462

For Spark:

SPARK Question 5

```
%spark.sql
select maker, model, avg(price_eur) as average from cars_db_spark.clean_used_cars group by maker,model order by average desc limit 10;
```

settings

maker	model	average
lamborghini	aventador	365960.99518518517
porsche	carrera-gt	302045.21664835163
bmw	z8	245118.60081081084
tesla	roadster	192880.28
tesla	model-x	176418.31999999998
bentley	brooklands	138501.302
rolls-royce	wraith	137663.46666666667
bentley	continental-gtc	129138.87816666665
bmw	i8	112273.42633663365
bentley	continental-gt	105946.88678807947

Question 6: Write Hive and SparkSQL queries to find the make and model for the cars with the top 10 lowest average price

Here as per below screenshots, results are same for both Hive and Spark. Avg. of ‘price_eur’, groupby and orderby (in ascending order) functions are used for the result. The range of result (average in EUR) is from 3071.80 to 3498.66.

For Hive:

Question 6: Write Hive and SparkSQL queries to find the make and model for the cars with the top 10 lowest average price

```
%Hive
select maker, model, avg(price_eur) as average from cars_db_hive.clean_used_cars Group by maker, model Order by average LIMIT 10
```

```
INFO  : Compiling command(queryId=hive_20220220075409_965b17ac-658d-4593-9ef0-93506e063310):
select maker, model, avg(price_eur) as Average from cars_db_hive.clean_used_cars Group by maker, model Order by Average LIMIT 10
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:maker, type:string, comment:null), FieldSchema(name:model, type:string, comment:null), FieldSchema(name:
```

settings

maker	model	average
skoda	galaxy	3071.800048828125
rover	streetwise	3187.8466796875
kia	retona	3200.2542898995534
bmw	transit	3290.159912109375
chevrolet	alero	3305.3701171875
hyundai	santamo	3391.010009765625
opel	kadett	3405.47998046875
fiat	128	3460.39990234375
nissan	frontier	3478.905029296875
seat	inca	3498.6566569010415

Took 20 sec. Last updated by anonymous at February 20 2022, 2:54:29 AM. (outdated)

For Spark:

SPARK Question 6

```
%spark.sql
select maker, model, avg(price_eur) as average from cars_db_spark.clean_used_cars Group by maker, model Order by average LIMIT 10
```

settings

maker	model	average
skoda	galaxy	3071.8
rover	streetwise	3187.8466666666664
kia	retona	3200.2542857142857
bmw	transit	3290.16
chevrolet	alero	3305.37
hyundai	santamo	3391.01
opel	kadett	3405.48
fiat	128	3460.3999999999996
nissan	frontier	3478.9049999999997
seat	inca	3498.6566666666663

Took 2 sec. Last updated by anonymous at February 20 2022, 3:47:51 AM. (outdated)

Question 7: Write Hive and SparkSQL query to recommend top five make and model for Economic segment customers (Top five manufacturers in the 3000 to 20,000 price range;3000≤price<20,000) - based on the top average price

Here, Hive and Spark queries are designed to get cars recommendation list for economic segment customers. The recommended results are based on grouping the maker and model in range of 3000 to 20000 price category. The top 5 range of the result (avg. price in EUR) is from 3071.8 to 3305.37 for economic segment customers. Results from both Hive and Spark queries are same.

For Hive:

Question 7: Write Hive and SparkSQL query to recommend top five make and model for Economic segment customers (Top five manufacturers in the 3000 to 20,000 price range;3000≤price<20,000) - based on the top average price

```
%Hive
select maker,model, avg(price_eur) as average from cars_db_hive.clean_used_cars Group by maker, model HAVING average BETWEEN 3000 AND 20000 Order by average LIMIT 5
```

```
INFO  : Total jobs = 1
INFO  : Launching Job 1 out of 1
INFO  : Starting task [Stage-1:MAPRED] in serial mode
INFO  : Subscribed to counters: [] for queryId: hive_20220220082152_f1ef92c0-9878-4542-98cd-ed9fd6e8a8e4
INFO  : Tez session hasn't been created yet. Opening session
```

settings

maker	model	average
skoda	galaxy	3071.800048828125
rover	streetwise	3187.8466796875
kia	retona	3200.2542898995534
bmw	transit	3290.159912109375
chevrolet	alero	3305.3701171875

Took 19 sec. Last updated by anonymous at February 20 2022, 3:22:11 AM. (outdated)

For Spark:

SPARK Question 7

```
%spark.sql
select maker,model, avg(price_eur) as average from cars_db_spark.clean_used_cars Group by maker, model HAVING average BETWEEN 3000 AND 20000 Order by average LIMIT 5
```

settings

maker	model	average
skoda	galaxy	3071.8
rover	streetwise	3187.8466666666664
kia	retona	3200.2542857142857
bmw	transit	3290.16
chevrolet	alero	3305.37

Took 2 sec. Last updated by anonymous at February 20 2022, 3:48:12 AM.

Question 8: Write Hive and SparkSQL queries to recommend the top five make and models for Intermediate segment customers (Top five manufacturers in the 20,000 to 300,000 price range; 20,000≤price<300,000) - based on the top average price

Here, Hive and Spark queries are designed to get cars recommendation list for intermediate segment customers. The recommended results are based on grouping the maker and model in range of 20,000 to 300,000 price category. The top 5 range of the result (avg. price in EUR) is from 20,045.61 to 20,518.04 for Intermediate segment customers. Results from both Hive and Spark queries are same.

For Hive:

Question 8: Write Hive and SparkSQL queries to recommend the top five make and models for Intermediate segment customers (Top five manufacturers in the 20,000 to 300,000 price range; 20,000≤price<300,000) - based on the top average price

```
%Hive
select maker,model, avg(price_eur) as average from cars_db_hive.clean_used_cars group by maker,model having average between 20000 and 300000 order by average limit 5;
```

```
INFO  : Compiling command(queryId=hive_20220220082359_98d86010-99cf-4903-95c1-ea6804710f37):
select maker,model, avg(price_eur) as average from cars_db_hive.clean_used_cars group by maker,model having average between 20000 and 300000 order by average limit 5
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:maker, type:string, comment:null), FieldSchema(name:model, type:string, comment:null), FieldSchema(name:ave
```

settings

maker	model	average
fiat	500x	20045.61346222573
volvo	960	20137.6201171875
fiat	freemont	20305.12878715386
mercedes-benz	126	20471.9458984375
mazda	cx-9	20518.044734954834

Took 23 sec. Last updated by anonymous at February 20 2022, 3:24:22 AM. (outdated)

For Spark:

SPARK Question 8

```
%spark.sql
select maker, model, avg(price_eur) as average from cars_db_spark.clean_used_cars Group by maker, model HAVING average BETWEEN 20000 and 300000 Order by average LIMIT 5
```

settings

maker	model	average
fiat	500x	20045.613449645403
volvo	960	20137.62
fiat	freemont	20305.12879237289
mercedes-benz	126	20471.946000000004
mazda	cx-9	20518.0446875

Took 2 sec. Last updated by anonymous at February 20 2022, 3:48:21 AM.

Question 9: Write Hive and SparkSQL queries to recommend the top five make and models for the Luxury segment customers (Top five manufacturers in the 300,000 to 2000,000 price range; 300,000≤price<2000,000) - based on the top average price

For Hive and Spark:

Here, Hive and Spark queries are designed to get cars recommendation list for Luxury segment customers. The recommended results are based on grouping the maker and model in range of 300,000 TO 2000,000 price category. We are getting only two cars for the recommendation for luxury segment customers. Results from both Hive and Spark queries are the same as per below.

Question 9: Write Hive and SparkSQL queries to recommend the top five make and models for the Luxury segment customers (Top five manufacturers in the 300,000 to 2000,000 price range; 300,000≤price<2000,000) - based on the top average price

```
%Hive
select maker,model, avg(price_eur) as average from cars_db_hive.clean_used_cars group by maker,model having average between 300000 and 2000000 order by average limit 5;
```

settings

maker	model	average
porsche	carrera-gt	302045.21671102336
lamborghini	aventador	365960.994212963

Took 23 sec. Last updated by anonymous at February 20 2022, 3:27:20 AM. (outdated)

SPARK Question 9

```
%spark.sql
select maker,model, avg(price_eur) as average from cars_db_spark.clean_used_cars Group by maker, model HAVING average BETWEEN 300000 AND 2000000 Order by average LIMIT 5
```

settings

maker	model	average
porsche	carrera-gt	302045.21664835163
lamborghini	aventador	365960.99518518517

Took 1 sec. Last updated by anonymous at February 20 2022, 3:48:43 AM. (outdated)

Final Answers:

1. Number of records in used_cars table
   3552912
2. Number of records in clean_used_cars table
   1322853
3. Make and model for the cars with the top 10 highest average prices and their average price

maker	model	average

lamborghini	aventador	365960.995
porsche	carrera-gt	302045.22
bmw	z8	245118.60
tesla	roadster	192880.28
tesla	model-x	176418.32
bentley	brooklands	138501.30
rolls-royce	wraith	137663.47
bentley	continental-gtc	129138.88
bmw	i8	112273.43
bentley	continental-gt	105946.89

4. Make and model for the cars with the top 10 lowest average prices and their average price

maker	model	average
skoda	galaxy	3071.8
rover	streetwise	3187.847
kia	retona	3200.254
bmw	transit	3290.16
chevrolet	alero	3305.37
hyundai	santamo	3391.01
opel	kadett	3405.48
fiat	128	3460.4
nissan	frontier	3478.905
seat	inca	3498.657

5. Write the name of the top five make and models for Economic segment customers

maker	model
skoda	galaxy

rover	streetwise
kia	retona
bmw	transit
chevrolet	alero

6. Write the name of the top five make and models for Intermediate segment customers

maker	model
fiat	500x
volvo	960
fiat	freemont
mercedes-benz	126
mazda	cx-9

7. Write the name of the top five make and models for Luxury segment customers

maker	model
porsche	carrera-gt
lamborghini	aventador