**Extraction, Transformation, and Load Technical Report**

**Project TrueSafeCars.com**

**Group 8 (NoFloods)**

Roopa Patel

Samuel Parks

Steven Lee

[(Table of contents page) 2](#_Toc41655116)

[1. Introduction 3](#_Toc41655117)

[1.1 Summary 3](#_Toc41655118)

[1.2 1.2 Scope 3](#_Toc41655119)

[1.3 Technologiesandresourcecontributions 3](#_Toc41655120)

[1.4 Definitions,AcronymsandAbbreviations 3](#_Toc41655121)

[2. ETL DETAILS 4](#_Toc41655122)

[2.1 DataImport/ExtractSourcesandMethod 4](#_Toc41655123)

[2.2 DataAcquisition 4](#_Toc41655124)

[2.3 DataTransform 4](#_Toc41655125)

[2.4 DataIntegrity 4](#_Toc41655126)

[2.5 DataRefreshFrequency 4](#_Toc41655127)

[2.6 DataSecurity 4](#_Toc41655128)

[2.7 DataLoadingandAvailability 5](#_Toc41655129)

[3. DATA QUALITY 6](#_Toc41655130)

# **Introduction**

## **Summary**

The objective of this ETL project was to provide our client, TrueSafeCars.com, with a database containing a consolidated inventory their cars. Our client was formed by buying inventory from two major used car dealership chains. To support their business goals, we were tasked with this ETL project with the following goals:

1. Consolidate the inventory into a central database
2. Assign the inventory to one of TrueSafeCars.com’s five main offices
3. Identify which vehicles should be sold at auction (vs. selling to the public)
4. Identify which vehicles meet our safety guidelines
5. Identify recall information per vehicle

**Extract Requirements**

Pull vehicle inventory data from the two dealerships we purchased:

* Data for purchased dealership #1: Kaggle.com (<https://www.kaggle.com/jpayne/852k-used-car-listings>)
* Data for purchased dealership #2: Kaggle.com (<https://www.kaggle.com/austinreese/craigslist-carstrucks-data>)

**Transform Requirements**

Once all of the vehicle inventory data has been extracted, perform the following transformations:

* Remove duplicate VIN numbers
* Assign the inventory to one of five main office locations. The client has provided a CSV file with the State-to-Main Office mapping. Our 5 main office locations are:
* Northeast Main Office
* Southeast Main Office
* Midwest Main Office
* Southwest Main Office
* West Main Office
* Assign each vehicle an overall safety rating using an API from NHTSA.gov (<https://webapi.nhtsa.gov/Default.aspx?SafetyRatings/API/5>)
* Determine the number of open recalls per vehicle using an API from NHTSA.gov (<https://webapi.nhtsa.gov/Default.aspx?Recalls/API/83>)

The vehicle inventory should have a column to indicate if the vehicle is ready to be sold to the public or not:

* Vehicles 20 years old or older should be sold at auction
* Vehicles with an overall poor safety rating should be sold at auction
* Vehicles in poor condition (if this information is available in the inventory data) should be sold at auction

**Load Requirements**

Load transformed data into a central database (NoSQL). The central database should have tables that can be easily queried to retrieve:

* Vehicles assigned to each main office
* Vehicles that are ready to be sold to the public
* Vehicles that should be sold at auction
* Overall safety rating for each vehicle
* Recalls that apply to each vehicle

## **Scope**

The data sources to be integrated are:

* Inventory data for the first dealership chain purchased by TrueSafeCars.com, represented by this Kaggle.com dataset: (<https://www.kaggle.com/jpayne/852k-used-car-listings>). This dataset was produced by scraping TrueCar.com
* Inventory data for the second dealership chain purchased by TrueSafeCars.com, represented by this Kaggle.com dataset: (<https://www.kaggle.com/austinreese/craigslist-carstrucks-data>). This dataset was produced by scraping Craigslist.com
* To assign inventory to one of TrueSafeCars.com’s main offices, we were provided CSV file with the State-to-Main Office mapping
* The VIN Decoder, Safety Rating and Recall data were obtained using the NHTSA.gov API

In Scope for this project:

* Perform extract of the entire dataset for both input files
* Perform data cleansing for the entire dataset for both input files
* Map vehicles to a Main Office location for the entire dataset
* Perform the VIN Decoder, Safety Rating, and Recall API calls for a subset of data

Out of Scope for this project:

* In order to reduce the number of API calls, performed the following for a small subset of the cleaned dataset
  + Used the VIN Decoder to determine if the VIN was authentic, as well as obtain a clean Make and Model from the VIN number
  + Retrieve Safety Rating data from the VIN number
  + Retrieve Recall data from the VIN number

## **Technologies and resource contributions**

**Team Members**

* Roopa Patel
  + Github repository manager
  + MongoDB database creation and load process
  + Client Acceptance website
* Samuel Parks
  + Extract data from input datasets and cleaning
  + Creating random test and production datasets
  + API calls and processing
* Steven Lee
  + Data cleaning
  + API calls and processing

**Tech Stack**

* Core data extraction and transformation
  + Python and pandas
* US State to TrueSafeCars.com Main Office mapping
  + Python and pandas
* API Requests
  + Python and pandas
* NoSQL Database
  + MongoDB
* TrueSafeCars Client Acceptance Website
  + Javascript
  + Node
  + HTML / CSS

## **Definitions, Acronyms and Abbreviations**

|  |  |
| --- | --- |
| API | Application Programming Interface |
| ETL | Extract, Transform, and Load |
| HTML | Hypertext Markup Language |
| MongoDB | MongoDB is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas |
| NaN | Not a Number |
| NHTSA | National Highway Traffic Safety Administration |
| pandas | pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series |
| PII | Personally Identifiable Information |
| Python | Python is an interpreted, high-level, general-purpose programming language |

# **ETL DETAILS**

## **Data Import / Extract Sources and Method**

Data for the two dealerships acquired were downloaded from Kaggle.com, then unzipped:

* <https://www.kaggle.com/jpayne/852k-used-car-listings>
* <https://www.kaggle.com/austinreese/craigslist-carstrucks-data>

The data for each were then loaded into pandas dataframes using the pandas .read\_csv method.

## **Data Acquisition**

The data needed for this project represents the entire car inventory for each of the dealership chains purchased by TrueSafeCars.com. When these dealership chains were purchased, all sales, purchases, etc… have stopped. Therefore, the data is static and does not need continuous update.

If additional inventory was found, it would simply be imported into Python in addition to the main input datasets and processed in the same way. In this way, any duplicates would be identified and reported for further analysis.

There were no issues or pre-requisites that needed to be cleared prior to getting the data.

## **Data Transform**

The following steps were performed to transform the data:

* Clean column names and determine final set of column names to apply to both datasets
  + Remove unneeded column names from each input dataset
    - Only keep **Price**, **Year**, **Mileage**, **City**, **State**, **Vin**, **Make**, and **Model**
  + Add an empty **Condition** column to the TrueCar dataset
  + Fix capitalization of column names so that they are exact matches across both datasets
* Clean data in any text columns. This includes removing any leading or trailing whitespaces, standardizing capitalization, and replacing NaNs where appropriate
  + Stripped whitespace from **State**, **Vin**, **Make**, **Model**, and **Condition** column data
  + Applied capitalization rules to **Make**, **Model**, **Condition**, and **State** column data
  + Replaced NaNs in the **Condition** column with the value “Unknown”
* Find “problem” rows for each dataset. These are records with NaNs that still remain in any column
  + Create a CSV of problem rows for each dataset so the TrueSafeCars.com staff can research further
  + Remove the problem rows from each dataset
* Look for duplicate VIN numbers
  + Create a CSV of rows with duplicate VINs for each dataset so the TrueSafeCars.com staff can research further
  + Remove the duplicate VINs (keep one original in the dataset) from each dataset
* Convert data types to be consistent across both datasets
  + Convert the **Year** and **Mileage** columns from the Craigslist input dataset to int64
* Create a Sales Channel column perform a first pass at assigning a sales channel to each car (Public or Auction)
  + For this first pass, cars that are 20 years old or older are assigned to the Auction sales channel. Otherwise, assign the car to the Public sales channel
* For the second pass of Sales Channel assignment, assign based on the **Condition** column
  + This only applies to the Craigslist input dataset as the TrueCar input dataset did not have Condition data for their cars
  + If the Condition found is not “New”, “Excellent”, “Like New”, “Good”, or “Unknown” then assign the car to the Auction sales channel
* Assign each vehicle to a Main Office location
  + Read in the client-provided CSV file using pandas .read\_csv
  + Use pandas .merge to join our vehicle dataframes with the client-provided State-to-Main Office mapping
* After determining a test sample/subset for API calls, we used the NHTSA.gov VIN decoder to get a clean “Make” and “Model” for each vehicle using the VIN
  + For the vehicles we were successful in getting clean Make and Model data, we added column to the dataframe “Vin\_Verified” and set the value to True
    - If there were any errors obtaining data for a VIN, we simply left Vin\_Verified set to the default value of False
  + For the Vin Verified records, we continued with more API calls
    - We obtained the number of recalls using Year, Make, and Model
      * We added column to the dataframe “Recall\_Count” to store the number of recalls found from the API call
      * If we experienced any errors with the Recall API call, we did not update Recall\_Count and left it as the default value of 0
    - We obtained the overall safety rating using Year, Make, and Model
      * We added column to the dataframe “Safety\_Rating” to store the overall safety rating found from the API call
      * If we experienced any errors with the Recall API call, we did not update Safety\_Rating and left it as the default value of blank

## **Data Integrity**

The extraction source data was fairly reliable. Any records with missing values were identified as part of the data cleansing process and exported as CSV as well as removed from the dataset. Also, any duplicate records were also identified, exported as CSV, and removed.

Since this was a one-time ETL project as requested by TrueSafeCars.com, we do not have to consider frequency of source data updates. The input datasets are final snapshots of car inventory data that TrueSafeCars.com is about to incorporate into their inventory.

As mentioned in the Out of Scope section of the report, we only performed VIN decoding and checking for a small subset of the dataset to reduce the number of API calls made. Ideally, we would run the VIN decoder on all VINs in our dataset. Additionally, the VIN decoder provided by NHTSA.gov is limited to cars 1981 and newer, so we would be unable to VIN decode and verify cars older than this.

## **Data Refresh Frequency**

Since this was a one-time ETL project as requested by TrueSafeCars.com, we do not have to consider frequency of source data updates. The input datasets are final snapshots of car inventory data that TrueSafeCars.com is about to incorporate into their inventory.

## **Data Security**

There is no sensitive or PII data in this dataset.

## **Data Loading and Availability**

Multiple datasets were produced from the Extract and Transform processes. All were exported to CSV:

* Large datasets:
  + Cleaned TrueCar dataframe (CleanedLargeDatasets/truecar\_df\_cleaned\_data.csv)
  + Cleaned Craigslist dataframe (CleanedLargeDatasets/craigslist\_df\_cleaned\_data.csv)
  + Combined TrueCar and Craigslist dataframes (CleanedLargeDatasets/combined\_df\_cleaned\_data.csv)
* Smaller datasets for testing and to simulate production:
  + Test TrueCar dataframe (50 records) (TrimmedDatasets/truecar\_test\_data.csv)
  + Test Craigslist dataframe (50 records) (TrimmedDatasets/craigslist\_test\_data.csv)
  + Test Combined TrueCar and Craigslist dataframes (100 records) (TrimmedDatasets/combined\_test\_data.csv)

Data is also available in MongoDB. For the schema, the field names come from the pandas dataframe column names, so they are very readable. See screenshot below, showing our combined data inserted into our database. As you can see, all 1,021,137 rows from our combined cars dataframe were successfully loaded.

Our client has many options to view the data:

* They can import the CSV file(s) into an application of their choosing
* They can use MongoDB Compass to look at the data
* They can use the website we have created to more easily look at the data

Screenshot of MongoDB Compass:A screenshot of a computer screen

Description automatically generated

Screenshots from the website we have created:

# **DATA QUALITY**

**For the TrueCar dataset:**

* We expected to read all rows into a pandas dataframe with the following total:
  + TrueCar dataframe beginning number of rows: 852,122
* We did a preliminary count of unique VINs:
  + TrueCar total unique VIN records: 852,075
  + We expect to remove the duplicates as shown below
* We did not find any problem rows (NaNs in columns)
  + Number of TrueCar problem rows found: 0
* We found duplicate VINs and removed them, so that our total rows equaled what we found earlier for number of unique VIN records
* Number of TrueCar duplicate VINs found: 47
* Number of TrueCar rows BEFORE removing duplicate VINs: 852122
* Number of TrueCar rows AFTER removing duplicate VINs: 852075
* We looked at a VIN count by sales channel and saw reasonable results:
  + Auction VINs: 4,557
  + Public VINs: 847,518

**For the Craigslist dataset:**

* We expected to read all rows into a pandas dataframe with the following total:
  + Craigslist dataframe beginning number of rows: 539759
* We did a preliminary count of unique VINs:
  + Craigslist total unique VIN records: 181,678
  + We expect to remove the duplicates as shown below. Unlike the TrueCar dataset, the number of rows that were left after removing duplicate VINs was smaller than 181,678. This was due to problem rows (with NaNs) being part of the unique VIN records count of 181,678.
* We found many problem rows, which was to be expected as Craigslist has many free-form text fields and the general public is responsible for creating their ads:
  + Number of Craigslist problem rows found: 248,071
  + Number of Craigslist rows BEFORE removing problem rows: 539,759
  + Number of Craigslist rows AFTER removing problem rows: 291,688
* We found duplicate VINs and removed them
  + Number of Craigslist duplicate VINs found: 122,626
  + Number of Craigslist rows BEFORE removing duplicate VINs: 291,688
  + Number of Craigslist rows AFTER removing duplicate VINs: 169,062
* We looked at a VIN count by sales channel and saw reasonable results:
  + Auction VINs: 6,233
  + Public VINs: 162,829
* After updating sales channel based on condition, the results seemed very reasonable in that the number of Auction VINs only went up slightly
  + Auction VINs: 7,360
  + Public VINs: 161,702

**User Acceptance Testing**

User acceptance testing for this project mainly consists of spot checks of our cleaned datasets. We recommend the spot checking to include:

* No NaN values exist
* Data for each column is formatted properly, e.g. two characters for the State column
* Check text in the Make and Model columns and ensure that they correspond to vehicles that exist
* Check the State to Main office assignments and ensure that the assignments make sense (e.g. a car in FL should be assigned to the Southeast Main Office)

As mentioned before, the client has many options to review the data. They can use the CSV files directly, use MongoDB compass to look at the database directly, or use the website we have created to review the data.