

Assignment 3

499100

1- Download the dataset from http://18.206.158.228/dataset/used_cars.csv into your home directory.

```
Last login: Fri Nov 12 22:30:14 2021 from 128.252.229.24
#####
#           Welcome to Big Data Server           #
#   You are now Connected to the server           #
#####
[baoxin.l@ip-172-31-95-86 ~]$ wget http://18.206.158.228/dataset/used_cars.csv
--2021-11-18 22:42:55--  http://18.206.158.228/dataset/used_cars.csv
Connecting to 18.206.158.228:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 69145610 (66M) [text/csv]
Saving to: 'used_cars.csv'

100%[=====] 69,145,610  260MB/s  in 0.3s

2021-11-18 22:42:55 (260 MB/s) - 'used_cars.csv' saved [69145610/69145610]
```

2- Using one line bash command print the number of columns.

```
[baoxin.l@ip-172-31-95-86 ~]$ head -1 used_cars.csv | tr "," "\n" | wc -l
23
```

3- Using the Linux commands, what are the three most frequent exterior colors? Did you expect these?

```
[baoxin.l@ip-172-31-95-86 ~]$ cut -d ',' -f 7 used_cars.csv | sort| uniq -c| sort -rn | head -3
1304 Black
840 White
481 Gray
```

4- Using the Linux commands, find the top 10 cars (combination of maker and model: like Ford F-150) that has “keyless entry”. Remember that they may have written “keyless entry” with either capital or small letters.

```
[baoxin.l@ip-172-31-95-86 ~]$ grep -i 'keyless entry' used_cars.csv | cut -d ',' -f 15,18 | sort | uniq -c | sort -rn | head -10
642 Jeep,Grand Cherokee
546 Ford,Fusion
457 Ford,F-150
340 Jeep,Compass
311 Ford,Escape
303 BMW,3 Series
295 Ford,Explorer
259 Chevrolet,Equinox
252 Hyundai,Tucson
245 Mazda,CX-5
```

5- Using the Linux commands, maximum mileage of Toyota SUVs. Make sure to keep track of words independent of capital/small letters.

```
[baoxin.l@ip-172-31-95-86 ~]$ grep -i 'Toyota' used_cars.csv | grep -i 'SUV' | cut -d ',' -f 17 | sort -rn | head -10
261711
227763
221396
216954
215251
206996
204124
197644
192402
190548
```

6- Find the average mileage per years that where at least more than 1000 cars are listed in those year. Sort the results to see the top average mileages first.

Impala

Add a name... Add a description...

1.18s Database default Type text

```

1 select year, avg(mileage), count(vin)
2 from used_cars
3 group by year
4 having count(vin) > 1000
5 order by avg(mileage) desc;
6

```

Query 634b3c2542dc6c6b:b4acd04500000000: 0% Complete (0/1)
 Query 634b3c2542dc6c6b:b4acd04500000000 100% Complete (1/1)

Query History Saved Queries Results (5)

| | year | avg(mileage) | count(vin) |
|---|------|--------------------|------------|
| 1 | 2017 | 34770.17920560748 | 4293 |
| 2 | 2018 | 28955.88382687927 | 1766 |
| 3 | 2019 | 19478.830746489282 | 1378 |
| 4 | 2020 | 662.3580426356589 | 6651 |
| 5 | 2021 | 10.422043010752688 | 1138 |

7- Find the average price and average mileage of cars per accident and salvage status.

1.22s Database default ▾ Type text ▾ ⚙ ?

```

1 select has_accidents, salvage, avg(price), avg(mileage)
2 from used_cars
3 group by has_accidents, salvage;
4

```

Query 1242ad7f2e5079c6:08a9c75a00000000: 0% Complete (0 out of 1)
 Query 1242ad7f2e5079c6:08a9c75a00000000 100% Complete (1 out of 1)
 Query 1242ad7f2e5079c6:08a9c75a00000000 100% Complete (1 out of 1)

Query History Saved Queries Results (5)

| | has_accidents | salvage | avg(price) | avg(mileage) |
|---|---------------|---------|--------------------|-------------------|
| 1 | | | 39698.9290276792 | 98.21245643279285 |
| 2 | FALSE | FALSE | 24948.56933346819 | 50084.37570188872 |
| 3 | TRUE | TRUE | 15030 | 94499.03703703704 |
| 4 | TRUE | FALSE | 17694.018842975205 | 75632.74461742298 |
| 5 | FALSE | TRUE | 13748.2 | 74094.4 |

8- First the total car per car maker where the average price of cars by the car maker is less than the average price of all cars without accident (accident as FALSE). Select the top 10 makes with the most cars. Hint: You may not be able to run it on both of Impala and Hive. Try that and let us know which one doesn't work.

Hive worked, Impala doesn't because it doesn't support having clause.

5.95s

Database default ▾

Type text ▾

```
1 select make_name, count(vin), avg(price)
2 from used_cars
3 group by make_name
4 having avg(price) < (select avg(price) from used_cars
5 where has_accidents = "FALSE")
6 order by count(vin) desc
7 limit 10;|
8
```



```
INFO : RAW_INPUT_SPLITS_Map_4: 1
INFO : savedToCache: true
INFO : Completed executing command(queryId=hive_20211118233155_e257b631-b644-49b0-bdde-60f96a7bb747); Time taken: 5.54 seconds
INFO : OK
```

Query History

Saved Queries

Results (10)


| | make_name | _c1 | _c2 |
|---|-----------|------|--------------------|
| 1 | Hyundai | 1371 | 22205.718453683443 |
| 2 | Toyota | 1332 | 23582.46921921922 |
| 3 | Chevrolet | 1239 | 22369.28410008071 |
| 4 | Nissan | 1027 | 16420.666017526775 |
| 5 | Honda | 917 | 16887.214830970555 |
| 6 | Kia | 897 | 20497.920847268673 |

Query History

Saved Queries


Results (10)

| | make_name | _c1 | _c2 |
|----|------------|------|--------------------|
| 1 | Hyundai | 1371 | 22205.718453683443 |
| 2 | Toyota | 1332 | 23582.46921921922 |
| 3 | Chevrolet | 1239 | 22369.28410008071 |
| 4 | Nissan | 1027 | 16420.666017526775 |
| 5 | Honda | 917 | 16887.214830970555 |
| 6 | Kia | 897 | 20497.920847268673 |
| 7 | Volkswagen | 556 | 23553.766187050358 |
| 8 | Mazda | 499 | 22642.875751503005 |
| 9 | Subaru | 412 | 19162.04126213592 |
| 10 | Chrysler | 252 | 22941.464285714286 |



Add a name...
Add a description...





0s Database default ▾ Type text ▾ ⚙️ ?

```

1 select make_name, count(vin), avg(price)
2 from used_cars
3 group by make_name
4 having avg(price) < (select avg(price) from used_cars
5 where has_accidents = "FALSE")
6 order by count(vin) desc
7 limit 10;
8

```

AnalysisException: Subqueries are not supported in the HAVING clause.

9- For each car maker find the average price and average days the car is on the market. List the worst 10 brands based on the average days on the market.

```

1 select make_name, avg(daysonmarket), avg(price)
2 from used_cars
3 group by make_name
4 order by avg(daysonmarket) desc
5 limit 10;
6
7

```

Query 6342331d3c3012b2:082617ad00000000: 0% Complete (6342331d3c3012b2:082617ad00000000)
Query 6342331d3c3012b2:082617ad00000000 100% Complete (1 out of 1)

Query History
Saved Queries
Results (10)

| | make_name | avg(daysonmarket) | avg(price) |
|----|--------------|--------------------|--------------------|
| 1 | McLaren | 740.5 | 122221 |
| 2 | Spyker | 463 | 305500 |
| 3 | Aston Martin | 291 | 89668 |
| 4 | Rolls-Royce | 272.64285714285717 | 216060.2142857143 |
| 5 | Mitsubishi | 212.69014084507043 | 13649.56338028169 |
| 6 | Ferrari | 208.75 | 434877.25 |
| 7 | Genesis | 185.0327868852459 | 42795.147540983606 |
| 8 | Bentley | 149.2 | 125571.25 |
| 9 | Alfa Romeo | 111.67521367521367 | 44807.470085470086 |
| 10 | Porsche | 100.34482758620689 | 76359.13793103448 |

10- Find the average price per maximum seat of SUV/Crossovers. List only valid seats (no null or incorrect value). Sort the results by average price.

1.18s Database default ▾ Type text ▾ ⚙ ?

```
1 select maximum_seating, avg(price)
2 from used_cars
3 where body_type="SUV / Crossover"
4 group by maximum_seating
5 having maximum_seating like "% seats"
6 order by avg(price);
7
8
```



Query 3c4d5383556c9708:4958b77100000000: 0% Complete (0 sec of 1)
Query 3c4d5383556c9708:4958b77100000000 100% Complete (1 out of 1)

Query History Saved Queries Results (6)

| | maximum_seating | avg(price) |
|---|-----------------|--------------------|
| 1 | 9 seats | 22537.363636363636 |
| 2 | 4 seats | 24381.875 |
| 3 | 5 seats | 29079.588467153284 |
| 4 | 7 seats | 33725.90569020021 |
| 5 | 8 seats | 36947.1986013986 |
| 6 | 6 seats | 46572.09090909091 |