INTERNSHIP TASK – 2
CORONA VIRUS ANALYSIS WITH SQL

By. ANNU JHA
Batch – MIP-DA-09

INTRODUCTION

- This project uses SQL to analyze COVID-19 data and understand the pandemic's impact. By exploring a dataset of COVID-19 cases, we answer important questions about confirmed cases, deaths, and recoveries across different countries and over time. This analysis helps us see how the virus has spread and affected different regions, demonstrating how SQL can be used to examine real-world data.

PROBLEM STATEMENT

- The COVID-19 pandemic has significantly impacted public health, creating an urgent need for data-driven insights to understand the virus's spread.

- As a data analyst, I have been tasked with analyzing a COVID-19 dataset to derive meaningful insights and present my findings. This analysis will help understand the virus's spread, impact, and trends over time, thereby aiding informed decision-making.

DATASET

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.

Table: corona_virus

Columns:

Province	text
Country_Region	text
Latitude	float
Longitude	float
Confirmed	int
Deaths	int
Recovered	int
Date	date

DATABASE CREATION

```
-- ----------------------CORONA VIRUS ANALYSIS ----------------
-- Creating Database for storing COVID-19 data
CREATE DATABASE IF NOT EXISTS CORONA_VIRUS_DATA;

-- Set this database as default database for further analysis
USE CORONA_VIRUS_DATA;
```

Note: Here, I created the database and made it the default for further analysis.

TABLE CREATION AND RECORD UPLOADING

```sql
-- Table Creation
CREATE TABLE CORONA_VIRUS
(
    Province TEXT,
    Country_Region TEXT,
    Latitude TEXT,
    Longitude TEXT,
    Date TEXT,
    Confirmed TEXT,
    Deaths TEXT,
    Recovered TEXT
);
```

Note: In the table, I define the data type of each field as "TEXT" so that I can change the data type after successfully uploading the records into the table.

```sql
-- Check the secure file possition in MySQL
SELECT @@secure_file_priv;

-- Load Data (Records) in tha table
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Corona_Virus_Dataset.csv'
INTO TABLE corona_virus
FIELDS TERMINATED BY ','
(Province, Country_Region, Latitude, Longitude, Date, Confirmed, Deaths, Recovered);
# Successfully loaded the records in the table
```

CHANGING DATATYPE

```
-- Change "Confirmed", "Deaths", "Recovered" columns Datatype
ALTER TABLE corona_virus
    MODIFY Confirmed INT,
    MODIFY Deaths INT,
    MODIFY Recovered INT;


-- Change "Latitude", "Longitude" columns Datatype
ALTER TABLE corona_virus
    MODIFY Latitude FLOAT,
    MODIFY Longitude FLOAT;
```

Here, I changed to the appropriate data types.

CHANGING DATE COLUMN DATA TYPE FROM TEXT TO DATE

```sql
-- Step 1: Disable safe update mode
SET SQL_SAFE_UPDATES = 0;

-- Step 2: Add a new DATE column
ALTER TABLE corona_virus ADD COLUMN NewDate DATE;

-- Step 3: Update the new column with converted values
UPDATE corona_virus
SET NewDate = STR_TO_DATE(Date, '%d-%m-%Y');

-- Step 4: Re-enable safe update mode
SET SQL_SAFE_UPDATES = 1;

 -- Step 5: Verify the data
SELECT * FROM corona_virus WHERE NewDate IS NULL AND Date IS NOT NULL;

-- Step 6: Drop the old column and rename the new column
ALTER TABLE corona_virus DROP COLUMN Date;
ALTER TABLE corona_virus CHANGE COLUMN NewDate Date DATE;
```

TABLE DETAILS

```
-- Diaplay table details
DESCRIBE corona_virus;
```

Field	Type	Null	Key	Default	Extra
Province	text	YES		NULL	
Country_Region	text	YES		NULL	
Latitude	float	YES		NULL	
Longitude	float	YES		NULL	
Confirmed	int	YES		NULL	
Deaths	int	YES		NULL	
Recovered	int	YES		NULL	
Date	date	YES		NULL	

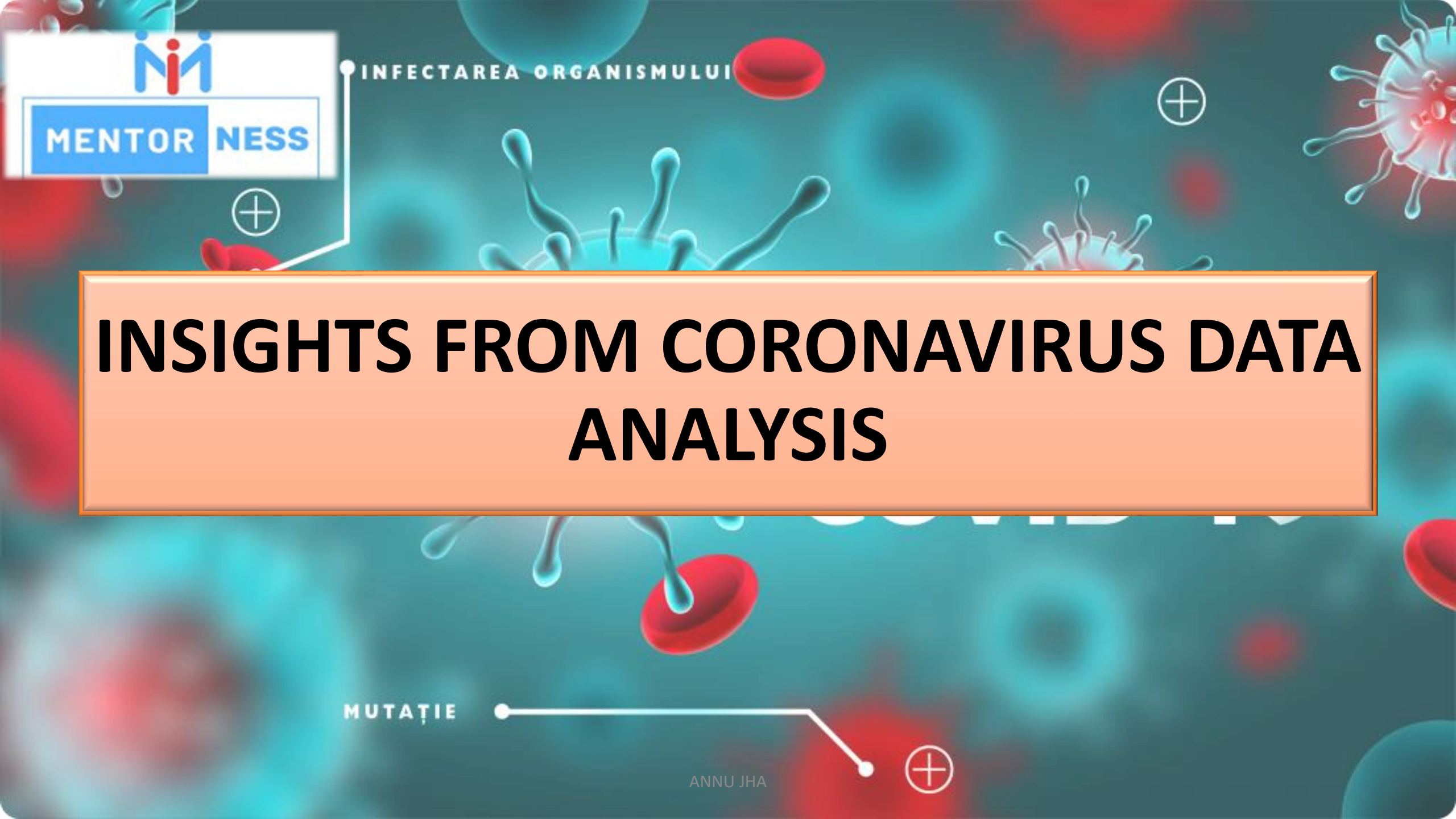
VIEWING THE TABLE DATA

```
-- Run Query to see records

SELECT * FROM CORONA_VIRUS;
```

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

Province	Country_Region	Latitude	Longitude	Confirmed	Deaths	Recovered	Date
Afghanistan	Afghanistan	33.9391	67.71	1842	51	409	2021-06-10
Afghanistan	Afghanistan	33.9391	67.71	1824	56	318	2021-06-11
Afghanistan	Afghanistan	33.9391	67.71	1724	54	302	2021-06-09
Afghanistan	Afghanistan	33.9391	67.71	1617	42	376	2021-06-07
Afghanistan	Afghanistan	33.9391	67.71	1509	34	74	2021-06-04
Afghanistan	Afghanistan	33.9391	67.71	1485	46	803	2021-01-04
Afghanistan	Afghanistan	33.9391	67.71	1485	64	571	2021-06-08
Afghanistan	Afghanistan	33.9391	67.71	1335	36	121	2021-06-05
Afghanistan	Afghanistan	33.9391	67.71	1261	41	357	2021-06-06
Afghanistan	Afghanistan	33.9391	67.71	1139	29	112	2021-06-01
Algeria	Algeria	28.0339	1.6596	1133	15	649	2020-11-24
Afghanistan	Afghanistan	33.9391	67.71	1121	78	471	2021-06-13
Algeria	Algeria	28.0339	1.6596	1103	12	619	2020-11-20
Afghanistan	Afghanistan	33.9391	67.71	1093	27	107	2021-06-03
Algeria	Algeria	28.0339	1.6596	1088	17	611	2020-11-22
Algeria	Algeria	28.0339	1.6596	1085	23	622	2020-11-26
Afghanistan	Afghanistan	33.9391	67.71	1077	25	179	2021-05-31

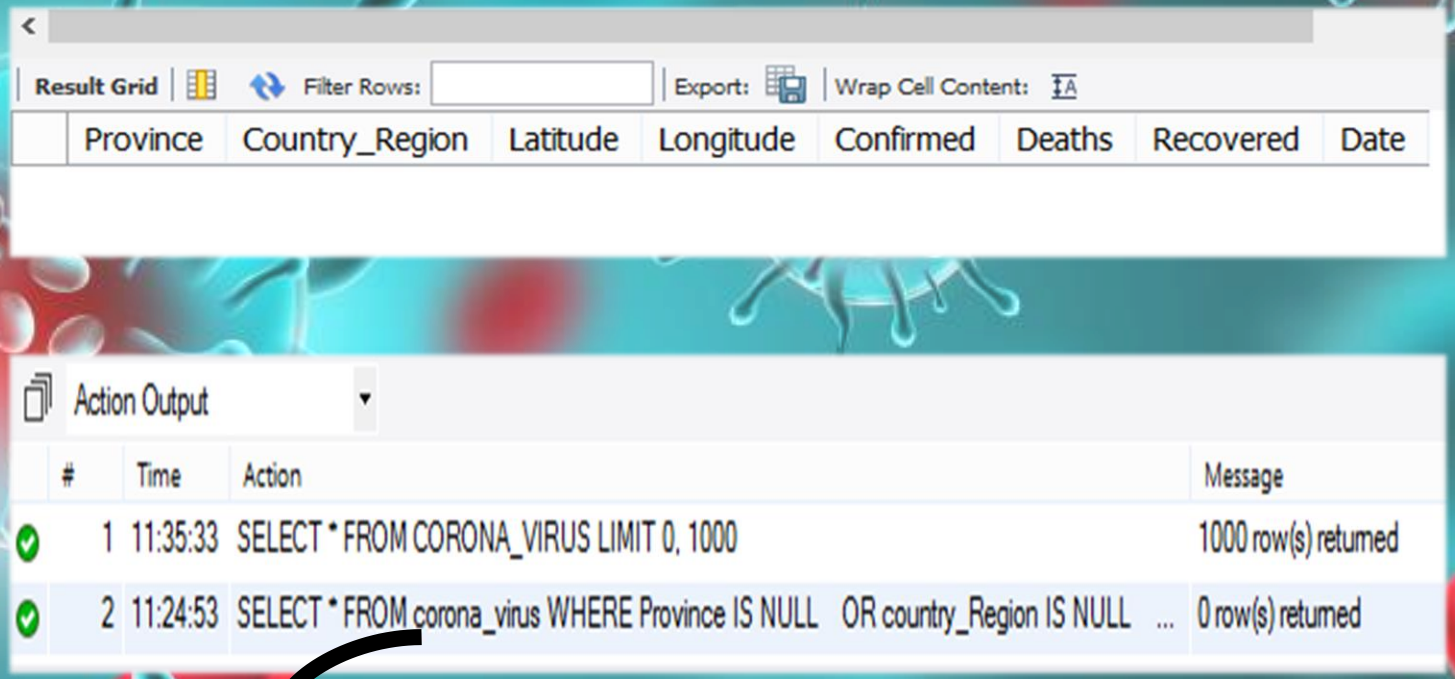


INSIGHTS FROM CORONAVIRUS DATA ANALYSIS

ANNU JHA

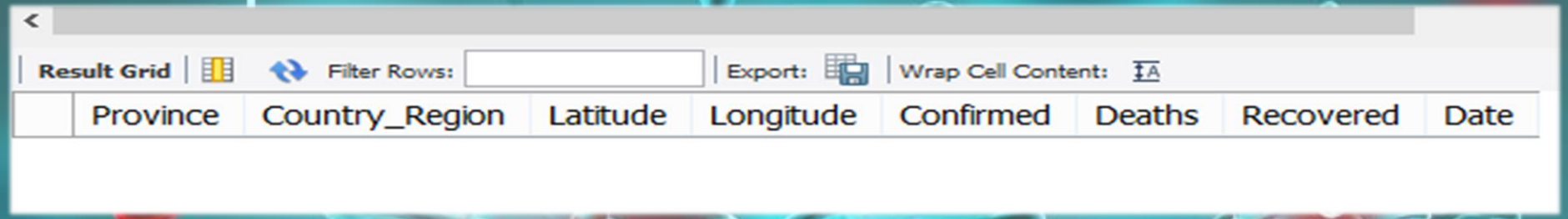
1. TO AVOID ANY ERRORS, CHECK MISSING VALUE / NULL VALUE

```sql
-- Q1. Write a code to check NULL values
SELECT *
FROM corona_virus
WHERE Province IS NULL
   OR country_Region IS NULL
   OR Latitude IS NULL
   OR Longitude IS NULL
   OR Confirmed IS NULL
   OR Deaths IS NULL
   OR Recovered IS NULL
   OR Date IS NULL;
```



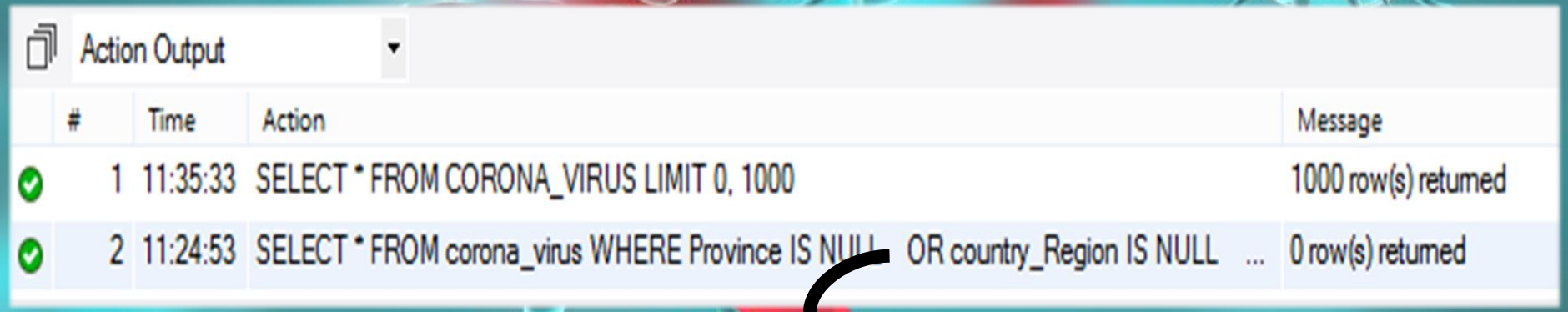
Insights: The analysis reveals that there are *no null values* present in the table.

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



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

Province	Country_Region	Latitude	Longitude	Confirmed	Deaths	Recovered	Date



Action Output

#	Time	Action	Message
1	11:35:33	SELECT * FROM CORONA_VIRUS LIMIT 0, 1000	1000 row(s) returned
2	11:24:53	SELECT * FROM corona_virus WHERE Province IS NULL OR country_Region IS NULL ...	0 row(s) returned

Insights: As the output indicates the absence of null values in the table, there is no requirement to update any null values.

3. Check total number of rows in the table.

```sql
-- Q3. check total number of rows

SELECT
    COUNT(*) AS Total_Rows
FROM
    corona_virus;
```

Result Grid | Filter

Total_Rows
78385

Insights: The table "corona_virus" contains 78,385 rows.

4. Verify the start and end dates of corona virus reporting.

```sql
-- Q4. Check what is start_date and end_date

SELECT
    MIN(date) AS START_DATE, MAX(date) AS END_DATE
FROM
    corona_virus;
```

Result Grid | Filter Rows:

START_DATE	END_DATE
2020-01-22	2021-06-13

Insights: According to the data, the start date of reporting coronavirus cases is 2020-01-22, and the end date is 2021-06-13.

5. Number of month present in dataset

```sql
-- Count the number of months year-wise
SELECT
    YEAR(Date) AS YEAR,
    COUNT(DISTINCT MONTH(Date)) AS TOTAL_MONTH
FROM
    corona_virus
GROUP BY
    year
ORDER BY
    year;
```

```sql
-- COUNT NUMBER OF MONTHS
SELECT
    COUNT(DISTINCT(DATE_FORMAT(Date, '%Y-%b'))) AS TOTAL_MONTHS
FROM
    corona_virus;
```

Result Grid | Filter Rows:

YEAR	TOTAL_MONTH
2020	12
2021	6

Result Grid | Filter Rows:

TOTAL_MONTHS
18

Insights: The data reveals a complete set of records for each month in 2020, totalling 12 months, enabling a comprehensive analysis of COVID-19 trends. However, for 2021, we only have data for 6 months, resulting in a total of 18 months of records available for analysis.

6. Find monthly average for confirmed, deaths, recovered

```sql
-- Q6. Find monthly average for confirmed, deaths, recovered

SELECT
    DATE_FORMAT(Date, '%b-%Y') AS YEAR_MONTHS,
    FLOOR(AVG(Confirmed)) AS AVG_CONFIRMED_CASES,
    FLOOR(AVG(Recovered)) AS AVG_RECOVERED_CASES,
    FLOOR(AVG(Deaths)) AS AVG_NO_OF_DEATHS
FROM
    corona_virus
GROUP BY
    YEAR(Date), MONTH(Date),YEAR_MONTHS
ORDER BY
    YEAR(Date), MONTH(Date), YEAR_MONTHS;
```

YEAR_MONTHS	AVG_CONFIRMED_CASES	AVG_RECOVERED_CASES	AVG_NO_OF_DEATHS
Jan-2020	4	0	0
Feb-2020	15	7	0
Mar-2020	161	27	8
Apr-2020	505	171	41
May-2020	574	318	30
Jun-2020	859	548	29
Jul-2020	1432	983	35

Result 8

Output

Action Output

#	Time	Action		Message
15	11:06:59	SELECT	DATE_FORMAT(Date, '%Y-%m') AS YEAR_MONTHS, DATE_FORMAT(DAT...	18 row(s) returned
16	11:10:42	SELECT	DATE_FORMAT(Date, '%b-%Y') AS YEAR_MONTHS, FLOOR(AVG(Confirme...	18 row(s) returned

Insights: The monthly average query shows the average number of confirmed cases, deaths, and recoveries each month. This helps identify trends and changes in the pandemic's impact over time.

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

```sql
-- Q7. Find most frequent value for confirmed, deaths, recovered each month
SELECT
    DATE_FORMAT(Date, '%Y-%b') AS YEAR_MONTHS,
    SUBSTRING_INDEX(GROUP_CONCAT(Confirmed ORDER BY Confirmed DESC), ',', 1) AS most_frequent_confirmed,
    SUBSTRING_INDEX(GROUP_CONCAT(Deaths ORDER BY Deaths DESC), ',', 1) AS most_frequent_deaths,
    SUBSTRING_INDEX(GROUP_CONCAT(Recovered ORDER BY Recovered DESC), ',', 1) AS most_frequent_recovered
FROM
    corona_virus
GROUP BY
    YEAR(DATE),MONTH(DATE),YEAR_MONTHS
ORDER BY
    YEAR(DATE),MONTH(DATE),YEAR_MONTHS;
```

YEAR_MONTHS	most_frequent_confirmed	most_frequent_deaths	most_frequent_recovered
2020-Jan	2131	49	51
2020-Feb	14840	242	3418
2020-Mar	26314	1085	4289
2020-Apr	50740	2607	33227
2020-May	34907	2309	51717
2020-Jun	54771	2003	94305

Result 10

Output

Action Output

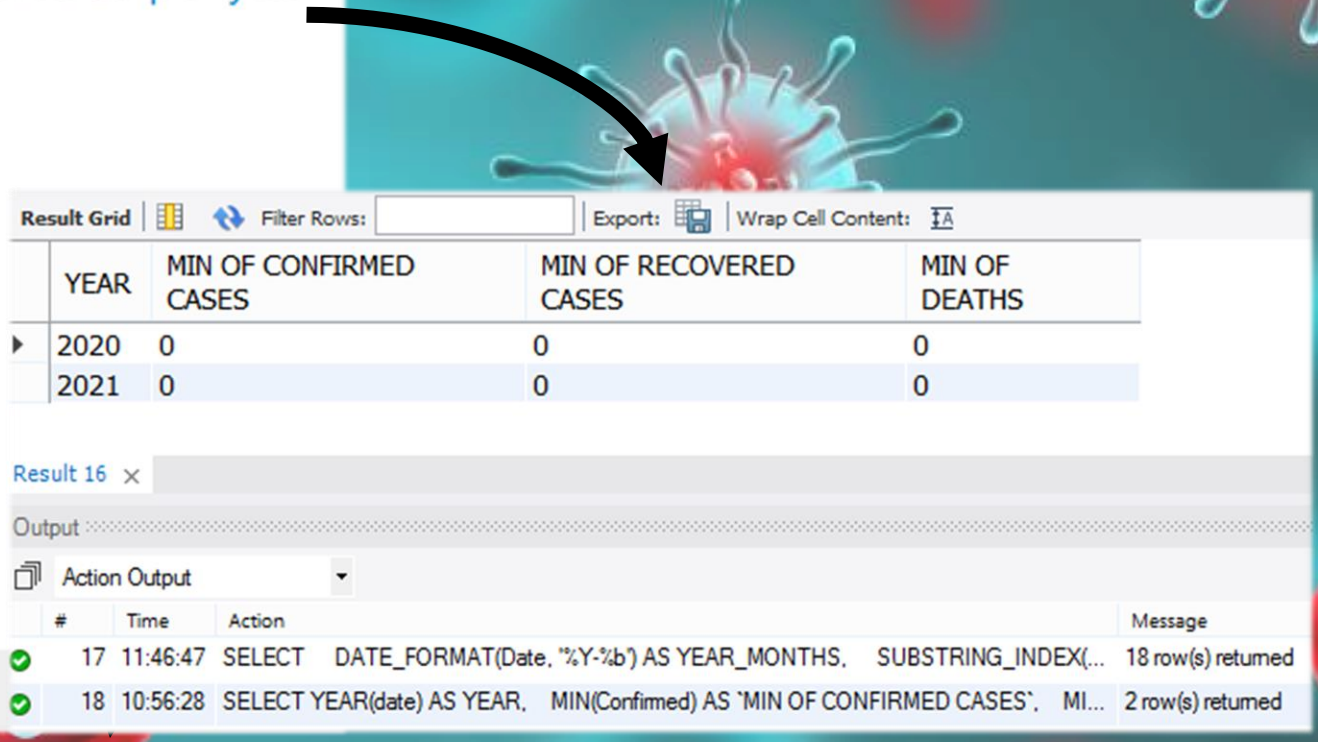
#	Time	Action			Message
17	11:16:56	SELECT	DATE_FORMAT(Date, '%Y-%b') AS YEAR_MONTHS,	SUBSTRING_INDEX(...	18 row(s) returned
18	11:19:04	SELECT	DATE_FORMAT(Date, '%Y-%b') AS YEAR_MONTHS,	SUBSTRING_INDEX(...	18 row(s) returned

Insights: The most frequent values for confirmed cases, deaths, and recoveries each month reveal the most common outcomes during the pandemic. This helps identify typical monthly figures and detect any recurring patterns or anomalies.

8. Find minimum values for confirmed, deaths, recovered per year.

```
-- Q8. Find minimum values for confirmed, deaths, recovered per year

SELECT YEAR(date) AS YEAR,
    MIN(Confirmed) AS `MIN OF CONFIRMED CASES`,
    MIN(Recovered) AS `MIN OF RECOVERED CASES`,
    MIN(Deaths) AS `MIN OF DEATHS`
FROM
    corona_virus
GROUP BY
    YEAR;
```



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

YEAR	MIN OF CONFIRMED CASES	MIN OF RECOVERED CASES	MIN OF DEATHS
2020	0	0	0
2021	0	0	0

Result 16

Output

Action Output

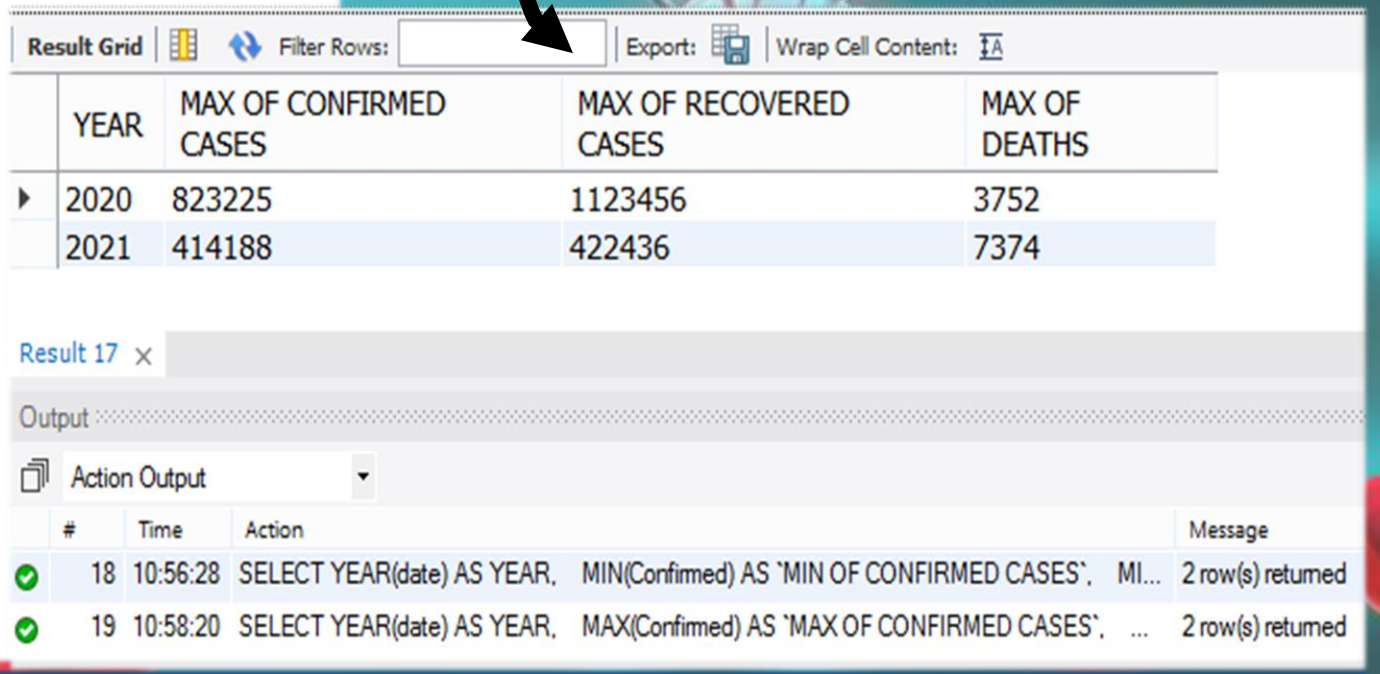
#	Time	Action	Message
17	11:46:47	SELECT DATE_FORMAT(Date, '%Y-%b') AS YEAR_MONTHS, SUBSTRING_INDEX(...	18 row(s) returned
18	10:56:28	SELECT YEAR(date) AS YEAR, MIN(Confirmed) AS `MIN OF CONFIRMED CASES`, MI...	2 row(s) returned

Insights: The minimum values for confirmed cases, deaths, and recoveries per year highlight the lowest recorded figures annually. These insights help identify the least severe periods of the pandemic each year.

9. Find maximum values of confirmed, deaths, recovered per year

```
-- Q9. Find maximum values of confirmed, deaths, recovered per year

SELECT YEAR(date) AS YEAR,
    MAX(Confirmed) AS `MAX OF CONFIRMED CASES`,
    MAX(Recovered) AS `MAX OF RECOVERED CASES`,
    MAX(Deaths) AS `MAX OF DEATHS`
FROM
    corona_virus
GROUP BY
    YEAR;
```



YEAR	MAX OF CONFIRMED CASES	MAX OF RECOVERED CASES	MAX OF DEATHS
2020	823225	1123456	3752
2021	414188	422436	7374

Insights: The maximum values for confirmed cases, deaths, and recoveries per year highlight the highest recorded figures annually. These insights help identify the most severe periods of the pandemic each year.

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

```sql
-- Q10. The total number of case of confirmed, deaths, recovered each month

SELECT
    date_format(DATE, '%Y-%b') AS MONTH,
    SUM(Confirmed) AS TOTAL_CONFIRMED_CASES,
    SUM(Recovered) AS TOTAL_RECOVERED_CASES,
    SUM(Deaths) AS TOTAL_DEATHS
FROM
    corona_virus
GROUP BY
    MONTH;
```

MONTH	TOTAL_CONFIRMED_CASES	TOTAL_RECOVERED_CASES	TOTAL_DEATHS
2020-Jan	6384	143	190
2020-Feb	68312	31405	2651
2020-Mar	769236	133070	41346
2020-Apr	2336798	792987	191833

Result 19

Output

Action Output

#	Time	Action		Message
20	11:00:18	SELECT	date_format(DATE, '%b') AS MONTH, SUM(Confirmed) AS TOTAL_CONFIR...	12 row(s) returned
21	11:02:17	SELECT	date_format(DATE, '%Y-%b') AS MONTH, SUM(Confirmed) AS TOTAL_CON...	18 row(s) returned

Insights: The total number of confirmed cases, deaths, and recoveries each month provides a comprehensive view of the pandemic's impact over time. These insights help track the monthly progression and severity of COVID-19.

11. Check how corona virus spread out with respect to confirmed case.

```
-- Q11. Check how corona virus spread out with respect to confirmed case
--      (Eg.: total confirmed cases, their average, variance & STDEV )

SELECT
    SUM(Confirmed) AS TOTAL_CONFIRMED_CASES,
    ROUND(AVG(Confirmed),2) AS AVG_CONFIRMED_CASES,
    ROUND(VARIANCE(Confirmed),2) AS VAR_CONFIRMED_CASES,
    ROUND(STDDEV(Confirmed),2) AS STDEV_CONFIRMED_CASES
FROM
    corona_virus;
```

TOTAL_CONFIRMED_CASES	AVG_CONFIRMED_CASES	VAR_CONFIRMED_CASES	STDEV_CONFIRMED_CASES
169065144	2156.86	157290872.35	12541.57

Result 21

Output

Action Output

#	Time	Action		Message
22	11:12:16	SELECT	SUM(Confirmed) AS TOTAL_CONFIRMED_CASES, ROUND(AVG(Confirmed)...	1 row(s) returned
23	11:13:40	SELECT	SUM(Confirmed) AS TOTAL_CONFIRMED_CASES, ROUND(AVG(Confirmed)...	1 row(s) returned

Insights: Analyzing the total confirmed cases, their average, variance, and standard deviation shows how COVID-19 cases are distributed. These insights help understand the overall spread, consistency, and volatility of the pandemic's impact over time.

12. Check how corona virus spread out with respect to death case per month.

```sql
-- Q12. Check how corona virus spread out with respect to death case per month
--      (Eg.: total confirmed cases, their average, variance & STDEV )
SELECT
    DATE_FORMAT(DATE, '%Y-%b') AS YEAR_MONTHS,
    SUM(Deaths) AS TOTAL_DEATHS,
    ROUND(AVG(Deaths),2) AS AVERAGE_DEATHS,
    ROUND(VARIANCE(Deaths),2) AS VARIANCE_DEATHS,
    ROUND(STDDEV(Deaths),2) AS STDDEV_DEATHS
FROM
    corona_virus
GROUP BY
    YEAR_MONTHS;
```

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

YEAR_MONTHS	TOTAL_DEATHS	AVERAGE_DEATHS	VARIANCE_DEATHS	STDDEV_DEATHS
2020-Jan	190	0.12	4.25	2.06
2020-Feb	2651	0.59	68.32	8.27
2020-Mar	41346	8.66	3900.79	62.46
2020-Apr	191833	41.52	40504.27	201.26
2020-May	144561	30.28	20684.91	143.82

Result 22

Output

Action Output

#	Time	Action		Message
32	11:33:08	SELECT	DATE_FORMAT(DATE, '%Y-%b') AS YEAR_MONTHS, SUM(Deaths) AS TOT...	18 row(s) returned
33	11:34:18	SELECT	DATE_FORMAT(DATE, '%Y-%b') AS YEAR_MONTHS, SUM(Deaths) AS TOT...	18 row(s) returned

Insights: Analyzing the total deaths, their average, variance, and standard deviation per month reveals how fatalities are distributed. These insights help understand the overall spread, consistency, and variability of COVID-19's impact on mortality each month.

13. Check how corona virus spread out with respect to recovered case.

```sql
-- Q13. Check how corona virus spread out with respect to recovered case
--      (Eg.: total confirmed cases, their average, variance & STDEV )

SELECT
    SUM(Recovered) AS TOTAL_RECOVERED_CASES,
    ROUND(AVG(Recovered),2) AS AVG_RECOVERED_CASES,
    ROUND(VARIANCE(Recovered),2) AS VAR_RECOVERED_CASES,
    ROUND(STDDEV(Recovered),2) AS STDEV_RECOVERED_CASES
FROM
    corona_virus;
```

TOTAL_RECOVERED_CASES	AVG_RECOVERED_CASES	VAR_RECOVERED_CASES	STDEV_RECOVERED_CASES
113089548	1442.74	107030862.14	10345.57

Result 23

Output

Action Output

#	Time	Action	Message
24	11:16:09	SELECT DATE_FORMAT(DATE, '%Y-%b') AS YEAR_MONTHS, SUM(Deaths) AS TOT...	18 row(s) returned
25	11:18:37	SELECT SUM(Recovered) AS TOTAL_RECOVERED_CASES, ROUND(AVG(Recover...	1 row(s) returned

Insights: Analyzing the total recoveries, their average, variance, and standard deviation per month reveals how recoveries are distributed. These insights help understand the overall spread, consistency, and variability of COVID-19 recovery rates over time.

14. Find Country having highest number of the Confirmed case.

```sql
-- Q14. Find Country having highest number of the Confirmed case

SELECT
    COUNTRY_REGION, SUM(CONFIRMED) AS TOTAL_CONFIRMED_CASES
FROM
    corona_virus
GROUP BY
    Country_Region
ORDER BY
    TOTAL_CONFIRMED_CASES DESC
LIMIT 1;
```

COUNTRY_REGION	TOTAL_CONFIRMED_CASES
US	33461982

Insights: The United States has the highest number of confirmed COVID-19 cases, with approximately 33,461,982 cases reported. This reflects the significant impact of the pandemic in the country, highlighting the importance of ongoing efforts to mitigate its spread and manage its effects.

15. Find Country having lowest number of the death case.

```sql
-- Q15. Find Country having lowest number of the death case

SELECT
    COUNTRY_REGION, SUM(Deaths) AS TOTAL_DEATHS
FROM
    corona_virus
GROUP BY
    Country_Region
ORDER BY
    TOTAL_DEATHS ASC
LIMIT 1;
```

	COUNTRY_REGION	TOTAL_DEATHS
▶	Dominica	0

Insights: Dominica stands out with the lowest number of reported COVID-19 deaths, recording none. This underscores the effectiveness of measures taken within the country to mitigate the impact of the pandemic and highlights the importance of proactive public health strategies in saving lives.

16. Find top 5 countries having highest recovered case.

```sql
-- Q16. Find top 5 countries having highest recovered case

SELECT
    COUNTRY_REGION,
    SUM(Recovered) AS TOTAL_RECOVERED
FROM
    corona_virus
GROUP BY
    Country_Region
ORDER BY
    TOTAL_RECOVERED DESC
LIMIT 5;
```

COUNTRY_REGION	TOTAL_RECOVERED
India	28089649
Brazil	15400169
US	6303715
Turkey	5202251
Russia	4745756

Insights: India, Brazil, the US, Turkey, and Russia are the top five countries with the most people recovering from COVID-19. This shows that efforts to help people get better are working in these places. It also reminds us that fighting the virus requires everyone to work together worldwide.

MENTOR NESS

INFECTAREA ORGANISMULUI

THANKYOU

MUTAȚIE