Data Warehouse Report

Federico Di Franco 263678

June 17, 2025

1 Dataset Description and Analyses

The *Vehicle Sales and Market Trends Dataset* from Kaggle [https://www.kaggle.com/datasets/syedanwarafridi/vehicle-sales-data] comprises tabular records of U.S. vehicle transactions, where each row details a single sale and columns capture attributes such as VIN, year, make, model, trim and body type, as well as transmission and exterior/interior colors. Usage and condition are documented via odometer readings and numeric condition ratings, while transaction specifics include sale dates and this enable temporal analysis, sale prices and Manheim Market Report (MMR) values. Seller names and states of registration enable regional analysis.

Data Quality Metrics

Metric	**Value (%)**
Validity	99.38 %
Completeness	98.62 %
Consistency	94.54 %
Uniqueness	100.00 %

Table 1: Data quality metrics computed on the dataset by the Python script

Calculation Methodology The four metrics were computed using the Python script `calculate_data_quality_metrics.py` as follows:

- **Validity:** percentage of cells whose values fall within predefined domains (year 1900–2025, VIN of exactly 17 characters, odometer 0–500.000 mi, condition score 1–50, sale price between $500 and $200,000).
- **Completeness:** ratio of non-null cells to total cells in the dataset.
- **Consistency:** percentage of records where sale price deviates by no more than 30% from the MMR value and the vehicle year is less than or equal to the year extracted from the sale date.
- **Uniqueness:** ratio of unique rows to total rows (1 minus the duplicate rate).

All scripts used for calculating data quality metrics and for data cleaning, transformation and integration are available on GitHub: https://github.com/fefe202/data_warehouse_project.git.

```python
# 3. CONSISTENCY
    consistency_checks = 0
    consistency_passed = 0

    # MMR vs Selling Price (differenza < 30%)
    if 'mmr' in df.columns and 'sellingprice' in df.columns:
        mmr_vals = pd.to_numeric(df['mmr'], errors='coerce')
        price_vals = pd.to_numeric(df['sellingprice'], errors='coerce')
        valid_pairs = (mmr_vals.notna() & price_vals.notna())

        if valid_pairs.sum() > 0:
            consistency_checks += valid_pairs.sum()
            diff_pct = abs((mmr_vals - price_vals) / mmr_vals) * 100
            consistency_passed += (diff_pct <= 30).sum()

    # Year vs Sale Date
    if 'year' in df.columns and 'saledate' in df.columns:
        years = pd.to_numeric(df['year'], errors='coerce')
        # Corretto: specificato utc=True e controllato se la conversione è riuscita
        sale_dates = pd.to_datetime(df['saledate'], errors='coerce', utc=True,
        ↪  infer_datetime_format=False)
        valid_pairs = (years.notna() & sale_dates.notna())

        if valid_pairs.sum() > 0:
            consistency_checks += valid_pairs.sum()
            valid_sale_dates = sale_dates[valid_pairs]
            valid_years = years[valid_pairs]
            if len(valid_sale_dates) > 0:
                sale_years = valid_sale_dates.dt.year
                consistency_passed += (sale_years >= valid_years).sum()

    consistency = (consistency_passed / consistency_checks * 100) if consistency_checks > 0
    ↪  else 85

```

Extract from the script used to calculate data quality metrics

2 Data Cleaning, Transformation and Integration

- **Removed records with null values:** All rows containing missing critical attributes were dropped to ensure completeness of analyses.
- **Filtered suspicious outliers:** Entries with implausible numeric values (likely typos or measurement errors) were excluded to improve data reliability.
- **Standardized `Trim`:** Any trim labels containing "+" or "!" were normalized to "Base" for consistency.
- **Normalized `Condition`:** Original scores (1–49) were binned into five semantic categories—*Extremely Used*, *Heavily Used*, *Moderately Used*, *Lightly Used*, *Like New*—to simplify downstream analysis.
- **Added `TypeSeller`:** A script assigns each seller to one of Remarketing, Rental/Leasing, Official Dealer, Bank/Financial or Independent Dealer by matching keywords in the vendor name.
- **Reformatted dates:** All sale dates were converted to `YYYY-MM-DD` to ensure seamless parsing in Tableau.
- **Expanded state codes:** Two-letter abbreviations (e.g. "CA") were mapped to full state names (e.g. "California") for clarity and compatibility in Tableau.
- **Corrected `Body` labels:** Variations like "sedan"/"Sedan" and "Regular Cab"/"regular-cab" were unified to a single standard form.
- **Grouped body types:** A mapping consolidated similar body variants (e.g. all cab styles → "Pickup Cab", multiple coupe names → "Coupe") for robust categorization.

- **Integrated type_land:** Using USGS National Land Cover [https://www.usgs.gov/centers/eros/science/national-land-cover-database#data] data (and LLM lookup for missing states), each record was enriched with land-cover classifications.
- **Integrated fuel_consumption:** External Kaggle fuel-consumption data [https://www.kaggle.com/datasets/ahmettyilmazz/fuel-consumption] were merged and missing values imputed in three passes (make–body–year, body–year, body) to complete the attribute.

3 Logical Schema

The logical schema decomposes the raw CSV into normalized entities to support the construction of the attribute tree.
Measure Additivity In this schema, **SellingPrice** and **Odometer** are *additive* across all dimensions (you can sum total revenue or total miles driven by any combination of vehicle, seller, location or date). In contrast, **MMR** and **Fuel_Consumption** are *non-additive*: it makes sense to compute their averages (for example, average MMR per month or weighted average fuel consumption by sales volume) but summing them across records does not yield a meaningful measure.

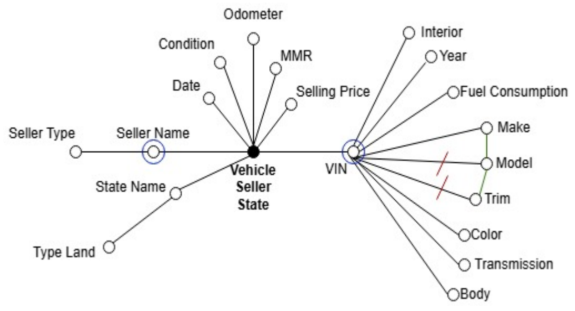
Entity	**Attributes (PK underlined, FKs in italics)**
Vehicle	VIN, Year, Make, Model, Trim, Body, Transmission, Fuel_Consumption, Color, Interior
Seller	Seller_Name, Seller_Type
Location	State_Name, Type_Land
Sale	*VIN*, *Seller_Name*, *State_Name*, Sale_Date, Condition, Odometer, MMR, SellingPrice

Relationships:

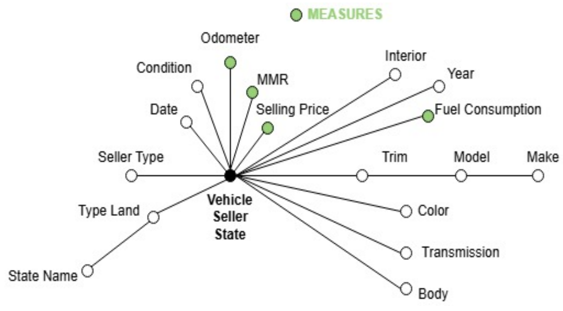
- Each *Sale* record is uniquely identified by the combination of (*VIN*, *Seller_Name*, *State_Name*) and references exactly one *Vehicle*, one *Seller* and one *Location*.
- A single *Vehicle* (VIN) can participate in multiple sales (e.g. remarketing events).
- A *Seller* may appear in many sales, and likewise each *Location* (state) can be associated with multiple transactions.
- The *Sale_Date* attribute allows temporal analyses directly within the 'Sale' table without a separate date dimension.

4 Attribute Tree

The attribute tree provides a hierarchical view of all the dataset's attributes, rooted at the composite entity (Vehicle, Seller, State). A pruned version highlights the measures in green, show the *Make, Model, Trim* hierarchy and clarify the dimensions of the fact.



(a) Initial Attribute Tree



(b) Pruned Attribute Tree

5 Fact Schema

All descriptive attributes (*VIN*, *Seller_Name*, *State_Name*) serve as foreign keys linking to the corresponding dimension tables (`Vehicle`, `Seller`, `Location`). Since we have aggregated individual sales by date—collapsing multiple raw events into a single "vehicle-by-date" record—this schema is a *temporal fact schema*. The grain differs from the original transactional source, enabling direct analysis of time-series trends (e.g. daily or monthly averages of price, odometer and condition) without further roll-up steps.

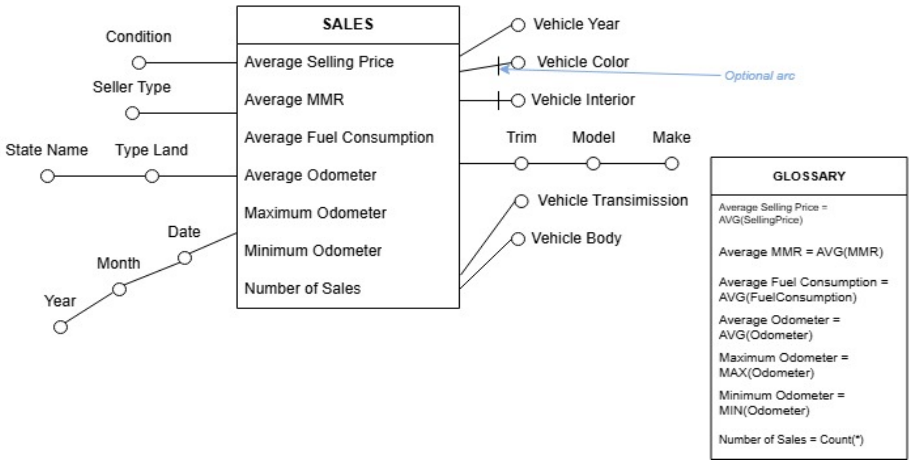


Figure 2: Fact Schema for the `Sale` fact and its links to Vehicle, Seller, Location and Date dimensions.

6 Star Schema

The star schema organizes the data for OLAP-style querying by placing the `Sale` fact table at the center, surrounded by its dimension tables. To streamline dimension management and reduce the number of small lookup tables, a *junk dimension* named `Vehicle_YCITBC` was created. This table aggregates several degenerate attributes—`Year`, `Color`, `Interior`, `Transmission`, `Body`, and `Condition`—into a single dimension. Since these attributes have no interdependencies and each has a limited set of distinct values, combining them avoids cluttering the schema with multiple trivial dimensions.

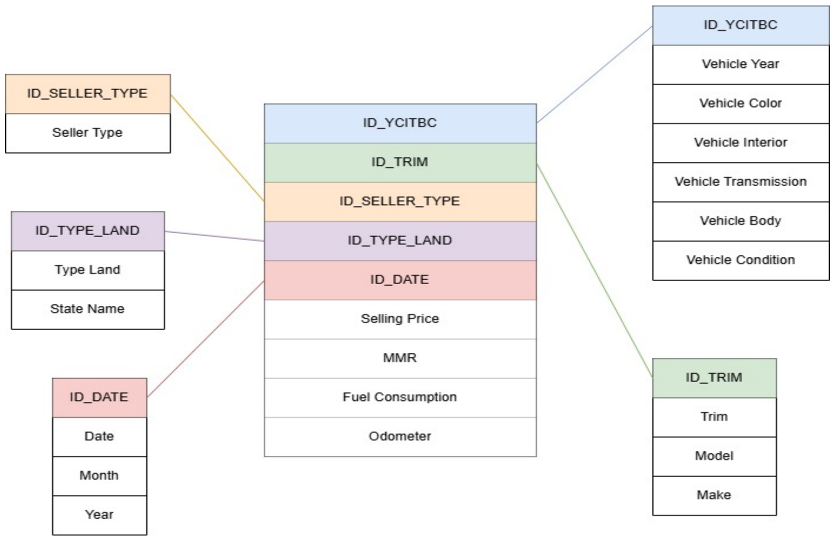


Figure 3: Star Schema and its links to the dimensions.