

VinoVoyage

Zora Mardjoko, Allison Mi, George Xue, Kerry Zhao

Group 50

Demo Video Link: https://youtu.be/Crfn_1TAyGY

1. Introduction

Motivation and Goals

Choosing the right bottle of wine can often be overwhelming, especially for those with limited knowledge about wine varieties and flavor profiles. The motivation behind this application stems from the desire to enhance the wine selection process for consumers by leveraging the vast amount of data available. We aim to streamline the wine selection process, allowing users to search by wine title and filter based on their desired price and points, and also find sommeliers to connect with on Twitter. Additionally, we aim to incorporate an educational component in our app, allowing users to learn more about wine. Overall, our application is designed to demystify the wine selection process, enrich the user's understanding of wine, and foster a more connected and informed community of wine enthusiasts.

Application Functionality

Our app offers a user-friendly interface where consumers can easily search for wines by title and apply filters such as price range and rating points to narrow down their options to suit their preferences and budget. This feature is particularly useful for those seeking to explore new wines or to ensure they get the best value for their money. Users can also view top sommeliers, ranked by the number of wines they reviewed, and access the sommeliers' Twitter pages through links on our site. Additionally, we implemented a random wine recommender, as well as a wine trivia game.

2. Architecture

System Architecture & Application

Our web application is designed using the client-server model, where the server-side is implemented using Node.js and the client-side is a React application written in JSX. Our application's deployment was facilitated using Vercel, and the live version of our site can be accessed here: <https://vino-voyage.vercel.app/>.

We store our data in one database on DataGrip, with 6 tables (refer to Appendix B) constructed from the 4 datasets (refer to Appendix A), providing data about Wine Reviews, GSOD (daily global surface summary), NOAA GSOD (station numbers and country abbreviations), and Countries ISO Code (country name and abbreviation). We use MySQL (MySQL-Connector hosted on AWS) since the relations between these tables are well-defined. For example, each wine has a country of origin, and those countries have data surrounding their country's climate.

3. Data

1. Wine Reviews Dataset: <https://www.kaggle.com/datasets/zynicide/wine-reviews>

This dataset includes the name of the wine coupled with a description of the flavor profile. It also includes the country of origin, price, and rating among other qualities that define the wine. The price is in dollars, and the wine rating system used is by Wine Enthusiast. The Wine Enthusiast scale ranges from 80-100, ranging from "acceptable" to "classic". Additionally, each review has a taster name and taster twitter handle, corresponding to the sommelier that tasted that wine.

2. NOAA GSOD Dataset: <https://www.kaggle.com/datasets/noaa/gsod>

The above dataset contains information on the daily global surface summary from over 9000 weather stations from 1929 to now, from mean temperature to visibility to precipitation. Each station has a unique number identifier.

3. NOAA GSOD Country Dataset: <https://www.kaggle.com/datasets/noaa/gsod?select=stations>

The above dataset contains information about every station, including its name, position (latitude and longitude), and the country it is located in. Notably, the country is provided as an abbreviation, not the full country name.

4. Countries ISO Codes Dataset: <https://www.kaggle.com/datasets/juanumusic/countries-iso-codes>

The above dataset relates the ISO Code to the country name. For example, "FR" corresponds to France. We use this dataset since the NOAA GSOD Dataset uses ISO Code for the country that a station is located in, and we wanted to query by country name.

Statistics

Wine Reviews Dataset

Size Statistics:

52.9 mb

129,970 rows

13 attributes

Summary Statistics:

Took the numeric values from the Wine Reviews Dataset:

Mean Price: 35.36864434814251

Standard Deviation Price: 41.03105236050748

Mean Points (out of 100): 88.42172251811175

Standard Deviation Points: 3.0449418777069135

Min Price: 4.0

Max Price: 3300.0

Min Points: 80

Max Points: 100

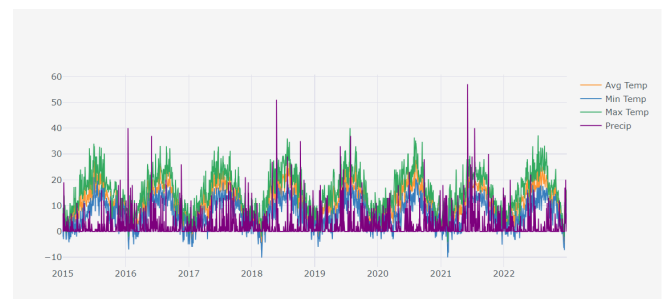
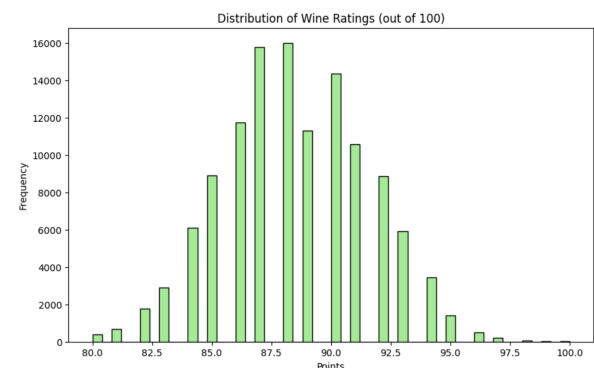
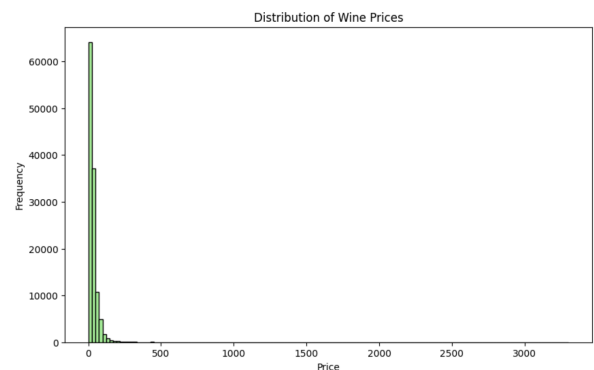
NOAA GSOD Dataset - 2019

Size Statistics: 3.29m rows.

Summary Statistics:

Avg Temp, Min temp, Max Temp, and Precipitation values for 'BRUSSELS NATL' station (daily values from years 2015-2023 shown here)

We were able to relate these databases through connecting the weather conditions in top wine producing countries with different wines and their reviews by sommeliers. For instance, the



country dataset was used to connect abbreviated country names with full country names, and country is also what connected the wine dataset with GSOD. We focused on top producing wine countries, and as such, for the GSOD dataset, used Argentina, Australia, Spain, Italy, New Zealand, United States, and South Africa.

As for the preprocessing, we filtered (choosing the year 2019 for the NOAA GSOD dataset, and also choosing the countries Argentina, Australia, Spain, Italy, New Zealand, United States, and South Africa to focus on). Next, we did normalization, where we split Wine_Reviews into Wine, Location, and Sommelier. For incorrect/missing data, we dropped rows (ex. such as those with missing or null values). For incorrect data, we checked ranges - for instance, the points scale for reviews was meant to be between 0-100, so anything outside of this range was dropped. Additionally, duplicates were removed and we standardized the format, ensuring consistency in naming (capitalization, US vs. United States), lowercases, and extra spaces).

To highlight various relationships, we created the 6 tables below (for instance, station (Stn) in GSOD refers to USAF in Country):

1. Country - USAF, country
2. Country_Abb - name, abbreviation
3. GSOD - Stn, Yr, Mo, Da, Temp, Dewp, Slp, Max, Min, Prcp
4. Location - Title, Country, Province
5. Sommelier - title, taster_name, taster_twitter_handle
6. Wine - title, description, designation, points, price, variety, winery

4. Database

Data Ingestion: We populated tables using our four datasets, using the Wine Reviews dataset to populate the Wine, Location, and Sommelier tables, the NOAA GSOD dataset to populate the GSOD table, the NOAA GSOD Countries dataset to populate the Countries table, and the Countries ISO Codes dataset to populate the Country_Abb table. We dropped rows containing null values.

ER Table:

1. Wine (title, description, designation, points, price, variety, winery)
 - Number of instances: 63,193 rows
 - Normal Form used: BCNF
 - Justification: There is only one candidate key, that being title, and there are no partial or transitive dependencies.
2. Location (title, country, province) FOREIGN KEY title REFERENCES Wine(title)
 - Number of instances: 63,193 rows
 - Normal Form used: 3NF
 - Justification: Candidate keys are title, country, and province. Here, we have partial dependencies, but the dependencies are all part of some key, so we are in 3NF.
3. Sommelier (title, taster_name, taster_twitter_handle) FOREIGN KEY title REFERENCES Wine(title)
 - Number of instances: 63,193 rows
 - Normal Form used: 3NF
 - Justification: Candidate keys are title, taster_name, and taster_twitter_handle. Here, we have partial dependencies, but the dependencies are all part of some key, so we are in 3NF.
4. GSOD (stn, yr, mo, da, temp, dewp, slp, max, min, prcp)

- Number of instances: 292,338 rows
- Normal Form used: 3NF
- Justification: Here, all non key attributes are dependent on the whole key, and there are no partial dependencies, so we are in Boyce-Codd.

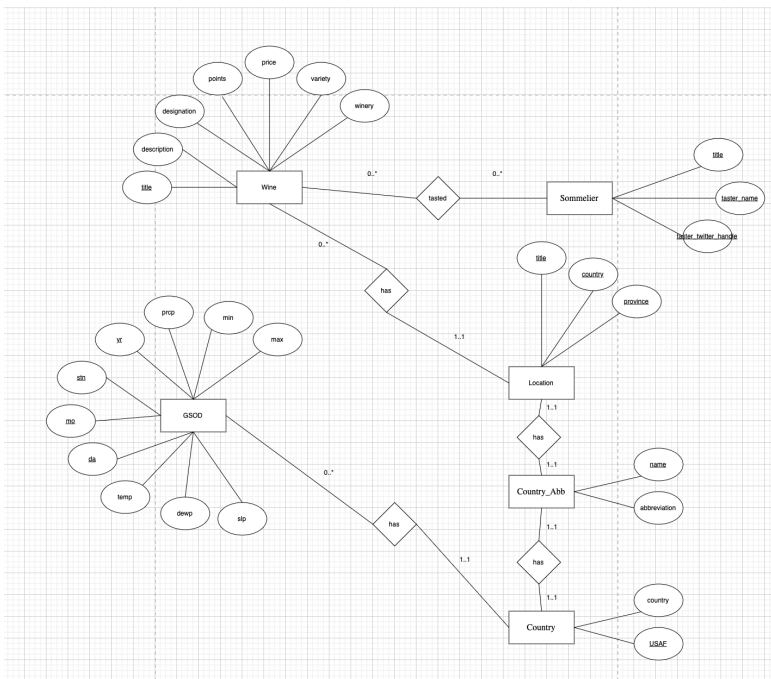
5. Country_Abb (name, abbreviation) FOREIGN KEY name REFERENCES Location(country)

- Number of instances: 247 rows
- Normal Form used: BCNF
- Justification: This is in BCNF because abbreviation is dependent on name, which is a candidate key, and there are no other dependencies.

6. Country (USAF, country) FOREIGN KEY USAF References GSOD stn, FOREIGN KEY country REFERENCES Country_Abb(abbreviation)

- Number of instances: 461 rows
- Normal Form used: BCNF
- Justification: This is in BCNF because country is dependent on USAF, which is a candidate key, and there are no other dependencies.

ER Diagram:



5. Web App description

Our web app has 4 pages: a home page, sommeliers page, wine search page, and trivia page. The motivation behind this was to capture various functionalities we wanted to implement in our web app. For example, we knew we wanted a search function for wines, but also be able to interact with the sommelier data. Photos can be found in Appendix E.

Home Page:

Our home page displays a random wine at the top, then has a grid layout of the top wines, ordered by points. Users can click on each wine card (or the random wine link) to open a pop up that displays more information about that wine, like its title, designation (which is the name of the wine given to it by the producer), price (in dollars), points (on the Wine Enthusiast scale), and variety and winery.

Sommeliers Page:

Our sommeliers page lists the sommeliers in order from most wines reviewed to fewest. Since our sommelier data includes their Twitter handle, we wanted a way to link this in our application, so that users can access Twitter profiles of their favorite sommeliers. Clicking on the sommelier's name takes a user to their Twitter page. Since our database contained the Twitter handles in the format "@mytwitterhandle", we used regex to match the "@" symbol and remove it, so we could easily concatenate the sommelier's Twitter handle to "<https://twitter.com/>" and route to the appropriate page.

Wine Search Page:

This page allows users to search for a wine, filtering by title, price, and points. The title search matches any wine that contains the substring of the user's search input, while the price and points slides match any wine that falls into that range. The results are clickable, and link to more information about the wine.

Trivia Page:

Our trivia page features 6 "static" and 6 "dynamic" questions in a fun quiz format to test the user's wine knowledge ("dynamic" = used query within the question). This page features the majority of our complex queries and some simple queries which are listed at the end (Appendix C & D)

- What is the term used for the year a wine's grapes were harvested?
- Which country is the largest producer of wine in the world?
- Select the wine that's < \$20 and from the U.S. (simple query)
- What type of wine is "Sherry"?
- Which red wine grape is most associated with the Bordeaux region of France?
- Which wine is not commonly described as citrusy? (simple query)
- What is the process called in which sugars in grape juice are converted to alcohol by yeast?
- Dynamic Question (Complex Query)
- What is the average point value of wines from Washington that have been tasted by sommeliers that have tasted over 10 wines? (Complex Query)
- Dynamic Question (Complex Query)
- Dynamic Question (Complex Query)
- Name a wine region in New Zealand famous for its Sauvignon Blanc.

6. API Specification

We used the following routes in our application:

- **1. Random Wine Title**
 - **Path:** /random

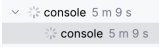
- Method: GET
- Description: Returns a random wine title and its associated winery.
- Request Parameters: None
- Response:
 - title: string, title of the wine.
 - winery: string, name of the winery.
- **2. Specific Wine Information**
 - Path: /wine/:title
 - Method: GET
 - Description: Fetches detailed information about a wine based on its title.
 - URL Parameters:
 - title: string, title of the wine.
 - Response:
 - JSON object containing all the fields from the Wine table for the specified title.
- **3. List of Sommeliers and their Wine Counts**
 - Path: /sommeliers
 - Method: GET
 - Description: Provides a list of sommeliers along with the number of wines they have tested.
 - Request Parameters: None
 - Response: Array of objects with each containing:
 - number_of_wines: number
 - sommelier: string, name of the sommelier.
 - taster_twitter_handle: string, Twitter handle of the sommelier without the '@' symbol.
- **4. Top Wines**
 - Path: /top_wines
 - Method: GET
 - Description: Retrieves a list of wines ordered by their rating points.
 - Query Parameters:
 - page: integer (optional), specifies the page of results to return.
 - page_size: integer (optional), number of results per page.
 - Response:
 - Array of wine objects sorted by points in descending order.
- **5. Search for Wines**
 - Path: /search_wines
 - Method: GET
 - Description: Allows for searching wines based on various attributes.
 - Query Parameters:
 - title: string (optional)
 - points_lower_bound: integer (optional)
 - points_upper_bound: integer (optional)
 - price_lower_bound: integer (optional)
 - price_upper_bound: integer (optional)
 - Response:
 - Array of wines that match the criteria.
- **6. Question One**

- Path: /question_one
- Method: GET
- Description: Retrieves a wine under \$20 from the US.
- Request Parameters: None
- Response: JSON object containing the title of the wine.
- **7. Question Two**
 - Path: /question_two
 - Method: GET
 - Description: Finds wines with descriptions that include the word 'citrus'.
 - Request Parameters: None
 - Response: Array of wine objects.
- **8. Question Three**
 - Path: /question_three
 - Method: GET
 - Description: Computes the average rating points of wines from Washington reviewed by sommeliers who have reviewed more than 10 wines.
 - Request Parameters: None
 - Response: JSON object with average points.
- **Question Four**
 - Path: /question_four
 - Method: GET
 - Description: optimizes database interactions by creating an index, fetching average ratings and maximum ratings of wines grouped by country; retrieves the details of the highest-rated wine from each country
 - Request Parameters: None
 - Response: JSON Array of objects containing country, avg_rating, title, and max_rating
- **Question Five**
 - Path: /question_five
 - Method: GET
 - Description: Retrieves the most relevant sommelier based on specific conditions such as the number of tastings and average wine ratings, excluding certain criteria. The endpoint also involves creating and managing temporary tables for optimized data processing
 - Request Parameters: None
 - Response: JSON object containing taster_name, num_tasting, and avg_wine_rating
- **Question Six**
 - Path: /question_six
 - Method: GET
 - Description: Calculates the average price of non-fruit described wines from countries with higher than average temperatures; leverages multiple data tables and custom indexes for optimized query performance.
 - Request Parameters: None
 - Response: JSON Array of objects containing country, avg_temp, and average price

7. Queries

Refer to Appendix C for simple queries and Appendix D for complex queries.

8. Performance evaluation

Query #	Time before Optimization	Time after Optimization	Description	Operations
9	39 s	1.3 s	Find the average wine price in Argentina for wineries where the average sea level pressure is lower than the mean sea level pressure at Nota Bene.	<p>indexing, caching, decreasing size of intermediate joins.</p> <p>Indexing on price in Wine and Slp in GSOD decreased the time required to group by and aggregate among those columns. Caching (representing the creation of the tables AvgSlp and GSOD_STN as views) allowed for reusing some intermediate results and joining on a heavily reduced table, decreasing the overall cost needed for the operation.</p>
10		929 ms	Average Prices of Wines from Countries with the highest average yearly temperatures. This query will calculate the average yearly temperature for each country with above mean average temperatures, and then find the average price of non-fruity wines for those countries.	<p>indexing, caching, using only relevant features from tables before performing operations.</p> <p>Created an index on idx_country_abbreviation on country in Country, which decreased the time required to group by, join, and aggregate that column. Caching using tables FruitWine and Countries_Temp allowed for joining on tables of decreased size.</p>
11	1 s 574 ms	670 ms	Identify sommeliers who have tasted wines with average ratings above 3.0 and whose Twitter handle is not "@vossroger", containing wines with descriptions containing "fruit", excluding those from Argentina or New Zealand, and with average ratings above 4.0.	<p>Indexing, caching, existential checks.</p> <p>Created an index on Location (Title, Country) which sped up joins and filtering based on these columns and used caching through the creation of the table FruitWine, which allowed for the amount of rows to be filtered down (as Fruit</p>

				Wines only contained description with fruit and not in the countries AR or NZ, so less was passed in to the later join with Fruit Wine and Sommelier, versus the optimized version.
12	15.9 s	630 ms	Find the average ratings of wines from each country and retrieve the details of the highest-rated wine from each country.	<p>Indexing, caching, pushing selections and projections down to the lowest level.</p> <p>Created index idx_country_abbreviation on Country(country), decreasing the cost of aggregating over that column. Used caching through the table MaxPoints representing a view, which pre-aggregates the data. Through subqueries and indexing, it limited the amount of data processed and sped up data retrieval. The unoptimized had performance issues due to direct repetitive calculation of large volumes of data during joins.</p>

9. Technical challenges

One technical challenge was downloading our dataset. The NOAA GSOD data was available on Kaggle as a BigQuery dataset, but this created a barrier for downloading the data. We couldn't just download a csv file straight from the website, and since the dataset was so large, we had to use BigQuery to specify countries and download country specific datasets.

Another technical challenge we had to figure out was deploying the website. We wanted to implement this extra credit functionality and were initially confused about how to set up the deployment due to our client server model, but eventually figured out that we needed to deploy the frontend and backend separately.

One of our biggest technical challenges was trying to optimize our queries. It took many tries to find what worked and what didn't, how to effectively create indices, how to use views effectively, and so on, especially with the representation of views as actual tables. Many of the implementations we thought would optimize runtime actually ended up not affecting or even increasing the runtime, but eventually we got through this. Additionally, if someone was working on the frontend website while someone else was working on the queries with the database, someone creating indexes by running the website (for instance) in the trivia game where we have complex queries would affect someone trying to run unoptimized code, as it would show the appearance of the index.

10. Extra Credit Features

```
PASS  __tests__/tests.js (13.458 s)
✓ GET /random (313 ms)
✓ GET /wine/Black Stallion 2013 Limited Release Merlot (Napa Valley) (48 ms)
✓ GET /wine/randomwordnotindata (81 ms)
✓ GET /sommeliers (94 ms)
✓ GET /top_wines top 12 (117 ms)
✓ GET /top_wines with pages (101 ms)
✓ GET /top_wines with pages error (51 ms)
✓ GET /search_wines one result (135 ms)
✓ GET /search_wines no results (208 ms)
✓ GET /question_one (52 ms)
✓ GET /question_two (1317 ms)
✓ GET /question_three (5639 ms)
✓ GET /question_four (452 ms)
✓ GET /question_five (1523 ms)
✓ GET /question_six (435 ms)

-----|-----|-----|-----|-----|-----
File      | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-----|-----|-----|-----|-----|-----
All files |    89.5 |    83.92 |     100 |    89.44 |
index.js  |     100 |       100 |       100 |       100 |
routes.js |    88.02 |    83.92 |       100 |    87.94 | ...154,202-203,228,267-268,293,367-368,393,432-433
-----|-----|-----|-----|-----|-----

Test Suites: 1 passed, 1 total
Tests:       15 passed, 15 total
Snapshots:   0 total
Time:        13.81 s
```

The first extra credit feature we implemented was backend unit testing. We achieved 89.44% code coverage for the backend.

Our second extra credit feature involved deploying the completed website to ensure it was accessible to users. For this, we utilized Vercel. The live version of the website can be accessed through the following link: <https://vino-voyage.vercel.app/>.

Appendix A - Datasets

1. Wine Reviews Dataset: <https://www.kaggle.com/datasets/zynicide/wine-reviews>

This dataset includes the name of the wine coupled with a description of the flavor profile. It also includes the country of origin, price, and rating among other qualities that define the wine. The price is in dollars, and the wine rating system used is by Wine Enthusiast. The Wine Enthusiast scale ranges from 80-100, ranging from “acceptable” to “classic”. Additionally, each review has a taster name and taster twitter handle, corresponding to the sommelier that tasted that wine.

2. NOAA GSOD Dataset: <https://www.kaggle.com/datasets/noaa/gsod>

The above dataset contains information on the daily global surface summary from over 9000 weather stations from 1929 to now, from mean temperature to visibility to precipitation. Each station has a unique number identifier.

3. NOAA GSOD Country Dataset: <https://www.kaggle.com/datasets/noaa/gsod?select=stations>

The above dataset contains information about every station, including its name, position (latitude and longitude), and the country it is located in. Notably, the country is provided as an abbreviation, not the full country name.

4. Countries ISO Codes Dataset: <https://www.kaggle.com/datasets/juanumusic/countries-iso-codes>

The above dataset relates the ISO Code to the country name. For example, “FR” corresponds to France. We use this dataset since the NOAA GSOD Dataset uses ISO Code for the country that a station is located in, and we wanted to query by country name.

Appendix B - Tables

1. Country - USAF, country
2. Country_Abb - name, abbreviation
3. GSOD - Stn, Yr, Mo, Da, Temp, Dewp, Slp, Max, Min, Prcp
4. Location - Title, Country, Province
5. Sommelier - title, taster_name, taster_twitter_handle
6. Wine - title, description, designation, points, price, variety, winery

Appendix C - Simple Queries

Query #1. Look for wines under \$20 from the US. Filter by price below \$20 (USD), where location = U.S. This query was used in our trivia game.

```
SELECT *
FROM Wine W
JOIN Location L on W.title = L.title
WHERE price < 20 AND L.country = 'US';
```

Query #2. Wine that matches a user-inputted description (here, description contains ‘flavorful’). This query was used in our search function.

```
SELECT *
FROM Wine
WHERE description LIKE '%flavorful%';
```

Query #3. Find wines from a country with a higher average temperature than Italy, where points are above 90 (this

requires joining the wine dataset with the weather and country datasets). Will need to find the average temperature of each country, compare it to Italy and filter by having points larger than 90.

```
WITH average_temp AS (  
  SELECT AVG(temp) AS avg_temperature_IT  
  FROM GSOD G  
  JOIN Country C ON G.stn = C.USAF  
  WHERE country = 'IT'  
)  
SELECT *  
FROM Wine W  
JOIN Location L ON W.title = L.title  
JOIN Country C ON L.country = C.country  
JOIN GSOD G ON G.stn = C.USAF  
WHERE G.temp > (SELECT avg_temperature_IT FROM average_temp)  
AND W.points > 90;
```

Query #4. Find the average points of wines from Washington that have been tasted by sommeliers that have tasted over 10 wines (this requires a subquery to find sommeliers that have tasted over 10 wines). This query was used in our trivia game.

```
CREATE INDEX idx_L_province ON Location (province);
```

```
CREATE TABLE IF NOT EXISTS Washington_Locations (  
  title VARCHAR(255),  
  province VARCHAR(255),  
  INDEX idx_L_province (province)  
)  
SELECT title, province  
FROM Location  
WHERE province = 'Washington';
```

```
SELECT AVG(W.points) AS average_points  
FROM Wine W  
JOIN Washington_Locations L ON W.title = L.title  
JOIN Sommelier S ON W.title = S.title  
WHERE S.taster_name IN (  
  SELECT taster_name  
  FROM Sommelier  
  GROUP BY taster_name  
  HAVING COUNT(title) > 10  
);
```

Query #5. Find the top 10 average wine score (points) for each country.

```
SELECT AVG(W.points) AS average_points
FROM Wine W
JOIN Location L ON W.title = L.title
GROUP BY L.country, L.province
ORDER BY average_points DESC
LIMIT 10;
```

Query #6. Find the average wine score for wines from Italy, correlating with the average daily temperature in June 2020.

```
SELECT L.Country, AVG(W.points) AS average_points, AVG(G.Temp) AS average_temperature
FROM Wine W
JOIN Location L ON W.title = L.Title
JOIN Country_Abb CA ON CA.name = L.Country
JOIN Country C ON CA.abbreviation = C.country
JOIN GSOD G ON C.USAF = G.Stn
WHERE L.Country = 'Italy'
AND G.Yr = '2020'
AND G.Mo = '6'
GROUP BY L.Country;
```

Query #7. Find the wines in the country with the highest precipitation.

```
SELECT DISTINCT W.title, L.country, W.points, G.Prcp
FROM Wine W
JOIN Location L ON W.title = L.Title
JOIN Country_Abb CA ON CA.name = L.Country
JOIN Country C ON C.country = CA.abbreviation
JOIN GSOD G ON C.USAF = G.Stn
WHERE G.Prcp =
    (SELECT MAX(G2.Prcp)
     FROM GSOD G2)
LIMIT 5;
```

Query #8. Identify the top 5 sommeliers who reviewed the most wines based in Australia.

```
SELECT S.taster_name, COUNT(*) AS number_reviews
FROM Sommelier S
JOIN Wine W ON W.title = S.title
JOIN Location L ON W.title = L.title
WHERE L.country = 'Australia'
GROUP BY S.taster_name
ORDER BY number_reviews ASC
LIMIT 5;
```

Appendix D - Complex Queries

Query #9. Find the average wine price in Argentina for wineries where the average sea level pressure is lower than the mean sea level pressure at Nota Bene. This query was used in our trivia game.

optimized -

```
CREATE INDEX idx_wine_price ON Wine(price);
CREATE INDEX idx_Slp ON GSOD(Slp);

CREATE TABLE AvgSlp AS (
  SELECT AVG(G.Slp)
  FROM Wine W
  JOIN Location L ON W.title = L.title
  JOIN Country C ON L.Country = C.country
  JOIN GSOD G ON G.Stn = C.USAF
  WHERE W.winery = ':Nota Bene'
);

CREATE TABLE GSOD_STN AS (
  SELECT G.Stn, AVG(G.Slp) AS Avg_Slp
  FROM GSOD G
  GROUP BY G.Stn
);

SELECT L.Country, AVG(price) AS avg_price
FROM Wine W
  JOIN Location L ON W.title = L.Title AND L.Country = 'Argentina'
  JOIN Country_Abb CA ON L.Country = CA.name
  JOIN Country C ON C.country = CA.abbreviation
  JOIN GSOD_STN G ON G.Stn = C.USAF
GROUP BY L.Country
HAVING AVG(G.Avg_Slp) < (SELECT * FROM AvgSlp);
```

unoptimized -

```
SELECT L.Country, AVG(W.price) AS AvgPrice
FROM Wine W
  JOIN Location L ON W.title = L.Title
  JOIN Country_Abb CA ON L.Country = CA.name
  JOIN Country C ON C.country = CA.abbreviation
  JOIN GSOD G ON G.Stn = C.USAF
WHERE C.country = 'AR'
HAVING AVG(G.Slp) < (
  SELECT AVG(G.Slp)
```

```

FROM Wine W
JOIN Location L ON W.title = L.title
JOIN Country C ON L.Country = C.country
JOIN GSOD G ON G.Stn = C.USAF
WHERE W.winery = ':Nota Bene'
)

```

Query #10. Average Prices of Wines from Countries with the highest average yearly temperatures. This query will calculate the average yearly temperature for each country with above mean average temperatures, and then find the average price of non-fruity wines for those countries. This query was used in our trivia game.

optimized:

```

CREATE INDEX idx_country_abbreviation ON Country(country);
WITH FruitWine AS (
    SELECT
        CA.abbreviation AS country,
        W.price
    FROM
        Location L
    JOIN
        Wine W ON L.Title = W.Title
    JOIN Country_Abb CA ON L.Country = CA.name
    WHERE
        W.description NOT LIKE '%fruit%'
),
Countries_Temp AS (
    SELECT C.country, AVG(G.temp) AS avg_temp
    FROM Country C
    JOIN GSOD G ON G.stn = C.USAF
    GROUP BY C.country
)
SELECT C.country, C.avg_temp, AVG(FW.price)
FROM
    Countries_Temp C
JOIN FruitWine FW ON C.country = FW.country
WHERE C.avg_temp > (SELECT AVG(temp) FROM GSOD)
GROUP BY
    C.country
ORDER BY
    C.avg_temp DESC

```

unoptimized:

```

WITH FruitWine AS (

```

```

SELECT
    L.Country,
    W.price
FROM
    Location L
JOIN
    Wine W ON L.Title = W.Title
WHERE
    W.description NOT LIKE '%fruit%'
)
SELECT
    C.country,
    AVG(G.temp) AS avg_temp,
    FW.price
FROM
    GSOD G
JOIN
    Country C ON G.stn = C.USAF
JOIN FruitWine FW ON C.country = FW.country
GROUP BY
    C.country
HAVING
    AVG(G.temp) > (SELECT AVG(temp) FROM GSOD)
ORDER BY
    AVG(G.temp) DESC

```

Query #11. Identify sommeliers who have tasted wines with average ratings above 3.0 and whose Twitter handle is not "@vossroger", containing wines with descriptions containing "fruit", excluding those from Argentina or New Zealand, and with average ratings above 4.0. This query was used in our trivia game.

optimized:

```

CREATE INDEX idx_location_title_country
ON Location (Title, Country);

```

```

CREATE TEMPORARY TABLE WineRatings AS
SELECT
    W.title,
    AVG(W.points) AS avg_rating
FROM
    Wine W

```



```

GROUP BY
    W.title;

CREATE TABLE FruitWine AS
SELECT
    L.Country,
    W.price,
    W.title,
    WR.avg_rating
FROM
    Location L
JOIN
    Wine W ON L.Title = W.title
JOIN
    WineRatings WR ON W.title = WR.title
WHERE
    W.description LIKE '%fruit%'
    AND L.Country NOT IN ('AR', 'NZ')

SELECT
    S.taster_name,
    COUNT(*) AS num_tastings,
    AVG(FW.avg_rating) AS avg_wine_rating
FROM
    Sommelier S
JOIN
    FruitWine FW ON FW.title = S.title
WHERE
    S.taster_twitter_handle NOT LIKE '@vossroger'
    AND NOT EXISTS (
        SELECT 1
        FROM WineRatings WR
        WHERE WR.title = S.title
        AND WR.avg_rating < 3.0
    )
GROUP BY
    S.taster_name
HAVING
    num_tastings > 3
ORDER BY

```

```
    num_tastings DESC
LIMIT 1;
```

unoptimized:

```
SELECT
S.taster_name,
COUNT(*) AS num_tastings,
AVG(sub.avg_rating) AS avg_wine_rating
FROM
(SELECT
    W.title,
    W.description,
    L.Country,
    WR.avg_rating
FROM
    Wine W
JOIN
    Location L ON W.title = L.Title
JOIN
    (SELECT
        W1.title,
        AVG(W1.points) AS avg_rating
    FROM
        Wine W1
    WHERE
        W1.description LIKE '%fruit%'
    GROUP BY
        W1.title
    ) WR ON W.title = WR.title
WHERE
    L.Country NOT IN ('AR', 'NZ')
    AND W.description LIKE '%fruit%'
) sub
JOIN
Sommelier S ON sub.title = S.title
WHERE
S.taster_twitter_handle NOT LIKE '@vossroger'
AND sub.avg_rating >= 3.0
GROUP BY
S.taster_name
HAVING
num_tastings > 3
```

```
ORDER BY
num_tastings DESC
LIMIT 1;
```

Query #12. Find the average ratings of wines from each country and retrieve the details of the highest-rated wine from each country. This query was used in our trivia game.

optimized:

```
CREATE INDEX idx_country_abbreviation ON Country(country);

CREATE TABLE MaxPoints AS (
    SELECT
        L.Country,
        MAX(W.points) AS max_points
    FROM Wine AS W
    JOIN Location AS L ON W.title = L.Title
    GROUP BY
        L.Country
);

SELECT C.country, AVG(W.points) AS avg_rating, W.title, W.points AS max_rating
FROM (SELECT title, points FROM Wine) AS W
JOIN (SELECT Title, country FROM Location) AS L ON W.title = L.Title
JOIN (SELECT name, abbreviation FROM Country_Abb) AS CA ON L.country = CA.name
JOIN (SELECT DISTINCT country FROM Country) AS C ON CA.abbreviation = C.country
JOIN (SELECT max_points, country FROM MaxPoints) AS MaxPoints ON W.points =
MaxPoints.max_points AND L.country = MaxPoints.country
GROUP BY C.country, W.title
ORDER BY C.country;
```

unoptimized:

```
SELECT C.country, AVG(W.points) AS avg_rating, W.title, W.points AS max_rating
FROM (SELECT title, points FROM Wine) AS W
JOIN (SELECT Title, Country FROM Location) AS L ON L.Title = W.Title
JOIN Country_Abb CA ON L.Country = CA.name
JOIN (SELECT country FROM Country) AS C ON CA.abbreviation = C.country
JOIN
    (SELECT country, MAX(points) AS max_points
     FROM Wine W1
     JOIN Location L1 ON W1.title = L1.Title
     GROUP BY country)
AS MaxPoints ON W.points = MaxPoints.max_points
```

```
AND L.country = MaxPoints.country
GROUP BY C.country, W.title
ORDER BY C.country;
```

Appendix E: Website Photos

Home Page:

VINOVOYAGEWINESSOMMELIERSTRIVIA

Wine of The Refresh: [Trapiche 2007 Broquel Torrontés \(Mendoza\)](#)

Top Wines in our Dataset!

Biondi Santi 2010
Riserva (Brunello di
Montalcino)
Price: \$550
Points: 100

Casa Ferreirinha 2008
Barca-Velha Red
(Douro)
Price: \$450
Points: 100

Cayuse 2008 Bionic
Frog Syrah (Walla
Walla Valley (WA))
Price: \$80
Points: 100

Chambers Rosewood
Vineyards NV Rare
Muscat (Rutherglen)
Price: \$350
Points: 100

Charles Smith 2006
Royal City Syrah
(Columbia Valley (WA))
Price: \$80
Points: 100

Krug 2002 Brut
(Champagne)
Price: \$259
Points: 100

Louis Roederer 2008
Cristal Vintage Brut
(Champagne)
Price: \$250
Points: 100

Quinta do Noval 2011
Nacional Vintage (Port)
Price: \$650
Points: 100

Wine Search Page:

VINOVOYAGEWINESSOMMELIERSTRIVIA

Search Wines

TitleDescription

PricePoints

SEARCH

Results

TitleDescription

No rows

Rows per page: 100-0 of 0<>

Sommeliers Page:

Top Sommeliers

#	Sommelier	Wines Reviewed
1	Roger Voss	12721
2	Michael Schachner	11078
3	Kerin O'Keefe	6202
4	Virginie Boone	6114
5	Paul Gregutt	6004
6	Matt Kettmann	4073
7	Sean P. Sullivan	3137
8	Anna Lee C. Iijima	3137
9	Joe Czerwinski	3090
10	Anne Krebiehl MW	2551

Trivia Page:

What is the term used for the year a wine's grapes were harvested?

Blend

Vintage

Reserve

Estate