Data Curation Project

Sales Transaction Data

Project By:
Supriya Hinge | Jay Patel | Hrishika Shelar

Introduction

- Sample Sales Data : https://www.kaggle.com/datasets/kyanyoga/sample-sales-data
- Data Comes from a Company related to Automobile Industry.
- Dataset comprises of sample sales data and has 25 columns and about 3000 unique records.
- Includes columns representing order details, customer information, sales and shipping details, etc.
- Dataset is useful for Segmentation, Customer Analytics, Clustering, Sale Forecasting, Inventory Management and more.

Data Definition

Data Definition

- The raw sales dataset consist of Order Details places for given products.
- The dataset consist of **25 Order Attributes** and **2823 Row Orders**
- The datatypes in the dataset includes INT, FLOAT, STRING, STRING, DATETIME

Column Name	Data Type	Short Description	Notes
ORDERNUMBER	int	Order Identifier	
QUANTITYORDERED	int	# of Units ordered	
PRICEEACH	float	Unit Price of Each	This can be pushed to Product Table, and references here
ORDERLINENUMBER	int	?	Can be dropped
SALES	float	Sale Price of the order	
ORDERDATE	datetime	Date on which order was placed	Remove time component
STATUS	string	Status of the order	
QTR_ID	int	Quarter Number	Can be dropped since we can derive it from Order Date
MONTH_ID	int	Month Number	Can be dropped since we can derive it from Order Date
YEAR_ID	int	Year Number	Can be dropped since we can derive it from Order Date
PRODUCTCODE	string	Product Identifier	Create Product Table using this ID
CUSTOMERNAME	string	Name of the Company	Create Customer Table
PHONE	string	Customer Phone Number	Push to Customer Table
ADDRESSLINE1	string	Customer Physical Address	Push to Customer Table
ADDRESSLINE2	string	Customer Physical Address Extended	Push to Customer Table
CITY	string	Name of the City	Standardize
STATE	string	Name of the State	Standardize, Default for Country with no State
POSTALCODE	string	ZipCode	Standardize
COUNTRY	string	Name of the country	
TERRITORY	string	Name of the territory	
CONTACTLASTNAME	string	Representative Last name of Customer	Push to Customer Table
CONTACTFIRSTNAME	string	Representative First name of Customer	Push to Customer Table
DEALSIZE	string	size of order	Can be populated using automatic function

Issues identified in the dataset

Identifying incorrect and irrelevant data from different columns of dataset and transforming it into a format that can be understood and analyze further.

1. **ORDERDATE** - The ORDERDATE column, along with the date, has a timestamp of 0:00 for all entries.
2. **Phone** - Customer's phone numbers (PHONE) are in different formats.
3. **STATE** - STATE has missing values as well as different formats - Some fields have the entire state name spelled out vs some have the standard two syllable abbreviation used in most addresses.
4. **ADDRESSLINE2** - ADDRESSLINE2 has missing values.

What issues are we solving?

1. Normalizing Data
    a. Split Dataset into smaller tables
2. Data Profiling
    a. Data Definition
    b. Data Type
    c. Constraints (Min, Max Values)
3. Data Imputation
    a. Filling missing values
4. Standardizing Data
    a. Reformat column names
    b. Standardize columns such as address, countries, etc.
    c. Added Data Versioning

Normalizing the Dataset

Splitting data into smaller tables for query optimization.

- One of the issues with this dataset is that, it's in a wide format.
- There are a lot of fields in this table that belongs to one group, that can be broken down into smaller tables.
- for this Project, we are attempting to break down the dataset into **3 Components**
  - Order Table
  - Customer Table
  - Product Table
- Even though there are specific columns such as Countries, and States that can have their own tables. Most of the columns fall under these categories, and using VizierDB's temp dataset, we can create these three tables, however, we will be providing the CREATE TABLE Script to go along with it as well

Normalizing the Dataset - Orders

- Dataset attributes consist of Order Details.
- Each Row Represents a product purchase with specific Orde ID
- Sales are dynamically calculated by PriceEach x Quantity Ordered

```
SELECT
ORDERNUMBER,
QUANTITYORDERED,
PRODUCTCODE,
ORDERLINENUMBER,
PRICEEACH,
SALES,
ORDERDATE,
STATUS,
DEALSIZE,
'CS520_TEST' AS CREATEDBY,
'CS520_TEST' AS UPDATEDBY,
current_timestamp() AS CREATED_DATETIME,
current_timestamp() AS UPDATED_DATETIME
FROM raw_data
```

Output Dataset Orders

Normalizing the Dataset - Product

- Dimension Table for Product Details
- Each Row is Unique Product
- PRICEEACH Fluctuate between Min and Max, indicating the price change in the product over the period of time
- Number of Unique products -109

```
1 SELECT PRODUCTCODE,
2 LAST(PRODUCTLINE) AS PRODUCTLINE,
3 MAX(PRICEEACH) AS MAX_UNIT_PRICE,
4 MIN(PRICEEACH) AS MIN_UNIT_PRICE,
5 MAX(MSRP) AS MAX_RETAIL_PRICE
6 -- MIN(MSRP) AS MIN_RETAIL_PRICE MSRP I
7 FROM raw_data
8 GROUP BY PRODUCTCODE
```

Output Dataset: Product

Normalizing the Dataset - Customer

- Customer Table is a Dimension Table representing customers
- A Customer in this Dataset is a organization/company
- 92 Customers in Dataset

[20]

```
SELECT DISTINCT
CUSTOMERNAME,
CONTACTFIRSTNAME,
CONTACTLASTNAME,
PHONE,
ADDRESSLINE1,
ADDRESSLINE2,
CITY,
STATE,
POSTALCODE,
COUNTRY
FROM raw_data
ORDER BY CUSTOMERNAME
```

Standardizing Data

- Standardizing is process to convert the data in a generalized format
- Standardization helps to maintain data integrity, and are easily processed and accepted by applications without overhead transformation.
- **Task Completed:**
  - Converted Countries column to ISO 3166 Standard Values
  - Phone Numbers
  - Converted Datetime to Date format
  - Rename Columns
  - Added UpdatedBy, CreatedBy, Update DT, Created DT columns
  - Added Active Status Code to Customers who are inactive > 12 Months.

Standardization Techniques: Standardize Country Names

- Designed function to **convert** unformatted **Country Names** to ISO Standard
- Available options to have 2 Letter, 3 Letter Abbreviations and Full Country Name.
- Works on any column with country field

7	Italy
8	Norway
9	Spain
10	Denmark
11	Ireland
12	USA
13	UK
14	Switzerland

```python
import pycountry

# Standardize Country Name
def clean_country_name(country_name, alpha=2):
    cleaned_output = pycountry.countries.search_fuzzy(country_name)[0].alpha_2
    return cleaned_output

ds = vizierdb.get_data_frame('raw_data')

ds['COUNTRY_A2'] = ds['COUNTRY'].apply(lambda x: clean_country_name(x))
print(ds.head(5))
```

Standardization Techniques: Cleanup Phone Numbers

```
|PREV FORMAT| NEW FORMAT| COUNTRY
2125557818 +1 212-555-7818 US
26471555 +33 26471555 FR
33146627555 +33 1 46 62 75 55 FR
6265557265 +1 626-555-7265 US
6505551386 +1 650-555-1386 US
6505556809 +1 650-555-6809 US
20161555 +33 20161555 FR
4722673215 +47 22 67 32 15 NO
6505555787 +1 650-555-5787 US
147556555 +33 1 47 55 65 55 FR
```

- Function **clean_phone_number** designed to standardize phone number by country.
- It follows International Phone Format Standard.
- Used Libraries - Pycountry, phone_numbers

```python
import phonenumbers
import pycountry
import re


def clean_phone_numbers(phone_number, country):
    '''Reformat's passed argument with corrected phone number format based on the country.'''
    formated_phone = phonenumbers.parse(phone_number, country)
    formated_phone = phonenumbers.format_number(formated_phone, phonenumbers.PhoneNumberFormat.INTERNATIONAL)

    return formated_phone

#phonenumbers.format_number(phonenumbers.parse("8006397663", 'US'), phonenumbers.PhoneNumberFormat.NATIONAL)
# get dataframe
ds = vizierdb.get_dataset('phone_number')
#print(phonenumbers.parse("8006397663", 'US'))
print("|PREV FORMAT| NEW FORMAT| COUNTRY")
for row in ds.rows[:10]:
    phone = re.sub("[^0-9]","",row[0])
    country = pycountry.countries.search_fuzzy(row[1])[0].alpha_2
    print(phone,clean_phone_numbers(phone, country), country)
```

Data Imputation and Cleaning

- Data can be incomplete due to various reasons such as manual entry, data, etc. **Data Imputation & Cleaning** is required to streamline the data.
- For this dataset, we want to ensure all the fields are intact, and follow defined constraints.
- Task Completed:
  - Defaulted Order Status to ‘UNKNOWN’ for NULL values
  - Designed Function to fill ORDERLINENUMBER values correctly.
  - Dynamically assign DEALSIZE based on Quantity ordered.

Conclusion

- Sourced Dataset consisted of many columns that has to be broken down into multiple tables.
- The Dataset consisted information about **Order Details**, **Product Details**, and **Customers**.
- **Standardizing** Columns such as **Phone Number** and **Country** benefits the application consuming data downstream.
- Certain Dataset fields required to be **defaulted** on **NULL**, and certain fields can be **calculated based on other fields** such as Sales and Deal Size.
- The outcome of the exercise is a **well-curated**, **normalized**, & **clean dataset**