Nagalaxmi Eepuri

HUBBLEMIND SQL Internship

SQL Project Report: Border_Crossing_Entry_Data

SQL Tasks – Solutions

Date: 5-11-2024

Project Description: Analysis of border crossing data between the U.S.-Canada and U.S.-Mexico borders. It will engage in tasks that involve data exploration, aggregations, and advanced SQL queries, simulating real-world data analysis scenarios

Data source: Dataset

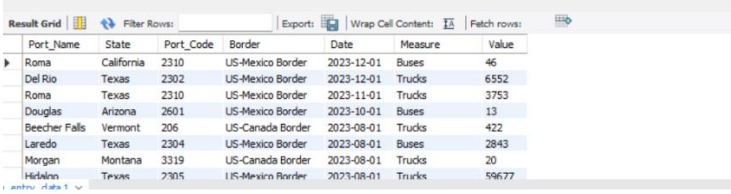
Database Environment: MySQL WorkBench

Analysis Techniques: SQL queries

First created a SCHEMA named Cross_Border_Crisis then created TABLE named Border_Crossing_Entry_Data with columns given in the dataset.

```
-- creating schema and table

CREATE DATABASE IF NOT EXISTS cross_border_crisis;

USE cross_border_crisis;


CREATE TABLE  border_crossing_entry_data(
         Port_Name VARCHAR(50),
         State VARCHAR(50),
        Port_Code int,
        Border VARCHAR(50),
        Date CHAR(10),
        Measure VARCHAR(50),
        Value int
        );
SELECT *FROM border_crossing_entry_data;


-- Then imported data by selecting Data Import Wizard in Server menu

SELECT * FROM border_crossing_entry_data;
```

OUTPUT:



Port_Name	State	Port_Code	Border	Date	Measure	Value
Roma	California	2310	US-Mexico Border	2023-12-01	Buses	46
Del Rio	Texas	2302	US-Mexico Border	2023-12-01	Trucks	6552
Roma	Texas	2310	US-Mexico Border	2023-11-01	Trucks	3753
Douglas	Arizona	2601	US-Mexico Border	2023-10-01	Buses	13
Beecher Falls	Vermont	206	US-Canada Border	2023-08-01	Trucks	422
Laredo	Texas	2304	US-Mexico Border	2023-08-01	Buses	2843
Morgan	Montana	3319	US-Canada Border	2023-08-01	Trucks	20
Hidalgo	Texas	2305	US-Mexico Border	2023-08-01	Trucks	59677

- -

Week 1: Data Exploration and Basic Queries

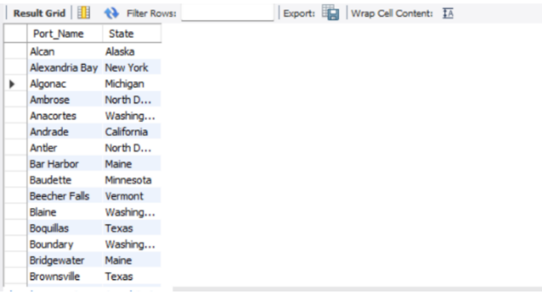
1. List all distinct port names and their corresponding states.

Here is an SQL query to list all distinct port names and their corresponding states from the Border Crossing table:

QUERY:

```
SELECT DISTINCT port_name, state FROM border_crossing_entry_data;
```

OUTPUT:



Port_Name	State
Alcan	Alaska
Alexandria Bay	New York
Algonac	Michigan
Ambrose	North D...
Anacortes	Washing...
Andrade	California
Antler	North D...
Bar Harbor	Maine
Baudette	Minnesota
Beecher Falls	Vermont
Blaine	Washing...
Boquillas	Texas
Boundary	Washing...
Bridgewater	Maine
Brownsville	Texas

DESCRIPTION:

1. SELECT DISTINCT: Ensures that the results contain only unique combinations of port names and states.
2. port_name, state: Specifies the columns to be included in the output.

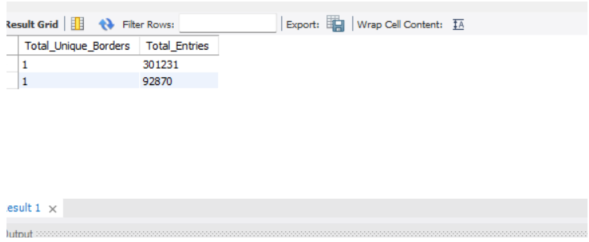
- -

2. Count the total number of unique Borders and the total number of entries associated with each Border.

QUERY:

```
SELECT COUNT(DISTINCT Border) AS Total_Unique_Borders, COUNT(*) AS Total_Entries

FROM border_crossing_entry_data

GROUP BY Border;
```

OUTPUT:



Total_Unique_Borders	Total_Entries
1	301231
1	92870

DESCRIPTION:

1. COUNT(DISTINCT Border) AS Total_Unique_Borders: Counts the unique borders and labels the result as Total_Unique_Borders.
2. COUNT(*) AS Total_Entries: Counts the total number of entries for each border and labels it as Total_Entries.
3. GROUP BY Border: Groups the results by each border.

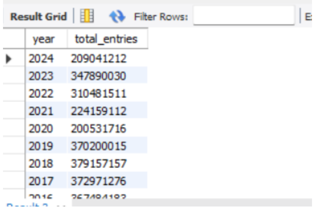
- -

3. Retrieve the total number of entries (crossings) for each year, sorted from most recent to oldest year.

QUERY:

```
SELECT YEAR(Date) AS year, SUM(Value) AS total_entries FROM border_crossing_entry_data

GROUP BY YEAR(Date)

ORDER BY year DESC;
```

OUPUT:



year	total_entries
2024	209041212
2023	347890030
2022	310481511
2021	224159112
2020	200531716
2019	370200015
2018	379157157
2017	372971276
2016	367484183

DESCRIPTION:

1. SELECT YEAR(Date) AS year: Extracts the year from the Date column and labels it as year.
2. SUM(Value) AS total_entries: Calculates the total number of entries (crossings) for each year and labels the result as total_entries.
3. FROM border_crossing_entry_data: Specifies the table from which to retrieve the data.
4. GROUP BY YEAR(Date): Groups the results by year, allowing the aggregation function SUM() to compute totals for each year.
5. ORDER BY year DESC: Sorts the results in descending order by year, so the most recent year appears first.

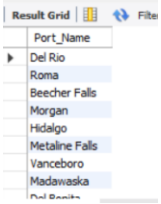
- -

4. Find all ports that have recorded more than 5000 crossings for the Trucks measure type.

QUERY:

```
SELECT Port_Name FROM border_crossing_entry_Data

WHERE Measure = 'Trucks'

GROUP BY Port_Name

HAVING SUM(Value) > 5000;
```

OUTPUT:



Port_Name
Del Rio
Roma
Beecher Falls
Morgan
Hidalgo
Metaline Falls
Vanceboro
Madawaska
Del Bonita

DESCRIPTION:

1. WHERE Measure = 'Trucks': Filters the data to include only entries related to truck crossings.
2. GROUP BY Port_Name: Groups the results by port name.
3. HAVING SUM(Value) > 5000: Ensures that only ports with a total of more than 5,000 crossings are included in the results.

- -

5. Identify the top 3 states with the highest total number of pedestrian crossings.

QUERY:

SELECT State, **SUM**(Value) **AS** total_pedestrian_crossings **FROM** border_crossing_entry_data

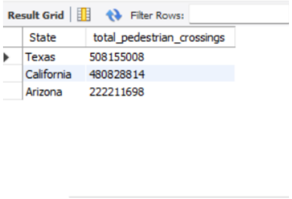
WHERE Measure = 'Pedestrians'

GROUP BY State

ORDER BY total_pedestrian_crossings **DESC**

LIMIT 3;

OUTPUT:



State	total_pedestrian_crossings
Texas	508155008
California	480828814
Arizona	222211698

DESCRIPTION:

1. WHERE Measure = 'Pedestrians': Filters the data to include only pedestrian crossings.
2. SUM(Value) AS total_pedestrian_crossings: Calculates the total number of pedestrian crossings for each state and labels it as total_pedestrian_crossings.
3. GROUP BY State: Groups the results by state.
4. ORDER BY total_pedestrian_crossings DESC: Sorts the states in descending order based on the total number of crossings.
5. LIMIT 3: Restricts the output to the top three states.

- -

6. For the year 2023, extract the total number of crossings per month, categorized by measure type.

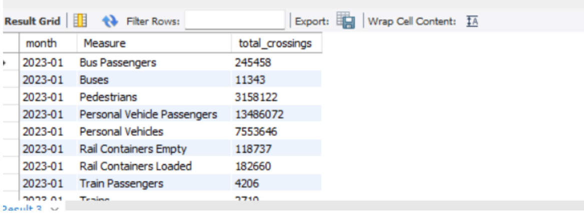
QUERY:

SELECT DATE_FORMAT (Date, '%Y-%m') **AS** month, Measure, **SUM**(Value) **AS** total_crossings

FROM border_crossing_entry_data

```
WHERE YEAR(Date) = 2023

GROUP BY month, Measure

ORDER BY month, Measure;
```

OUTPUT:



month	Measure	total_crossings
2023-01	Bus Passengers	245458
2023-01	Buses	11343
2023-01	Pedestrians	3158122
2023-01	Personal Vehicle Passengers	13486072
2023-01	Personal Vehicles	7553646
2023-01	Rail Containers Empty	118737
2023-01	Rail Containers Loaded	182660
2023-01	Train Passengers	4206
2023-01	Trains	2710

DESCRIPTION:

1. DATE_FORMAT(Date, '%Y-%m') AS month: Formats the Date column to show the year and month and labels it as month.
2. Measure: Includes the type of measure (e.g., Trucks, Pedestrians).
3. SUM(Value) AS total_crossings: Calculates the total number of crossings for each measure and month, labeling it as total_crossings.
4. WHERE YEAR(Date) = 2023: Filters the data to include only entries from the year 2023.
5. GROUP BY month, Measure: Groups the results by formatted month and measure type.
6. ORDER BY month, Measure: Sorts the results first by month and then by measure type

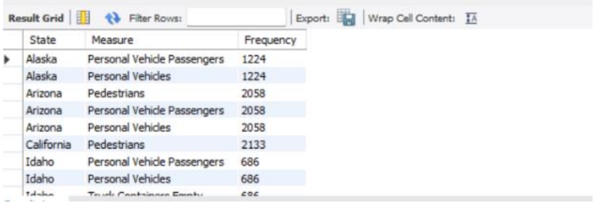
- -

7. Find which measure type is the most frequently recorded for each state.

QUERY:

```
SELECT State, Measure, COUNT(*) AS Frequency

FROM border_crossing_entry_data

GROUP BY State, Measure

HAVING COUNT(*) = (

            SELECT MAX(sub_count)

            FROM (

                SELECT State, Measure, COUNT(*) AS sub_count

                FROM border_crossing_entry_data

                GROUP BY State, Measure

            ) AS subquery

            WHERE subquery.state = border_crossing_entry_data.state

        )

ORDER BY State;
```

OUTPUT:



State	Measure	Frequency
Alaska	Personal Vehicle Passengers	1224
Alaska	Personal Vehicles	1224
Arizona	Pedestrians	2058
Arizona	Personal Vehicle Passengers	2058
Arizona	Personal Vehicles	2058
California	Pedestrians	2133
Idaho	Personal Vehicle Passengers	686
Idaho	Personal Vehicles	686
Idaho	Truck Containers Empty	686

DESCRIPTION:

1. SELECT State, Measure, COUNT(*) AS Frequency: Selects the state and measure type while counting the number of entries for each combination, labeling it as Frequency.
2. GROUP BY State, Measure: Groups the results by state and measure type.
3. HAVING COUNT(*) = (...): Filters the results to only include those combinations where the count matches the maximum count for that specific state.
4. The subquery calculates the maximum count of entries (MAX(sub_count)) for each state and measure combination.
5. ORDER BY State: Sorts the results alphabetically by state.

- -

8. Generate a summary report showing the total number of crossings for each measure type, grouped by border.

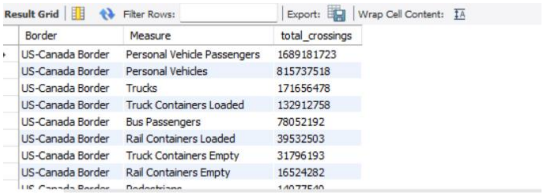
QUERY:

SELECT Border, Measure, **SUM**(Value) **AS** total_crossings **FROM** border_crossing_entry_data

GROUP BY Border, Measure

ORDER BY Border, total_crossings **DESC**;

OUTPUT:



Border	Measure	total_crossings
US-Canada Border	Personal Vehicle Passengers	1689181723
US-Canada Border	Personal Vehicles	815737518
US-Canada Border	Trucks	171656478
US-Canada Border	Truck Containers Loaded	132912758
US-Canada Border	Bus Passengers	78052192
US-Canada Border	Rail Containers Loaded	39532503
US-Canada Border	Truck Containers Empty	31796193
US-Canada Border	Rail Containers Empty	16524282
US-Canada Border	Pedestrians	14077540

DESCRIPTION:

1. SELECT Border, Measure, SUM(Value) AS total_crossings: Selects the border and measure type while summing the number of crossings, labeling the result as total_crossings.
2. GROUP BY Border, Measure: Groups the results by border and measure type.
3. ORDER BY Border, total_crossings DESC: Sorts the results first by border (alphabetically) and then by the total number of crossings in descending order.

- -

Week 2: Intermediate Queries with Aggregations

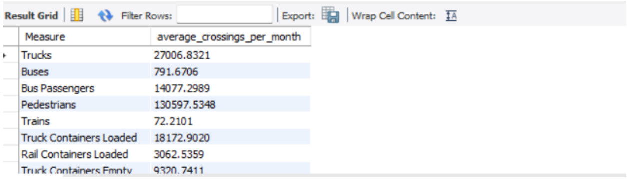
1. For Texas, calculate the average number of crossings per month for each measure type.

QUERY:

SELECT Measure, **AVG**(Value) **AS** average_crossings_per_month **FROM** border_crossing_entry_data

```
WHERE State = 'Texas'

GROUP BY Measure;
```

OUTPUT:



Measure	average_crossings_per_month
Trucks	27006.8321
Buses	791.6706
Bus Passengers	14077.2989
Pedestrians	130597.5348
Trains	72.2101
Truck Containers Loaded	18172.9020
Rail Containers Loaded	3062.5359
Truck Containers Empty	9320.7411

DESCRIPTION:

1. SELECT Measure, AVG(Value) AS average_crossings_per_month: Selects the measure type and computes the average crossings, labeling it as average_crossings_per_month.
2. WHERE State = 'Texas': Filters the data to include only entries related to Texas.
3. GROUP BY Measure: Groups the results by measure type.

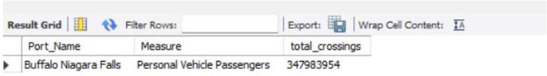
- -

2. Find the port on the U.S.-Canada border with the highest number of crossings. Include the measure type and total crossings.

QUERY:

```
SELECT Port_Name, Measure, SUM(Value) AS total_crossings FROM border_crossing_entry_data

WHERE Border = 'US-Canada border'

GROUP BY Port_Name, Measure

ORDER BY total_crossings DESC

LIMIT 1;
```

OUTPUT:



Port_Name	Measure	total_crossings
Buffalo Niagara Falls	Personal Vehicle Passengers	347983954

DESCRIPTION:

1. SELECT Port_Name, Measure, SUM(Value) AS total_crossings: Selects the port name and measure type while calculating the total number of crossings, labeling it as total_crossings.
2. WHERE Border = 'US-Canada border': Filters the data to include only entries for the U.S.-Canada border.
3. GROUP BY Port_Name, Measure: Groups the results by port name and measure type.
4. ORDER BY total_crossings DESC: Sorts the results in descending order based on the total crossings.
5. LIMIT 1: Restricts the output to the port with the highest total crossings.

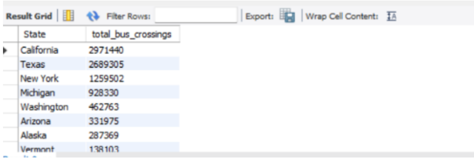
- -

3. Calculate the total number of crossings for the "Buses" measure type in each state, ordered by total crossings in descending order.

QUERY:

```
SELECT State, SUM(Value) AS total_bus_crossings FROM border_crossing_entry_data

WHERE Measure = 'Buses'

GROUP BY State

ORDER BY total_bus_crossings DESC;
```

OUTPUT:



State	total_bus_crossings
California	2971440
Texas	2689305
New York	1259502
Michigan	928330
Washington	462763
Arizona	331975
Alaska	287369
Vermont	138103

DESCRIPTION:

1. WHERE Measure = 'Buses': Filters the data to include only bus crossings.
2. SUM(Value): Calculates the total bus crossings for each state.
3. GROUP BY State: Groups the data by state.
4. ORDER BY total_bus_crossings DESC: Sorts the results with the highest totals first.

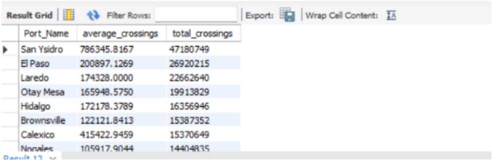
- -

4. For the U.S.-Mexico border, calculate the average and total number of crossings for each port in the year 2022.

QUERY:

```
SELECT Port_Name, AVG(Value) AS average_crossings, SUM(Value) AS total_crossings

FROM border_crossing_entry_data

WHERE Border = 'US-Mexico border' AND YEAR(Date) = 2022

GROUP BY Port_Name

ORDER BY total_crossings DESC;
```

OUTPUT:



Port_Name	average_crossings	total_crossings
San Ysidro	786345.8167	47180749
El Paso	200897.1269	26920215
Laredo	174328.0000	22662640
Otay Mesa	165948.5750	19913829
Hidalgo	172178.3789	16356946
Brownsville	122121.8413	15387352
Calexico	415422.9459	15370649
Nogales	105917.9044	14404835

DESCRIPTION:

1. WHERE Border = 'US-Mexico border' AND YEAR(Date) = 2022: Filters data for crossings on the U.S.-Mexico border in 2022.
2. AVG(Value): Computes the average number of crossings per port.
3. SUM(Value): Calculates the total number of crossings per port.

4. GROUP BY Port_Name: Groups data by each port.
5. ORDER BY total_crossings DESC: Sorts ports by total crossings in descending order.

---

5. List all ports that reported pedestrians as a crossing measure in 2023 and show their total number of pedestrian crossings.

QUERY:

SELECT Port_Name, **SUM**(Value) **AS** total_pedestrian_crossings

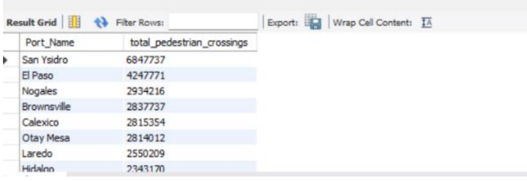
FROM border_crossing_entry_data

WHERE Measure = 'Pedestrians' **AND** Year(Date) = 2023

GROUP BY Port_Name

ORDER BY total_pedestrian_crossings **DESC**;

OUTPUT:



Port_Name	total_pedestrian_crossings
San Ysidro	6847737
El Paso	4247771
Nogales	2934216
Brownsville	2837737
Calexico	2815354
Otay Mesa	2814012
Laredo	2550209
Hidalgo	2343170

DESCRIPTION:

1. WHERE Measure = 'Pedestrians' AND Year(Date) = 2023: Filters the data for pedestrian crossings in 2023.
2. SUM(Value): Calculates the total pedestrian crossings for each port.
3. GROUP BY Port_Name: Groups the data by port.
4. ORDER BY total_pedestrian_crossings DESC: Sorts the ports by total crossings, with the highest numbers first.

---

6. Extract the total number of crossings in each border for every year available in the dataset.

QUERY:

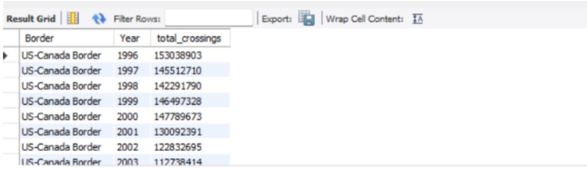
SELECT Border, Year(Date) **AS** Year, **SUM**(Value) **AS** total_crossings

FROM border_crossing_entry_data

GROUP BY Border, Year(Date)

ORDER BY Border, Year;

OUTPUT:



Border	Year	total_crossings
US-Canada Border	1996	153038903
US-Canada Border	1997	145512710
US-Canada Border	1998	142291790
US-Canada Border	1999	146497328
US-Canada Border	2000	147789673
US-Canada Border	2001	130092391
US-Canada Border	2002	122832695
US-Canada Border	2003	112738414

DESCRIPTION:

1. SUM(Value): Computes total crossings per border per year.
2. GROUP BY: Groups data by border and year.
3. ORDER BY: Sorts the output by border and then by year.

- -

7. Identify the month in 2023 with the highest number of truck crossings. List the month and total truck crossings.

QUERY:

SELECT DATE_FORMAT(Date, '%Y-%m') **AS** Month**, SUM**(Value) **AS** total_truck_crossings

FROM border_crossing_entry_data

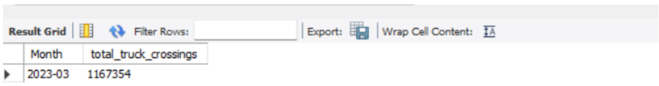
WHERE YEAR(Date) = 2023 **AND** Measure = 'Trucks'

GROUP BY DATE_FORMAT(Date, '%Y-%m')

ORDER BY total_truck_crossings **DESC**

LIMIT 1;

OUTPUT:



Month	total_truck_crossings
2023-03	1167354

DESCRIPTION:

1. DATE_FORMAT(date, '%Y-%m') formats the date to extract the year and month in YYYY-MM format.
2. The WHERE clause filters the data to only include records from the year 2023 and where the measure is for trucks.
3. SUM(value) calculates the total number of truck crossings for each month.
4. The results are ordered by the total truck crossings in descending order to find the month with the highest value.
5. LIMIT 1 ensures that only the month with the highest number of truck crossings is returned.

- -

8. List the top 5 ports with the highest crossing activity (all measure types) in 2021, showing the measure type and total crossings for each port.

QUERY:

SELECT Port_Name, Measure, **SUM**(Value) **AS** total_crossings

FROM border_crossing_entry_data

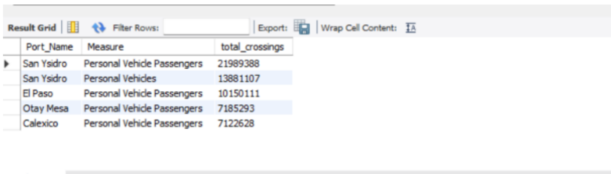
WHERE YEAR(Date) = 2021

GROUP BY Port_Name, Measure

ORDER BY total_crossings DESC

LIMIT 5;

OUTPUT:



Port_Name	Measure	total_crossings
San Ysidro	Personal Vehicle Passengers	21989388
San Ysidro	Personal Vehicles	13881107
El Paso	Personal Vehicle Passengers	10150111
Otay Mesa	Personal Vehicle Passengers	7185293
Calexico	Personal Vehicle Passengers	7122628

DESCRIPTION:

1. SUM(value) AS total_crossings: Calculates the total number of crossings for each port and measure type.
2. WHERE YEAR(date) = 2021: Filters the data to include only entries from the year 2021.
3. GROUP BY port_name, measure: Groups the results by each port and measure type.
4. ORDER BY total_crossings DESC: Sorts the results in descending order based on the total number of crossings.
5. LIMIT 5: Returns the top 5 ports with the highest crossing activity.

- -

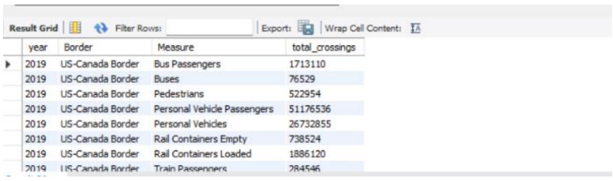
WEEK 3: Advanced Queries and Research-Like Tasks

1. Calculate the total number of crossings for each year from 2019 to 2023, grouped by border and measure type.

QUERY:

```
SELECT YEAR(Date) AS year, Border, Measure, SUM(Value) AS total_crossings

FROM border_crossing_entry_data

WHERE YEAR(Date) BETWEEN 2019 AND 2023

GROUP BY year,Border, Measure

ORDER BY year, Border, Measure
```

OUTPUT:



year	Border	Measure	total_crossings
2019	US-Canada Border	Bus Passengers	1713110
2019	US-Canada Border	Buses	76529
2019	US-Canada Border	Pedestrians	522954
2019	US-Canada Border	Personal Vehicle Passengers	51176536
2019	US-Canada Border	Personal Vehicles	26732855
2019	US-Canada Border	Rail Containers Empty	738524
2019	US-Canada Border	Rail Containers Loaded	1886120
2019	US-Canada Border	Train Passengers	284546

DESCRIPTION:

1. YEAR(date) AS year: Extracts the year from the date column.
2. SUM(value) AS total_crossings: Calculates the total number of crossings for each combination of year, border, and measure type.
3. WHERE YEAR(date) BETWEEN 2019 AND 2023: Filters the data to include only the years from 2019 to 2023.
4. GROUP BY year, border, measure: Groups the results by year, border, and measure type.

5. ORDER BY year, border, measure: Sorts the results by year, then by border, and finally by measure type.

- -

2. For Texas, find the most frequently recorded measure types for 2023. Rank the measure types by the number of entries without using RANK().

Since you can't use the RANK() function, we'll use COUNT() and ORDER BY to achieve the ranking.

QUERY:

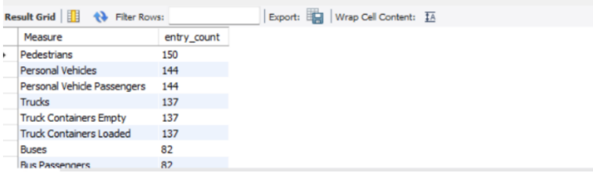
SELECT Measure, **COUNT**(*) **AS** entry_count **FROM** border_crossing_entry_data

WHERE State = 'Texas' **AND** YEAR(Date) = 2023

GROUP BY Measure

ORDER BY entry_count **DESC**;

OUTPUT:



Measure	entry_count
Pedestrians	150
Personal Vehicles	144
Personal Vehicle Passengers	144
Trucks	137
Truck Containers Empty	137
Truck Containers Loaded	137
Buses	82
Bus Passengers	82

DESCRIPTION:

- COUNT(*) calculates the number of entries for each measure.
- GROUP BY measure groups the results by each measure type.
- ORD+ER BY entry_count DESC sorts the measure types in descending order based on the number of entries, giving you the most frequently recorded measure types at the top.

- -

3. Compare the total number of container crossings over the last 3 years for each border.

QUERY:

SELECT Border, YEAR(Date) **AS** Year, **SUM**(Value) **AS** total_container_crossings

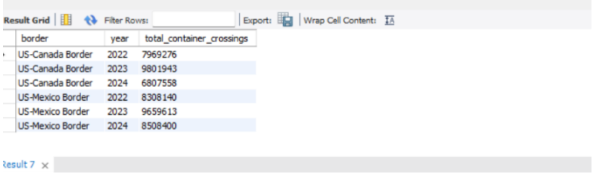
FROM border_crossing_entry_data

WHERE Measure **LIKE** '% Containers %' **AND** YEAR(Date) **BETWEEN** YEAR(CURDATE()) - 2 **AND** YEAR(CURDATE()) - 0

GROUP BY Border, YEAR(Date)

ORDER BY Border, Year;

OUTPUT:



border	year	total_container_crossings
US-Canada Border	2022	7969276
US-Canada Border	2023	9801943
US-Canada Border	2024	6807558
US-Mexico Border	2022	8308140
US-Mexico Border	2023	9659613
US-Mexico Border	2024	8508400

DESCRIPTION:

1. YEAR(date) extracts the year from the date.
2. The WHERE clause filters the data to include only records where the measure is 'Containers' and limits the years to the last three years.
3. SUM(value) calculates the total number of container crossings for each year and border.
4. The results are grouped by border and year to display totals for each combination.
5. The results are ordered by border and then by year to make the comparison clear.

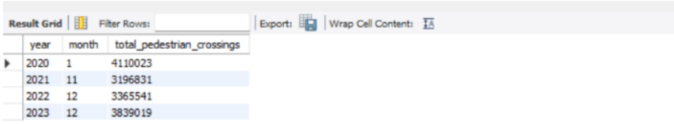
- -

4. Identify the busiest month of each year (2020-2023) in terms of pedestrian crossings. Show the year, month, and total pedestrian crossings

QUERY:

```
WITH PedestriansMonthlyTotal AS (
        SELECT YEAR(Date) AS year, MONTH(Date) AS month, SUM(Value) AS total_pedestrian_crossings
        FROM border_crossing_entry_data
        WHERE Measure = 'Pedestrians' AND YEAR(Date) BETWEEN 2020 AND 2023
        GROUP BY YEAR(Date), MONTH(Date))


SELECT year, month, total_pedestrian_crossings FROM PedestriansMonthlyTotal
WHERE total_pedestrian_crossings = (
        SELECT MAX(total_pedestrian_crossings)
        FROM  PedestriansMonthlyTotal AS subquery
        WHERE subquery.year = PedestriansMonthlyTotal.year)
ORDER BY year, month;
```

OUTPUT:



year	month	total_pedestrian_crossings
2020	1	4110023
2021	11	3196831
2022	12	3365541
2023	12	3839019

DESCRIPTION:

1. **Common Table Expression (CTE)**: We use a CTE named PedestrianMonthlyTotals to calculate the total pedestrian crossings for each month of each year.

2. SUM(value) calculates the total pedestrian crossings for each month.

3. **Filtering**: The WHERE measure = 'Pedestrians' clause ensures that only pedestrian crossing data is included.

4. **Finding the Busiest Month**: The main query selects the busiest month for each year using a subquery to get the maximum monthly crossings for each year.

5. The results are ordered by year and month.

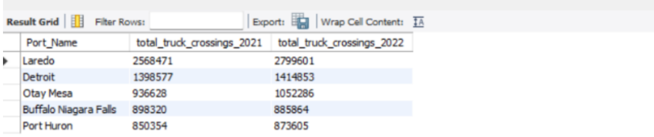
- -

5. Compare the total number of truck crossings in 2021 and 2022 at the top 5 busiest ports for trucks. Display both years' totals side by side.

QUERY:

```
WITH TruckCrossings AS (

        SELECT Port_Name, YEAR(Date) AS year, SUM(Value) AS total_truck_crossings

        FROM border_crossing_entry_data

        WHERE Measure = 'Trucks' AND YEAR(Date) IN (2021, 2022)

        GROUP BY Port_Name, YEAR(Date)),


Top5Ports AS (

        SELECT Port_Name, SUM(total_truck_crossings) AS total_crossings

        FROM TruckCrossings

        GROUP BY Port_Name

        ORDER BY total_crossings DESC

        LIMIT 5)


SELECT t.Port_Name,

  SUM(CASE WHEN t.year = 2021 THEN t.total_truck_crossings ELSE 0 END) AS total_truck_crossings_2021,

  SUM(CASE WHEN t.year = 2022 THEN t.total_truck_crossings ELSE 0 END) AS total_truck_crossings_2022

FROM TruckCrossings t


JOIN Top5Ports tp ON t.Port_Name=tp.Port_Name

GROUP BY t.Port_Name

ORDER BY total_truck_crossings_2021 + total_truck_crossings_2022 DESC;
```

OUTPUT:



Port_Name	total_truck_crossings_2021	total_truck_crossings_2022
Laredo	2568471	2799601
Detroit	1398577	1414853
Otay Mesa	936628	1052286
Buffalo Niagara Falls	898320	885864
Port Huron	850354	873605

DESCRIPTION:

1. TruckCrossings CTE: This common table expression calculates the total number of truck crossings for each port for the years 2021 and 2022.

2. Top5Ports CTE: This CTE identifies the top 5 ports with the highest total truck crossings across both years.
3. The main query joins the TruckCrossings data with the Top5Ports to display only the data for the busiest ports.
4. The CASE statements in the main query help to display the total truck crossings for 2021 and 2022 side by side.
5. The results are ordered by the combined total truck crossings for both years, showing the busiest ports at the top.

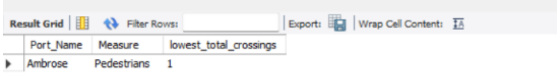
---

6. Find the port with the lowest total crossings on the U.S.-Canada border for any measure type in 2023.

QUERY:

```
SELECT Port_Name, Measure, SUM(Value) AS lowest_total_crossings

FROM border_crossing_entry_data

WHERE Border = 'US-Canada Border' AND YEAR(Date) = 2023

GROUP BY Port_Name, Measure

ORDER BY lowest_total_crossings ASC

LIMIT 1;
```

OUTPUT:



Port_Name	Measure	lowest_total_crossings
Ambrose	Pedestrians	1

DESCRIPTION:

- SUM(value) calculates the total number of crossings for each port.
- WHERE border = 'U.S.-Canada' AND YEAR(date) = 2023 filters the data to include only crossings on the U.S.-Canada border in 2023.
- GROUP BY port_name groups the results by port.
- ORDER BY total_crossings ASC sorts the ports in ascending order based on the number of crossings.
- LIMIT 1 ensures that only the port with the lowest total crossings is returned.

---

7. List the monthly total number of crossings for buses across all states in 2022, sorted in ascending order.

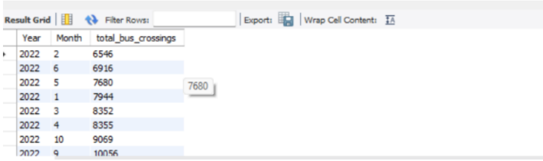
QUERY:

```
SELECT YEAR(Date) AS Year, MONTH(Date) AS Month, SUM(Value) AS total_bus_crossings

FROM border_crossing_entry_data

WHERE Measure = 'Buses' AND YEAR(Date) = 2022
```

GROUP BY YEAR(Date), MONTH(Date)

ORDER BY total_bus_crossings **ASC**;

OUTPUT:



Year	Month	total_bus_crossings
2022	2	6546
2022	6	6916
2022	5	7680
2022	1	7944
2022	3	8352
2022	4	8355
2022	10	9069
2022	9	10056

DESCRIPTION:

- SUM(value) calculates the total number of bus crossings for each month.
- WHERE measure = 'Buses' AND YEAR(date) = 2022 filters the data to include only bus crossings in the year 2022.
- GROUP BY YEAR(date), MONTH(date) groups the results by month.
- ORDER BY total_bus_crossings ASC sorts the monthly totals in ascending order, showing the months with the fewest bus crossings first.

- -

8. Display the sum and average number of crossings for each state, grouped by measure type and year. Only show entries where the average crossings exceed 500.

QUERY:

SELECT State, Measure, YEAR(Date) **AS** Year, SUM(Value) **AS** total_crossings, **AVG**(Value) **AS** average_crossings

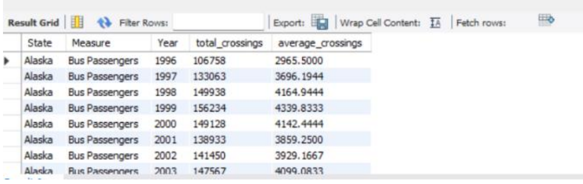
FROM border_crossing_entry_data

GROUP BY State, Measure, YEAR(Date)

HAVING AVG(Value) > 500

ORDER BY State, Measure, Year;

OUTPUT:



State	Measure	Year	total_crossings	average_crossings
Alaska	Bus Passengers	1996	106758	2965.5000
Alaska	Bus Passengers	1997	133063	3696.1944
Alaska	Bus Passengers	1998	149938	4164.9444
Alaska	Bus Passengers	1999	156234	4339.8333
Alaska	Bus Passengers	2000	149128	4142.4444
Alaska	Bus Passengers	2001	138933	3859.2500
Alaska	Bus Passengers	2002	141450	3929.1667
Alaska	Bus Passengers	2003	147567	4099.0833

DESCRIPTION:

- SUM(value) calculates the total number of crossings for each state, measure type, and year.
- AVG(value) calculates the average number of crossings for each state, measure type, and year.
- GROUP BY state, measure, YEAR(date) groups the results by state, measure, and year.
- HAVING AVG(value) > 500 filters the groups to include only those where the average number of crossings exceeds 500.
- ORDER BY state, measure, year sorts the results for easier readability.

Nagalaxmi Eepuri

- -

Key Findings and Insights from the dataset, such as

The busiest ports and states for crossings.

Top busiest ports and states for crossings are San Ysidro -California, El Paso -Texas, Laredo-Texas, Hidalgo -Texas, Calexico-California, Buffalo Nagar falls-New York, Brownsville-Texas, Otay Mesa-California.

Patterns and trends by border (U.S.-Mexico vs U.S.-Canada) and How the number of crossings varies by measure type, year, and month

Measure Type Patterns: Understanding which crossing types are dominant helps align border resources with actual traffic, supporting better service delivery. From the dataset, there are higher in personal vehicle passengers, personal vehicle and trucks that cross US-Canada and US- Mexico borders.

Peak Crossing Months: Identifying high-traffic months for each border aids in seasonal workforce planning and infrastructure utilization. By analysing the dataset, I found the peak crossing months are March, May and December.

Peak Crossing Months: Identifying high-traffic months for each border aids in seasonal workforce planning and infrastructure utilization.