Data Curation | Projects | Branches | Notebook | Dataset | Caveats | Settings

On branch **Untitled Branch**

```
[1] LOAD DATASET raw_data AS csv FROM Raw_Data.csv @ artifact file 2
```

Console | Timing | Datasets | Charts

raw_data (**2823** rows)

Views

	ORDERNUMBER (short)	QUANTITYORDERED (short)	PRICEEACH (float)	ORDERLINENUMBER (short)	SALES (float)	ORDERDATE (string)	STATUS (string)	QTR
0	10107	30	95.69999694824219	2	2871	2/24/2003 0:00	Shipped	1
1	10121	34	81.3499984741211	5	2765.89990234375	5/7/2003 0:00	Shipped	2
2	10134	41	94.73999786376953	2	3884.340087890625	7/1/2003 0:00	Shipped	3
3	10145	45	83.26000213623047	6	3746.699951171875	8/25/2003 0:00	Shipped	3
4	10159	49	100	14	5205.27001953125	10/10/2003 0:00	Shipped	4
5	10168	36	96.66000366210938	1	3479.760009765625	10/28/2003 0:00	Shipped	4
6	10180	29	86.12999725341797	9	2497.77001953125	11/11/2003 0:00	Shipped	4
7	10188	48	100	1	5512.31982421875	11/18/2003 0:00	Shipped	4
8	10201	22	98.56999969482422	2	2168.5400390625	12/1/2003 0:00	Shipped	4
9	10211	41	100	14	4708.43994140625	1/15/2004 0:00	Shipped	1
10	10223	37	100	1	3965.659912109375	2/20/2004 0:00	Shipped	1
11	10237	23	100	7	2333.1201171875	4/5/2004 0:00	Shipped	2
12	10251	28	100	2	3188.639892578125	5/18/2004 0:00	Shipped	2
13	10263	34	100	2	3676.760009765625	6/28/2004 0:00	Shipped	2
14	10275	45	92.83000183105469	1	4177.35009765625	7/23/2004 0:00	Shipped	3
15	10285	36	100	6	4099.68017578125	8/27/2004 0:00	Shipped	3
16	10299	23	100	9	2597.389892578125	9/30/2004 0:00	Shipped	3
17	10309	41	100	5	4394.3798828125	10/15/2004 0:00	Shipped	4
18	10318	46	94.73999786376953	1	4358.0400390625	11/2/2004 0:00	Shipped	4
19	10329	42	100	1	4396.14013671875	11/15/2004 0:00	Shipped	4

```
[2]
```

Console | Timing | Datasets | Charts

Fixing Sales Data

CS520: Data Curation Project

Hrishika Shelar | Supriya Hinge | Jay Patel

Overview

Data Curation is an essential practice to clean data for business purposes. In this project, will apply the techniques learned in class to clean and integrate one or more real world datasets. The project includes following elements.

1. Extracting Data
2. Loading Data in VizierDB Environment
3. Data Exploration
4. Data Definition
5. Data Normalization
- Create STAR Schema Model
6. Data Cleaning
- Impute Null Values
- Drop Unwanted Columns
- Convert Data Types with datatype mismatch
7. Data Standardization
- Standardize Time Fields
- Standardize Geolocation Fields
- Add Versioning such as Created By, Updated By, Created Datetime, Updated Datetime.

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

[4]

Console | Timing | Datasets | Charts

Initialize Data Exploration

In this section, we will familiarize ourselves with the core dataset at hand, and try to figure out all the setbacks, problems and complications.

The process involves using both SQL and Python Capabilities of Vizier. Our intention is to show the team's strong-hand at data handling in any programming language.

[5]

```
-- Get Account of Dateoutliers & Date Range
SELECT
ORDERDATE,
COUNT(1)
FROM raw_data
GROUP BY ORDERDATE
ORDER BY ORDERDATE DESC;
```

Console | Timing | Datasets | Charts

temporary_dataset (252 rows)

Views

	ORDERDATE (string)	count(1) (long)
0	9/9/2004 0:00	9
1	9/8/2004 0:00	26
2	9/7/2004 0:00	2

4	9/30/2004 0:00	11
5	9/3/2004 0:00	4
6	9/3/2003 0:00	2
7	9/28/2003 0:00	13
8	9/27/2004 0:00	2
9	9/25/2003 0:00	4
10	9/21/2003 0:00	10
11	9/19/2003 0:00	11
12	9/16/2004 0:00	7
13	9/15/2004 0:00	14
14	9/12/2003 0:00	11
15	9/11/2003 0:00	14
16	9/10/2004 0:00	6
17	9/1/2004 0:00	14
18	8/9/2004 0:00	6
19	8/8/2003 0:00	16

[6]

```
-- Get Min and Max Dates
SELECT MIN(ORDERDATE), MAX(ORDERDATE) FROM raw_data;
```

Console Timing Datasets Charts

temporary_dataset (1 rows)

Views

	min(ORDERDATE) (string)	max(ORDERDATE) (string)
0	1/10/2003 0:00	9/9/2004 0:00

[7]

```
SELECT DISTINCT STATUS FROM raw_data
```

Console Timing Datasets Charts

temporary_dataset (6 rows)

Views

	STATUS (string)
0	Shipped
1	On Hold
2	Cancelled
3	Resolved
4	In Process
5	Disputed

[8]

Console Timing Datasets Charts

We can notice that we are dealting with the Sales data between January 10, 2003 and September 9, 2004. 1 Yr 9 Mnths. Within which, the orders were places for 252 days. As we can see, from the data values, there is an unnecessary dangling H:MM values along with the date field.

TO DO- Remove Time Aspect from the **ORDERDATE** field, to make the data look cleaner, minimize complexity**. TO DO-** Standaridize Status into *OrderStatusCode* that can be consumed in downstream data flow**.**

```
ds = vizierdb.get_data_frame("raw_data")

print("Null Values by Column")
print(ds.isna().sum())
```

Console Timing Datasets Charts

```
Null Values by Column
ORDERNUMBER           0
QUANTITYORDERED       0
PRICEEACH             0
ORDERLINENUMBER       0
SALES                 0
ORDERDATE             0
STATUS                0
QTR_ID                0
MONTH_ID              0
YEAR_ID               0
PRODUCTLINE           0
MSRP                  0
PRODUCTCODE           0
CUSTOMERNAME          0
PHONE                 0
ADDRESSLINE1          0
ADDRESSLINE2       2521
CITY                  0
STATE              1486
POSTALCODE          541
COUNTRY               0
TERRITORY             0
CONTACTLASTNAME       0
CONTACTFIRSTNAME      0
DEALSIZE              0
dtype: int64
```

[10]

```
SELECT PHONE, COUNTRY FROM raw_data
```

Console Timing Datasets Charts

phone_number (**2823** rows)

Views

	PHONE (string)	COUNTRY (string)
0	2125557818	USA
1	26.47.1555	France
2	+33 1 46 62 7555	France
3	6265557265	USA
4	6505551386	USA
5	6505556809	USA
6	20.16.1555	France
7	+47 2267 3215	Norway
8	6505555787	USA
9	(1) 47.55.6555	France
10	03 9520 4555	Australia
11	2125551500	USA
12	2015559350	USA
13	2035552570	USA
14	40.67.8555	France
15	6175558555	USA

17	07-98 9555	Norway
18	2155551555	USA
19	2125557818	USA

[11]

```
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('phone_number')

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

Console Timing Datasets Charts

```
  PHONE COUNTRY COUNTRY_A2
  0        2125557818     USA         US
  1        26.47.1555  France         FR
  2  +33 1 46 62 7555  France         FR
  3        6265557265     USA         US
  4        6505551386     USA         US
```

[12]

```
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)
```

Console Timing Datasets Charts

```
  |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
```

Normalizing the Data Set

One of the issues with this dataset is that, its 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.

Even though there are specific columns such as Countries, and States that can have their own tables, for this Project, we are attempting to break down the dataset into **3 Components** 1. Order Table 2. Customer Table 3. Product Table

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

[14]

Console | Timing | Datasets | Charts

ORDER TABLE

The Order Table Consists of the following Schema

Table Name: Orders

Columns:

1. id, identity, int
2. ORDERNUMBER, int PK
3. QUANTITYORDERED, int
4. PRODUCTID, int FK
5. ORDERLINENUMBER, int
6. PRICEEACH, int
7. SALES, float {Although this can be dynamically calculated)
8. ORDERDATE, date
9. STATUS, string
10. DEALSIZE, string
11. CUSTOMERID, string FK

[15]

```
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
```

Console | Timing | Datasets | Charts

orders (2823 rows)

Views

	ORDERNUMBER (short)	QUANTITYORDERED (short)	PRODUCTCODE (string)	ORDERLINENUMBER (short)	PRICEEACH (float)	SALES (float)	ORDERDATE (strin
0	10107	30	S10_1678	2	95.69999694824219	2871	2/24/2003 0:00
1	10121	34	S10_1678	5	81.3499984741211	2765.89990234375	5/7/2003 0:00
2	10134	41	S10_1678	2	94.73999786376953	3884.340087890625	7/1/2003 0:00
3	10145	45	S10_1678	6	83.26000213623047	3746.699951171875	8/25/2003 0:00
4	10159	49	S10_1678	14	100	5205.27001953125	10/10/2003 0:00
5	10168	36	S10_1678	1	96.66000366210938	3479.760009765625	10/28/2003 0:00

7	10188	48	S10_1678	1	100	5512.31982421875	11/18/2003 0:00
8	10201	22	S10_1678	2	98.56999969482422	2168.5400390625	12/1/2003 0:00
9	10211	41	S10_1678	14	100	4708.43994140625	1/15/2004 0:00
10	10223	37	S10_1678	1	100	3965.659912109375	2/20/2004 0:00
11	10237	23	S10_1678	7	100	2333.1201171875	4/5/2004 0:00
12	10251	28	S10_1678	2	100	3188.639892578125	5/18/2004 0:00
13	10263	34	S10_1678	2	100	3676.760009765625	6/28/2004 0:00
14	10275	45	S10_1678	1	92.83000183105469	4177.35009765625	7/23/2004 0:00
15	10285	36	S10_1678	6	100	4099.68017578125	8/27/2004 0:00
16	10299	23	S10_1678	9	100	2597.389892578125	9/30/2004 0:00
17	10309	41	S10_1678	5	100	4394.3798828125	10/15/2004 0:00
18	10318	46	S10_1678	1	94.73999786376953	4358.0400390625	11/2/2004 0:00
19	10329	42	S10_1678	1	100	4396.14013671875	11/15/2004 0:00

[16]

Console Timing Datasets Charts

PRODUCT

The Order Table consist of the following Schema

Table Name: Product

Columns: 1. ID, Identity, int 2. PRODUCTCODE, int 3. PRICEEACH, float 4. PRODUCTLINE, string 5. MSRP, float

[17]

```
SELECT PRODUCTCODE,
LAST(PRODUCTLINE) AS PRODUCTLINE,
MAX(PRICEEACH) AS MAX_UNIT_PRICE,
MIN(PRICEEACH) AS MIN_UNIT_PRICE,
MAX(MSRP) AS MAX_RETAIL_PRICE
-- MIN(MSRP) AS MIN_RETAIL_PRICE MSRP IS SAME
FROM raw_data
GROUP BY PRODUCTCODE
```

Console Timing Datasets Charts

product (**109** rows)

Views

	PRODUCTCODE (string)	PRODUCTLINE (string)	MAX_UNIT_PRICE (float)	MIN_UNIT_PRICE (float)	MAX_RETAIL_PRICE (short)
0	S18_4600	Trucks and Buses	100	57.529998779296875	121
1	S18_1749	Vintage Cars	100	74.04000091552734	170
2	S12_3891	Classic Cars	100	62.09000015258789	173
3	S18_2248	Vintage Cars	100	49.040000915527344	60
4	S700_1138	Ships	100	54	66
5	S32_1268	Trucks and Buses	100	80.9000015258789	96
6	S12_1099	Classic Cars	100	71.47000122070312	194
7	S18_2795	Vintage Cars	100	47.18000030517578	168
8	S24_1937	Vintage Cars	100	26.8799991607666	33
9	S32_3522	Trucks and Buses	100	54.939998626708984	64
10	S18_1097	Trucks and Buses	100	40.25	116
11	S18_1662	Planes	100	54.56999969482422	157
12	S12_1666	Trucks and Buses	100	63.20000076293945	136
13	S24_3969	Vintage Cars	97.44000244140625	33.22999954223633	41
14	S24_1578	Motorcycles	100	84.38999938964844	112

16	S18_3320	Vintage Cars	100	73.08000183105469	99
17	S24_3816	Vintage Cars	100	59.36000061035156	83
18	S18_3136	Vintage Cars	100	39.79999923706055	104
19	S32_2509	Trucks and Buses	100	43.290000915527344	54

[18]

Console Timing Datasets Charts

CUSTOMER Table

- The customer table consist of information about customer. A customer in this dataset is an Organization/Company.
- Customer related fields are, Address, Phone number, Contact person etc.
- Each Row of Customer Table signifies a CustomerEmployee (Contact Person)
- Below is the defined table schema

Table Name: Customer
Columns: 1. CUSTOMERID. int, IDENTITY PK
2. CUSTOMERNAME, string
3. PHONE, string
4. ADDRESSLINE1, String
5. ADDRESSLINE2, String
6. CITY, String
7. STATE, String
8. POSTALCODE, String
9. COUNTRY, String
10. COUNTRYCODE, String
11. CONTACTLASTNAME
12. CONTACTFIRSTNAME

[19]

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

Console Timing Datasets Charts

temporary_dataset (**92** rows)

Views

	CUSTOMERNAME (string)	CONTACTFIRSTNAME (string)	CONTACTLASTNAME (string)	PHONE (string)	ADDRESSLINE1 (string)	ADDRESSLINE2
0	AV Stores, Co.	Victoria	Ashworth	(171) 555-1555	Fauntleroy Circus	
1	Alpha Cognac	Annette	Roulet	61.77.6555	1 rue Alsace-Lorraine	
2	Amica Models & Co.	Paolo	Accorti	011-4988555	Via Monte Bianco 34	
3	Anna's Decorations, Ltd	Anna	O'Hara	02 9936 8555	201 Miller Street	Level 15
4	Atelier graphique	Carine	Schmitt	40.32.2555	54, rue Royale	
5	Australian Collectables, Ltd	Sean	Connery	61-9-3844-6555	7 Allen Street	
6	Australian Collectors, Co.	Peter	Ferguson	03 9520 4555	636 St Kilda Road	Level 3
7	Australian Gift Network, Co	Tony	Calaghan	61-7-3844-6555	31 Duncan St. West End	
8	Auto Assoc. & Cie.	Daniel	Tonini	30.59.8555	67, avenue de l'Europe	
9	Auto Canal Petit	Dominique	Perrier	(1) 47.55.6555	25, rue Lauriston	
10	Auto-Moto Classics Inc.	Leslie	Taylor	6175558428	16780 Pompton St.	

12	Bavarian Collectables Imports, Co.	Michael	Donnermeyer	+49 89 61 08 9555	Hansastr. 15	
13	Blauer See Auto, Co.	Roland	Keitel	+49 69 66 90 2555	Lyonerstr. 34	
14	Boards & Toys Co.	Leslie	Young	3105552373	4097 Douglas Av.	
15	CAF Imports	Jesus	Fernandez	+34 913 728 555	Merchants House, 27-30 Merchant's Quay	
16	Cambridge Collectables Co.	Kyung	Tseng	6175555555	4658 Baden Av.	
17	Canadian Gift Exchange Network	Yoshi	Tannamuri	(604) 555-3392	1900 Oak St.	
18	Classic Gift Ideas, Inc	Francisca	Cervantes	2155554695	782 First Street	
19	Classic Legends Inc.	Maria	Hernandez	2125558493	5905 Pompton St.	Suite 750

[20]

```
SELECT DISTINCT PHONE FROM raw_data
```

Console Timing Datasets Charts

temporary_dataset (91 rows)

Views

	PHONE (string)
0	86 21 3555
1	2035554407
2	03 9520 4555
3	6175557555
4	2125557413
5	02 9936 8555
6	4085553659
7	6505555787
8	61-9-3844-6555
9	(198) 555-8888
10	61-7-3844-6555
11	(02) 5554 67
12	7605558146
13	(071) 23 67 2555
14	6505556809
15	(171) 555-7555
16	9145554562
17	6265557265
18	3105552373
19	2125551957

[21]

```
import pandas as pd
df = vizierdb.get_data_frame('raw_data')

# change ORDERSTATUS field to Datetime
df['ORDERDATE_DATETIME'] = pd.to_datetime(df['ORDERDATE'], format="%m/%d/%Y %H:%M")

## Extract DATE
df['ORDERDATE_CLEAN'] = df['ORDERDATE_DATETIME'].dt.date

print("ORDERDATE Comparision")
print("*"*20)
print(df[['ORDERDATE','ORDERDATE_DATETIME','ORDERDATE_CLEAN',]])
```

Console Timing Datasets Charts

```
**********
ORDERDATE ORDERDATE_DATETIME ORDERDATE_CLEAN
0      2/24/2003 0:00         2003-02-24      2003-02-24
1       5/7/2003 0:00         2003-05-07      2003-05-07
2       7/1/2003 0:00         2003-07-01      2003-07-01
3      8/25/2003 0:00         2003-08-25      2003-08-25
4     10/10/2003 0:00         2003-10-10      2003-10-10
...             ...                ...             ...
2818  12/2/2004 0:00         2004-12-02      2004-12-02
2819  1/31/2005 0:00         2005-01-31      2005-01-31
2820   3/1/2005 0:00         2005-03-01      2005-03-01
2821  3/28/2005 0:00         2005-03-28      2005-03-28
2822   5/6/2005 0:00         2005-05-06      2005-05-06

[2823 rows x 3 columns]
```

[22]

Console Timing Datasets Charts

Postal Code

[23]

```
from pyzipcode import ZipCodeDatabase
zcdb = ZipCodeDatabase()

print(zcdb.find_zip(city="Chicago")[0].zip)
```

Console Timing Datasets Charts

```
60601
```

[24]

```
from pyzipcode import ZipCodeDatabase
zcdb = ZipCodeDatabase()
df = vizierdb.get_data_frame('raw_data')

def get_zip_from_city(city):
    try:
        zip = zcdb.find_zip(city=city)[0].zip
    except Exception as e:
        zip = 'NA'

    return str(zip)

df['POSTALCODE_DERIVED'] = df['CITY'].apply(lambda x: get_zip_from_city(x))

print(df[['CITY','POSTALCODE','POSTALCODE_DERIVED']])
```

Console Timing Datasets Charts

```
CITY  POSTALCODE POSTALCODE_DERIVED
0              NYC     10022.0               NA
1            Reims     51100.0               NA
2            Paris     75508.0            04271
3          Pasadena     90003.0            21122
4     San Francisco         NaN            94101
...             ...         ...              ...
2818         Madrid     28034.0            13660
2819           Oulu     90110.0               NA
2820         Madrid     28034.0            13660
2821       Toulouse     31000.0               NA
2822         Boston     51003.0            02101
```

[25]

Console Timing Datasets Charts

Missing Value Operation

1. Filling OrderStatus to Unknown
2. Rearrange Item Number

[26]

```
df = vizierdb.get_data_frame('raw_data')

# fill order status with UNKNOWN

df['STATUS'] = df['STATUS'].fillna('UNKNOWN')
```

Console Timing Datasets Charts

[27]

```
df = vizierdb.get_data_frame('raw_data')

df['ORDERLINENUMBER_2'] = df.sort_values(by='ORDERLINENUMBER').groupby(['ORDERNUMBER'], as_index=False).cumcount()+1
print(df[['ORDERNUMBER','ORDERLINENUMBER','ORDERLINENUMBER_2']][df['ORDERNUMBER'] == 10350])
```

Console Timing Datasets Charts

ORDERNUMBER	ORDERLINENUMBER	ORDERLINENUMBER_2	
126	10350	5	5
955	10350	6	6
1008	10350	1	1
1085	10350	2	2
1310	10350	3	3
1611	10350	7	7
1840	10350	9	9
1964	10350	10	10
2270	10350	14	14
2399	10350	8	8
2426	10350	17	17
2478	10350	11	11
2529	10350	4	4
2609	10350	12	12
2712	10350	13	13
2738	10350	16	16
2818	10350	15	15