A perfect blend of flavor and code

Wine And Python

PROJECT 3

Presentation By: Team 2

June 2024

MEET OUR TEAM

Ria Arora

●●●

Analyst

Rob Molenda

●●●

Analyst

Avinash

●●●

Analyst

Ricardo Varela

●●●

Analyst

MEET OUR TEAM

Ria Arora

●●●

Analyst

Rob Molenda

●●●

Analyst

Avinash

●●●

Analyst

Ricardo Varela

●●●

Analyst

Presentation Summary

TOPICS TO COVER

1. Motivation
2. Problem Statement
3. Challenges and Solutions
4. Use Cases
5. Data Collection
6. Approach
7. Workflow Diagram
8. Tools and Packages Applied
9. Limitations/Considerations
10. Future Work Scope
11. Ethical Considerations
12. References

Motivation

Why we chose this data set

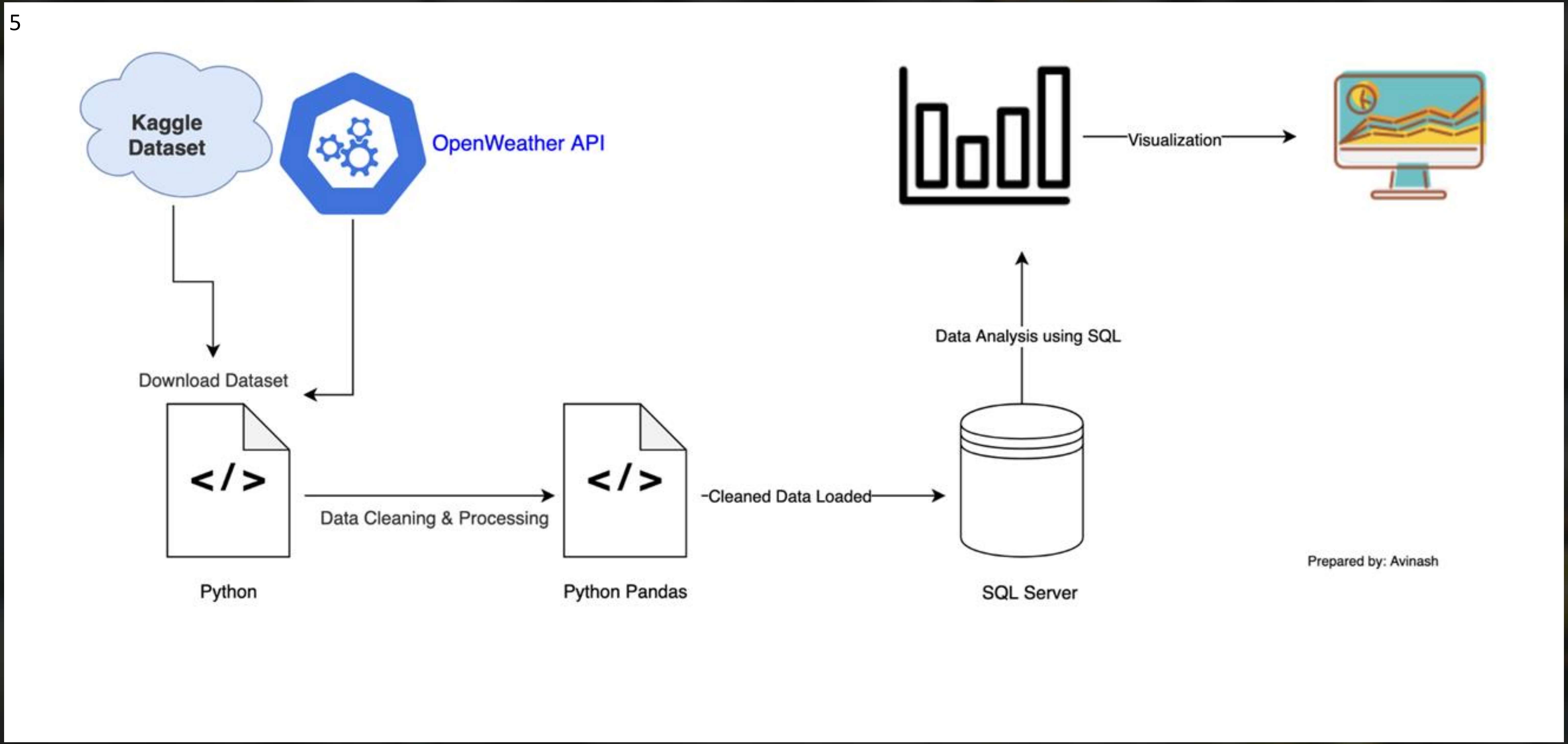
Our group was interested in analyzing the effect that climate played on different industries. We found a data set that focused on a number of wines from around the world, their price, region, variety and their reviewed score. We felt that this data set would be interesting to examine, and since it was raw and unstructured, it would provide a good challenge for us as data engineers.

Approach

OUR PATH

- Problem Framing
- Data Extraction
- Data Transformation
- Data Modeling

Workflow Diagram



Problem Statement

Wine producers and investors would be interested in understanding how climate variables such as temperature and precipitation impact the quality of wines produced in different regions, or temperature conditions where specific wines tend to be of highest quality. Wine enthusiasts may be interested to seek the best quality wines by region.

Our data set would provide interested parties with useful data and analysis that would be helpful for them as decision makers.

CHALLENGES

- OpenWeatherMap historical data is a premium feature
- Data set did not include latitude and longitude for each region
- Inconsistency is regions in our database

Solutions

- We used a second API called MeteoStat, that we were able to collect historical climate data from
- Using OpenWeatherMap, we were able to retrieve lat and lng coordinates for many of the regions in our data set (a happy accident)
- We were forced to reduce the number of regions in our database as some regions did not return lat and lng coordinates.

Use Cases

The impact of climate on wine ratings and price

- Analyzing correlations between temperature, precipitation, price and quality would provide valuable insights for winemakers and marketing strategies

Wine variety and regional characteristics

- Our database can identify regional preferences for wine varieties, and also identify which categories perform best in certain climates.

Predictive modeling for wine quality

Our database could help new vineyards identify the wine variety that would perform best given their regional climate

Data Collection

OUR DATA

We found our wine database on Kaggle. This database was scraped from WineEnthusiast in 2017, it includes 14 columns and 130,000 rows. (https://www.kaggle.com/datasets/zynicide/wine-reviews)

We used OpenWeatherAPI to assign latitude and lng values to the regions in our database

We used meteostat API to collect average temperature and precipitation values.

	A	B	C	D	E	F	G	H
1		country	descriptio	designatic	points	price	province	region_1
2	0	Italy	Aromas in	VulkÃ Bia	87		Sicily & Sa	Etna
3	1	Portugal	This is ripe	Avidagos	87	15	Douro	
4	2	US	Tart and snappy, the		87	14	Oregon	Willamett
5	3	US	Pineapple	Reserve La	87	13	Michigan	Lake Michi
6	4	US	Much like	Vintner's I	87	65	Oregon	Willamett
7	5	Spain	Blackberry	Ars In Vitr	87	15	Northern	Navarra
8	6	Italy	Here's a b	Belsito	87	16	Sicily & Sa	Vittoria
9	7	France	This dry and restrain		87	24	Alsace	Alsace
10	8	Germany	Savory dri	Shine	87	12	Rheinhessen	
11	9	France	This has gı	Les Nature	87	27	Alsace	Alsace
12	10	US	Soft, supp	Mountain	87	19	California	Napa Vall
13	11	France	This is a dry wine, ve		87	30	Alsace	Alsace
14	12	US	Slightly reduced, thi		87	34	California	Alexander
15	13	Italy	This is dor	Rosso	87		Sicily & Sa	Etna
16	14	US	Building on 150 year		87	12	California	Central Co
17	15	Germany	Zesty orar	Devon	87	24	Mosel	
18	16	Argentina	Baked plu	Felix	87	30	Other	Cafayate
19	17	Argentina	Raw black	Winemak	87	13	Mendoza	Mendoza
20	18	Spain	Desiccate	Vendimia	87	28	Northern	Ribera del

Data Extraction

	Unnamed: 0	country	points	price	province	title	variety	winery
0	0	Italy	87	NaN	Sicily & Sardinia	Nicosia 2013 Vulkà Bianco (Etna)	White Blend	Nicosia
1	1	Portugal	87	15.0	Douro	Quinta dos Avidagos 2011 Avidagos Red (Douro)	Portuguese Red	Quinta dos Avidagos
2	2	US	87	14.0	Oregon	Rainstorm 2013 Pinot Gris (Willamette Valley)	Pinot Gris	Rainstorm

```python
#Creating a list of unique provinces
province_list = wine_df["province"].unique().tolist()

#Iterating through our list and appending values
city_list = []

for city in province_list:
    print(city)
    city_list.append(city)
```

✓ 0.0s

- First we read in the data
- We then created a list of regions by looping through our data frame
- We attempted to retrieve historical weather data using OpenWeatherMap API, however this was unsuccessful so we pivoted to using MeteoStat API instead

```python
#Function to retrieve weather data from meteostat given lat/lng coordinates
def get_weather_data(lat, lng):
    location = Point(lat, lng) # Creating a Point object for the given coordinates
    try:
        data = Monthly(location, start, end)
        data = data.fetch() # Fetching monthly weather data
        return data[['tavg', 'prcp']] # Returning average temp and total precipitation
    except:
        return None

# Define the time period for the data
start = datetime(2023, 1, 1)
end = datetime(2023, 12, 31)

# Initialize lists to hold the weather data
avg_temps = []
total_prcps = []

# Iterate over the rows in the dataframe
for index, row in city_data_df.iterrows():
    weather_data = get_weather_data(row['Lat'], row['Lng'])
    if weather_data is not None and not weather_data.empty:
        avg_temps.append(weather_data['tavg'].mean())
        total_prcps.append(weather_data['prcp'].sum())
    else:
        avg_temps.append(None)
        total_prcps.append(None)

# Add the weather data to the dataframe
city_data_df['Avg_Temp_2023'] = avg_temps
city_data_df['Total_Prcp_2023'] = total_prcps
```

Add reference of table

Data Transformation

Dropping null values

```
# Dropping null values from dataframe
wine_df.dropna(inplace=True)
#Deleting Unnamed column
del wine_df ["Unnamed: 0"]
```

✓ 0.0s

Removing special characters with regex

```
# Define a function to remove special characters
def remove_special_characters(df):
    # Using regex to replace special characters with empty string
    df = df.replace(r'[^A-Za-z0-9 ]+', '', regex=True)
    return df

# Apply the function to the dataframe
cleaned_wine_df = remove_special_characters(wine_df)

print(cleaned_wine_df)
```

✓ 0.6s

Converting data types to prepare for SQL loading

```
# Convert 'regionID', 'wineID and 'price' columns to int64 to match SQL ERD
ordered_wine_df['region_id'] = ordered_wine_df['region_id'].astype('int64')
ordered_wine_df['price'] = ordered_wine_df['price'].astype('int64')
ordered_wine_df['wine_id'] = ordered_wine_df['wine_id'].astype('int64')
```

✓ 0.0s

Reordering data frame columns

```
# Reorder dataframe columns
city_order = ['region_id', 'region', 'lat', 'lng', 'temp', 'prcp']

ordered_city_df = city_df[city_order]
```

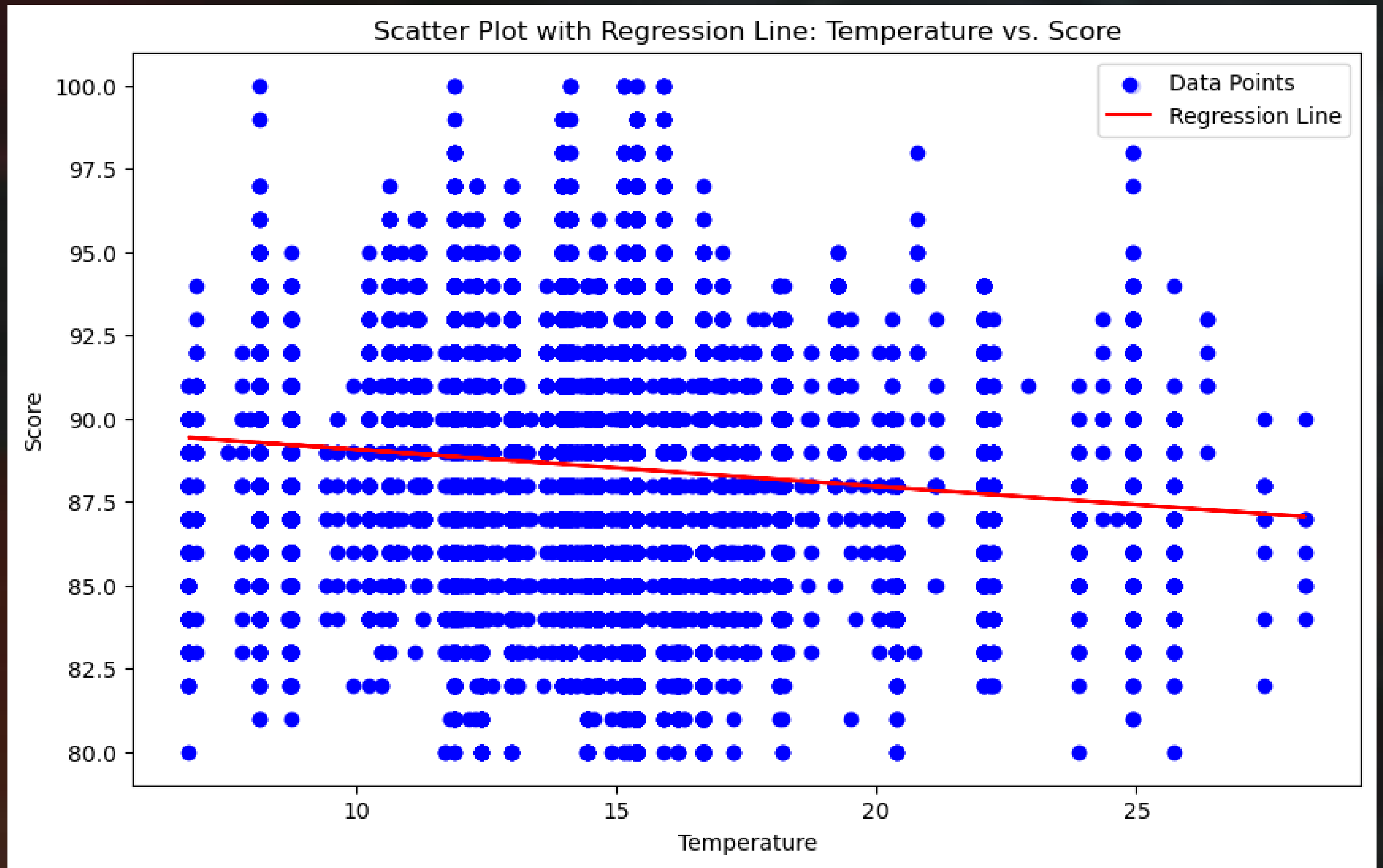
✓ 0.0s

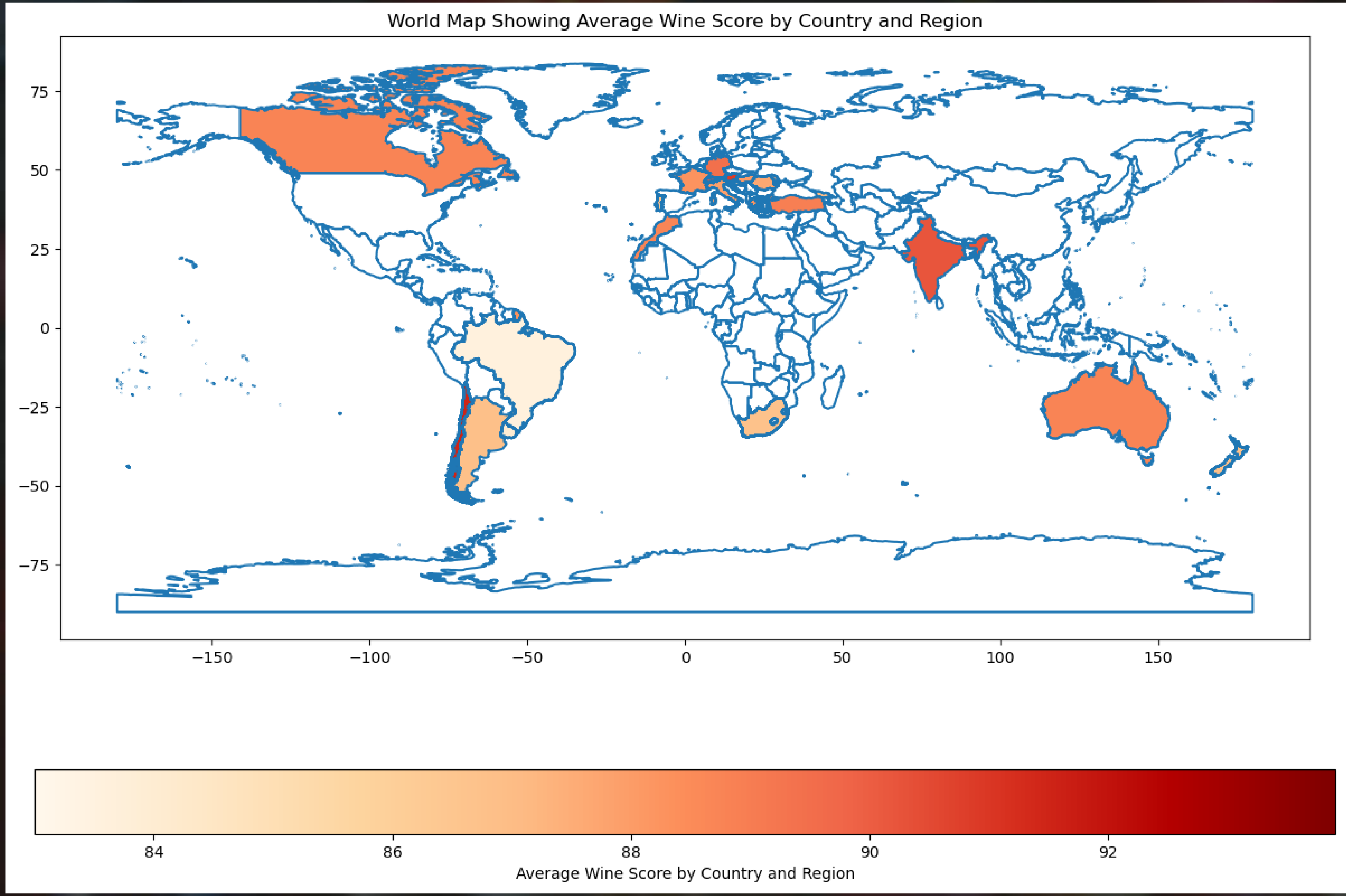
Creating a unique ID for our regions and wine titles

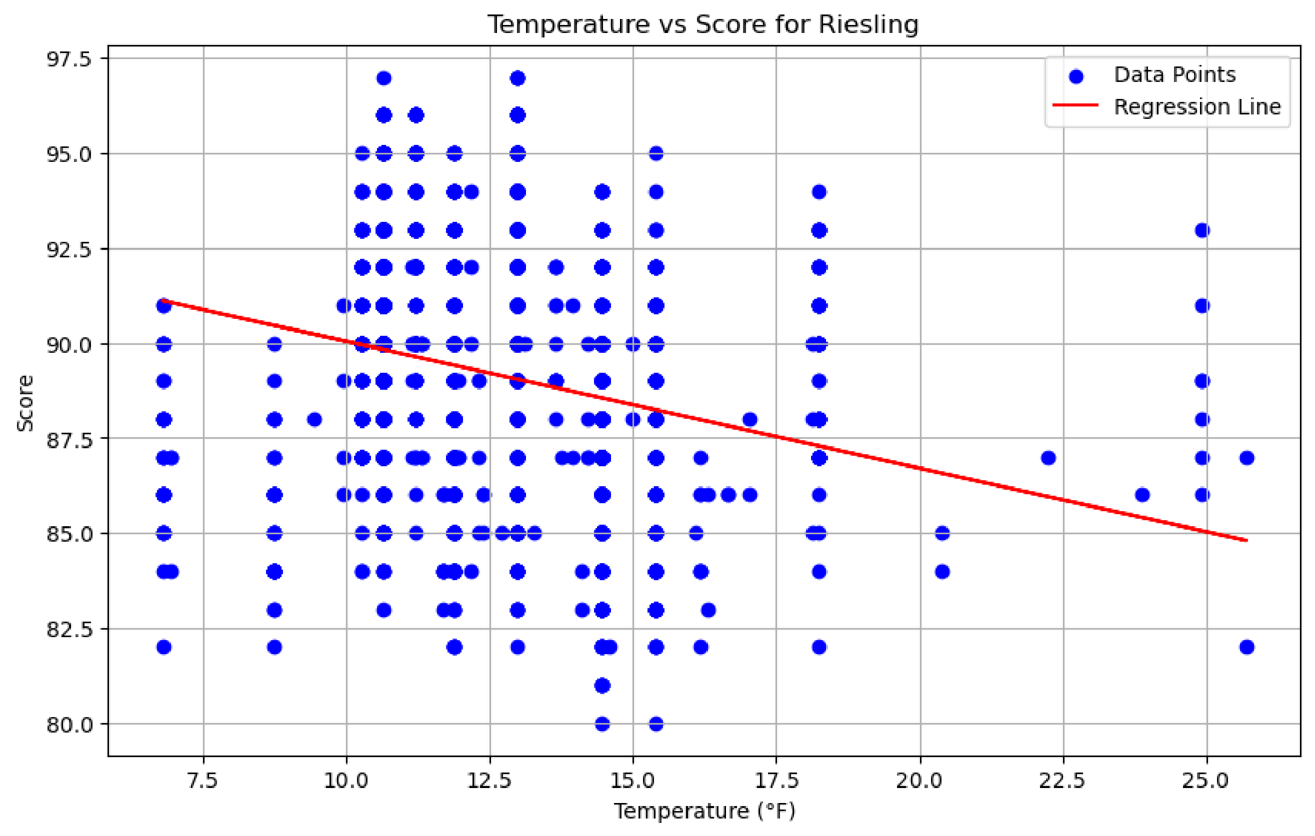
```
# Create a unique regionID for each unique region, and a wineID for each wine title
cleaned_wine_df['regionID'] = cleaned_wine_df['region'].astype('category').cat.codes
cleaned_wine_df['wineID'] = cleaned_wine_df['title'].astype('category').cat.codes
```

✓ 0.1s

Pandas Data Analysis







Data Modeling and Validation

[8]
```
wine_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72703 entries, 0 to 72702
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   wine_id    72703 non-null  int64
 1   title      72703 non-null  object
 2   category   72703 non-null  object
 3   winery     72703 non-null  object
 4   region_id  72703 non-null  int64
dtypes: int64(2), object(3)
memory usage: 2.8+ MB
```

[10]
```
region_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 154 entries, 0 to 153
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   region_id  154 non-null    int64
 1   region     154 non-null    object
 2   lat        154 non-null    float64
 3   lng        154 non-null    float64
 4   temp       154 non-null    float64
 5   prcp       154 non-null    float64
dtypes: float64(4), int64(1), object(1)
memory usage: 7.3+ KB
```

[12]
```
price_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72703 entries, 0 to 72702
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   wine_id    72703 non-null  int64
 1   region_id  72703 non-null  int64
 2   price      72703 non-null  int64
 3   score      72703 non-null  int64
dtypes: int64(4)
memory usage: 2.2 MB
```

```
wine_df.region_id.nunique()
```

153

```
price_df['region_id'].nunique()
```

153

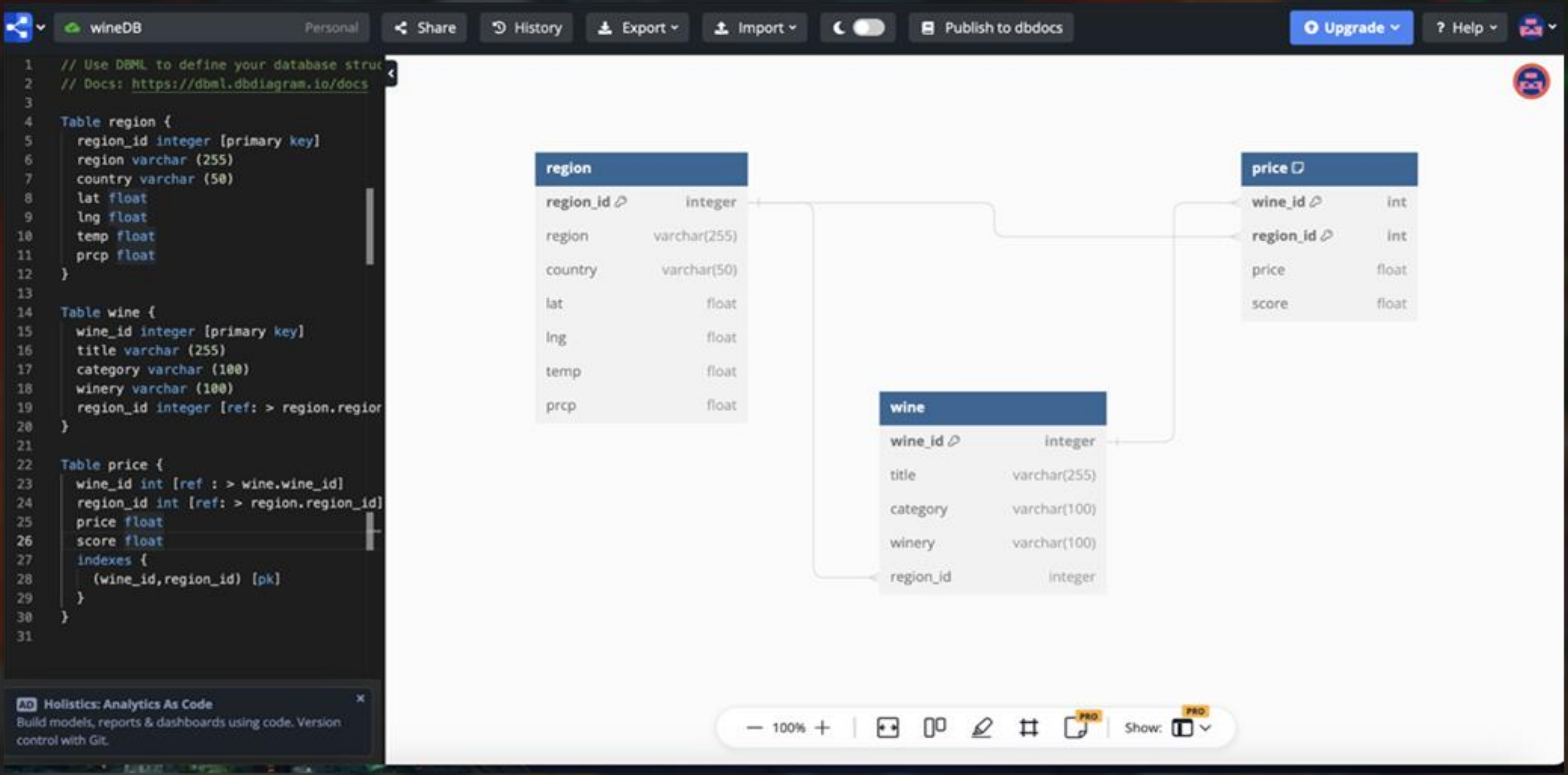
```
price_df['wine_id'].nunique()
```

72703

```
wine_df.wine_id.nunique()
```

72703

ERD Diagram


wineDB
Personal
Share
History
Export
Import
Publish to dbdocs
Upgrade
Help
// Use DBML to define your database struc
// Docs: https://dbml.dbdiagram.io/docs
Table region {
region_id integer [primary key]
region varchar (255)
country varchar (50)
lat float
lng float
temp float
prcp float
}
Table wine {
wine_id integer [primary key]
title varchar (255)
category varchar (100)
winery varchar (100)
region_id integer [ref: > region.regior
}
Table price {
wine_id int [ref : > wine.wine_id]
region_id int [ref: > region.region_id]
price float
score float
indexes {
(wine_id,region_id) [pk]
}
}
Holistics: Analytics As Code
Build models, reports & dashboards using code. Version control with Git.
region
region_id integer
region varchar(255)
country varchar(50)
lat float
lng float
temp float
prcp float
price
wine_id int
region_id int
price float
score float
wine
wine_id integer
title varchar(255)
category varchar(100)
winery varchar(100)
region_id integer
100%
Show:

Create Database and Table Scheme

```sql
DROP TABLE IF EXISTS region;
CREATE TABLE  "region" (
  "region_id" integer PRIMARY KEY,
  "region" varchar(255),
  "country" varchar(150),
  "lat" float,
  "lng" float,
  "temp" float,
  "prcp" float
);

DROP TABLE IF EXISTS wine;
CREATE TABLE "wine" (
  "wine_id" integer PRIMARY KEY,
  "title" varchar (255),
  "category" varchar(100),
  "winery" varchar(100),
  "region_id" integer
);

DROP TABLE IF EXISTS price;
CREATE TABLE "price" (
  "wine_id" int,
  "region_id" int,
  "price" float,
  "score" float,
  PRIMARY KEY ("wine_id", "region_id")
);

ALTER TABLE "wine" ADD FOREIGN KEY ("region_id") REFERENCES "region" ("region_id");

ALTER TABLE "price" ADD FOREIGN KEY ("wine_id") REFERENCES "wine" ("wine_id");

ALTER TABLE "price" ADD FOREIGN KEY ("region_id") REFERENCES "region" ("region_id");
```

SQLite Connection and Analysis

```python
# SQLite connection
import sqlite3
conn = sqlite3.connect('wine.db')
```
✓ 0.0s

```python
def import_csv_to_sqlite(csv_file, table_name):
    df = pd.read_csv(csv_file)
    df.to_sql(table_name, conn, if_exists='replace', index=False)
```
✓ 0.0s

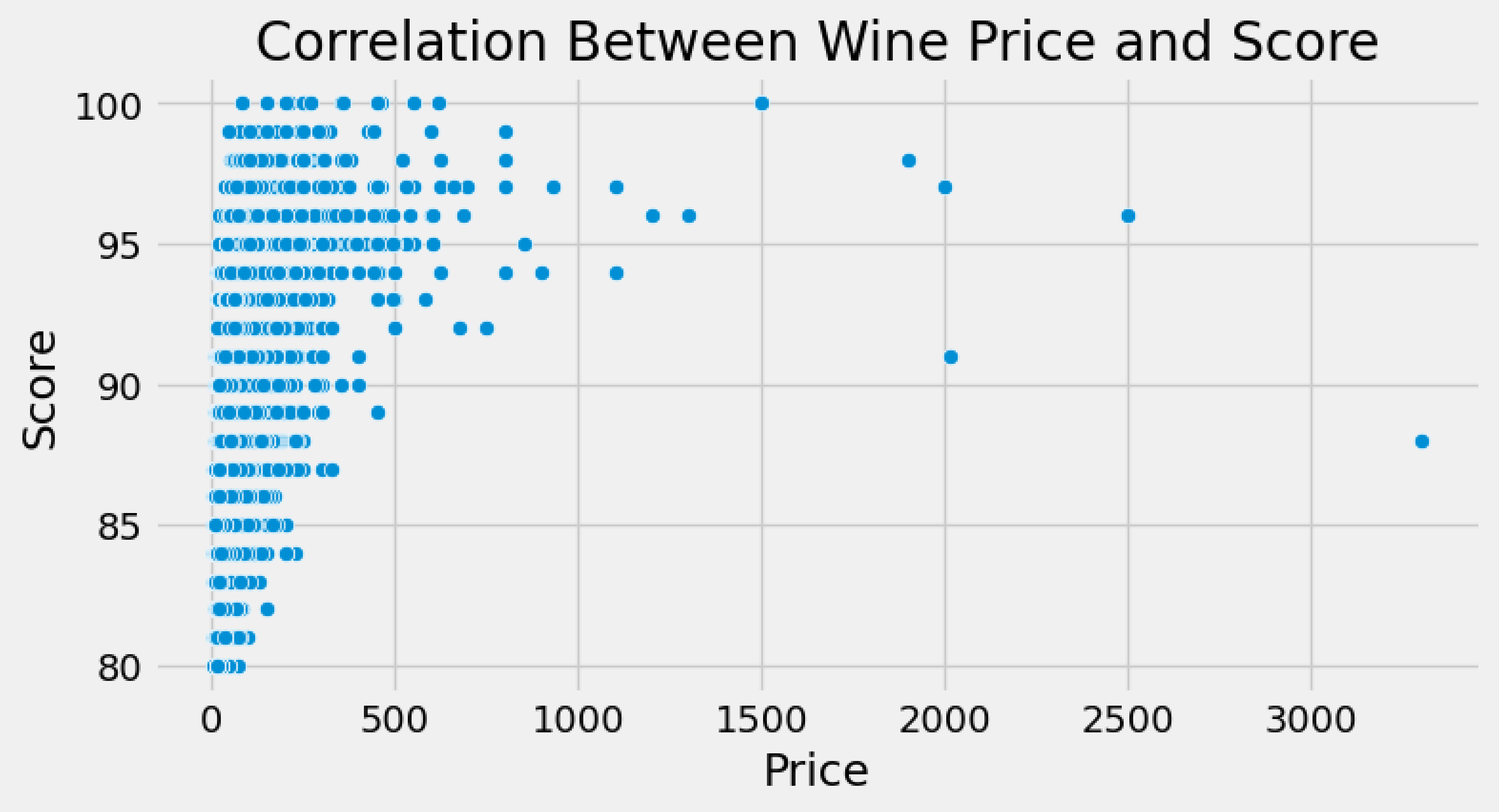
```python
# File paths and table names
files_and_tables = {
    "Resources/CityData_sql.csv": "region",
    "Resources/Wine_new.csv": "wine",
    "Resources/Price_new.csv": "price"
}
```
✓ 0.0s

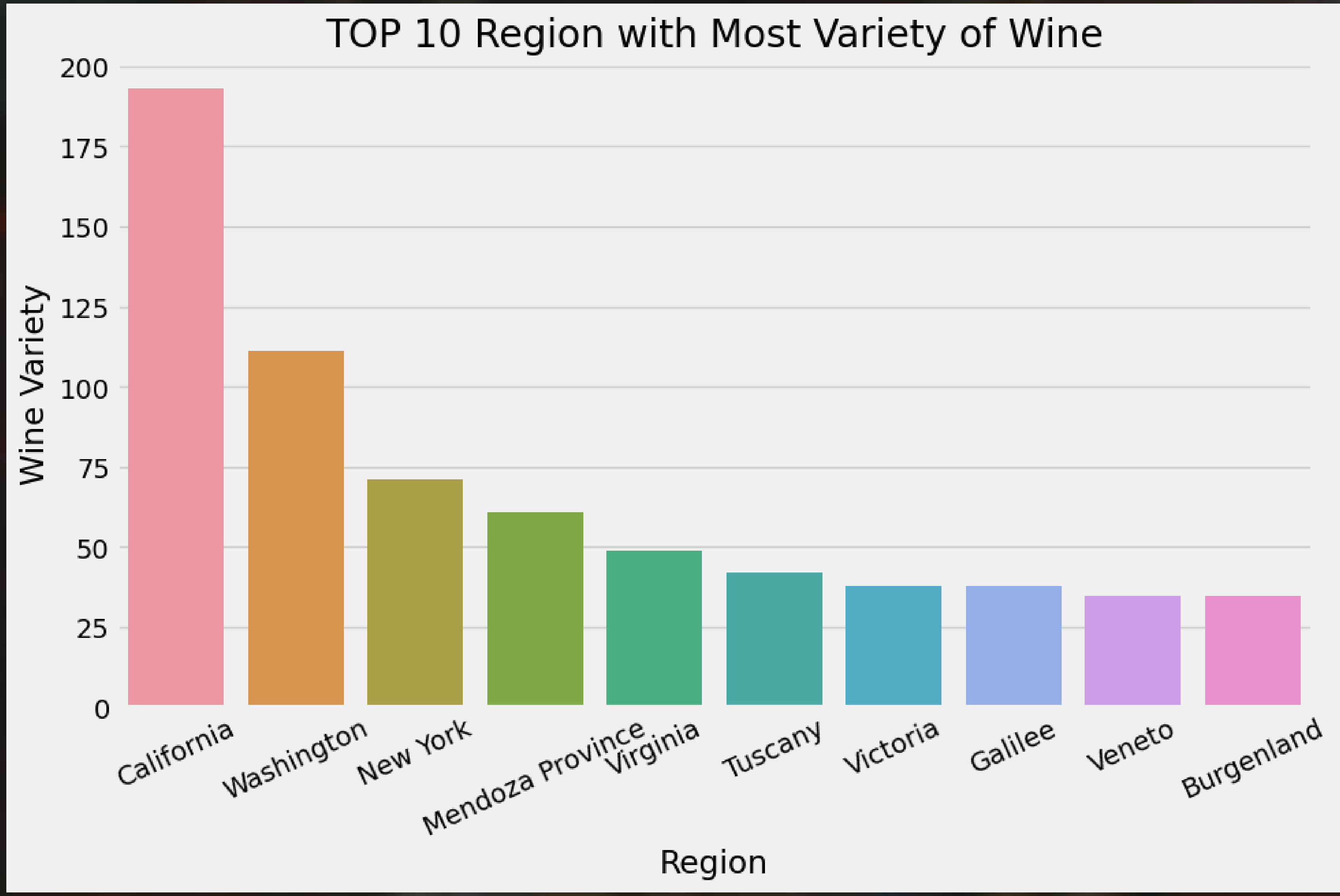
```python
# Import each CSV file into its corresponding SQLite table
for csv_file, table_name in files_and_tables.items():
    import_csv_to_sqlite(csv_file, table_name)
```
✓ 0.3s

```python
# # # create engine to wine.sqlite

engine = create_engine("sqlite:///wine.db", echo=True)
```

```python
query = """
SELECT p.price, p.score
FROM price p;
"""
price_score_data = pd.read_sql_query(query, conn)

plt.figure(figsize=(8, 4))
sns.scatterplot(x='price', y='score', data=price_score_data)
plt.title('Correlation Between Wine Price and Score')
plt.xlabel('Price')
plt.ylabel('Score')
plt.show()
```

```python
regionWithMostVariety = pd.read_sql_query("""SELECT r.region, COUNT(DISTINCT w.category) AS wine_varieties
FROM wine w
JOIN region r ON w.region_id = r.region_id
GROUP BY r.region
ORDER BY wine_varieties DESC LIMIT 10;""",conn)

plt.figure(figsize=(10, 6))
sns.barplot(y='wine_varieties', x='region', data=regionWithMostVariety)
plt.title('TOP 10 Region with Most Variety of Wine')
plt.ylabel('Wine Variety')
plt.xlabel('Region')
plt.tight_layout()
plt.xticks(rotation = 25)
plt.show()
```





SQL Query

```sql
----Top 10 wines by scores
Select p.wine_id, w.title, p.score
FROM price p
Join wine w on p.wine_id = w.wine_id
order by p.score desc
limit 10
```

wine_id	title	score
9140	Biondi Santi 2010 Riserva Brunello di Montalcino	100
63620	Louis Roederer 2008 Cristal Vintage Brut Champagne	100
97221	Tenuta dellOrnellaia 2007 Masseto Merlot Toscana	100
16455	Casa Ferreirinha 2008 BarcaVelha Red Douro	100
15828	Cardinale 2006 Cabernet Sauvignon Napa Valley	100
22680	Chteau Loville Barton 2010 SaintJulien	100
19446	Chambers Rosewood Vineyards NV Rare Muscat Rutherglen	100
5750	Avignonesi 1995 Occhio di Pernice Vin Santo di Montepulciano	100
57709	Krug 2002 Brut Champagne	100
87562	Salon 2006 Le Mesnil Blanc de Blancs Brut Chardonnay Champag	100

```sql
---Bottom 10 wines and their scores
Select p.wine_id, w.title, p.score
FROM price p
Join wine w on p.wine_id = w.wine_id
order by p.score asc
limit 10
```

wine_id	title	score
42058	Finca El Origen 2007 Gran Reserva Malbec Uco Valley	80
105282	Vina Robles 2004 Cabernet Sauvignon Paso Robles	80
84478	Ricardo Santos 2009 Smillon Mendoza	80
2242	Alma del Sur 2009 Coleccin Cabernet SauvignonMalbec Mendoza	80
76526	Pascual Toso 2007 Torronts Maip	80
39718	Esser Cellars 2005 Zinfandel California	80
9785	Bodega Carmine Granata 2009 Smillon Mendoza	80
29345	Cruz Alta 2007 Grand Reserve Malbec Mendoza	80
45289	Gardel 2009 Torronts Mendoza	80
78491	Pianetta 2004 Cabernet Sauvignon Monterey	80

SQL Query

```
--- average temperature and impact on wine score
----- Case 1:
Select  r.region, r.temp, r.prcp, avg(p.score) as avg_score
FROM price p
Join wine w on p.wine_id = w.wine_id
join region r on w.region_id = r.region_id
group by r.region, r.temp, r.prcp
order by avg_score desc
limit 10
```

region	temp	prcp	avg_score
Madeira	20.775	576.1	93.90909091
Puente Alto	14.51818182	471.8	91.85714286
Wachau	11.2	740.4	91.79166667
England	19.27272727	956.6	91.76271186
Santa Cruz	26.34166667	1476.3	91.5
Eisenberg	10.89166667	677.3	91.2
Buin	14.51818182	398.3	91.14285714
Gladstone	22.9	711	91
Wagram	11.13636364	1061.4	90.82758621
Champagne	14.11818182	694.8	90.52902622

TOOLS AND PACKAGES APPLIED

- Python
- Pandas
- Numpy
- Pathlib
- matplotlib
- Requests
- Datetime
- Config

- PostgreSql

- Jupyter Notebook

- Meteostat
- dbd digagram
- GeoPandas
- Openweathermap

LIMITATIONS

- Region data is inconsistent
- Dataset is from 2017 and might not be an accurate reflection of current markets
- Pricing format is unknown

CONSIDERATIONS

- Weather data was taken from a 1 year period

Future Work Scope

Other Factors

Other factors such as soil quality, topography, agriculture would be interesting to explore.

Geojson

Use geojson to fetch the lat and lng coordinates for the missing regions.

Year of Production

We would have liked to take the year that the wine was produced into our analysis.

References

- Kaggle wine data set - https://www.kaggle.com/datasets/zynicide/wine-reviews
- OpenWeatherAPI - https://openweathermap.org/api
- MeteoStatAPI - https://dev.meteostat.net/python/

Thank You

Hope you enjoyed it