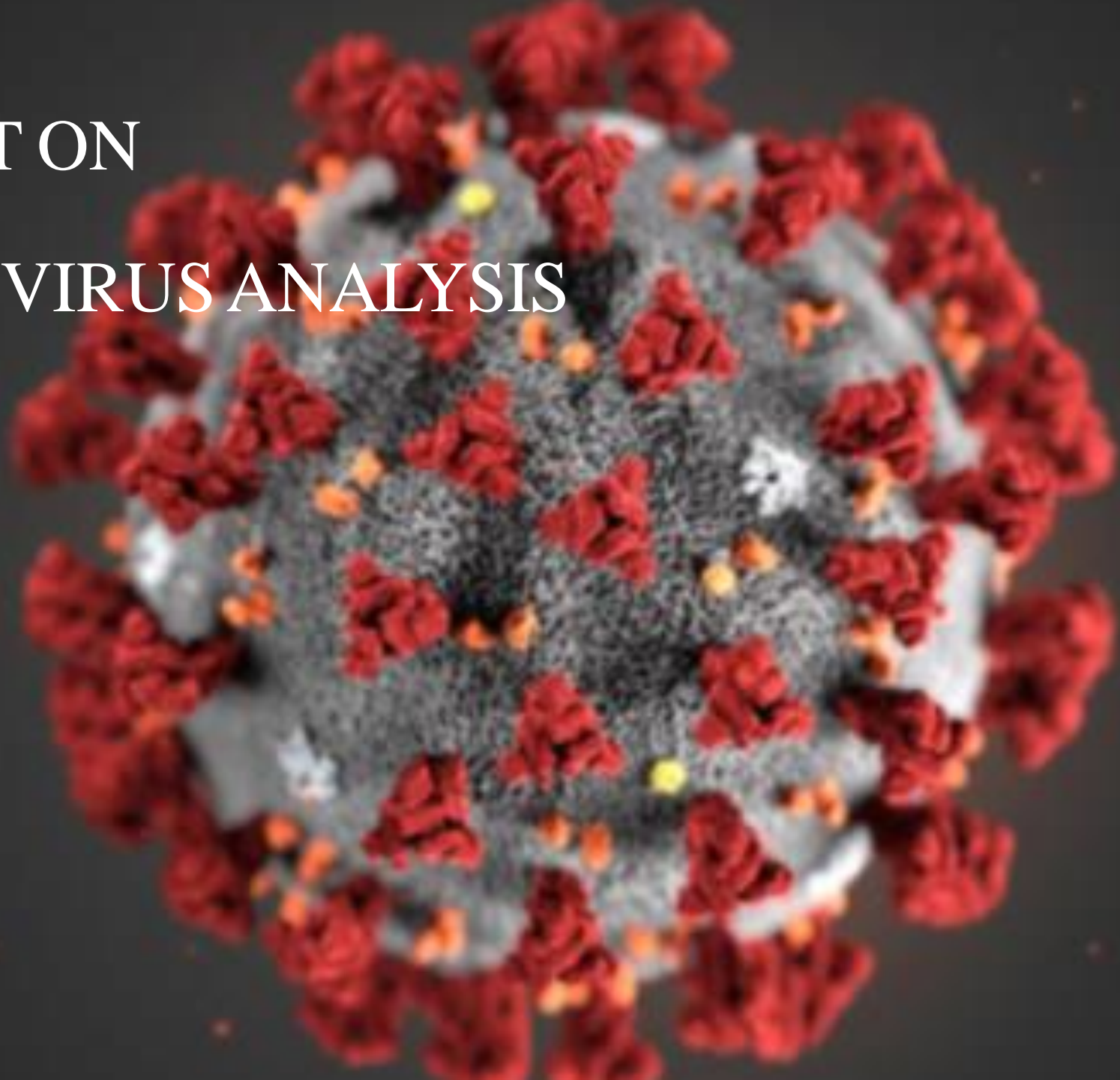


PROJECT ON CORONA VIRUS ANALYSIS



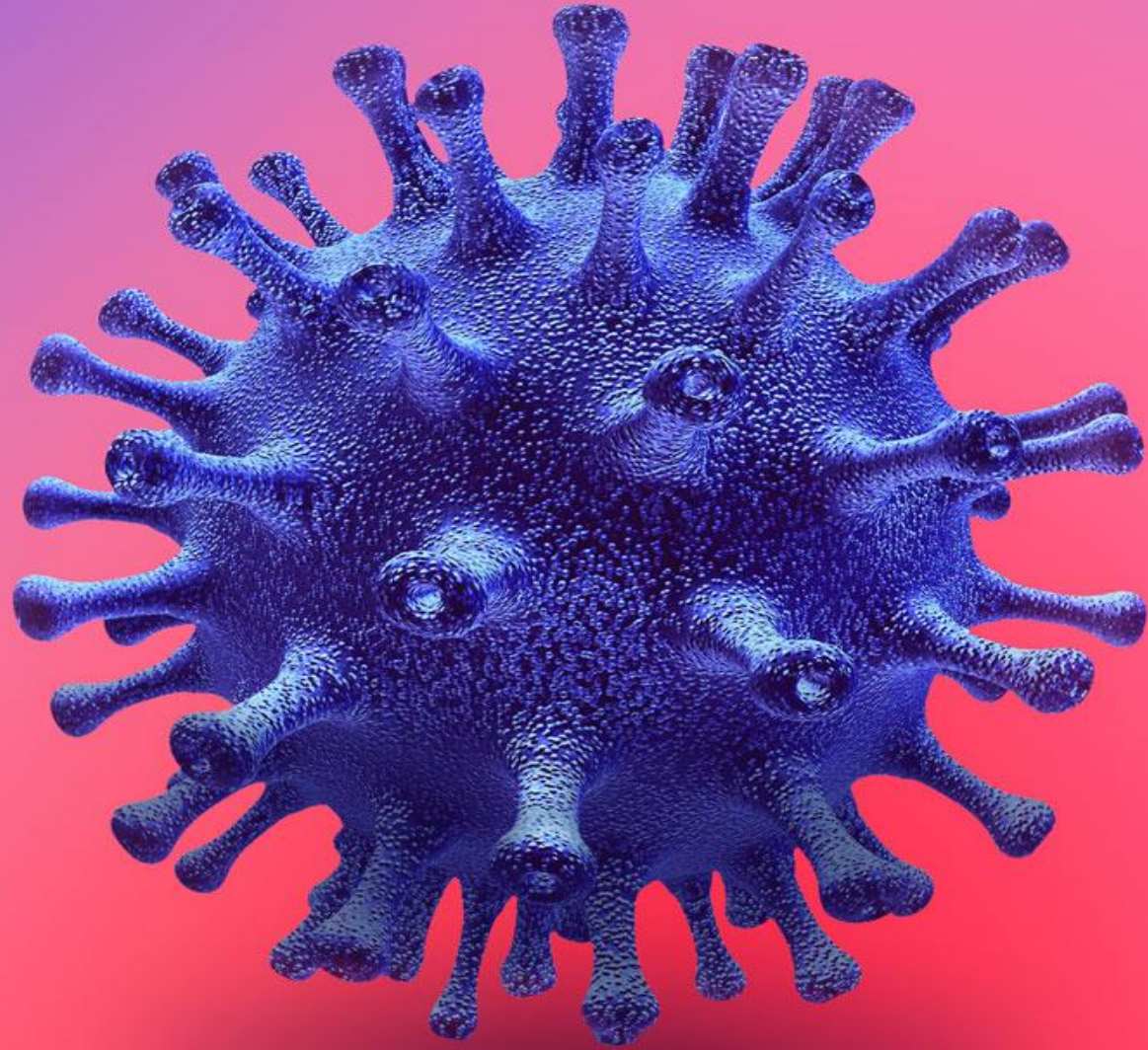
Overview Dataset

Problem statement query with output

Conclusion



CORONA VIRUS ANALYSIS PROJECT

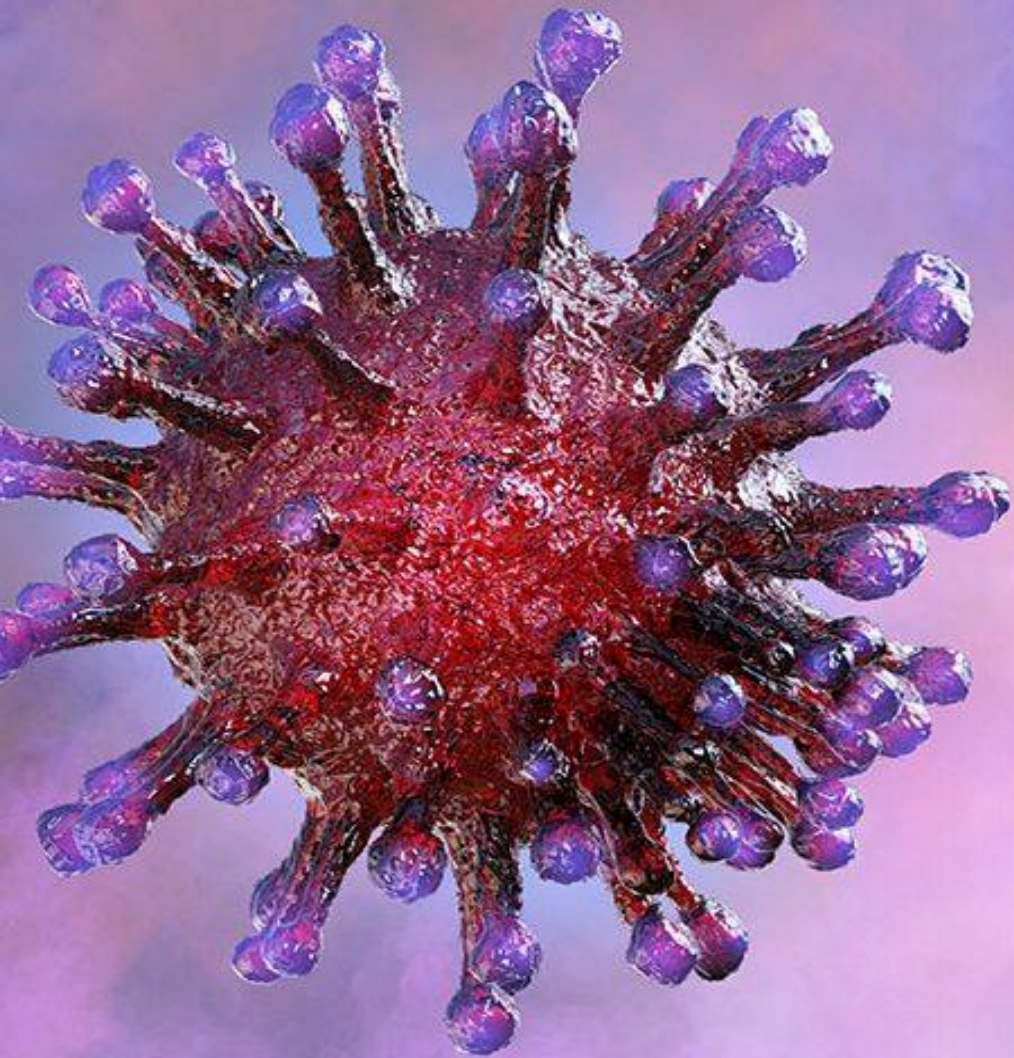


THE CORONA VIRUS PANDEMIC HAS HAD A SIGNIFICANT IMPACT ON PUBLIC HEALTH AND HAS CREATED AN URGENT NEED FOR DATA-DRIVEN INSIGHTS TO UNDERSTAND THE SPREAD OF THE VIRUS. AS A DATA ANALYST, YOU HAVE BEEN TASKED WITH ANALYZING A CORONA VIRUS DATASET TO DERIVE MEANINGFUL INSIGHTS AND PRESENT YOUR FINDINGS.





DESCRIPTION OF EACH
COLUMN IN DATASET:



PROVINCE: GEOGRAPHIC SUBDIVISION WITHIN A

COUNTRY/REGION.

COUNTRY/REGION: GEOGRAPHIC ENTITY WHERE DATA IS

RECORDED.

LATITUDE: NORTH-SOUTH POSITION ON EARTH'S

SURFACE.

LONGITUDE: EAST-WEST POSITION ON EARTH'S SURFACE.

DATE: RECORDED DATE OF CORONA VIRUS DATA.

CONFIRMED: NUMBER OF DIAGNOSED CORONA VIRUS

CASES.

DEATHS: NUMBER OF CORONA VIRUS RELATED DEATHS.

RECOVERED: NUMBER OF RECOVERED CORONA VIRUS

CASES.



DATA-DRIVEN INSIGHTS

Critical Need for Data

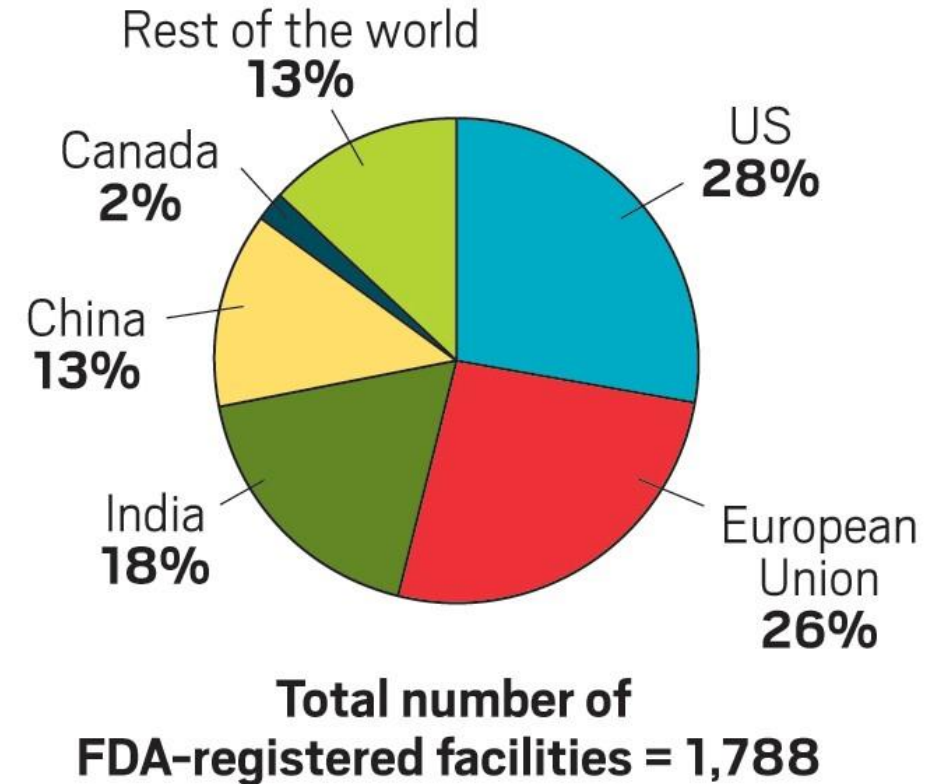
Data-driven insights are crucial to accurately assess the spread and impact of the virus.

Real-time Analysis

Real-time data analysis aids in understanding transmission patterns and predicting trends.

Policy Decision Support

Data insights guide policymakers in implementing effective mitigation strategies.



ANALYZING COVID-19 DATASET

Healthcare Resource Allocation

Insights help in allocating resources to areas with the highest impact and need.

Epidemiological Trends

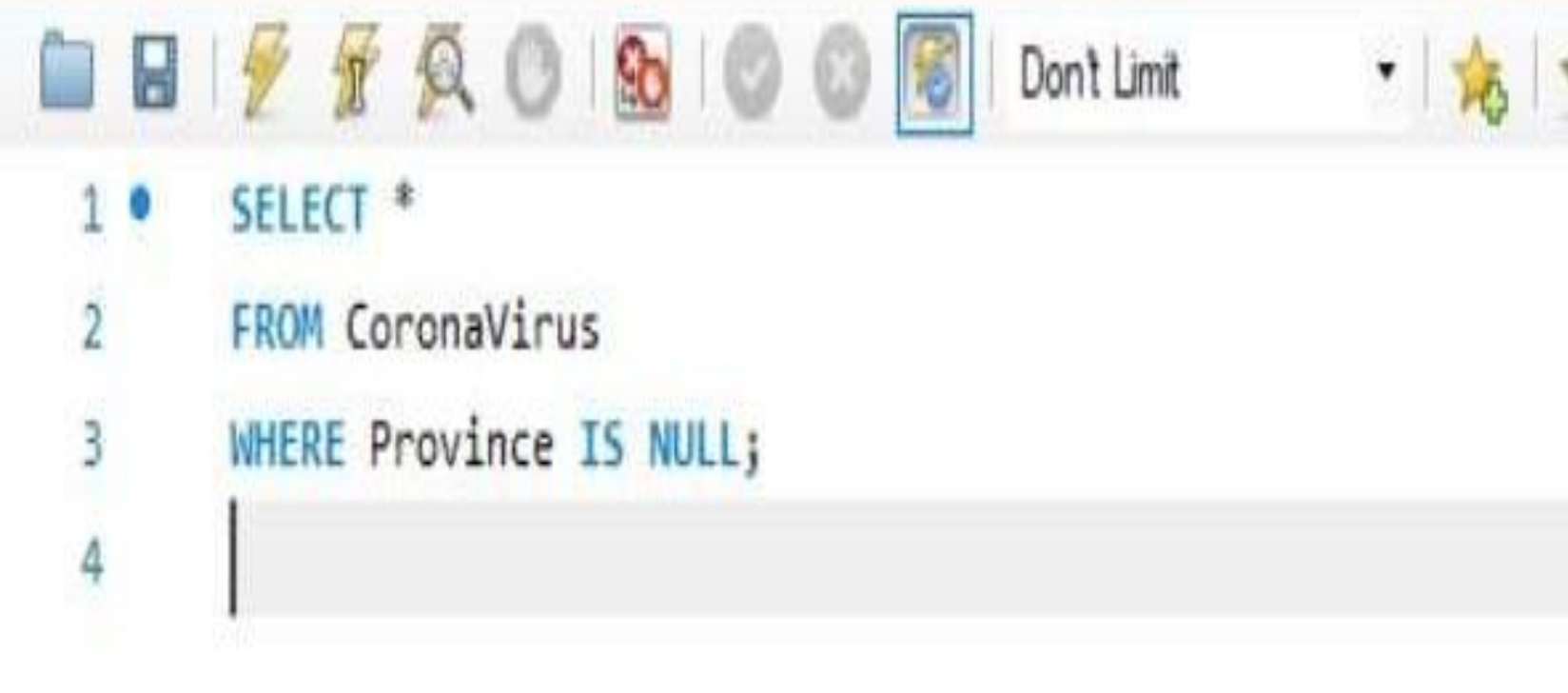
Analyzing the dataset unveils geographical and demographic patterns of the virus.

Risk Factor Identification

Identifying high-risk groups and factors contributing to virus transmission.

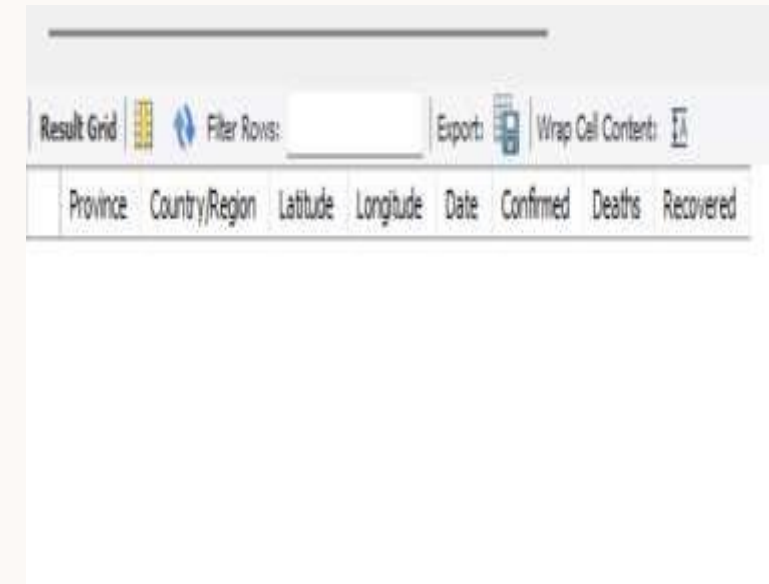
Q1. Write a code to check NULL values

output



The image shows a toolbar with various icons for file operations, execution, and search. Below the toolbar, a SQL query is entered in a text area with line numbers 1 through 4 on the left. The query is: `SELECT *
FROM CoronaVirus
WHERE Province IS NULL;`

```
1 SELECT *  
2 FROM CoronaVirus  
3 WHERE Province IS NULL;  
4
```



The image shows a 'Result Grid' interface. At the top, there are buttons for 'Filter Rows', 'Export', and 'Wrap Cell Contents'. Below these is a table with the following columns: Province, Country/Region, Latitude, Longitude, Date, Confirmed, Deaths, and Recovered. The table is currently empty.

Province	Country/Region	Latitude	Longitude	Date	Confirmed	Deaths	Recovered
----------	----------------	----------	-----------	------	-----------	--------	-----------

Province is not null

```
4 SELECT *
5 FROM CoronaVirus
6 WHERE Province IS NOT NULL;
```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

Fetch rows:




	Province	Country/Region	Latitude	Longitude	Date	Confirmed	Deaths	Recovered
▶	Afghanistan	Afghanistan	33.93911	67.709953	22-01-2020	0	0	0
	Afghanistan	Afghanistan	33.93911	67.709953	23-01-2020	0	0	0
	Afghanistan	Afghanistan	33.93911	67.709953	24-01-2020	0	0	0
	Afghanistan	Afghanistan	33.93911	67.709953	25-01-2020	0	0	0
	Afghanistan	Afghanistan	33.93911	67.709953	26-01-2020	0	0	0
	Afghanistan	Afghanistan	33.93911	67.709953	27-01-2020	0	0	0
	Afghanistan	Afghanistan	33.93911	67.709953	28-01-2020	0	0	0
	Afghanistan	Afghanistan	33.93911	67.709953	29-01-2020	0	0	0
	Afghanistan	Afghanistan	33.93911	67.709953	30-01-2020	0	0	0
	Afghanistan	Afghanistan	33.93911	67.709953	31-01-2020	0	0	0
	Afghanistan	Afghanistan	33.93911	67.709953	01-02-2020	0	0	0
	Afghanistan	Afghanistan	33.93911	67.709953	02-02-2020	0	0	0
	Afghanistan	Afghanistan	33.93911	67.709953	03-02-2020	0	0	0

CoronaVirus 4

CoronaVirus 5 x

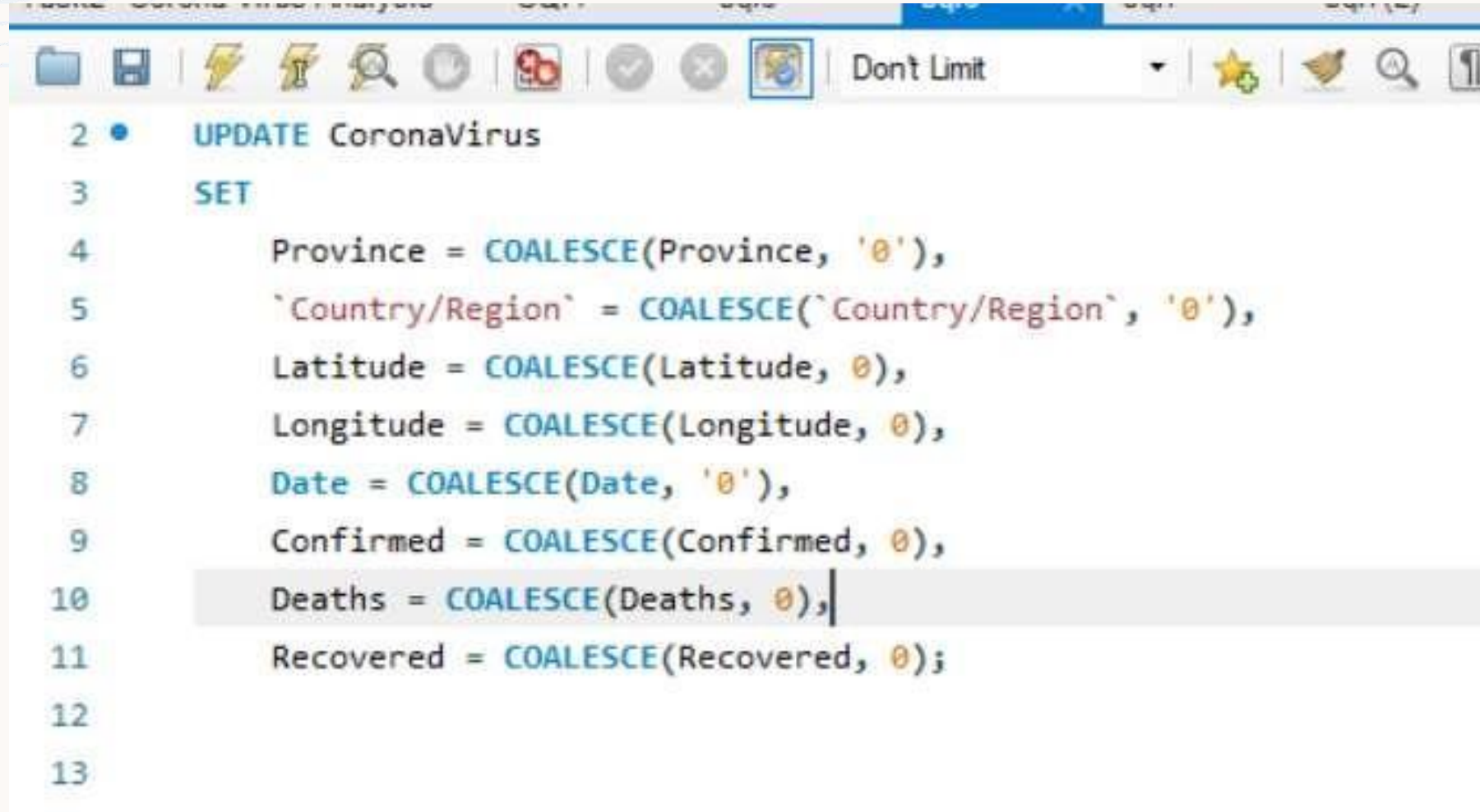
Count all and Province is null

```
9 • SELECT COUNT(*)
10 FROM CoronaVirus
11 WHERE Province IS NULL;
```

Result Grid |  Filter Rows: | Export:  | Wrap Cell Content: 

	COUNT(*)
▶	0

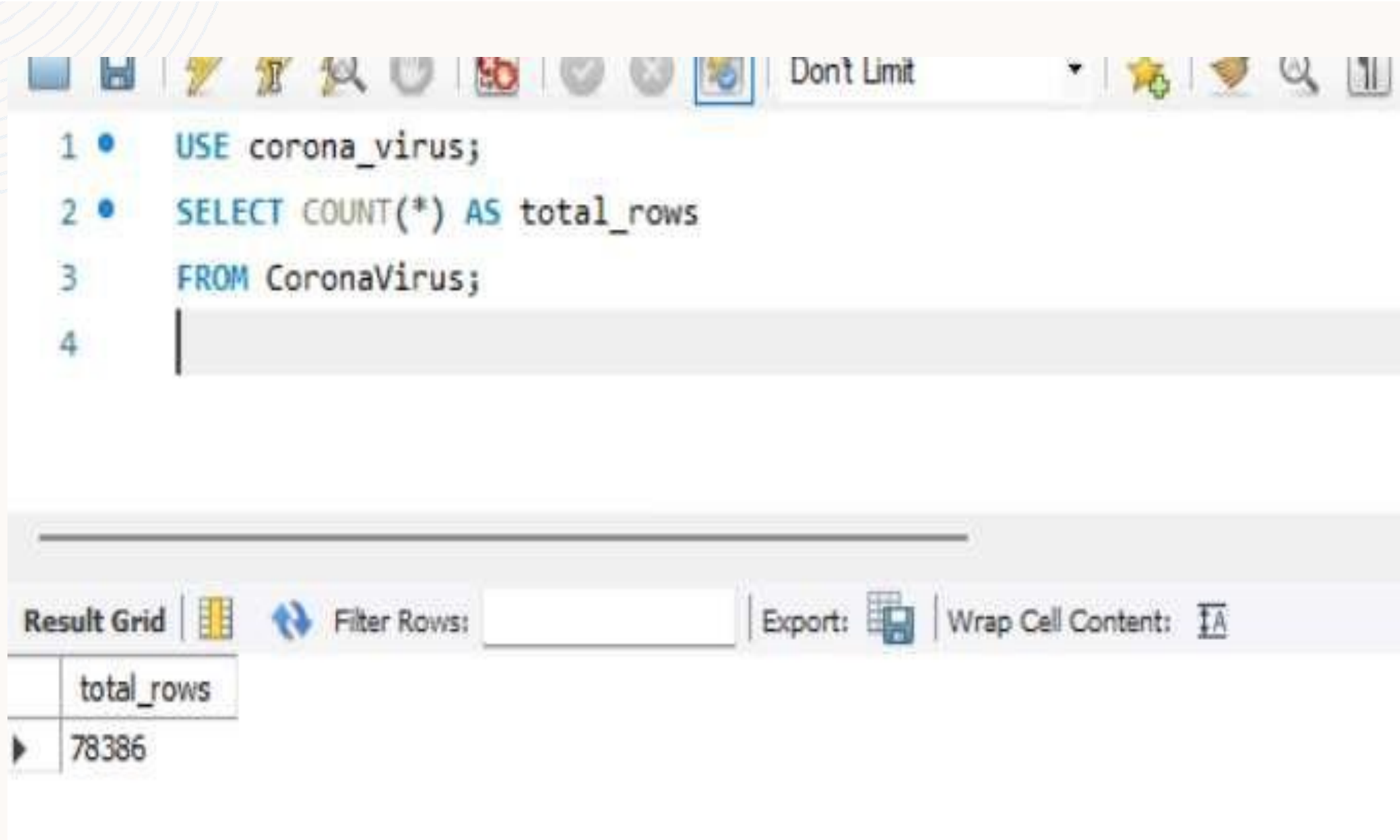
If NULL values are present, update them with zeros for all columns.



```
2 • UPDATE CoronaVirus
3 SET
4     Province = COALESCE(Province, '0'),
5     `Country/Region` = COALESCE(`Country/Region`, '0'),
6     Latitude = COALESCE(Latitude, 0),
7     Longitude = COALESCE(Longitude, 0),
8     Date = COALESCE(Date, '0'),
9     Confirmed = COALESCE(Confirmed, 0),
10    Deaths = COALESCE(Deaths, 0),
11    Recovered = COALESCE(Recovered, 0);
12
13
```

0 Row
Effecte
d

check total number of rows



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Don't Limit' dropdown. The SQL editor contains the following code:

```
1 • USE corona_virus;  
2 • SELECT COUNT(*) AS total_rows  
3 FROM CoronaVirus;  
4 |
```

Below the editor is a horizontal scrollbar. The bottom toolbar features a 'Result Grid' icon, a 'Filter Rows' input field, an 'Export' icon, and a 'Wrap Cell Content' icon. The 'Result Grid' is displayed as a table with one row and one column.

	total_rows
▶	78386

Check what is start_date and end_date

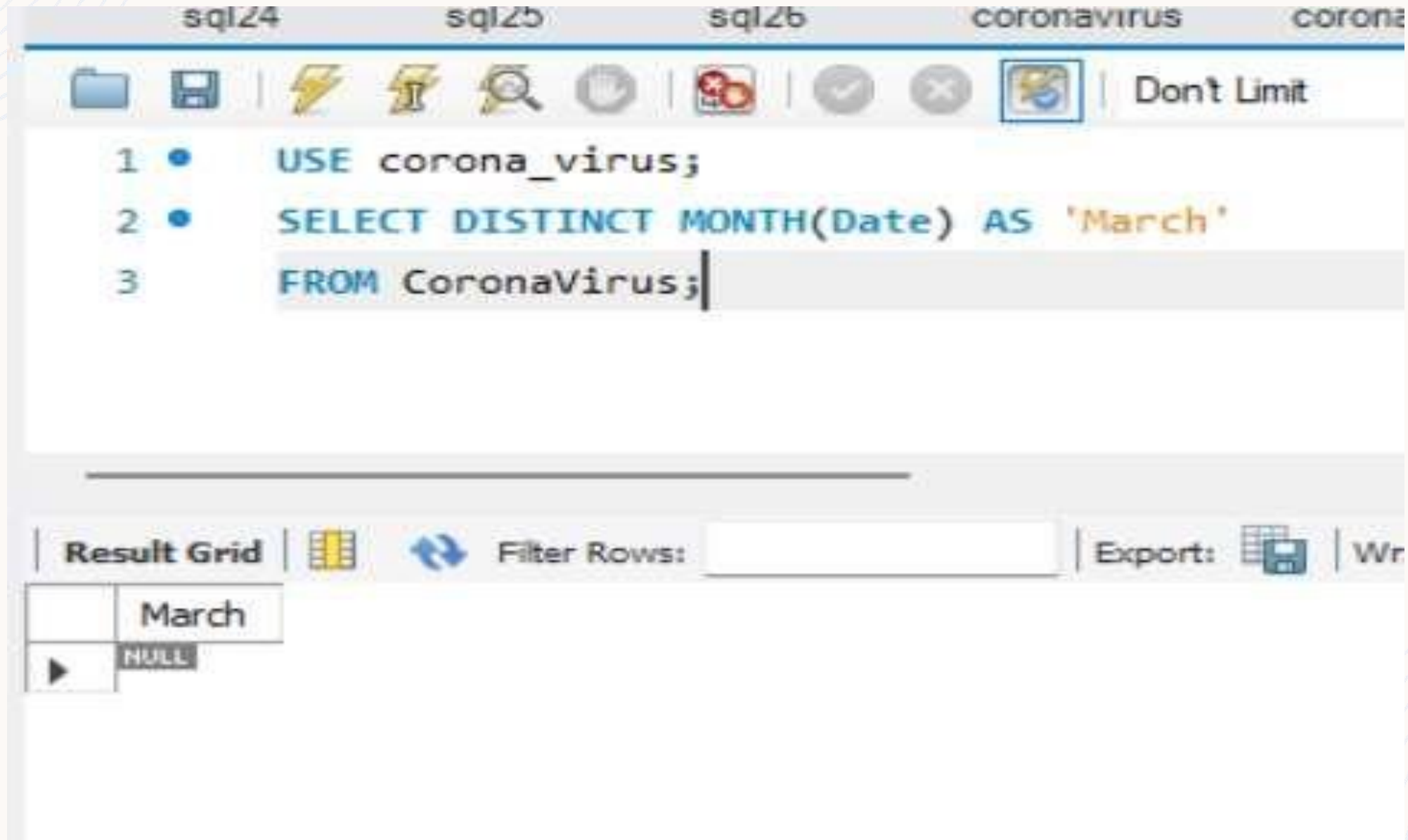
```
4 DESCRIBE CoronaVirus;
```

```
5
```

Result Grid					
Filter Rows:					
	Field	Type	Null	Key	Default
▶	Province	text	YES		NULL
	Country/Region	text	YES		NULL
	Latitude	double	YES		NULL
	Longitude	double	YES		NULL
	Date	text	YES		NULL
	Confirmed	int	YES		NULL
	Deaths	int	YES		NULL
	Recovered	int	YES		NULL

Number of month present in dataset

5



The screenshot shows a SQL IDE interface with multiple tabs at the top: 'sql24', 'sql25', 'sql26', 'coronavirus', and 'corona'. The 'sql26' tab is active, displaying a SQL query with three lines:

```
1 • USE corona_virus;  
2 • SELECT DISTINCT MONTH(Date) AS 'March'  
3 FROM CoronaVirus;
```

Below the query editor, there is a toolbar with icons for file operations, execution, and a 'Don't Limit' button. The 'Result Grid' tab is selected, showing a table with two rows:

	March
▶	NULL

The interface also includes a 'Filter Rows' input field and an 'Export' button.

Find monthly average for confirmed, deaths, recovered

```
13 • SELECT
14     MONTH(date) AS month,
15     YEAR(date) AS year,
16     AVG(confirmed) AS avg_confirmed,
17     AVG(deaths) AS avg_deaths,
18     AVG(recovered) AS avg_recovered
19 FROM
20     CoronaVirus
21 GROUP BY
22     YEAR(date),
23     MONTH(date);
24
```

Result Grid		Filter Rows:	Export:		
	month	year	avg_confirmed	avg_deaths	avg_recovered
▶	2020-01	2020	2156.8283	46.5376	1442.7264

Find most frequent value for confirmed, deaths, recovered each month

```
1 SELECT
2     YEAR(date) AS year,
3     MONTH(date) AS month,
4     (SELECT confirmed FROM CoronaVirus WHERE date = '2020-04-02' GROUP BY confirmed ORDER BY COUNT(*) DESC LIMIT 1) AS most_frequent_confirmed
5     (SELECT deaths FROM CoronaVirus WHERE date = '2020-04-02' GROUP BY deaths ORDER BY COUNT(*) DESC LIMIT 1) AS most_frequent_deaths,
6     (SELECT recovered FROM CoronaVirus WHERE date = '2020-04-02' GROUP BY recovered ORDER BY COUNT(*) DESC LIMIT 1) AS most_frequent_recovered
7 FROM
8     CoronaVirus
9 WHERE
10     date = '2020-04-02';
11
```

Find minimum values for confirmed, deaths, recovered per year

```
1 SELECT
2     YEAR(2024-04-02) AS year,
3     MIN(confirmed) AS min_confirmed,
4     MIN(deaths) AS min_deaths,
5     MIN(recovered) AS min_recovered
6 FROM
7     CoronaVirus
8 GROUP BY
9     YEAR(2024-04-02);
```

0
confirmed,
deaths,
recovered

Find maximum values of confirmed, deaths, recovered per year

```
1 • SELECT
2     YEAR(2024-04-12) AS year,
3     MAX(confirmed) AS max_confirmed,
4     MAX(deaths) AS max_deaths,
5     MAX(recovered) AS max_recovered
6 FROM
7     CoronaVirus
8 GROUP BY
9     YEAR(2024-04-12);
10
```

year	max_confirmed
NULL	823225

max_deaths	max_recovered
7374	1123456

The total number of case of confirmed, deaths, recovered each month

```
1 • SELECT
2     YEAR(Date) AS year,
3     MONTH(Date) AS month,
4     SUM(confirmed) AS total_confirmed,
5     SUM(deaths) AS total_deaths,
6     SUM(recovered) AS total_recovered
7 FROM
8     CoronaVirus
9 GROUP BY
10     YEAR(Date), MONTH(Date);
11
```

year	month	total_confirmed
NULL	NULL	169065144

total_deaths	total_recovered
3647894	113089548

Check how corona virus spread out with respect to confirmed case

(Eg.: total confirmed cases, their average, variance & STDEV)

```
-- Total confirmed cases
SELECT
    SUM(confirmed) AS total_confirmed_cases
FROM
    CoronaVirus;
```

total_confirmed_cases
169065144

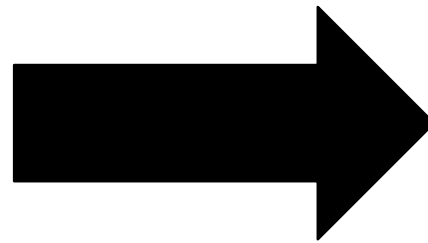

```
-- Average confirmed cases
```

```
SELECT
```

```
    AVG(confirmed) AS average_confirmed_cases
```

```
FROM
```

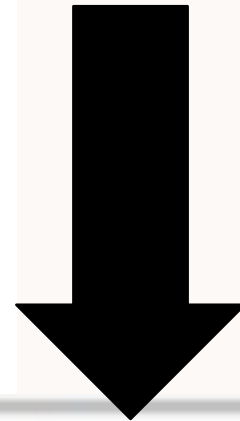
```
    CoronaVirus;
```



average_confirmed_cases

2156.8283

```
-- Variance of confirmed cases  
SELECT  
    VARIANCE(confirmed) AS variance_confirmed_cases  
FROM  
    CoronaVirus;
```



variance_confirmed_cases
157288925.07796532

```
-- Standard deviation of confirmed cases
```

```
SELECT
```

```
STDDEV_POP(confirmed) AS std_dev_confirmed_cases
```

```
FROM
```

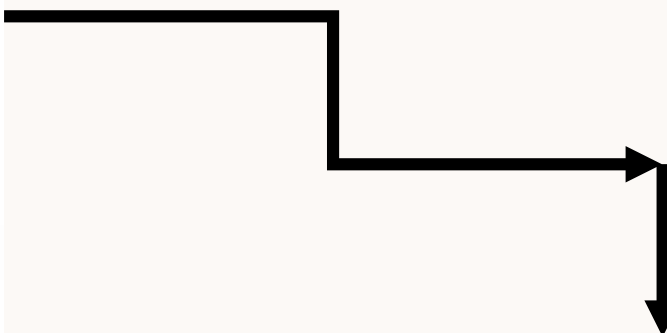
```
CoronaVirus;
```



std_dev_confirmed_cases
12541.488152446875

Check how corona virus spread out with respect to death case per month -- (Eg.: total confirmed cases, their average, variance & STDEV)

```
-- Total death cases per month
SELECT
    YEAR(Date) AS year,
    MONTH(Date) AS month,
    SUM(deaths) AS total_death_cases
FROM
    CoronaVirus
GROUP BY
    YEAR(Date), MONTH(Date);
```



year	month	total_death_cases
NULL	NULL	3647894

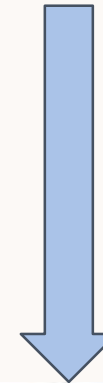
```
-- Average death cases per month  
SELECT  
    YEAR(Date) AS year,  
    MONTH(Date) AS month,  
    AVG(deaths) AS average_death_cases  
FROM  
    CoronaVirus  
GROUP BY  
    YEAR(Date), MONTH(Date);
```



average_death_cases

46.5376

```
-- Variance of death cases per month  
SELECT  
    YEAR(Date) AS year,  
    MONTH(Date) AS month,  
    VARIANCE(deaths) AS variance_death_cases  
FROM  
    CoronaVirus  
GROUP BY  
    YEAR(Date), MONTH(Date);
```



<u>variance_death_cases</u>
45892.01885355753


```
-- Standard deviation of death cases per month
```

```
SELECT
```

```
    YEAR(Date) AS year,
```

```
    MONTH(Date) AS month,
```

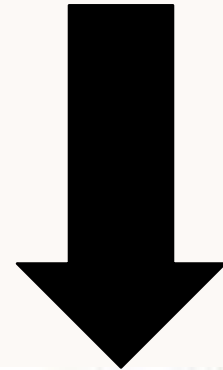
```
    STDDEV_POP(deaths) AS std_dev_death_cases
```

```
FROM
```

```
    CoronaVirus
```

```
GROUP BY
```

```
    YEAR(Date), MONTH(Date);
```



std_dev_death_cases

214.22422564583476

Check how corona virus spread out with respect to recovered case -- (Eg.: total confirmed cases, their average, variance & STDEV)

```
-- Total recovered cases
```

```
SELECT
```

```
SUM(recovered) AS total_recovered_cases
```

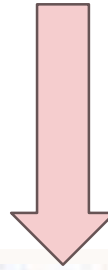
```
FROM
```

```
CoronaVirus;
```



total_recovered_cases
113089548

```
-- Average recovered cases  
SELECT  
    AVG(recovered) AS average_recovered_cases  
FROM  
    CoronaVirus;
```



<u>average_recovered_cases</u>
1442.7264

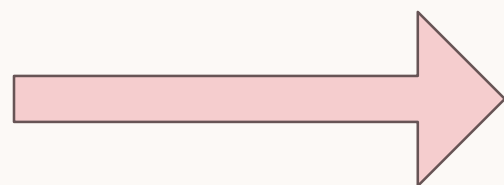

```
-- Variance of recovered cases
```

```
SELECT
```

```
    VARIANCE(recovered) AS variance_recovered_cases
```

```
FROM
```

```
    CoronaVirus;
```



variance_recovered_cases
107029523.26229636

```
-- Standard deviation of recovered cases
```

```
SELECT
```

```
STDDEV_POP(recovered) AS std_dev_recovered_cases
```

```
FROM
```


```
CoronaVirus;
```



std_dev_recovered_cases
10345.507395110999

Find Country having highest number of the Confirmed case

```
SELECT  
    `Country/Region`  
FROM  
    CoronaVirus  
WHERE  
    confirmed = (  
        SELECT  
            MAX(confirmed)  
        FROM  
            CoronaVirus  
    );
```



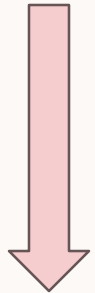
Country/Region
Turkey

Find Country having lowest number of the death case

```
SELECT
    `Country/Region`
FROM
    CoronaVirus
WHERE
    deaths = (
        SELECT
            MIN(deaths)
        FROM
            CoronaVirus
    );
```



Country/Region
Afghanistan
Afghanistan
Afghanistan
Afghanistan
Afghanistan
Afghanistan
Afghanistan
Afghanistan
Afghanistan
Afghanistan
Afghanistan



43082
Country/
Region

Find top 5 countries having highest recovered case

```
SELECT
    `Country/Region`,
    recovered
FROM
    CoronaVirus
ORDER BY
    recovered DESC
LIMIT
    5;
```



Country/Region	recovered
Turkey	1123456
India	422436
India	389851
Brazil	388340
India	386404

Conclusion

The SQL analysis of the corona dataset provides insights into the pandemic's spread, including trends over time, monthly averages, and country-specific metrics. It reveals patterns of infection, recovery, and mortality, aiding in understanding and managing the crisis efficiently.



Stay
H O M E



STAY SAFE
— LET'S STOP —
CORONAVIRUS

THANK YOU