On branch **default**

[1]

Console | Timing | Datasets | Charts

Introduction to the Business Dataset

The dataset selected for the data curation project is from Yelp, which is a business directory service and crowd-sourced review forum. It contains information about various businesses, capturing details such as the business's name, address, city, state, postal code, latitude, longitude, star rating, review count, attributes, categories, and operational hours

[2]
```
LOAD DATASET Business_Data AS csv FROM yelp_academic_dataset_business_project.csv @ artifact file 41
```

Console | Timing | Datasets | Charts

business_data (10000 rows)

Views

	business_id (string)	name (string)	address (string)	city (string)	state (string)	postal_code (int)	latitude (float)	
0	TacYUYhU3HpLHF9Rs6fW2w	Steps to Learning Montessori Preschool	6901 Phelps Rd	Goleta	CA	93117	34.423309326171875	-1
1	RnExaICvIeXxFpbIKEqJsQ	Breeze Blow Dry Bar	9916 Clayton Rd	St. Louis	MO	63124	38.636714935302734	-9
2	pjtjBeZC3gvmtIiIQt-DFA	Impact Guns	11655 W Executive Dr	Boise	ID	83713	43.608699798583984	-1
3	OMl5pTUBVUjW_jvDKLvYtw	Palms Primary Care	1615 Pasadena Ave S, Ste 430	Saint Petersburg	FL	33707	27.752653121948242	-8
4	MO-LTDfO843xaRtW0bx6jQ	J&Q Nails	9655 E US Hwy 36, Unit H	Avon	IN	46123	39.763057708740234	-8
5	jFq8QSWDwtAWMA1FaNvRhw	Candy Barrel	735 Dodecanese Blvd	Tarpon Springs	FL	34689	28.155075073242188	-8
6	t53MqkLTtWxJ7jMSqXYz-g	Luminosity	690 W Dekalb Pike	King of Prussia	PA	19406	40.08949661254883	-7
7	dY6rzL7Gw1U5afOskTlMmg	Nail Care Salon	12337 Olive Blvd	Creve Coeur	MO	63141	38.67354202270508	-9
8	VIJ4wKPf2TmKbaOTKYHROg	Architectural Antiques of Indianapolis	5000 W 96th St	Indianapolis	IN	46268	39.92622375488281	-8
9	luKCyfSY7AKhRbRA1JPlPw	Aster's Floral Shop	41 Haddon Ave	Collingswood	NJ	8108	39.91556167602539	-7
10	6XOn1p3sbO22UjJGpmCgxg	China Wok	4319 Telegraph Rd	Saint Louis	MO	63129	38.486759185791016	-9
11	c8d8h47cogM_B_ZMC-h3zg	Brandon Family Medical Care	1218 Millennium Pkwy	Brandon	FL	33511	27.929439544677734	-8
12	YEwmI50bso_LYtuivsdwiA	7-Eleven	13151 Race Track Rd	Tampa	FL	33626	28.07027244567871	-8
13	6IG4SysBKyRnFW_e22q13A	Uber		Philadelphia	PA	19107	39.955928802490234	-7
14	bNBi-RVlx71bugXY0GRLtQ	Chestnut St. Cafe	4403 Chestnut St	Philadelphia	PA	19104	39.95672607421875	-7
15	rbPK4jSyFS10zhWYvo_Srg	Cafe Porche and snowbar	1625 Baronne St	New Orleans	LA	70113	29.939315795898438	-9
16	lrO5vOwa7gE0qAo0UQJYlA	Eyeglass World	13002 Seminole Blvd, Ste 10-11	Largo	FL	33778	27.891815185546875	-8
17	thpDDcdKLPzSFldLLBULLA	Callahan's Corner	914 Edwardsville Rd	Troy	IL	62294	38.732933044433594	-8
18	-EyRrBY1d-EQZrTlXWH4BQ	Spa Guy Dave		Pennsauken	NJ	8109	39.967105865478516	-7
19	EkthrfcRWCVYy-NuvfNmPg	FroYo Frozen Yogurt	4663 Maryland Ave	Saint Louis	MO	63108	38.64484405517578	-9

[3]

Console | Timing | Datasets | Charts

Preliminary Data Characteristics

- Some basic statistics about the dataset such as the number of rows (entries), number of attributes (columns), and the data format.
- Checking how many missing values are present in each column.
- Checking if there are any duplicate entries in the dataset based on the business_id, which should be unique for each business.

[4]
```
# reading the dataset as a datafram in python
dataset = vizierdb.get_data_frame('Business_Data')
print(dataset.head())
print("-" * 100,"\n")

# calculating the dataset characteristics
total_rows = len(dataset)
total_columns = len(dataset.columns)
data_types = dataset.dtypes

print('The total number of rows are - ',total_rows)
print('The total number of features are - ',total_columns)
print('The datatype of various features is -\n',data_types)
```

Console | Timing | Datasets | Charts

```
business_id  ...                                              hours
0  TacYUYhU3HpLHF9Rs6fW2w  ...  Education, Elementary Schools, Child Care & Da...
1  RnExaICvIeXxFpbIKEqJsQ  ...                                'validated': False
2  pjtjBeZC3gvmtIiIQt-DFA  ...                                'validated': False
3  OMl5pTUBVUjW_jvDKLvYtw  ...  {'Monday': '8:30-17:0', 'Tuesday': '8:30-17:0'...
4  MO-LTDfO843xaRtW0bx6jQ  ...                                'validated': False
```

```
[5 rows x 14 columns]

--------------------------------------------------------------------------------------------------------------

The total number of rows are -  10000
The total number of features are -  14
The datatype of various features is -
business_id      object
name             object
address          object
city             object
state            object
postal_code     float64
latitude        float32
longitude       float32
stars           float32
review_count      int16
is_open            bool
attributes       object
categories       object
hours            object
dtype: object
```

[5]

```
dataset = vizierdb.get_data_frame('Business_Data')
# checking missing vals
missing_values = dataset.isnull().sum()
print('Checking the Null/Missing values Feature-wise \n', missing_values)

# checking for duplicate rows based on the business_id
duplicate_rows = dataset[dataset.duplicated(subset='business_id')]
duplicate_rows_count = len(duplicate_rows)
print('Checking the duplicate rows \n',duplicate_rows_count)
```

Console | Timing | Datasets | Charts

```
Checking the Null/Missing values Feature-wise
business_id       0
name              0
address         349
city              0
state             0
postal_code     372
latitude          0
longitude         0
stars             0
review_count      0
is_open           0
attributes      964
categories        4
hours           649
dtype: int64

Checking the duplicate rows
0
```

[6]

Console | Timing | Datasets | Charts

Data Cleaning Strategies

1. Handling the Missing Values for Column 'Attribute and Hours'

Imputation Based on Business Categories

- For Attributes and Hours: For each category c in our DataFrame, find the most common value v in attributes and hours. For all rows r where r.categories = c and r.attribute (or r.hours) is null, assign v to r.attribute (or r.hours). If no common value is found, assign 'Not specified'.

2. Inference from Existing Data

- For all pairs of rows x and y in our DataFrame, if the postal_code of x is null and x and y have the same city and state, then assign the postal_code of y to x.

3. Consistency in Data Types

- For all rows in our DataFrame that have a postal_code, ensure that the data type of postal_code is a string.

4. Dropped rows with missing 'categories'

- Given the significance of the 'categories' column in the dataset and the low number of missing values (only 4), it is decided to drop these rows to maintain data integrity.

5. Geocoding for Missing Addresses and Postal Codes

- For Address and Postal Code: For each row r in our DataFrame, if r.address or r.postal_code is null, use the OpenStreetMap Nominatim API to perform reverse geocoding based on r.latitude and r.longitude. Assign the obtained address to r.address and the postal code to r.postal_code.

[7]

```
SELECT categories, attributes, COUNT(*) AS frequency
FROM business_data
WHERE attributes IS NOT NULL
GROUP BY categories, attributes
ORDER BY categories, frequency DESC;
```

Console | Timing | Datasets | Charts

common_attributes (5985 rows)

Views

	categories (string)	attributes (string)
0	'AcceptsInsurance': 'False'	"{'WiFi': "u'free'""
1	'AcceptsInsurance': 'False'	"{'BikeParking': 'True', 'ByAppointmentOnly': 'True', 'WiFi': "u'free'""
2	'AcceptsInsurance': 'False'	"{'BusinessAcceptsCreditCards': 'True', 'WiFi': "u'no'""
3	'AcceptsInsurance': 'False'	"{'WiFi': "u'paid'""
4	'AcceptsInsurance': 'False'	"{'BusinessAcceptsCreditCards': 'True', 'GoodForKids': 'True', 'ByAppointmentOnly': 'True', 'WiFi': "u'free'""
5	'AcceptsInsurance': 'False'	"{'BusinessAcceptsCreditCards': 'True', 'WheelchairAccessible': 'True', 'BikeParking': 'True', 'WiFi': "u'free'""
6	'AcceptsInsurance': 'False'	"{'RestaurantsPriceRange2': '1', 'WheelchairAccessible': 'True', 'OutdoorSeating': 'False', 'Caters': 'True', 'RestaurantsTakeOut': 'True', 'RestaurantsDeliver
7	'AcceptsInsurance': 'False'}"	"{'BusinessAcceptsCreditCards': 'True', 'ByAppointmentOnly': 'True', 'WiFi': "u'no'""
8	'AcceptsInsurance': 'True'	"{'ByAppointmentOnly': 'True', 'WiFi': "u'free'""
9	'AcceptsInsurance': 'True'	"{'BusinessAcceptsCreditCards': 'True', 'WiFi': "u'free'""
10	'AcceptsInsurance': 'True'	"{'WiFi': "u'free'""
11	'AcceptsInsurance': 'True'	"{'RestaurantsPriceRange2': '3', 'BikeParking': 'True', 'BusinessAcceptsCreditCards': 'True', 'WheelchairAccessible': 'True', 'ByAppointmentOnly': 'True', 'W
12	'AcceptsInsurance': 'True'	"{'ByAppointmentOnly': 'False', 'BusinessAcceptsCreditCards': 'True', 'BikeParking': 'False', 'WiFi': "u'no'""
13	'AcceptsInsurance': 'True'}"	"{'ByAppointmentOnly': 'True', 'BusinessAcceptsCreditCards': 'True', 'WiFi': "u'free'""
14	'AcceptsInsurance': 'True'}"	"{'ByAppointmentOnly': 'False', 'BusinessAcceptsCreditCards': 'True', 'WiFi': "u'free'""
15	'AcceptsInsurance': 'True'}"	"{'BusinessAcceptsCreditCards': 'True', 'ByAppointmentOnly': 'False', 'WiFi': "u'no'""
16	'AcceptsInsurance': 'True'}"	"{'BusinessAcceptsCreditCards': 'True', 'ByAppointmentOnly': 'True', 'WiFi': "u'free'""
17	'AcceptsInsurance': 'True'}"	"{'WheelchairAccessible': 'True', 'BusinessAcceptsCreditCards': 'True', 'BikeParking': 'True', 'WiFi': "u'no'""
18	'AcceptsInsurance': 'True'}"	"{'BusinessAcceptsCreditCards': 'True', 'ByAppointmentOnly': 'True', 'WiFi': "u'paid'""
19	'Alcohol': "'beer_and_wine'""	"{'BusinessAcceptsCreditCards': 'True', 'RestaurantsAttire': "u'casual'""

[8]

```
SELECT categories, hours, COUNT(*) AS frequency
FROM business_data
WHERE hours IS NOT NULL
GROUP BY categories, hours
ORDER BY categories, frequency DESC;
```

Console | Timing | Datasets | Charts

common_hours (3951 rows)

Views

	categories (string)	hours (string)	frequency (long)
0	'AcceptsInsurance': 'False'	'BusinessAcceptsCreditCards': 'True'	3
1	'AcceptsInsurance': 'False'	'BikeParking': 'True'	2
2	'AcceptsInsurance': 'False'	'BusinessParking': "{'garage': False	2
3	'AcceptsInsurance': 'False'	'RestaurantsPriceRange2': '2'	1
4	'AcceptsInsurance': 'False'	'ByAppointmentOnly': 'True'}"	1
5	'AcceptsInsurance': 'False'	'ByAppointmentOnly': 'False'	1
6	'AcceptsInsurance': 'False'}"	Chiropractors, Health & Medical	1
7	'AcceptsInsurance': 'True'	'BusinessAcceptsCreditCards': 'True'}"	4
8	'AcceptsInsurance': 'True'	'ByAppointmentOnly': 'True'}"	2
9	'AcceptsInsurance': 'True'	'BusinessAcceptsBitcoin': 'False'}"	2
10	'AcceptsInsurance': 'True'}"	Health & Medical, Dentists, Orthodontists	1
11	'AcceptsInsurance': 'True'}"	Massage, Skin Care, Health & Medical, Hair Removal, Waxing, Beauty & Spas, Massage Therapy	1
12	'AcceptsInsurance': 'True'}"	Cosmetic Dentists, Health & Medical, Dentists, General Dentistry, Orthodontists	1
13	'AcceptsInsurance': 'True'}"	Doctors, Health & Medical, Physical Therapy, Massage Therapy, Chiropractors	1
14	'AcceptsInsurance': 'True'}"	Health & Medical, Chiropractors, Massage Therapy, Physical Therapy, Doctors	1
15	'AcceptsInsurance': 'True'}"	Periodontists, Cosmetic Dentists, Dentists, General Dentistry, Health & Medical	1
16	'AcceptsInsurance': 'True'}"	Dentists, Cosmetic Dentists, Health & Medical, General Dentistry	1
17	'AcceptsInsurance': 'True'}"	Chiropractors, Health & Medical	1
18	'Alcohol': "'beer_and_wine'""	'NoiseLevel': "u'average'""	1
19	'Alcohol': "'beer_and_wine'""	'BikeParking': 'True'	1

[9]

```
import pandas as pd
```

```python
business_df = vizierdb.get_data_frame('Business_Data')

# func to impute values based on business categories
def impute_by_category(df, column_name):
    placeholder = "Not specified"
    imputed_count = 0
    p_assigned_count = 0

    # group by categories then find the most common value in each group
    most_common_per_category = df.groupby('categories')[column_name].agg(
        lambda x: x.mode()[0] if not x.mode().empty else placeholder
    )

    # apply the imputation
    for category, most_common_value in most_common_per_category.items():
        missing_in_category = (df['categories'] == category) & (df[column_name].isnull())
        imputed_count += missing_in_category.sum()
        df.loc[missing_in_category, column_name] = most_common_value

    # rest are assigned the placeholder
    remaining_missing = df[column_name].isnull()
    p_assigned_count = remaining_missing.sum()
    df[column_name].fillna(placeholder, inplace=True)

    return imputed_count, p_assigned_count

# counting the values
attributes_imputed, attributes_p_assigned = impute_by_category(business_df, 'attributes')
hours_imputed, hours_p_assigned = impute_by_category(business_df, 'hours')
print(f"'attributes' - Imputed: {attributes_imputed}, Assigned to Placeholder: {attributes_p_assigned}")
print(f"'hours' - Imputed: {hours_imputed}, Assigned to Placeholder: {hours_p_assigned}")

# saving the dataset
business_df.to_csv('Business_Data_II.csv', index=False)
```

Console | Timing | Datasets | Charts

```
'attributes' - Imputed: 960, Assigned to Placeholder: 4
'hours' - Imputed: 645, Assigned to Placeholder: 4
```

[10]
```
LOAD DATASET Business_Data_II AS csv FROM Business_Data_II.csv @ url 'Business_Data_II.csv'
```

Console | Timing | Datasets | Charts

business_data_ii (10000 rows)

Views

	business_id (string)	name (string)	address (string)	city (string)	state (string)	postal_code (float)	latitude (float)
0	TacYUYhU3HpLHF9Rs6fW2w	Steps to Learning Montessori Preschool	6901 Phelps Rd	Goleta	CA	93117	34.423309326171875
1	RnExalCvleXxFpbIKEqJsQ	Breeze Blow Dry Bar	9916 Clayton Rd	St. Louis	MO	63124	38.636714935302734
2	pjtjBeZC3gvmtIilQt-DFA	Impact Guns	11655 W Executive Dr	Boise	ID	83713	43.608699798583984
3	OMl5pTUBVUjW_jvDKLvYtw	Palms Primary Care	1615 Pasadena Ave S, Ste 430	Saint Petersburg	FL	33707	27.752653121948242
4	MO-LTDfO843xaRtW0bx6jQ	J&Q Nails	9655 E US Hwy 36, Unit H	Avon	IN	46123	39.763057708740234
5	jFq8QSWDwtAWMA1FaNvRhw	Candy Barrel	735 Dodecanese Blvd	Tarpon Springs	FL	34689	28.155075073242188
6	t53MqkLTtWxJ7jMSqXYz-g	Luminosity	690 W Dekalb Pike	King of Prussia	PA	19406	40.08949661254883
7	dY6rzL7Gw1U5afOskTIMmg	Nail Care Salon	12337 Olive Blvd	Creve Coeur	MO	63141	38.67354202270508
8	VlJ4wKPf2TmKbaOTKYHROg	Architectural Antiques of Indianapolis	5000 W 96th St	Indianapolis	IN	46268	39.92622375488281
9	IuKCyfSY7AKhRbRA1JPlPw	Aster's Floral Shop	41 Haddon Ave	Collingswood	NJ	8108	39.91556167602539
10	6XOn1p3sbO22UjJGpmCgxg	China Wok	4319 Telegraph Rd	Saint Louis	MO	63129	38.486759185791016
11	c8d8h47cogM_B_ZMC-h3zg	Brandon Family Medical Care	1218 Millennium Pkwy	Brandon	FL	33511	27.929439544677734
12	YEwml50bso_LYtuivsdwiA	7-Eleven	13151 Race Track Rd	Tampa	FL	33626	28.07027244567871
13	6lG4SysBKyRnFW_e22q13A	Uber		Philadelphia	PA	19107	39.955928802490234
14	bNBi-RVlx71bugXY0GRLtQ	Chestnut St. Cafe	4403 Chestnut St	Philadelphia	PA	19104	39.95672607421875
15	rbPK4jSyFS10zhWYvo_Srg	Cafe Porche and snowbar	1625 Baronne St	New Orleans	LA	70113	29.939315795898438
16	lrO5vOwa7gE0qAo0UQJYlA	Eyeglass World	13002 Seminole Blvd, Ste 10-11	Largo	FL	33778	27.891815185546875
17	thpDDcdKLPzSFldLLBULLA	Callahan's Corner	914 Edwardsville Rd	Troy	IL	62294	38.732933044433594
18	-EyRrBY1d-EQZrTlXWH4BQ	Spa Guy Dave		Pennsauken	NJ	8109	39.967105865478516
19	EkthrfcRWCVYy-NuvfNmPg	FroYo Frozen Yogurt	4663 Maryland Ave	Saint Louis	MO	63108	38.64484405517578

[11]
```
SELECT city, state, postal_code, COUNT(*) AS frequency
FROM business_data
WHERE postal_code IS NOT NULL
GROUP BY city, state, postal_code
ORDER BY city, state, COUNT(*) DESC;
```

Console | Timing | Datasets | Charts

common_postal_codes (1084 rows)

Views

	city (string)	state (string)	postal_code (int)	frequency (long)
0	Abington	PA	19001	11
1	Aldan	PA	19018	1
2	Alton	IL	62002	11
3	Ambler	PA	19002	12
4	Antioch	TN	37013	14
5	Antioch	TN	37203	1
6	Apollo Beach	FL	33572	4
7	Arabi	LA	70032	4
8	Ardmore	PA	19003	27
9	Ardmore	PA	19010	1
10	Arnold	MO	63010	8
11	Ashland City	TN	37015	5
12	Aston	PA	19014	14
13	Atco	NJ	8004	2
14	Audubon	NJ	8106	5
15	Audubon	PA	19403	4
16	Avon	IN	46123	15
17	Avondale	LA	70094	1
18	Bala Cynwyd	PA	19004	14
19	Ballwin	MO	63011	10

[12]

```
import requests
import pandas as pd

# function to get the missing address and postal code based on the openstreetmap api
def reverse_geocode(lat, lon):
    url = f"https://nominatim.openstreetmap.org/reverse?lat={lat}&lon={lon}&format=json"
    response = requests.get(url)
    if response.status_code == 200:
        data = response.json()
        address = data.get('display_name', 'Address not found')
        postal_code = data.get('address', {}).get('postcode', 'Postal code not found')
        return address, postal_code
    else:
        return 'Address not found', 'Postal code not found'

business_df = vizierdb.get_data_frame('Business_Data_II')

# counters
address_assigned_count = 0
postal_code_assigned_count = 0

for index, row in business_df.iterrows():
    if pd.isnull(row['postal_code']) or pd.isnull(row['address']):
        full_address, postal_code = reverse_geocode(row['latitude'], row['longitude'])

        if pd.isnull(row['address']):
            business_df.at[index, 'address'] = full_address
            if full_address != 'Address not found':
                address_assigned_count += 1

        if pd.isnull(row['postal_code']):
            business_df.at[index, 'postal_code'] = postal_code
            if postal_code != 'Postal code not found':
                postal_code_assigned_count += 1

# dropped rows where 'categories' is missing
initial_row_count = len(business_df)
business_df.dropna(subset=['categories'], inplace=True)
print(f"Dropped {initial_row_count - len(business_df)} rows where 'categories' was missing.")

# saving the df
business_df.to_csv('Business_Data_III.csv', index=False)

# printing remaining null values
remaining_null_values = business_df.isnull().sum()
print("Remaining null values in each column:")
print(remaining_null_values)

# printing the count of updates
print(f"Number of rows assigned with new address: {address_assigned_count}")
print(f"Number of rows assigned with new postal code: {postal_code_assigned_count}")
```

Console | Timing | Datasets | Charts

```
Dropped 4 rows where 'categories' was missing.
Remaining null values in each column:
business_id     0
name            0
address         0
city            1
state           0
```

```
postal_code     0
latitude        0
longitude       0
stars           0
review_count    0
is_open         1
attributes      0
categories      0
hours           0
dtype: int64

Number of rows assigned with new address: 348
Number of rows assigned with new postal code: 371
```

[13]
```
LOAD DATASET Business_Data_III AS csv FROM Business_Data_III.csv @ url 'Business_Data_III.csv'
```

Console ▾ Timing Datasets ▾ Charts ▾

business_data_iii (9996 rows)

Views

	business_id (string)	name (string)	address (string)	
0	TacYUYhU3HpLHF9Rs6fW2w	Steps to Learning Montessori Preschool	6901 Phelps Rd	Gc
1	RnExalCvleXxFpbIKEqJsQ	Breeze Blow Dry Bar	9916 Clayton Rd	St
2	pjtjBeZC3gvmtliIQt-DFA	Impact Guns	11655 W Executive Dr	Bc
3	OMl5pTUBVUjW_jvDKLvYtw	Palms Primary Care	1615 Pasadena Ave S, Ste 430	Sa
4	MO-LTDfO843xaRtW0bx6jQ	J&Q Nails	9655 E US Hwy 36, Unit H	Av
5	jFq8QSWDwtAWMA1FaNvRhw	Candy Barrel	735 Dodecanese Blvd	Ta
6	t53MqkLTtWxJ7jMSqXYz-g	Luminosity	690 W Dekalb Pike	Ki
7	dY6rzL7Gw1U5afOskTIMmg	Nail Care Salon	12337 Olive Blvd	Cr
8	VlJ4wKPf2TmKbaOTKYHROg	Architectural Antiques of Indianapolis	5000 W 96th St	In
9	IuKCyfSY7AKhRbRA1JPlPw	Aster's Floral Shop	41 Haddon Ave	Cc
10	6XOn1p3sbO22UjJGpmCgxg	China Wok	4319 Telegraph Rd	Sa
11	c8d8h47cogM_B_ZMC-h3zg	Brandon Family Medical Care	1218 Millennium Pkwy	Br
12	YEwml50bso_LYtuivsdwiA	7-Eleven	13151 Race Track Rd	Ta
13	6lG4SysBKyRnFW_e22q13A	Uber	6th District Police Lot, Race Street, Chinatown, Center City, Philadelphia, Pennsylvania, 19103, United States	Ph
14	bNBi-RVlx71bugXY0GRLtQ	Chestnut St. Cafe	4403 Chestnut St	Ph
15	rbPK4jSyFS10zhWYvo_Srg	Cafe Porche and snowbar	1625 Baronne St	Ne
16	lrO5vOwa7gE0qAo0UQJYlA	Eyeglass World	13002 Seminole Blvd, Ste 10-11	La
17	thpDDcdKLPzSFldLLBULLA	Callahan's Corner	914 Edwardsville Rd	Tr
18	-EyRrBY1d-EQZrTlXWH4BQ	Spa Guy Dave	2649, Hadley Drive, Jordantown, Pennsauken Township, Camden County, New Jersey, 08109, United States	Pe
19	EkthrfcRWCVYy-NuvfNmPg	FroYo Frozen Yogurt	4663 Maryland Ave	Sa

[14]

Console ▾ Timing Datasets ▾ Charts ▾

Analyzing the Dataset

[15]
```
SELECT stars, city FROM Business_Data_III;
```

Console ▾ Timing Datasets ▾ Charts ▾

stars_and_city (9996 rows)

Views

	stars (float)	city (string)
0	4.5	Goleta
1	4	St. Louis
2	3	Boise
3	4	Saint Petersburg
4	3.5	Avon
5	3	Tarpon Springs
6	4.5	King of Prussia
7	4	Creve Coeur
8	4	Indianapolis
9	4	Collingswood
10	4.5	Saint Louis
11	2.5	Brandon
12	3.5	Tampa
13	3	Philadelphia
14	4.5	Philadelphia
15	5	New Orleans
16	5	Largo

17	3	Troy
18	4.5	Pennsauken
19	3.5	Saint Louis

[16]

```
SELECT city, AVG(stars) as avg_stars
FROM stars_and_city
WHERE city IS NOT NULL
GROUP BY city
ORDER BY avg_stars DESC;
```

Console | Timing | Datasets | Charts

city_avg_stars (548 rows)

Views

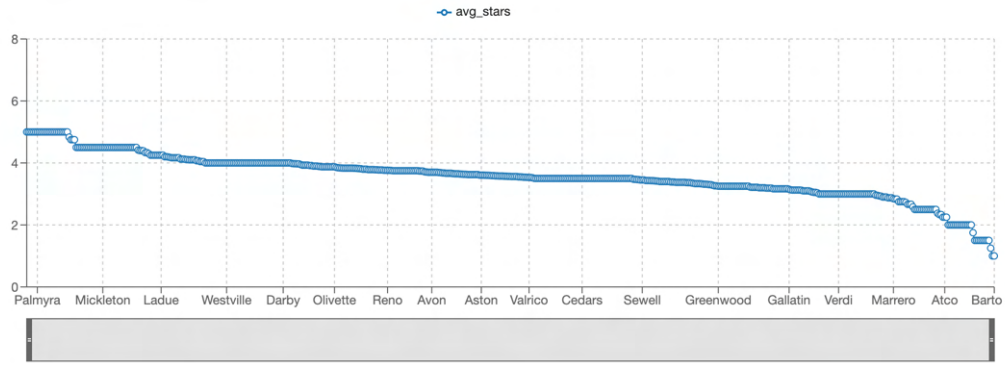
	city (string)	avg_stars (real)
0	Truckee	5
1	St. Charles	5
2	Erial	5
3	Boothwyn	5
4	Pegram	5
5	Picture Rocks	5
6	Spring House	5
7	Palmyra	5
8	King of Prussi	5
9	Virtual	5
10	Belleair	5
11	Indianpolis	5
12	saint ann	5
13	Skippack Village	5
14	Treasure Island	5
15	Greenville	5
16	West Hill	5
17	Clermont	5
18	Waterford Works	5
19	Bellefonte	5

[17] CREATE Scatter Plot Average Star Rating FOR "city_avg_stars"

Console | Timing | Datasets | Charts

Average Star Rating

Charts: Line Chart with Points



Download Chart

[18]

```
SELECT business_id, categories, city FROM Business_Data_III;
```

Console | Timing | Datasets | Charts

categories_and_city (9996 rows)

Views

	business_id (string)	categories (string)	city (string)
0	TacYUYhU3HpLHF9Rs6fW2w	"'WiFi': """""""u'no'"""""""""	Goleta
1	RnExalCvleXxFpbIKEqJsQ	'BusinessAcceptsCreditCards': 'True'	St. Louis
2	pjtjBeZC3gvmtliilQt-DFA	'street': False	Boise
3	OMl5pTUBVUjW_jvDKLvYtw	Internal Medicine, Doctors, Health & Medical	Saint Petersburg
4	MO-LTDfO843xaRtW0bx6jQ	'street': False	Avon
5	jFq8QSWDwtAWMA1FaNvRhw	'street': False	Tarpon Springs
6	t53MqkLTtWxJ7jMSqXYz-g	'BusinessAcceptsCreditCards': 'True'	King of Prussia
7	dY6rzL7Gw1U5afOskTlMmg	'street': False	Creve Coeur
8	VlJ4wKPf2TmKbaOTKYHROg	'BusinessAcceptsCreditCards': 'True'	Indianapolis
9	luKCyfSY7AKhRbRA1JPlPw	'BusinessAcceptsCreditCards': 'True'	Collingswood
10	6XOn1p3sbO22UjJGpmCgxg	'RestaurantsDelivery': 'False'	Saint Louis
11	c8d8h47cogM_B_ZMC-h3zg	Medical Centers, Health & Medical	Brandon
12	YEwml50bso_LYtuivsdwiA	'street': False	Tampa
13	6lG4SysBKyRnFW_e22q13A	Hotels & Travel, Taxis, Transportation, Local Services, Automotive	Philadelphia
14	bNBi-RVlx71bugXY0GRLtQ	'street': True	Philadelphia
15	rbPK4jSyFS10zhWYvo_Srg	'street': True	New Orleans
16	lrO5vOwa7gE0qAo0UQJYlA	Optometrists, Health & Medical, Eyewear & Opticians, Ophthalmologists, Doctors, Shopping	Largo
17	thpDDcdKLPzSFldLLBULLA	"'RestaurantsAttire': """""""u'casual'"""""""""	Troy
18	-EyRrBY1d-EQZrTlXWH4BQ	Home Services, Pool & Hot Tub Service	Pennsauken
19	EkthrfcRWCVYy-NuvfNmPg	'RestaurantsPriceRange2': '1'	Saint Louis

[19]

```
SELECT categories, city, COUNT(business_id) as business_count
FROM categories_and_city
GROUP BY categories, city
ORDER BY business_count DESC;
```

Console | Timing | Datasets | Charts

city_category_counts (6207 rows)

Views

	categories (string)	city (string)	business_count (long)
0	'BusinessAcceptsCreditCards': 'True'	Philadelphia	69
1	"'BusinessParking': """""""{'garage': False"""	Philadelphia	65
2	'BusinessAcceptsCreditCards': 'True'	Tampa	61
3	'street': True	Philadelphia	59
4	'BusinessAcceptsCreditCards': 'True'	Indianapolis	53
5	'street': False	Philadelphia	49
6	'RestaurantsPriceRange2': '2'	Philadelphia	45
7	'street': False	Tucson	42
8	"'BusinessParking': """""""{'garage': False"""	Tucson	41
9	'BikeParking': 'True'	Philadelphia	40
10	'BusinessAcceptsCreditCards': 'True'	Nashville	39
11	"'BusinessParking': """""""{'garage': False"""	Edmonton	38
12	'BusinessAcceptsCreditCards': 'True'	Tucson	35
13	'BusinessAcceptsCreditCards': 'True'	New Orleans	32
14	'street': False	Tampa	31
15	"'BusinessParking': """""""{'garage': False"""	Saint Louis	30
16	'street': False	Indianapolis	30
17	'RestaurantsTakeOut': 'True'	Philadelphia	30
18	"'BusinessParking': """""""{'garage': False"""	New Orleans	30
19	"'BusinessParking': """""""{'garage': False"""	Indianapolis	29

[20]

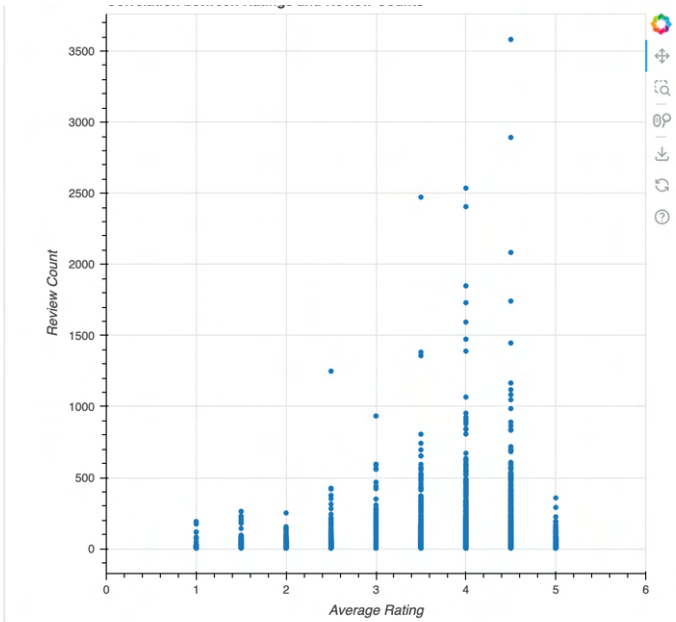
```
from bokeh.io import show
from bokeh.plotting import figure

ds = vizierdb.get_dataset('Business_Data_III')

plot = figure(title="Correlation between Ratings and Review Counts",
              x_axis_label='Average Rating',
              y_axis_label='Review Count',
              x_range=(0, 6))
plot.scatter(
  x='stars',
  y='review_count',
  source=ds.to_bokeh()
)
show(plot)
```

Console | Timing | Datasets | Charts

Correlation between Ratings and Review Counts



[21]

Console | Timing | Datasets | Charts

Preliminary Analysis - Reviews Data Schema

- Some basic statistics about the dataset such as the number of rows (entries), number of attributes (columns), and the data format.
- Checking if there are any duplicate entries in the dataset based on the reviews_id, which should be unique for each business.

[22]
```
LOAD DATASET Reviews_Data AS csv FROM yelp_academic_dataset_checkin.csv @ artifact file 70
```

Console | Timing | Datasets | Charts

reviews_data (28224 rows)

Views

	review_id (string)	user_id (string)	business_id (string)	stars (long)	useful (long)	funny (long)	
0	FCXselWrkSUzC5wHZxVXrw	NDkwKnvjhBbjCh1cNlBoAw	t1qF12NdW8KvCqxqbvy-Hg	4	1	0	
1	jTGp3mbMA8w_Prg-Ufxfkg	YAlp90pskaKL1_Wtn7kbKQ	kqAa2CtPGA-QsZhhbzpzUQ	5	0	0	
2	Great to find a restaurant close by that's both healthy and affordable."	1454131476000					
3	RxJw4NR37bcVKQj_BoG7-w	pzfZBclLOqf5wMj0O5yLlg	MaYb7qMN6BomP1zQGj3Wjg	5	1	0	
4							
5	Please don't go ""big-timer"" on us	Pi; you're far too awesome for that."	1294720909000				
6	zPt6CuKvx1v24BXMCwHzRA	p2PwR7oPLDo_vbo8gCCa7A	pSmOH4a3HNNpYM82J5ycLA	5	0	0	
7	6g39Ku2yBsC4-l9N3prEaA	YsqK0URRY3oRwC3wTRtd7Q	h4C7s_Go9UKo5twaLIWc8A	5	0	0	
8	I will be back!"	1488733986000					
9	qY11QD0JVNNuB9ObWDzBsQ	Hwzn3_MuTD_ZhY5-pjURUQ	8eDkw7CE0NKqMknPlu26fw	3	0	0	
10	NbvR8lib1vdbXDMeXbhnVA	fEtWwhNKSoTqIWTSEwGlvg	KVFZ6yiM1DgiEGK-Jn80ig	2	0	0	
11	UbhAJpJvtt4jaOv-dpSAyg	kMT7Hb8zRubKuissGbjcfw	0sr1EyOc6Td1C-962QW88w	1	3	1	
12	vuS2mktaoBZo77XdLJjjNA	kmXNP537i0dyYOFF-sxtdg	rG0UTjvbmVVsh9-kGzeliQ	4	0	0	
13	Rt-BrvgR8k_mbmazrPoXbw	6cUt5rA5EY8-mH4q-v239Q	yeHLiKNp0hyR-ig4M6us-w	5	2	1	
14	AGgFJ8VPX7kAwdy2XTh95w	3MYdpmHeNwC6FquRWi3YOg	uEWsfftrJ7ukPv1xyMVcrg	5	6	0	
15	No3OGeXLydmnOYqG3FguUw	Wl30_VevEQt4IPbhqtNhrA	3SHw4aZW_muE1kRSLQQjjA	3	1	0	
16	d7qA-vF3ICEYg0pzxP33sg	mIeNRnJ7fdEEpUp8nvTSrw	jgcQGltZISMAwzo10XOjOw	5	2	1	
17	ij3w_OPSeHZflospE4jLNQ	XSFkFLUC9dFvvyJPJHhg_A	2oav5QoWgnvTl2gO5xFMjw	3	1	0	
18	xtoFKDQVvrQv6LRnmJF9Nw	6cFqRc7XOZlrQJ2f0pWvDw	vsrryxBR2Jykl71qa1H6SQ	4	1	0	
19							

[23]
```
# reading the dataset as a datafram
dataset = vizierdb.get_data_frame('Reviews_Data')

# displaying head
print(dataset.head())
print("-" * 100,"\n")

# computing the dataset characteristics
total_rows = len(dataset)
total_columns = len(dataset.columns)
data_types = dataset.dtypes
```

```
print('The total number of rows are - ',total_rows)
print('The total number of features are - ',total_columns)
print('The datatype of various features is -\n',data_types)

# checking for duplicate rows based on the review_id
duplicate_rows = dataset[dataset.duplicated(subset='review_id')]

duplicate_rows_count = len(duplicate_rows)
print('Computing the duplicate rows \n',duplicate_rows_count)
```

Console | Timing | Datasets | Charts

```
  review_id  ...          date
  0                           FCXseIWrkSUzC5wHZxVXrw  ...  1.363797e+12
  1                           jTGp3mbMA8w_Prg-Ufxfkg  ...           NaN
  2  Great to find a restaurant close by that's bot...  ...           NaN
  3                           RxJw4NR37bcVKQj_BoG7-w  ...           NaN
  4                                             None  ...           NaN

  [5 rows x 9 columns]

  ----------------------------------------------------------------------------------------------------

  The total number of rows are -  28224
  The total number of features are -  9
  The datatype of various features is -
  review_id       object
  user_id         object
  business_id     object
  stars          float64
  useful         float64
  funny          float64
  cool           float64
  text            object
  date           float64
  dtype: object

  Computing the duplicate rows
  8266
```

[24]

```
#Importing the dataset as a datafram
dataset = vizierdb.get_data_frame('Reviews_Data')

#Checking for missing values in each column
missing_values = dataset.isnull().sum()

print('Computing the Null/Missing values Feature-wise \n', missing_values)
```

Console | Timing | Datasets | Charts

```
  Computing the Null/Missing values Feature-wise
  review_id       7773
  user_id        10459
  business_id    13670
  stars          17787
  useful         18019
  funny          18103
  cool           18172
  text           17808
  date           21962
  dtype: int64
```

[25]

```
# load the datasets
business_df = vizierdb.get_data_frame('Business_Data_III')
reviews_df = vizierdb.get_data_frame('Reviews_Data')

# check common column
business_columns = ', '.join(business_df.columns.tolist())
reviews_columns = ', '.join(reviews_df.columns.tolist())

vizierdb.show("Business Dataset Columns: " + business_columns)
vizierdb.show("Reviews Dataset Columns: " + reviews_columns)
```

Console | Timing | Datasets | Charts

```
  Business Dataset Columns: business_id, name, address, city, state, postal_code, latitude, longitude, stars, review_count, is
  Reviews Dataset Columns: review_id, user_id, business_id, stars, useful, funny, cool, text, date
```

[26]

```
reviews_df = vizierdb.get_data_frame('Reviews_Data')

# Calculate the initial number of rows (before any filtering)
```

```
initial_row_count = len(reviews_df)
print(f"Initial number of rows: {initial_row_count}")

# Drop rows where 'review_id', 'user_id', or 'business_id' are missing
reviews_df.dropna(subset=['review_id', 'user_id', 'business_id'], inplace=True)

# Calculate the number of rows after dropping rows with missing IDs
post_id_drop_row_count = len(reviews_df)
print(f"Number of rows after dropping missing IDs: {post_id_drop_row_count}")

# Define the threshold for maximum number of nulls in a row
max_nulls = 4

# Calculate the number of nulls in each row
reviews_df['null_count'] = reviews_df.isnull().sum(axis=1)

# Filter out rows that exceed the threshold and create an explicit copy
filtered_reviews_df = reviews_df[reviews_df['null_count'] < max_nulls].copy()

# Drop the 'null_count' column from the filtered DataFrame
filtered_reviews_df.drop(columns=['null_count'], inplace=True)

# Calculate the number of rows after filtering for null counts
post_filter_row_count = len(filtered_reviews_df)
print(f"Number of rows after filtering based on null count: {post_filter_row_count}")

# Show the number of rows that were removed
rows_removed = initial_row_count - post_filter_row_count
print(f"Total number of rows removed: {rows_removed}")

# Save the DataFrame as a CSV file
filtered_reviews_df.to_csv('Filtered_Reviews_Data.csv', index=False)
```

Console ▾ Timing Datasets ▾ Charts ▾

```
Initial number of rows: 28224
Number of rows after dropping missing IDs: 14553
Number of rows after filtering based on null count: 10000
Total number of rows removed: 18224
```

[27]
```
LOAD DATASET Filtered_Reviews_Data AS csv FROM Filtered_Reviews_Data.csv @ url 'Filtered_Reviews_Data.csv'
```

Console ▾ Timing Datasets ▾ Charts ▾

filtered_reviews_data (10000 rows)

Views

	review_id (string)	user_id (string)	business_id (string)	stars (float)	useful (float)	funny (float)	cool (float)	
0	FCXselWrkSUzC5wHZxVXrw	NDkwKnvjhBbjCh1cNlBoAw	t1qF12NdW8KvCqxqbvy-Hg	4	1	0	0	Good croissant, brie, egg sar
1	jTGp3mbMA8w_Prg-Ufxfkg	YAlp90pskaKL1_Wtn7kbKQ	kqAa2CtPGA-QsZhhbzpzUQ	5	0	0	0	Wow! We had the most amaz
2	RxJw4NR37bcVKQj_BoG7-w	pzfZBcILOqf5wMj0O5yLlg	MaYb7qMN6BomP1zQGj3Wjg	5	1	0	0	"""Pi has shrunk their salads.
3	zPt6CuKvx1v24BXMCwHzRA	p2PwR7oPLDo_vbo8gCCa7A	pSmOH4a3HNNpYM82J5ycLA	5	0	0	0	Worth the wait. I always get t
4	6g39Ku2yBsC4-l9N3prEaA	YsqK0URRY3oRwC3wTRtd7Q	h4C7s_Go9UKo5twaLIWc8A	5	0	0	0	Pad kee mau was authentic a
5	qY11QD0JVNNuB9ObWDzBsQ	Hwzn3_MuTD_ZhY5-pjURUQ	8eDkw7CE0NKqMknPlu26fw	3	0	0	0	The food was just okay. I had
6	NbvR8lib1vdbXDMeXbhnVA	fEtWwhNKSoTqIWTSEwGlvg	KVFZ6yiM1DgiEGK-Jn80ig	2	0	0	0	Came over from Philly to get
7	UbhAJpJvtt4jaOv-dpSAyg	kMT7Hb8zRubKuissGbjcfw	0sr1EyOc6Td1C-962QW88w	1	3	1	0	Rude staff. Can't even get a s
8	vuS2mktaoBZo77XdLIjjNA	kmXNP537i0dyYOFF-sxtdg	rG0UTjvbmVVsh9-kGzeliQ	4	0	0	0	City Limits is a cute little plac
9	Rt-BrvgR8k_mbmazrPoXbw	6cUt5rA5EY8-mH4q-v239Q	yeHLiKNp0hyR-ig4M6us-w	5	2	1	1	I would say that Livery is one
10	AGgFJ8VPX7kAwdy2XTh95w	3MYdpmHeNwC6FquRWi3YOg	uEWsfftrJ7ukPv1xyMVcrg	5	6	0	3	I can't believe I haven't yelpe
11	No3OGeXLydmnOYqG3FguUw	WI30_VevEQt4IPbhqtNhrA	3SHw4aZW_muE1kRSLQQjjA	3	1	0	0	It's IHOP so,I don't expect m
12	d7qA-vF3ICEYg0pzxP33sg	mIeNRnJ7fdEEpUp8nvTSrw	jgcQGltZISMAwzo10XOjOw	5	2	1	0	Hammy isn't too sure about t
13	ij3w_OPSeHZfIospE4jLNQ	XSFkFLUC9dFvvyJPJHhg_A	2oav5QoWgnvTl2gO5xFMjw	3	1	0	0	"""Don't come!!! My husband
14	xtoFKDQVvrQv6LRnmJF9Nw	6cFqRc7XOZlrQJ2f0pWvDw	vsrryxBR2Jykl71qa1H6SQ	4	1	0	1	Cafe primo is one of my new
15	Xl2uJjEr5OXIlt7WwRpdkQ	w3jpHKubt8S8M1U7pCvImw	KP5OncF2jhT7_J1phHPPww	4	0	0	0	I had lunch. I really like it. Me
16	Pr208PmLFtNLoRxpvSH_rw	SOJwtIVajM8SKZB6ARnq2Q	fitSn2LBLb5OXV7hdl86Kw	5	0	0	0	Best put together rooms I've
17	HFKBH1MFxG3CKuo7jas44w	eTvp_hYnsrI5-ow_sQ31_g	GV-e7Op4aU98xkjjoKlRhA	5	4	0	2	Went with girlfriends for happ
18	3qlSHiBXApIMiH7D_g9GnQ	RBQkmdMQJAMJUK6O_J_J7g	EtKSTHV5Qx_Q7Aur9o4kQQ	3	0	0	0	I do not recommend going fo
19	boD0dt5gi0lMJd_RKzBelg	rsDS74FGRL414i19kaVALQ	ZBcgfFw2ZVNNVbBzsz4hsw	5	1	0	0	Matt C was awesome to work

[28]
```
# Load the filtered dataset from VizierDB
filtered_reviews_df = vizierdb.get_data_frame('Filtered_Reviews_Data')

# Calculate the number of null values in each column
null_values_count = filtered_reviews_df.isnull().sum()

# Display the count of null values for each column
vizierdb.show(null_values_count)
```

Console ▾ Timing Datasets ▾ Charts ▾

```
review_id         0
user_id           0
business_id       0
stars             0
useful            0
funny             0
cool              0
text              0
date           3838
dtype: int64
```

[29]

Console | Timing | Datasets | Charts

Transformation of Timestamp in Reviews Dataset

In the reviews dataset, the time information was initially stored as an epoch timestamp, which represents the number of milliseconds since January 1, 1970, UTC. This format, while efficient for storage and computation, is not human-readable and thus not suitable for analysis that requires interpretation of dates and times.

To address this, we executed the following steps:

1. **Converting Epoch to Datetime:**
   - The 'date' column, which contained the epoch timestamps, was converted to Python datetime objects. This conversion was achieved using the `pd.to_datetime` function in pandas, specifying the unit as 'ms' (milliseconds).
   - The conversion interprets the epoch timestamp as the number of milliseconds since the Unix epoch start (1970-01-01 00:00:00 UTC) and converts it to a standard datetime object.
2. **Updating the DataFrame:**
   - The transformed date-time information was saved back into the DataFrame.
   - We then saved the updated DataFrame back into VizierDB, ensuring that all changes were persisted.

By transforming the epoch timestamps into a human-readable format, we made the dataset more intuitive and accessible for any subsequent analysis or reporting that involves date and time information. This step is often essential in data pre-processing, especially when dealing with time-series data or records that are time-stamped.

[30]

```python
import pandas as pd

reviews_df = vizierdb.get_data_frame('Filtered_Reviews_Data')

# convert to datetime object
reviews_df['date'] = pd.to_datetime(reviews_df['date'], unit='ms')
reviews_df.sort_values(by='date', inplace=True, ascending=True)
reviews_df.to_csv('Reviews_Data_2.csv', index=False)
```

Console | Timing | Datasets | Charts

[31]

```
LOAD DATASET Updated_Reviews_Data AS csv FROM Reviews_Data_2.csv @ url 'Reviews_Data_2.csv'
```

Console | Timing | Datasets | Charts

updated_reviews_data (10000 rows)

Views

	review_id (string)	user_id (string)	business_id (string)	stars (float)	useful (float)	funny (float)	cool (float)	
0	TYslH-CAecjJxLNs96KduA	XCsZ3hWa_6oP1WkWvK7pmg	29YqJwOGEuAWqlHZxMc1OA	5	0	0	0	Lovely little restaurant whic
1	Uc1C5BBmYOw6ZyaDm2MMbw	3MYdpmHeNwC6FquRWi3YOg	5vpqoHruyjHALHjxJLSJbA	1	0	1	0	I once went to this Gavi and
2	9eeoAUQEJDw0KYV5NPYH-A	-qoyKSF2G3PkR_7XNoJfpQ	0ZETsmrLSGWaBu1NNdcmYw	2	2	0	0	Good luck finding anyone t
3	DigjNdnkH35l4NVcwXqT5Q	W4Ez5lpm377893t1XhLquw	nRKndeZLQ3eDL10UMwS2rQ	5	0	0	1	First, Ted Drewes is more th
4	WBmqTb2lrJu9_nIWgZu21g	n-IBS02-3yvlY5Q91mmwDA	grjFEAN9gXOKQFEta4yBJg	3	0	0	0	Tropical Isle is tacky, but th
5	ayL25w-ao4lj3c2_7_bXug	3MYdpmHeNwC6FquRWi3YOg	-pRU97z3uPU_M7yUf5yzMg	4	0	1	0	Great food and handled a la
6	IY7i04LMohgO7Hlk5pYvew	_BHTC7nyCBoZcfiiD5cOXg	YtSqYv1Q_pOltsVPSx54SA	5	2	2	2	Despite the fact it's part of
7	AGgFJ8VPX7kAwdy2XTh95w	3MYdpmHeNwC6FquRWi3YOg	uEWsfftrJ7ukPv1xyMVcrg	5	6	0	3	I can't believe I haven't yelp
8	-JNEW5LMvfMG1a2oBuRrCg	3MYdpmHeNwC6FquRWi3YOg	PsHzOS-Jao5CpRl5LKfvlg	2	5	0	0	I don't know, I was told abc
9	-sydClhmvq5JJDsWW5Mo4Q	jTXj5uf5ER5siDnQK_y4aA	d_tRshM-w6S4QxE4VVi8tQ	4	3	0	1	One of the best brunch plac
10	c79fKwD2-OkFAUyT4oiKJw	iw_6JAAw9FAlv2YkbLcMFA	PY9GRfzr4nTZeINf346QOw	4	1	0	0	One of the best hotels in Re
11	UP_c95ne3ts3lx2E-rJrfQ	3xX-Et9l4Bi4wdRPWJkAFg	KA9WEoGHr9Ae0pFXZqzylQ	4	1	0	0	Pool hall, bar, cocktail loun
12	rsH4l8qyCgsMl2es2pgHfw	cOSEvHWqoT5B4d3RcZthyw	XleIcqVgh3iBeM4s-rdy6g	5	1	1	1	This place has the best foo
13	0VFjGL_5PkO1GZQ-EGrEcA	QBHfjwsZF0XUJBqdz_547Q	jxAJrlQ2JjjTs8ZzEytP-w	5	0	0	0	Not only is this place comfo
14	0X0k2Rf2whgkQY6X6yMYTQ	tiLm3eFfpxhwV6O9DEaneQ	jxAJrlQ2JjjTs8ZzEytP-w	5	0	0	0	i love this place for any type
15	pHCO77mLsu-7FRRkLCNMug	HLNNvVX-lZztLUuzTnN73w	fnO03-RX7UDC1TzXETWEsQ	5	2	0	1	This is by far our family's fa
16	FyNLkuX23vWn0sLO1PbSxw	Ct456aElCzDfn03MHZCzsw	iJoGdgWECMNpe-ajs0rfxA	4	0	0	0	Closed Tuesday. Open 10ar
17	ZysvcYxaeY22yBvUR14d2A	A94ivfwchCxwN4rcdoeuJQ	1wC41D8PLqOldFPUVAZ3tA	5	0	1	0	Great looking building walk
18	nXNtmRDfHJvu_j2-3EObcw	38zhUVlh6t_-mpcmbWYmiw	Y3ZCO17N1_T_Ms1JmswwzA	5	0	0	0	Incredibly good and totally
19	XfYXaBr3llgl8shztTpK3A	VQC12BRAJoGvGkiRyDYv3A	SZU9c8V2GuREDN5KgyHFJw	5	1	1	1	I adore you Shellfish Compa

[32]

```
SELECT date, COUNT(*) as dates_count
FROM Updated_Reviews_Data
GROUP BY date
ORDER BY dates_count DESC
LIMIT 50
```

Console | Timing | Datasets | Charts

dates_data (50 rows)

Views

	date (date)	dates_count (long)
0		3838
1	2018-02-04	25
2	2017-02-05	24
3	2018-03-28	22
4	2018-06-17	21
5	2015-07-31	20
6	2017-06-29	20
7	2018-02-17	19
8	2018-00-21	19
9	2018-04-19	19
10	2016-00-30	19
11	2018-06-08	18
12	2016-06-16	18
13	2018-04-09	18
14	2016-00-31	18
15	2017-00-28	18
16	2016-02-07	18
17	2016-05-12	18
18	2017-07-29	17
19	2017-04-28	17

[33]

```
SELECT EXTRACT(YEAR FROM date) AS year, COUNT(*) as dates_count
FROM Updated_Reviews_Data
WHERE date IS NOT NULL
GROUP BY year
ORDER BY dates_count DESC
```

Console | Timing | Datasets | Charts

most_frequent_years (14 rows)

Views

	year (int)	dates_count (long)
0	2017	1218
1	2016	1107
2	2015	1088
3	2018	911
4	2014	705
5	2013	447
6	2012	273
7	2011	205
8	2010	84
9	2009	51
10	2008	44
11	2007	10
12	2005	10
13	2006	9

[34]

```
SELECT MIN(date) as MinDate, MAX(date) as MaxDate
FROM Updated_Reviews_Data;
```

Console | Timing | Datasets | Charts

date_range (1 rows)

Views

	MinDate (date)	MaxDate (date)
0	2005-02-01	2018-09-04

[35]

Console | Timing | Datasets | Charts

Handling Missing Time Values: Approach and Implementation

Addressing missing time values in the Reviews dataset was executed with a strategic approach to maintain data integrity for temporal analysis.

Imputation Strategy

- **Year-Based Imputation**: Missing dates were imputed based on the year distribution within the dataset, reflecting the true temporal nature of the data.

Implementation

- **Frequency Distribution Calculation**: The distribution of years from review dates was used to guide the imputation process.
- **Random Assignment**: Missing dates were imputed by randomly selecting a year from this distribution.
- **Transparency**: A new column, `date_imputed`, was introduced to flag imputed dates, distinguishing them from original dates.

Significance

- **Preservation of Structure**: This method helped maintain the dataset's overall trends and patterns.
- **Transparency in Data Curation**: Clear indication of imputed dates ensures transparency in analyses.
- **Balanced Approach**: The strategy provided a reasoned solution to the challenge of missing temporal data.

---

[36]

```python
import pandas as pd
import numpy as np

# Load the Most_Frequent_Years dataset from VizierDB
most_frequent_years_df = vizierdb.get_data_frame('Most_Frequent_Years')

# Calculate the total count of reviews across all years
total_reviews = most_frequent_years_df['dates_count'].sum()

# Calculate the probability of each year
most_frequent_years_df['probability'] = most_frequent_years_df['dates_count'] / total_reviews

# Create a year distribution dictionary from the DataFrame
year_distribution = most_frequent_years_df.set_index('year')['probability'].to_dict()

# Load the Filtered_Reviews_Data dataset from VizierDB
reviews_df = vizierdb.get_data_frame('Updated_Reviews_Data')

# Function to randomly select a year based on the distribution
def impute_year(year_distribution):
    year = np.random.choice(list(year_distribution.keys()), p=list(year_distribution.values()))
    return int(year)  # Convert the year to an integer

# Initialize the 'date_imputed' column to False for all rows
reviews_df['date_imputed'] = False

# Counter for the number of imputed values
num_imputed = 0

# Impute missing dates based on the year distribution
missing_date_indices = reviews_df[reviews_df['date'].isnull()].index
for index in missing_date_indices:
    imputed_year = impute_year(year_distribution)
    reviews_df.at[index, 'date'] = pd.Timestamp(year=imputed_year, month=1, day=1)
    reviews_df.at[index, 'date_imputed'] = True
    num_imputed += 1  # Increment the counter

# Ensure all dates are now Timestamps
reviews_df['date'] = pd.to_datetime(reviews_df['date'])

# Save the DataFrame as a CSV file
reviews_df.to_csv('Reviews_Data_With_Imputed_Dates.csv', index=False)

# Output the number of imputed values
print(f"Number of dates imputed: {num_imputed}")
```

Console | Timing | Datasets | Charts

```
Number of dates imputed: 3838
```

[37]

```
LOAD DATASET Reviews_Data_With_Imputed_Dates AS csv FROM Reviews_Data_With_Imputed_Dates.csv @ url
'Reviews_Data_With_Imputed_Dates.csv'
```

Console | Timing | Datasets | Charts

reviews_data_with_imputed_dates (10000 rows)

Views

	review_id (string)	user_id (string)	business_id (string)	stars (float)	useful (float)	funny (float)	cool (float)	
0	TYslH-CAecjJxLNs96KduA	XCsZ3hWa_6oP1WkWvK7pmg	29YqJwOGEuAWqlHZxMc1OA	5	0	0	0	Lovely little restaurant whic

1	Uc1C5BBmYOw6ZyaDm2MMbw	3MYdpmHeNwC6FquRWi3YOg	5vpqoHruyjHALHjxJLSJbA	1	0	1	0	I once went to this Gavi and
2	9eeoAUQEJDw0KYV5NPYH-A	-qoyKSF2G3PkR_7XNoJfpQ	0ZETsmrLSGWaBu1NNdcmYw	2	2	0	0	Good luck finding anyone t
3	DigjNdnkH35l4NVcwXqT5Q	W4Ez5lpm377893t1XhLquw	nRKndeZLQ3eDL10UMwS2rQ	5	0	0	1	First, Ted Drewes is more th
4	WBmqTb2lrJu9_nIWgZu21g	n-lBS02-3yvlY5Q91mmwDA	grjFEAN9gXOKQFEta4yBJg	3	0	0	0	Tropical Isle is tacky, but th
5	ayL25w-ao4lj3c2_7_bXug	3MYdpmHeNwC6FquRWi3YOg	-pRU97z3uPU_M7yUf5yzMg	4	0	1	0	Great food and handled a la
6	IY7i04LMohgO7HIk5pYvew	_BHTC7nyCBoZcfiiD5cOXg	YtSqYv1Q_pOltsVPSx54SA	5	2	2	2	Despite the fact it's part of
7	AGgFJ8VPX7kAwdy2XTh95w	3MYdpmHeNwC6FquRWi3YOg	uEWsfftrJ7ukPv1xyMVcrg	5	6	0	3	I can't believe I haven't yelp
8	-JNEW5LMvfMG1a2oBuRrCg	3MYdpmHeNwC6FquRWi3YOg	PsHzOS-Jao5CpRl5LKfvlg	2	5	0	0	I don't know, I was told abc
9	-sydClhmvq5JJDsWW5Mo4Q	jTXj5uf5ER5siDnQK_y4aA	d_tRshM-w6S4QxE4VVi8tQ	4	3	0	1	One of the best brunch plac
10	c79fKwD2-OkFAUyT4oiKJw	iw_6JAAw9FAIv2YkbLcMFA	PY9GRfzr4nTZeINf346QOw	4	1	0	0	One of the best hotels in Re
11	UP_c95ne3ts3lx2E-rJrfQ	3xX-Et9l4Bi4wdRPWJkAFg	KA9WEoGHr9Ae0pFXZqzylQ	4	1	0	0	Pool hall, bar, cocktail loung
12	rsH4l8qyCgsMl2es2pgHfw	cOSEvHWqoT5B4d3RcZthyw	XleIcqVgh3iBeM4s-rdy6g	5	1	1	1	This place has the best foo
13	0VFjGL_5PkO1GZQ-EGrEcA	QBHfjwsZF0XUJBqdz_547Q	jxAJrlQ2JjjTs8ZzEytP-w	5	0	0	0	Not only is this place comfo
14	0X0k2Rf2whgkQY6X6yMYTQ	tiLm3eFfpxhwV6O9DEaneQ	jxAJrlQ2JjjTs8ZzEytP-w	5	0	0	0	i love this place for any type
15	pHCO77mLsu-7FRRkLCNMug	HLNNvVX-lZztLUuzTnN73w	fnO03-RX7UDC1TzXETWEsQ	5	2	0	1	This is by far our family's fa
16	FyNLkuX23vWn0sLO1PbSxw	Ct456aElCzDfn03MHZCzsw	iJoGdgWECMNpe-ajs0rfxA	4	0	0	0	Closed Tuesday. Open 10ar
17	ZysvcYxaeY22yBvUR14d2A	A94ivfwchCxwN4rcdoeuJQ	1wC41D8PLqOldFPUVAZ3tA	5	0	1	0	Great looking building walk
18	nXNtmRDfHJvu_j2-3EObcw	38zhUVlh6t_-mpcmbWYmiw	Y3ZCO17N1_T_Ms1JmswwzA	5	0	0	0	Incredibly good and totally
19	XfYXaBr3llgl8shztTpK3A	VQC12BRAJoGvGkiRyDYv3A	SZU9c8V2GuREDN5KgyHFJw	5	1	1	1	I adore you Shellfish Compa

[38]

```
import pandas as pd

business_df = vizierdb.get_data_frame('Business_Data_III')
reviews_df = vizierdb.get_data_frame('Reviews_Data_With_Imputed_Dates')
# Perform the joinreviews_data_with_imputed_dates
integrated_df = pd.merge(business_df, reviews_df, on='business_id', how='inner')
print(integrated_df)

integrated_df.to_csv('Integrated_Data.csv', index=False)
```

Console | Timing | Datasets | Charts

```
     business_id                name  ...        date date_imputed
0     IuKCyfSY7AKhRbRA1JPlPw  Aster's Floral Shop  ...  2018-01-01         True
1     vD2jzpPv4iyOLKzITscGvA  Farmhaus Restaurant  ...  2011-08-01        False
2     vD2jzpPv4iyOLKzITscGvA  Farmhaus Restaurant  ...  2014-12-03        False
3     vD2jzpPv4iyOLKzITscGvA  Farmhaus Restaurant  ...  2015-04-15        False
4     vD2jzpPv4iyOLKzITscGvA  Farmhaus Restaurant  ...  2015-01-01         True
..                       ...                 ...  ...         ...          ...
683   mQvRi0nm84Www71d4qOheQ  Ekta Indian Cuisine  ...  2018-01-01         True
684   mQvRi0nm84Www71d4qOheQ  Ekta Indian Cuisine  ...  2015-01-01         True
685   aiggx_mzALX3kZOGYTPdjg       Hotel Mazarin  ...  2016-01-01         True
686   aiggx_mzALX3kZOGYTPdjg       Hotel Mazarin  ...  2015-01-01         True
687   aiggx_mzALX3kZOGYTPdjg       Hotel Mazarin  ...  2016-01-01         True

[688 rows x 23 columns]
```

[39] `LOAD DATASET Integrated_Data AS csv FROM Integrated_Data.csv @ url 'Integrated_Data.csv'`

Console | Timing | Datasets | Charts

integrated_data (688 rows)

Views

	_c0 (short)	business_id (string)	name (string)	address (string)	city (string)	state (string)	postal_code (float)	latitude (float)
0	9	IuKCyfSY7AKhRbRA1JPlPw	Aster's Floral Shop	41 Haddon Ave	Collingswood	NJ	8108	39.91556167602539
1	59	vD2jzpPv4iyOLKzITscGvA	Farmhaus Restaurant	3257 Ivanhoe Ave	Saint Louis	MO	63139	38.604129791259766
2	59	vD2jzpPv4iyOLKzITscGvA	Farmhaus Restaurant	3257 Ivanhoe Ave	Saint Louis	MO	63139	38.604129791259766
3	59	vD2jzpPv4iyOLKzITscGvA	Farmhaus Restaurant	3257 Ivanhoe Ave	Saint Louis	MO	63139	38.604129791259766
4	59	vD2jzpPv4iyOLKzITscGvA	Farmhaus Restaurant	3257 Ivanhoe Ave	Saint Louis	MO	63139	38.604129791259766
5	59	vD2jzpPv4iyOLKzITscGvA	Farmhaus Restaurant	3257 Ivanhoe Ave	Saint Louis	MO	63139	38.604129791259766
6	59	vD2jzpPv4iyOLKzITscGvA	Farmhaus Restaurant	3257 Ivanhoe Ave	Saint Louis	MO	63139	38.604129791259766
7	59	vD2jzpPv4iyOLKzITscGvA	Farmhaus Restaurant	3257 Ivanhoe Ave	Saint Louis	MO	63139	38.604129791259766
8	90	8OPE0UaZHf7vdYkgblj8vg	Fox and Hound English Pub and Grille	4901 E 82nd St, Ste 900	Indianapolis	IN	46250	39.90424728393555
9	90	8OPE0UaZHf7vdYkgblj8vg	Fox and Hound English Pub and Grille	4901 E 82nd St, Ste 900	Indianapolis	IN	46250	39.90424728393555
10	367	SRb3xScVOeYfhZP4U8LMdA	Paradise Bakery	8540 Castleton Corner Dr	Indianapolis	IN	46250	39.910667419433594
11	367	SRb3xScVOeYfhZP4U8LMdA	Paradise Bakery	8540 Castleton Corner Dr	Indianapolis	IN	46250	39.910667419433594
12	371	zjGHI3SlpU-N3ZPC3skhSg	Grapevine Wines & Spirits	309 S Kirkwood Rd	Saint Louis	MO	63122	38.578548431396484
13	376	QeO2pk5Bg_olt8-j7wg_uQ	Bayshore Pet Resort	4804 S Manhattan Ave	Tampa	FL	33611	27.895435333251953
14	394	zSEFzAbxnO44w8ACQ_XkAA	Akira Hibachi & Sushi Bar	13420 Boyette Rd	Riverview	FL	33569	27.852144241333008
15	394	zSEFzAbxnO44w8ACQ_XkAA	Akira Hibachi & Sushi Bar	13420 Boyette Rd	Riverview	FL	33569	27.852144241333008
16	422	-7GjicSH_rM8JeZGCXGcUg	Double Decker	1721 E 7th Ave	Tampa	FL	33605	27.960084915161133
17	445	0hlXH9jMdHov1VrLC8ujUg	Al's Finger Licking Good Bar-B-Que	1609 Angel Oliva Sr St	Tampa	FL	33605	27.95924949645996
18	445	0hlXH9jMdHov1VrLC8ujUg	Al's Finger Licking Good Bar-B-Que	1609 Angel Oliva Sr St	Tampa	FL	33605	27.95924949645996
19	445	0hlXH9jMdHov1VrLC8ujUg	Al's Finger Licking Good Bar-B-Que	1609 Angel Oliva Sr St	Tampa	FL	33605	27.95924949645996