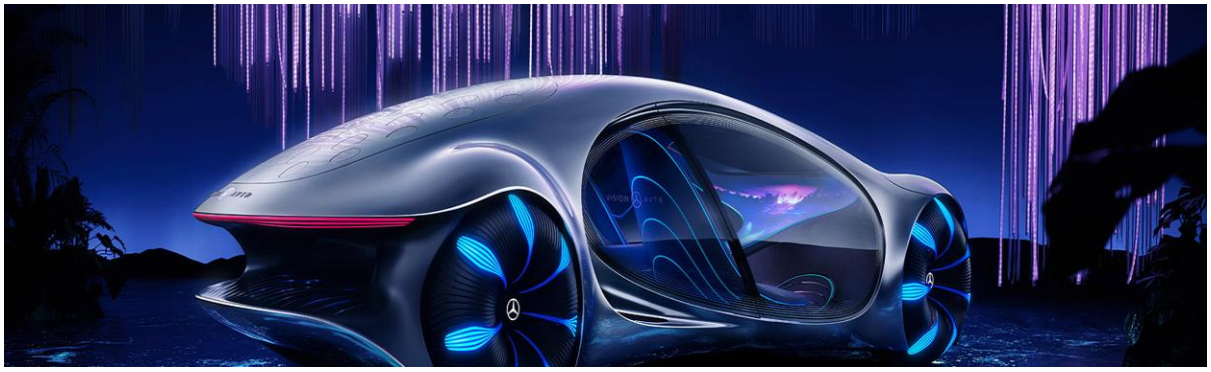
Big-Data Architecture and Governance

Used Cars Price Analysis

Us (United States) Used Cars Dataset

Group 4: Surbhi Wagh, Rohit Singh, Sumit Patil



Project Overview:

Due to increasing demand in cars by the people for commuting in US there are group of people who are even interested in purchasing used cars for their use.

In this case it's necessary to analyze the condition brand type model and so on for the car and check the price of cars as per purchaser needs.

This project will help to get an idea of the price of the used cars in United states

Objective and Tools used:

Objective:

The goal of this project is to find patterns in the data and build a dashboard that help a customer to get an idea regarding the price of the used car according to his requirement and colors and as per his location to get idea about what's the used card available around his area.

Tools Used:

Data Extraction, Profiling, Cleaning, Visualization: Python(Jupyter notebook)

Graph Creation: Arrow. App

Graph Database Management System: Neo4j

Project, Resource and Risk Management: Velero ETP

About this file

vin:	Vehicle Identification Number is a unique encoded string for every vehicle.
back_legroom:	Legroom in the rear seat.
bed:	Category of bed size(open cargo area) in pickup truck. Null usually means the vehicle isn't a pickup truck
bed_height:	Height of bed in inches
bed_length:	Length of Bed in inches
body_type :	Body Type of the vehicle
cabin:	Category of cabin size
city :	City where the car is listed
city_fuel_economy:	Fuel economy in city traffic in km per litre
combine_fuel_economy :	Combined fuel economy is a weighted average of City and Highway fuel economy in km per litre
daysonmarket:	Days since the vehicle was first listed on the website
dealer_zip:	Zipcode of the dealer
description :	Vehicle description on the vehicle's listing page
engine_cylinders :	Engine configuration.
engine_displacement :	Measure of the cylinder volume

engine_type :	Type of Engine
exterior_color :	Exterior color of the vehicle
fleet :	Whether the vehicle was previously part of a fleet
frame_damaged :	Whether the vehicle has a damaged frame.
franchise_dealer :	Whether the dealer is a franchise dealer.
franchise_make :	Company that owns the franchise
front_legroom :	The legroom in inches for the passenger seat
fuel_tank_volume :	Fuel tank's filling capacity in gallons
fuel_type :	Dominant type of fuel ingested by the vehicle.
has_accidents :	Whether the vin has any accidents registered
height :	Height of the vehicle in inches
highway_fuel_economy :	Fuel economy in highway traffic in km per litre
horsepower :	Horsepower is the power produced by an engine
interior_color :	Interior color of the vehicle
isCab :	Whether the vehicle was previously taxi/cab
is_certified :	Whether the vehicle is certified
is_cpo :	Certified vehicles warranty for free repairs flag
is_new :	Is the vehicle launched less than 2 years ago
is_oemcpo :	Pre-owned cars certified by the manufacturer.
latitude :	Latitude from the geolocation of the dealership
length :	Length of the vehicle in inches
listed_date :	Listing date of the Vehicle
listing_color :	Dominant color group from the exterior colo
listing_id :	Listing id from the website
longitude :	Longitude from the geolocation of the dealership.
main_picture_url :	Url of the picture of the used Vehicles
major_options :	Major options available for the vehice
make_name :	Brand of the vehicle
maximum_seating:	Maximum number of seats
mileage:	Mileage of the car
model_name :	Model name of the cars
owner_count :	No of previous owner
power :	power configuration of the car
price :	price of the car
salvage :	Whether car has been damaged or not
savings_amount :	amount you will save buying car
seller_rating :	Rating of the seller
sp_id :	Seller Id
sp_name :	Seller name
theft_title :	Whether the the vehicle has theft recovery
torque :	Torque of the vehicle
transmission :	Transmission type which is CVT or automatic
transmission_display :	Displaying the transmission type of the display
trimId :	Version of model with configuration
trim_name :	Name of the version
vehicle_damage_category :	Any damage to the vehicle

wheel_system :	Types of wheel system
wheel_system_display :	wheel drive systems
wheelbase :	wheel base of the vehicle in inches
width :	Width of the vehicle in inches
year :	The year on which the vehicle was made.

Risk & Issues of Project:

Type	Risk/Issue-Description	Mitigation
Issue	Improper Environment setup for Data cleansing and data validation i.e., insufficient packages to run python scripts	Make sure all packages are installed and proper environment is set up to run all scripts error for data cleansing/wrangling process
Risk	Many fields needed to track cars details has missing data which might be an analytical risk leading to biased visualization	Deriving new columns by understanding existing database attributes to generate reports without errors.
Risk	Inconsistent entries in data available for cars classification and thus analysis might be biased eventually which further leads to incomplete data merging or error in the creating graph database	Categorization of the range for available data records is to be done before loading data into the database.
Risk	Lack of Knowledge in terms of cars dataset among team members could lead to incorrect analysis and interpretation	Prior research/study of cars and features for understanding dataset

Data Profiling Instructions:

1) Download Python (follow instructions on https://www.python.org/downloads/ based on operating system
2) Install pandas library by running pip install pandas
3) Install pandas profiling to be able to create an html profile report by running the command pip install pandas-profiling
4) Run profiling.py script and output.html file will be created
5) Open output.html on browser to view data profile report

In [4]:
```
#importing all the libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import statistics
import pandas_profiling
```

In [5]:
```
df = pd.read_csv('used_cars_data.csv')
```

```
C:\Users\Surbhi\AppData\Local\Temp\ipykernel_14832\1206154635.py:1: DtypeWarning: Columns (11) have mixed types. Specify dtype
option on import or set low_memory=False.
  df = pd.read_csv('used_cars_data.csv')
```

In [6]:
```
#Reading the first 5 rows of the dataset
df.head()
```

Out[6]:

	vin	back_legroom	bed	bed_height	bed_length	body_type	cabin	city	city_fuel_economy	combine_fuel_economy	...	transmission
0	ZACNJABB5KPJ92081	35.1 in	NaN	NaN	NaN	SUV / Crossover	NaN	Bayamon	NaN	NaN	...	A
1	SALCJ2FX1LH858117	38.1 in	NaN	NaN	NaN	SUV / Crossover	NaN	San Juan	NaN	NaN	...	A

Data Wrangling/Cleansing Instructions

1) Make sure pandas library is installed (should have been installed during data profiling)
2) Run exploration.py script
3) Once script is finished running, “old-car-dataset.csv” file will be created
4) Use this new filtered dataset for database/visualization

In [15]:
```
df_select = df_select.drop('bed', axis=1)
df_select = df_select.drop('bed_height', axis=1)
df_select = df_select.drop('bed_length', axis=1)
df_select = df_select.drop('cabin', axis=1)
df_select = df_select.drop('combine_fuel_economy', axis=1)
df_select = df_select.drop('fleet', axis=1)
df_select = df_select.drop('frame_damaged', axis=1)
df_select = df_select.drop('has_accidents', axis=1)
df_select = df_select.drop('isCab', axis=1)
df_select = df_select.drop('owner_count', axis=1)
df_select = df_select.drop('salvage', axis=1)
df_select = df_select.drop('theft_title', axis=1)
df_select = df_select.drop('vehicle_damage_category', axis=1)
df_select = df_select.drop('is_certified', axis=1)
df_select = df_select.drop('is_oemcpo', axis=1)
df_select = df_select.drop('is_cpo', axis=1)
```

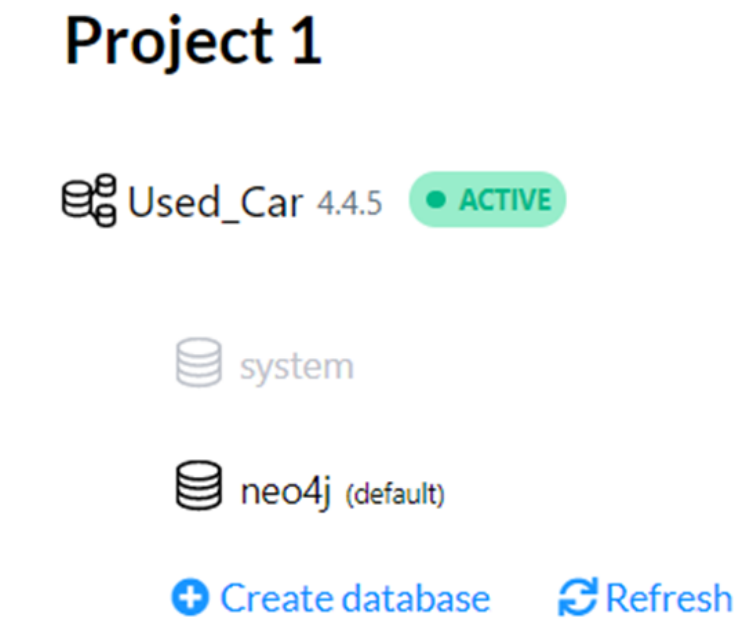
In [16]:
```
#Checking missing values again

percent_missing = df_select.isnull().sum() * 100 / len(df_select)
missing_value_df = pd.DataFrame({'column_name': df_select.columns,
                                 'percent_missing': percent_missing})

missing_value_df
```

Neo4j Instruction:

1. Create a new database in Neo4j
2. Copy the CSV file into the dbms import folder of the database created
3. Start the database and open the database



Constraint Creation:

```
CREATE CONSTRAINT ON (Oldcars:Oldcars) ASSERT Oldcars.oldcarid IS UNIQUE;
CREATE CONSTRAINT ON (city:city) ASSERT city.city IS UNIQUE;
CREATE CONSTRAINT ON (vin:vin) ASSERT vin.vin IS UNIQUE;
CREATE CONSTRAINT ON (daysonmarket: daysonmarket) ASSERT daysonmarket. daysonmarket IS UNIQUE;
CREATE CONSTRAINT ON maximum_seating: maximum_seating) ASSERT maximum_seating. maximum_seating IS UNIQUE;
```

Node Creation:

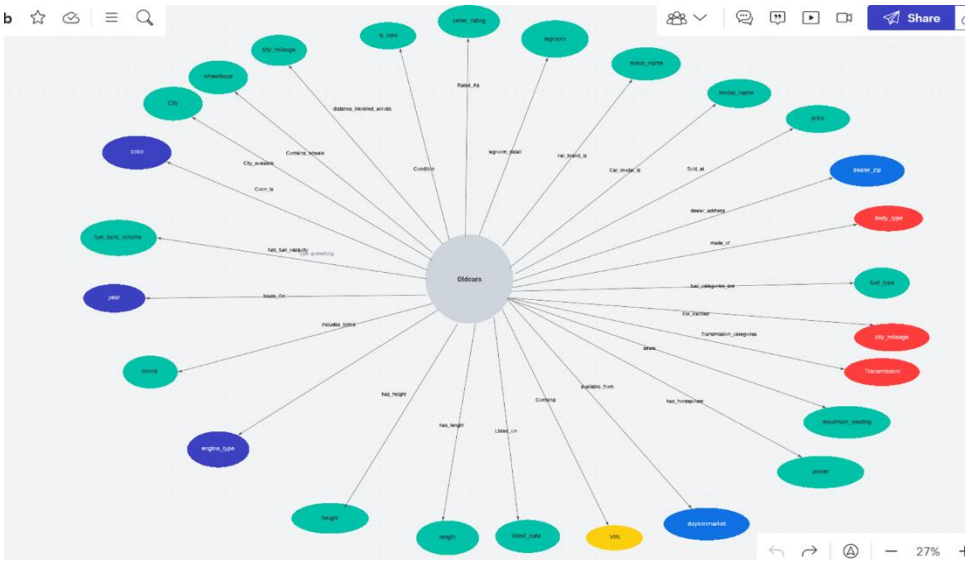
```
// Creating Maximum_seating//////
:auto USING PERIODIC COMMIT 500
LOAD CSV With HEADERS FROM 'file:///used_car_clean_CSV.csv' AS row
MERGE
 (maximum_seating: maximum_seating{ maximum_seating:row. maximum_seating});
```

Relationship Creation:

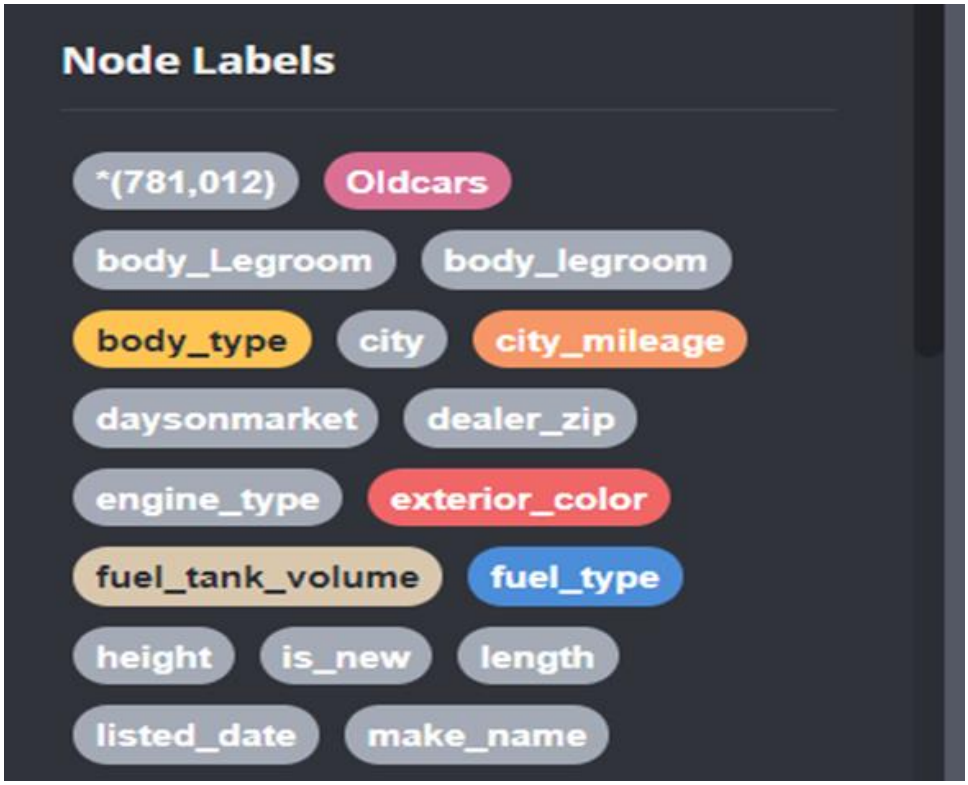
```
:auto USING PERIODIC COMMIT 500
LOAD CSV With HEADERS FROM 'file:///used_car_clean_CSV.csv' AS row
    MATCH (Oldcars:Oldcars { oldcarid:row.oldcarid})
    MATCH (engine_type: engine_type { engine_type:row. engine_type })
    MERGE (Oldcars)-[:type_of_engine]->( engine_type);
```

- Copy the cypher code(projectQuery_script.txt) to create the constraints, nodes and relationships between them
- Run the code to create the graph database
- The scripts will run one by one and will take a few minutes due to the large dataset size The graph database should be prepared and can be tested by queries

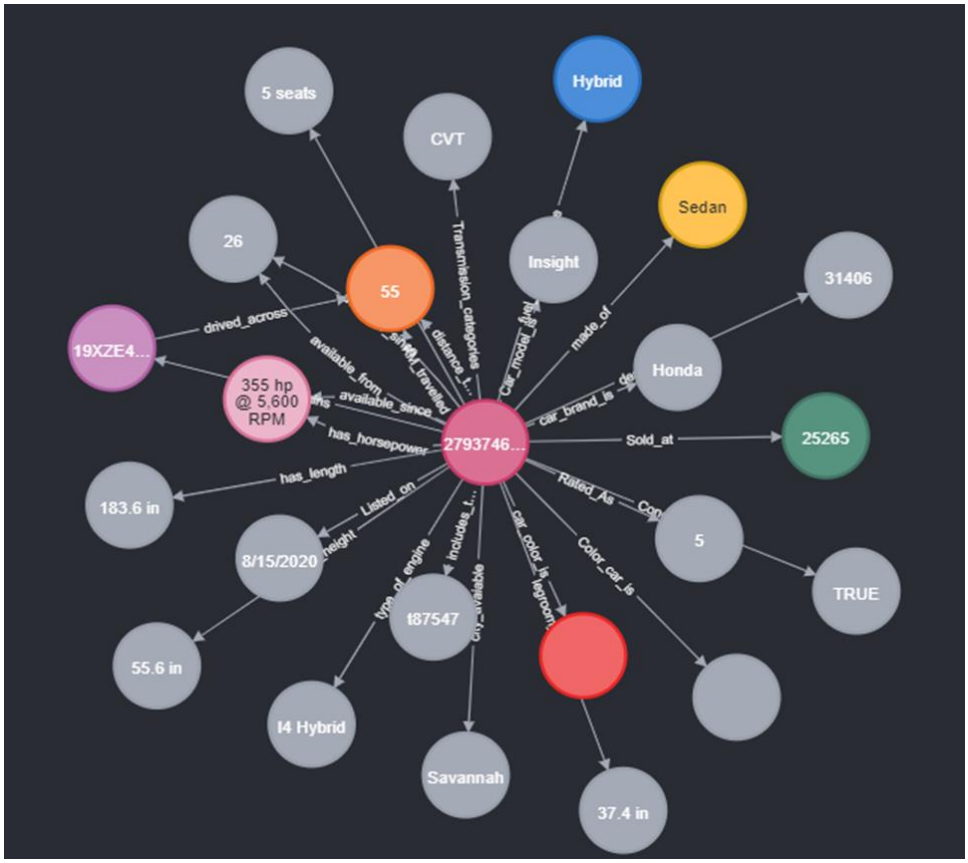
Graph Database:



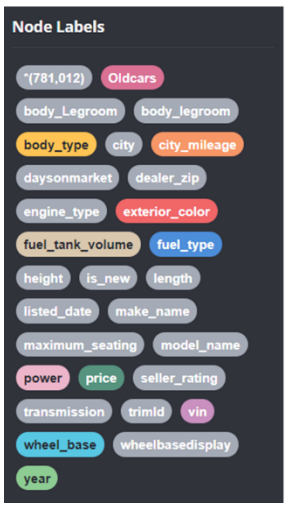
Neo4j nodes:

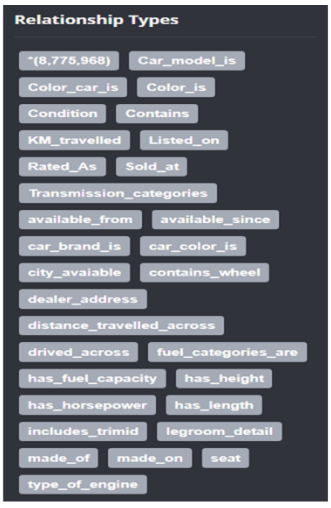

Node Labels
*(781,012)
Oldcars
body_Legroom
body_legroom
body_type
city
city_mileage
daysonmarket
dealer_zip
engine_type
exterior_color
fuel_tank_volume
fuel_type
height
is_new
length
listed_date
make_name

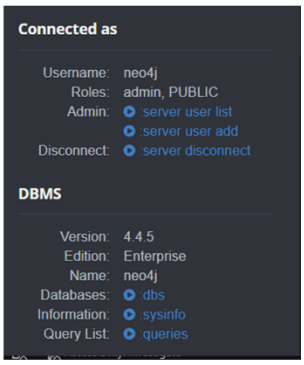

Relationship Types
*(8,775,968)
Car_model_is
Color_car_is
Color_is
Condition
Contains
KM_travelled
Listed_on
Rated_As
Sold_at
Transmission_categories
available_from
available_since
car_brand_is
car_color_is
city_avaiable
contains_wheel



Complete Nodes and relationship in Neo4j







Analytics Dashboard Documentation:

1) Download Python if not already downloaded (follow instructions on https://www.python.org/downloads/ based on operating system
2) Install Anaconda Distribution (follow instructions on https://docs.anaconda.com/anaconda/install/index.html based on operating system)
3) Once installed, open Anaconda command prompt (anaconda3)
4) Install jupyter notebook by running command pip install notebook
5) Then, install dash framework by running pip command pip install jupyter-dash
6) After installing dash framework, install other required packages: · Pandas: pip install pandas · Matplotlib: pip install matplotlib · Plotly: pip install plotly=5.7.0 · NumPy: pip install numpy · Py2Neo: pip install py2neo
7) Open jupyter notebook by running command jupyter notebook on Anaconda command prompt
8) Jupyter Notebook web page should appear
9) Navigate to the folder containing dataset and app.ipynb. Ensure both files are in the same folder 10) Open app.ipynb on Jupyter Notebook
11)Run app.ipynb script
12)Wait until code fully loads and IP address appears at the bottom of the page
13) Navigate to IP address and wait for dashboard to fully load
14) Dashboard is displayed!

Technical Metadata:

Technical metadata consists of metadata that is associated with data transformation rules, data storage structures, semantic layers, and interface layers. Provides information on the format and structure of the data as needed by computer systems:

File Name	used_cars_data.csv
File size	3.12 GB
Date/Time created	21 March,2022
Types of Compression	Zip
OS	Windows
Hardware processor name	Intel® core i7
Hardware RAM	16 GB
Tools used	Anaconda Navigator, Velero ETP, Lucid Chart, Neo4j, Microsoft Excel

Python Data Cleaning

Original Csv file	used_cars_data.csv
Cleaned CSV file	used_car_clean_CSV.csv
Original CSV file records	1 Million records
Cleaned csv file records	300 K records
Cleaned csv file size	63 MB

Column Names	GroupNumber	Column Description	DataType
vin	4	Vehicle Identification Number is a unique encoded string for every vehicle.	String
body_legroom	4	Legroom in the rear seat.	String
body_type	4	Body Type of the vehicle	String
city	4	City where the car is listed	String
daysonmarket	4	Days since the vehicle was first listed on the website	Integer
dealer_zip	4	Zipcode of the dealer	Integer
engine_type	4	Type of Engine	String
exterior_color	4	Exterior color of the vehicle	String
fuel_tank_volume	4	Fuel tank's filling capacity in gallons	String
fuel_type	4	Dominant type of fuel ingested by the vehicle.	String
height	4	Height of the vehicle in inches	String
is_new	4	Is the vehicle launched less than 2 years ago	Boolean
length	4	Length of the vehicle in inches	String
oldcarid	4	Unique identifcation for the old car	String
listed_date	4	Listing date of the Vehicle	String
make_name	4	Brand of the vehicle	String
maximum_seating	4	Maximum number of seats	String
mileage	4	Mileage of the car	Float
model_name	4	Model name of the cars	String
power	4	power configuration of the car	String
price	4	price of the car	Float
seller_rating	4	Rating of the seller	Float
transmission	4	Types of transmission	String
trimId	4	Version of model with particular configuration	String
wheelbase	4	wheel base of the vehicle in inches	String
year	4	The year on which the vehicle was made.	Integer

Business Metadata:

Business metadata is data that adds business context to other data. It provides information authored by business people and/or used by businesspeople.

Vin along with dealer zip, city and old Carid helps in identifying and filtering the used cars based on their location.

1.Number of used cars in the city

2.Various dealers in the city

3.Number of used cars sold and manufactured based on the dealer zip code.

·Length, Height, and Color are the dimensions of the used cars, and the car can be filtered based on this category by the business.

1.Filtering based on color, length, and height.

·Model name, make name, price, power, mileage, and year can be used by business to identify the total sales by model name and make name. The business can use the data to filter the number of used cars sold by their price, power, mileage, and year. Find the cars which have sold more that year on a certain price range.

·Body type, Engine type and legroom can be used by the business to find how many cars have which engine type and body type.

·Listing date, and Days on Market will be used by the business to find listing id and listing data of the vehicle on the website. Days on Market will be the days since the vehicle was first listed on the website.

·Fuel type and Fuel tank volume data can be used by businesses to find the car's fuel tanking capacity in gallons and filter the dominant fuel ingested by the vehicle.

Testing and Validation:

System Integration and User Acceptance Testing

Test 1: Check distinct values for each node

- Distinct values for each node were calculated after creating the nodes and compared it to the values in the cleaned dataset file which was inserted into Neo4j.Made sure that the column counts for the unique value in the CSV file and the one loaded in the Neo4j database is the same.

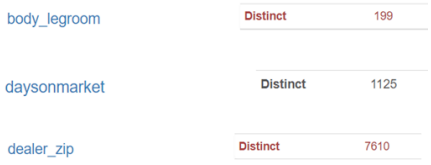
Neo4j Count

```
q = '''CALL apoc.meta.stats() YIELD labels
RETURN labels;'''
sit_data = DataFrame([dict(_) for _ in connect.query(q, db='neo4j')])
x = sit_data['labels'].values
d = dict(x[0])
```

```
df2.nunique()
```

```
vin                299996
body_legroom          199
body_type               9
city                 4366
city_mileage           94
daysonmarket         1125
dealer_zip          13013
engine_type            32
exterior_color       9440
fuel_tank_volume      167
fuel_type               7
height                440
is_new                  2
length                769
listed_date          1114
oldcarid           299999
make_name              71
maximum_seating        12
model_name           1071
power                1603
price               53046
seller_rating         249
transmission            4
trimId              23497
wheel_base              5
year                   85
dtype: int64
```

csv count



Test 2 : Verification of datatypes

Verified data type of each node property by comparing data type of each node with cleaned dataset file.

Neo4j column data types

```
dtf_data.values
```

```
array([['oldcarid', 'STRING'],
       ['city', 'STRING'],
       ['vin', 'STRING'],
       ['body_legroom', 'STRING'],
       ['body_type', 'STRING'],
       ['city_mileage', 'STRING'],
       ['daysonmarket', 'STRING'],
       ['dealer_zip', 'STRING'],
       ['engine_type', 'STRING'],
       ['exterior_color', 'STRING'],
       ['fuel_tank_volume', 'STRING'],
       ['fuel_type', 'STRING'],
       ['height', 'STRING'],
       ['is_new', 'STRING'],
       ['length', 'STRING'],
       ['listed_date', 'STRING'],
       ['make_name', 'STRING'],
       ['maximum_seating', 'STRING'],
       ['model_name', 'STRING'],
       ['power', 'STRING'],
       ['seller_rating', 'STRING'],
       ['transmission', 'STRING'],
       ['wheel_base', 'STRING'],
       ['year', 'STRING'],
       ['trimId', 'STRING'],
       ['price', 'STRING']], dtype=object)
```

Test 3 : Verifying counts in Neo 4j and the counts via python

The query helps to get the count of the records in neo4j

Neo4j count

```
neo4j$ MATCH (Oldcars)-[:Condition]→(is_new) RETURN COUNT(Oldcars.oldcarid), is_new.is_new
```

	COUNT(Oldcars.oldcarid)	is_new.is_new
1	147426	"TRUE"
2	152573	"FALSE"

Python Count

```
data.columns = ['Count of cars', 'Condition']
data
```

	Count of cars	Condition
0	147426	TRUE
1	152573	FALSE

Test 4 : Check number of null values

```
df_select.isnull().sum()
```

```
back_legroom            16104
bed                    298110
bed_height             257154
bed_length             257154
body_type                1330
cabin                  293726
city                        0
city_fuel_economy       49458
combine_fuel_economy   300000
daysonmarket                0
dealer_zip                  0
```

Test 5 : Finding the missing value percentage in the dataset

```
percent_missing = df_select.isnull().sum() * 100 / len(df_select)
missing_value_df = pd.DataFrame({'column_name': df_select.columns,
                                 'percent_missing': percent_missing})

missing_value_df
```

	column_name	percent_missing
vin	vin	0.000000
back_legroom	back_legroom	5.368000
bed	bed	99.370000
bed_height	bed_height	85.718000
bed_length	bed_length	85.718000
body_type	body_type	0.443333
cabin	cabin	97.908667
city	city	0.000000

Test 6 : Null values replaced

Explanation: These metrics were used in our data analysis and we wanted to account for null values instead of just removing them from the dataset.

```
df_select.columns = [c.replace(' ', '_') for c in df_select.columns]
df_select
```

	vin	back_legroom	bed	bed_height	bed_length	body_type	cabin	city	city_fuel_economy	comb
2264373	5UXCR6C09M9D85675	NaN	NaN	NaN	NaN	SUV / Crossover	NaN	Phoenix	NaN	
2500767	3FA6P0HD9LR156514	38.3 in	NaN	NaN	NaN	Sedan	NaN	Spring	23.0	
2287253	1FMSK7DHXLGC91805	39 in	NaN	NaN	NaN	SUV / Crossover	NaN	Gilbert	21.0	
1976977	3GTP9EEL6LG420561	43.4 in	NaN	--	69.9 in	Pickup Truck	NaN	Helena	15.0	
472200	2T3RFREV0JW813348	37.2 in	NaN	NaN	NaN	SUV / Crossover	NaN	Streamwood	22.0	
2156875	WA1ANAFY5L2091374	37.8 in	NaN	NaN	NaN	SUV / Crossover	NaN	Phoenix	NaN	
1119418	1GCGSBENXM1108891	35.8 in	NaN	--	61.7 in	Pickup Truck	NaN	Lawrenceville	18.0	
1452100	1N4AL3AP5HC116903	36.1 in	NaN	NaN	NaN	Sedan	NaN	Dundas	27.0	

300000 rows × 66 columns Open in new tab

Test 7: Dropped Columns Not Being Used for Analysis

Explanation: Columns dropped were sigma values or scientific values that we could not fully understand scientifically for use in our data analysis.

```
df_select = df_select.drop('bed', axis=1)
df_select = df_select.drop('bed_height', axis=1)
df_select = df_select.drop('bed_length', axis=1)
df_select = df_select.drop('cabin', axis=1)
df_select = df_select.drop('combine_fuel_economy', axis=1)
df_select = df_select.drop('fleet', axis=1)
df_select = df_select.drop('frame_damaged', axis=1)
df_select = df_select.drop('has_accidents', axis=1)
df_select = df_select.drop('isCab', axis=1)
df_select = df_select.drop('owner_count', axis=1)
df_select = df_select.drop('salvage', axis=1)
df_select = df_select.drop('theft_title', axis=1)
df_select = df_select.drop('vehicle_damage_category', axis=1)
df_select = df_select.drop('is_certified', axis=1)
df_select = df_select.drop('is_oemcpo', axis=1)
df_select = df_select.drop('is_cpo', axis=1)
```

Test 8: Count of percent missing values will be zero

```
#Checking missing values again

percent_missing = df_select.isnull().sum() * 100 / len(df_select)
missing_value_df = pd.DataFrame({'column_name': df_select.columns,
                                 'percent_missing': percent_missing})

missing_value_df
```

	column_name	percent_missing
vin	vin	0.0
back_legroom	back_legroom	0.0
body_type	body_type	0.0
city	city	0.0
city_fuel_economy	city_fuel_economy	0.0
daysonmarket	daysonmarket	0.0
dealer_zip	dealer_zip	0.0
description	description	0.0

50 rows × 2 columns Open in new tab

Neo4j Testing screenshots using Cypher Queries

Steps for testing and verification of data using Neo4j:
- Copy the cypher code(neo4j_testing_script.txt) containing the test queries
- Run the code to view graphs and tables verifying data load to Neo4j
- The scripts will run one by one and will take a few minutes displaying the below results

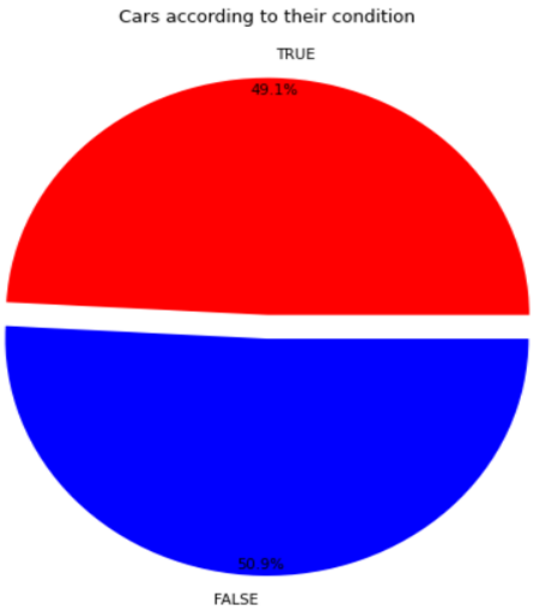
Nodes Verification :

- Distinct values for each nodes were calculated after creating the nodes and compared it to the values in the cleaned dataset file which was inserted into Neo4j
- Made sure that the column counts for the unique value in the CSV file and the one loaded in the Neo4j database is the same

Dashboard Interpretation/Findings:

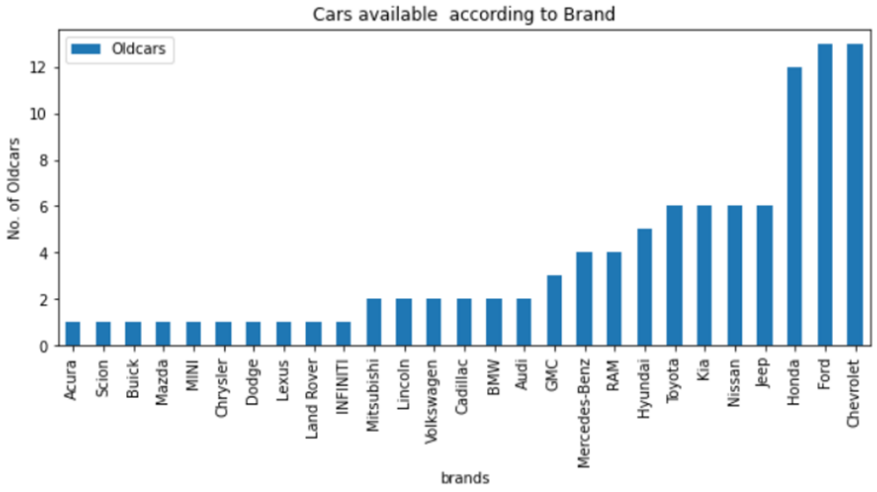
1. Number of Cars according to the condition of the car

- Highlights percentage of Used car condition that are in new condition
- The chart shows around 50.9% of cars are not new.
- This chart will give an idea to the buyer to get details regarding the cars that are in new condition at your price
- 49.1% of cars are in good condition, which means they are like new cars



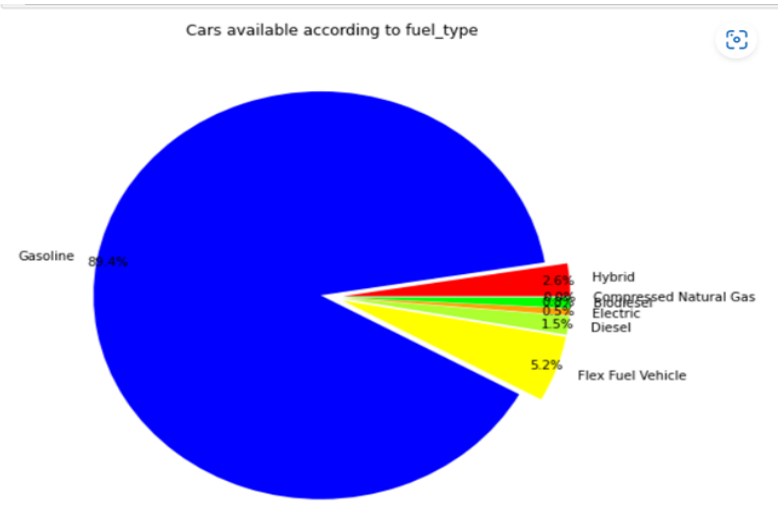
2. Count of cars available according to year of manufacture

- Dashboard depicts the count of used cars available in market for sale and the year of manufacture of car.
- This will give an idea regarding the cars available according to the year of manufacturing to the buyer so that he can decide of more to which kind of car he can go for.
- Using limit 100, we can find what is the count of the cars available which are manufactured recently and get more details regarding those cars.



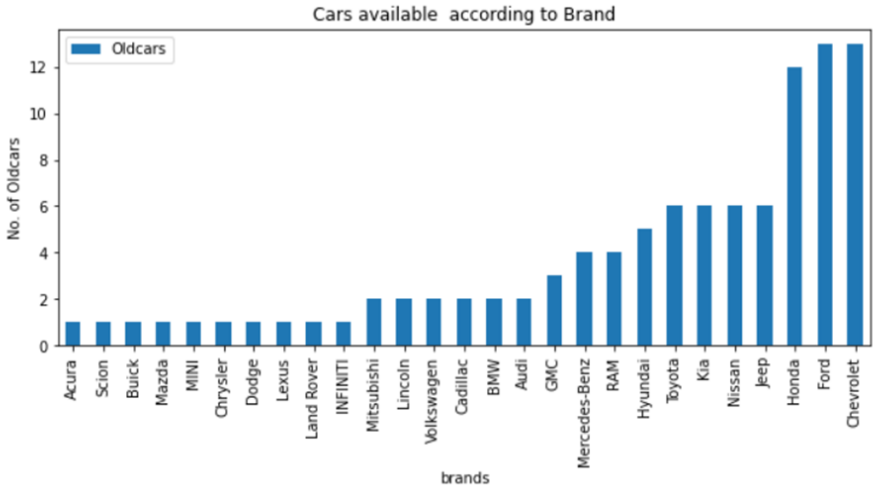
3. Count of cars available according to fuel type

- Highlights percentage of cars available according fuel type
- Chart shows that most 89.4%of cars are considered using gasoline and flex-fuel becomes 2nd highest as 5%
- This will give idea to buyers if there are any cars available according to their fuel preference.



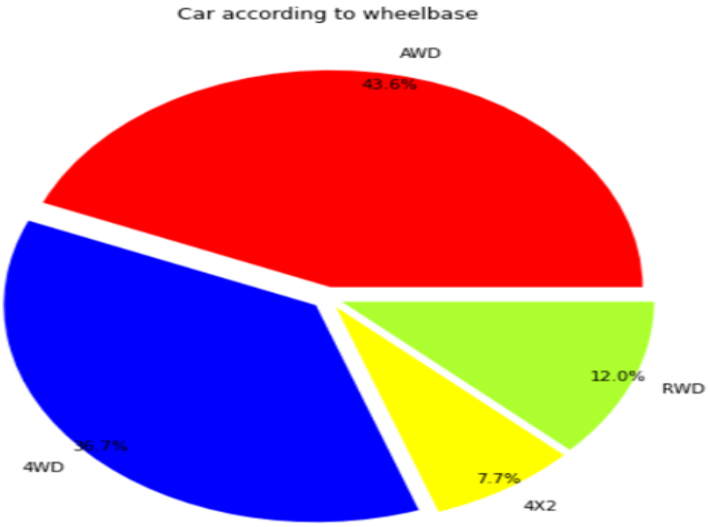
4. Count of cars available according to Brands

- Dashboard depicts the count of used cars available in market for sale and their brands
- This will give an idea regarding the cars available according to manufacturing brand so that they can decide as buyers have brand preference so it will help them to figure of the cars according to the brands
- Using limit 100, we can find what is the count of the cars available according to their brands



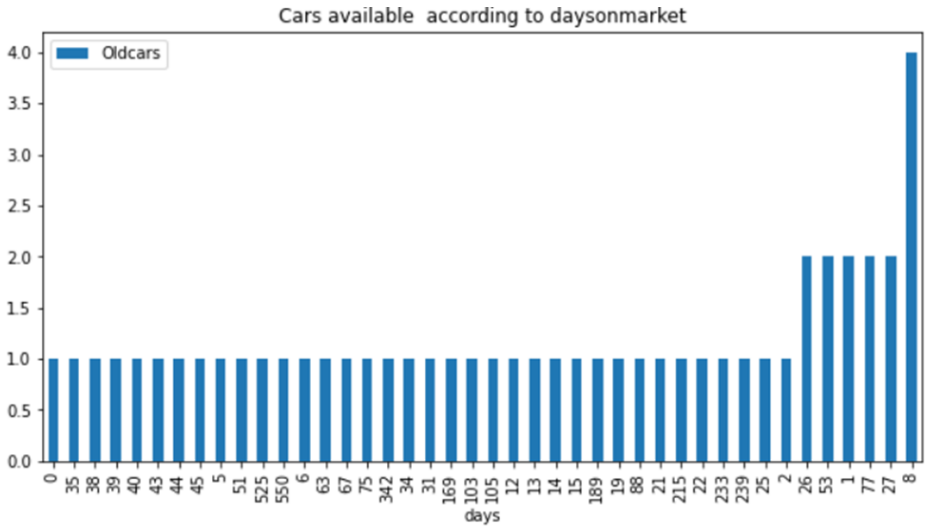
5. Percentage of cars available according to Wheelbase

- Highlights percentage of asteroids that are/are not potentially hazardous
- Chart shows that at most 43.6%of cars are considered having AWD wheels and 4WD being 36.7%
- This will give idea to buyers if there are any cars available according to their Wheelbase preference



6. Count of cars available according to Days on market

- Dashboard depicts the count of used cars available in market and the days they are available since markets.
- This helps the buyers to get idea about what are the latest cars available for purchasing also the cars which are market since long.
- For the cars which are in market since long even the seller can get an idea if his car will be in demand or not or else which location has that demand.
- Cars and the days on the markets are the main interpretation



7. Count of cars available according to Seller rating

- Highlights number of cars available with the seller having high ratings.
- If the seller is having good rating this helps the buyer and the seller both to get idea regarding the seller's service and they can go-ahead with that seller for either buying or selling cars
- Ratings help to get idea and decide as to which seller should proceed with the purchase
- Have below count of cars with sellers having with good and bad ratings

COUNT(Oldcars.oldcarid)	seller_rating
22746	5

Count of cars	seller rating
699	1
29	1.33
14	1.4
36	1.8
51	1.67
46	1.75
22	1.5

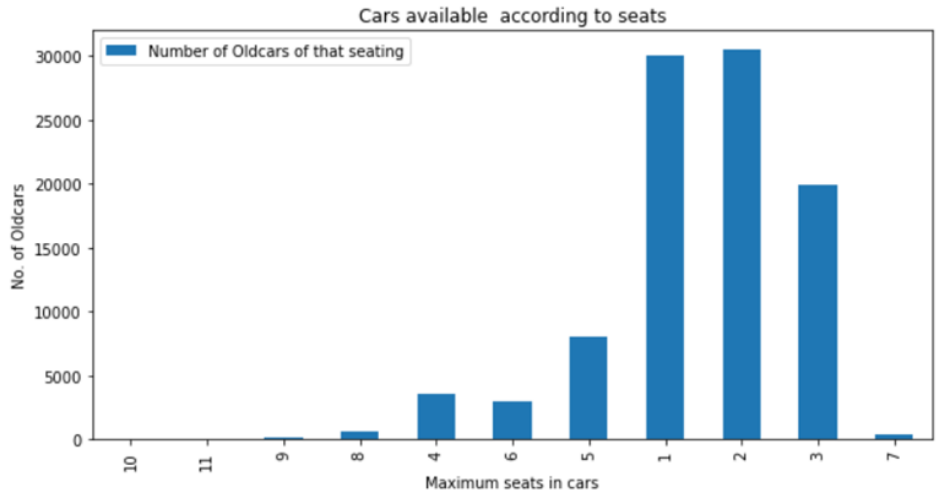
8. Getting the car id and price of the car

- Simply gives the idea of the price of the car
- Can sort the price desc and decide as which is the carid and then get details regarding the car may it be its color or type of interior or date of manufacturing
- Old car id and the price of the same for 10 sample old car is shown in the dashboard

Oldcarid	Price of cars
279374649	25265
272898693	28521
279718872	26675
279449435	47163
281719831	35384
277051529	7700
280116206	43145
273989662	63765
254387257	16244
263299746	12975

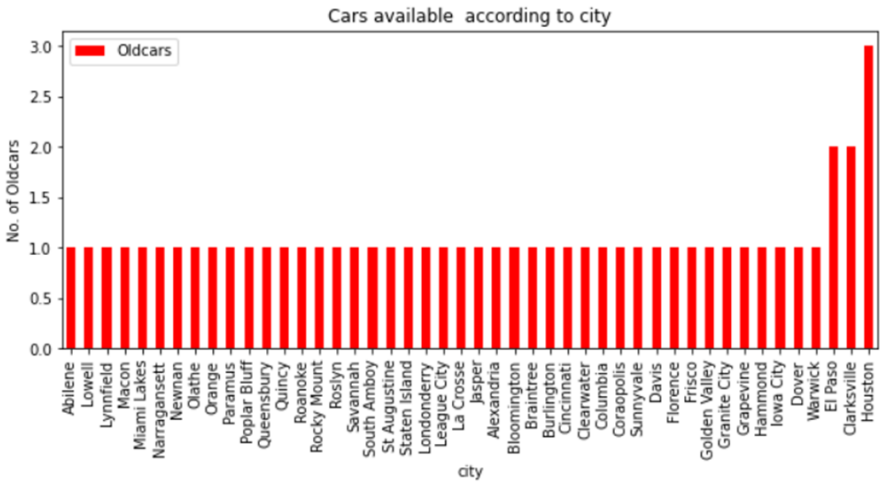
9. Count of cars available according to seats in the car

- Highlights number of cars available according to the seats in the cars
- As per the buyers requirements if the need 4/6 seaters cars he can get idea about the counts of cars
- Accordingly the buyer can decide if the cars which he wants to buy as per the number of seats is available in his area or not



10. Count of cars available according to City

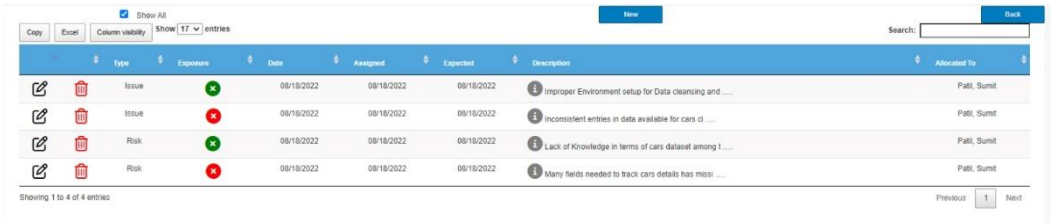
- Dashboard depicts the count of used cars available in market according to your nearest city
- The dashboard have 50 cities and the counts of cars available around them.
- This gives idea to the buyer to get the counts of the car available in their area so that they can search for the cars available and its for them to buy the same.


Cars available according to city
Oldcars
No. of Oldcars
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Abilene
Lowell
Lynnfield
Macon
Miami Lakes
Narragansett
Newnan
Olathe
Orange
Paramus
Poplar Bluff
Queensbury
Quincy
Roanoke
Rocky Mount
Roslyn
Savannah
South Amboy
St Augustine
Staten Island
Londonderry
League City
La Crosse
Jasper
Alexandria
Bloomington
Braintree
Burlington
Cincinnati
Clearwater
Columbia
Coraopolis
Sunnyvale
Davis
Florence
Frisco
Golden Valley
Granite City
Grapevine
Hammond
Iowa City
Dover
Warwick
El Paso
Clarksville
Houston
city

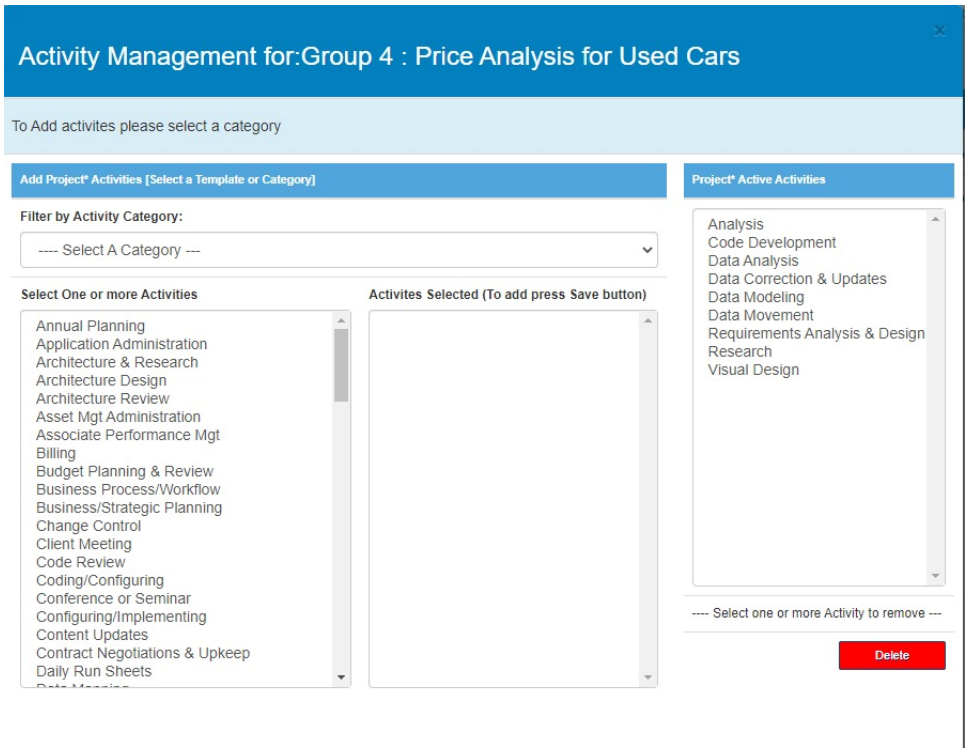
Velero Screenshot:

1. Resource Management

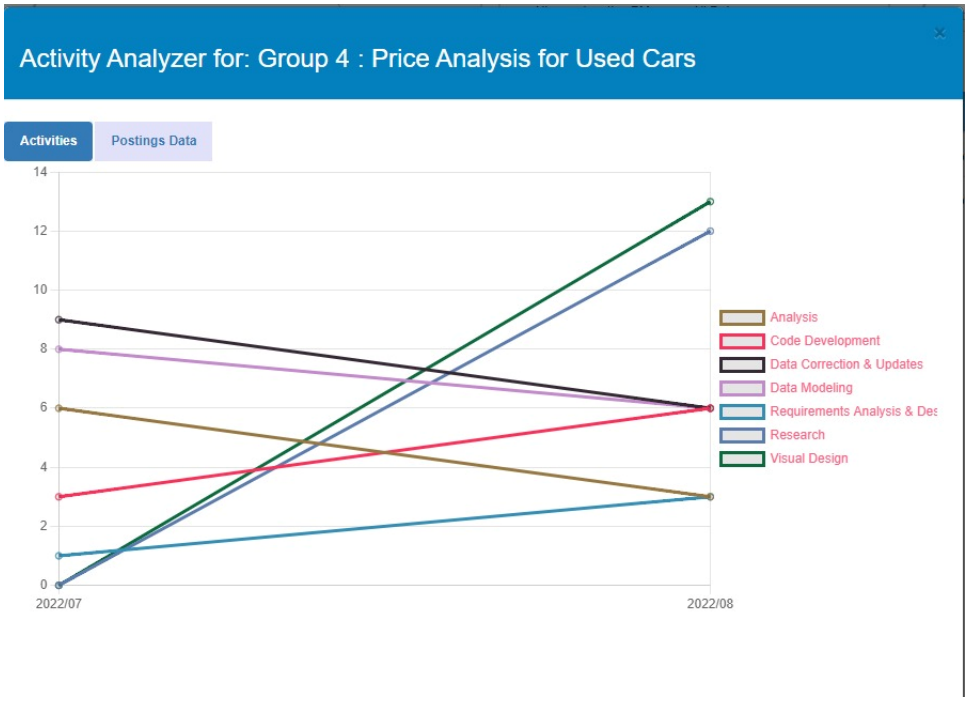
2. Risk & Issues:



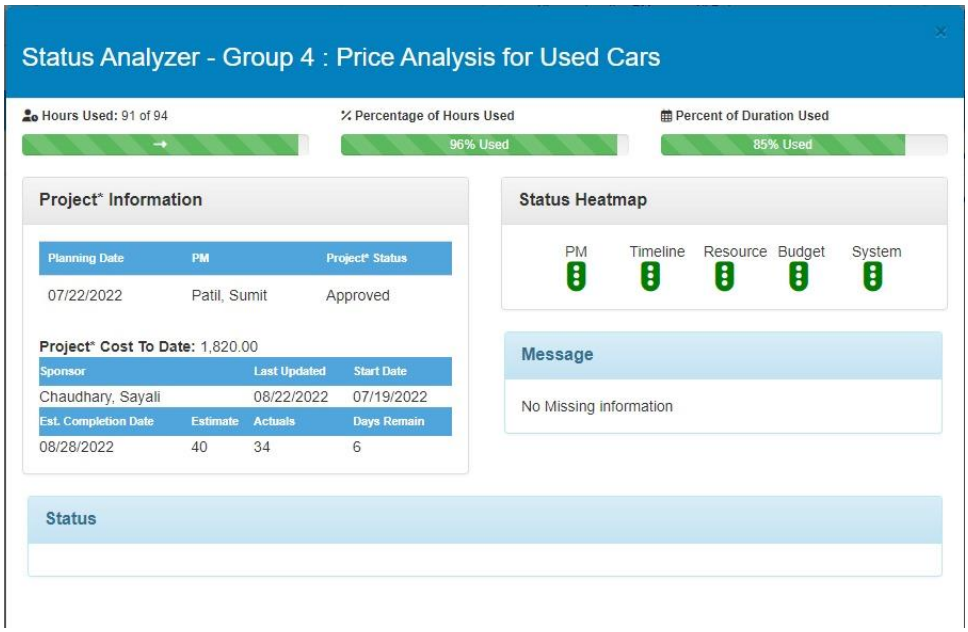
3. Activity Management



4. Activity Analyzer

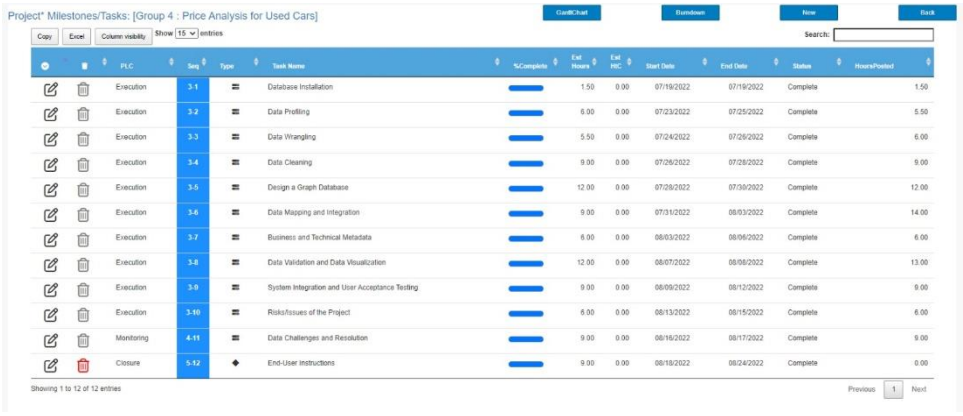


5. Status Analyzer

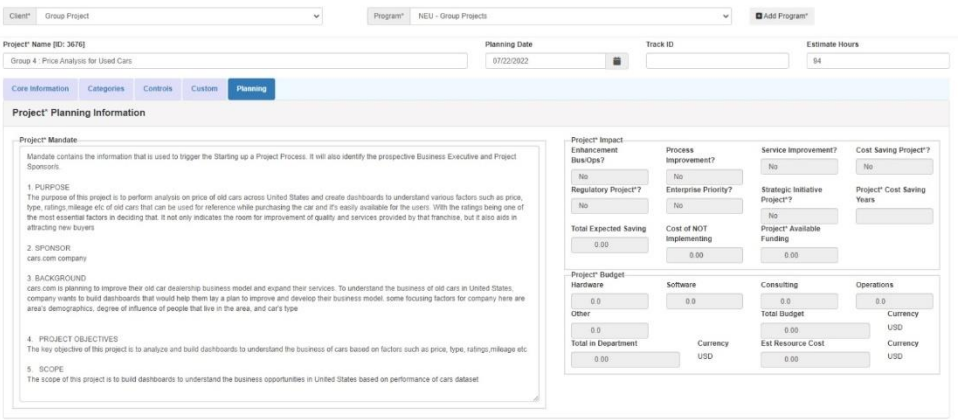


6. Gantt Chart

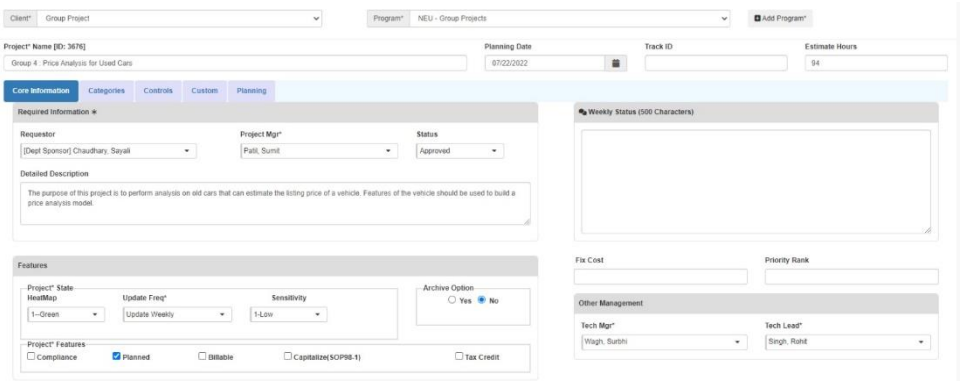
7. Management Plan



8. Mandates



9. Overview


Client*
Group Project
Program*
NEU - Group Projects
Add Program*
Project* Name [ID: 3676]
Group 4 : Price Analysis for Used Cars
Planning Date
07/22/2022
Track ID
Estimate Hours
94
Core Information
Categories
Controls
Custom
Planning
Required Information *
Requestor
[Dept Sponsor] Chaudhary, Sayali
Project Mgr*
Patil, Sumit
Status
Approved
Detailed Description
The purpose of this project is to perform analysis on old cars that can estimate the listing price of a vehicle. Features of the vehicle should be used to build a price analysis model.
Weekly Status (500 Characters)
Features
Project* State
HeatMap
1--Green
Update Freq*
Update Weekly
Sensitivity
1-Low
Archive Option
Yes
No
Project* Features
Compliance
Planned
Billable
Capitalize(SOP98-1)
Tax Credit
Fix Cost
Priority Rank
Other Management
Tech Mgr*
Wagh, Surbhi
Tech Lead*
Singh, Rohit