CORONA VIRUS ANALYSIS WITH SQL

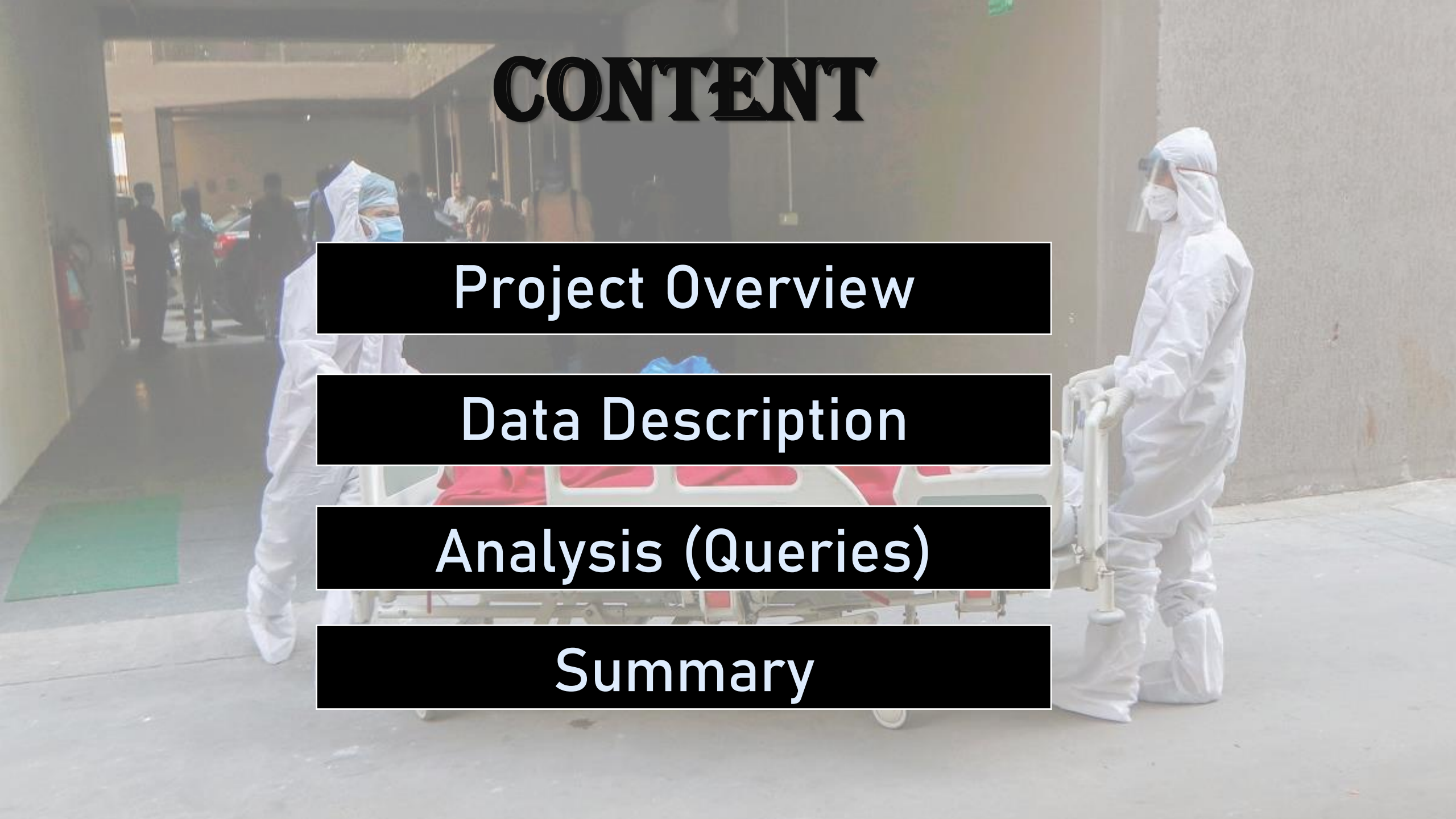
MENTORNESS

INTERNSHIP PROJECT BY

PRIYANSHU BHATIA

BATCH NAME: MIP-DA-07

CONTENT



- Project Overview
- Data Description
- Analysis (Queries)
- Summary

PROJECT OVERVIEW

- Analyzing a Corona Virus dataset using "Corona_virus_dataset" table is necessary to "comprehend Coronavirus." Its goal is to gather information about impacted and recovered cases. Our goal is to comprehend the global impact of the coronavirus using SQL queries.
- Important queries include which country is most and least impacted by the coronavirus, how long ago occurrence occurred , and all the details regarding confirmed ,recovered and fatal cases. Our objective is to offer practical insights so that future decisions can be made with knowledge.
- The project encompasses data explorations , query formulation , result interpretation and data visualization techniques. Through concise presentation

DATA DESCRIPTION

The dataset include 1 table: "Corona_Virus_Dataset"

"CORONA_VIRUS_DATASET"

- **Province**
- **Country_region**
- **Latitude**
- **Longitude**
- **Date**
- **Confirmed**
- **Deaths**
- **Recovered**

province	country_region	latitude	longitude	date	confirmed	deaths	recovered
Afghanistan	Afghanistan	33.93911	67.709953	2020-01-22	0	0	0
Afghanistan	Afghanistan	33.93911	67.709953	2020-01-23	0	0	0
Afghanistan	Afghanistan	33.93911	67.709953	2020-01-24	0	0	0
Afghanistan	Afghanistan	33.93911	67.709953	2020-01-25	0	0	0
Afghanistan	Afghanistan	33.93911	67.709953	2020-01-26	0	0	0
Afghanistan	Afghanistan	33.93911	67.709953	2020-01-27	0	0	0
Afghanistan	Afghanistan	33.93911	67.709953	2020-01-28	0	0	0
Afghanistan	Afghanistan	33.93911	67.709953	2020-01-29	0	0	0
Afghanistan	Afghanistan	33.93911	67.709953	2020-01-30	0	0	0
Afghanistan	Afghanistan	33.93911	67.709953	2020-01-31	0	0	0
Afghanistan	Afghanistan	33.93911	67.709953	2020-02-01	0	0	0
Afghanistan	Afghanistan	33.93911	67.709953	2020-02-02	0	0	0
Afghanistan	Afghanistan	33.93911	67.709953	2020-02-03	0	0	0
Afghanistan	Afghanistan	33.93911	67.709953	2020-02-04	0	0	0

```sql
##Q1. Write a code to check NULL values
SELECT * FROM corona_virus_dataset WHERE province IS NULL;
SELECT * FROM corona_virus_dataset WHERE country_region IS NULL;
SELECT * FROM corona_virus_dataset WHERE latitude IS NULL;
SELECT * FROM corona_virus_dataset WHERE longitude IS NULL;
SELECT * FROM corona_virus_dataset WHERE date IS NULL;
SELECT * FROM corona_virus_dataset WHERE confirmed IS NULL;
SELECT * FROM corona_virus_dataset WHERE deaths IS NULL;
SELECT * FROM corona_virus_dataset WHERE  recovered IS NULL;
```

corona_virus_dataset (0r × 8c) | corona_virus_dataset (0r × 8c) | corona_virus_dataset (0r × 8c) | corona_virus_dataset (0r × 8c) | corona_virus_dataset (0r × 8c) | corona_virus_dataset (0r × 8c) | corona_virus_dataset (0r ×

province	country_region	latitude	longitude	date	confirmed	deaths	recovered

Analysis:-

- To find the NULL values in the coronavirus table, it employs the IS NULL logical operators.
- The result is that there is no NULL values present in the corona_virus_dataset.

```sql

##Q2. If NULL values are present, update them with zeros for all columns.
update corona_virus_dataset
SET province = COALESCE(province, 0),
    country_region = COALESCE(country_region, 0),
    latitude = COALESCE(latitude, 0),
    longitude = COALESCE(longitude,0),
    DATE = COALESCE(DATE,0),
    confirmed = COALESCE(confirmed,0),
    deaths = COALESCE(deaths,0),
    recovered = COALESCE(recovered,0)
WHERE province IS NULL OR country_region IS NULL OR latitude
      IS NULL OR longitude IS NULL OR DATE IS NULL OR confirmed
      IS NULL OR deaths IS NULL OR recovered IS NULL;

```

Analysis:-

- It performs COALESCE function to replace the NULL values with Zeros present in the corona_virus_dataset table.
- The result is that there is none NULL values present in the corona_virus_dataset that should be replaced with zeros.

```sql
28
29 -- Q3. check total number of rows
30
31 SELECT COUNT(*) as no_of_rows from corona_virus_dataset;
32
```

corona_virus_dataset (1r × 1c)

no_of_rows
78,386

Analysis:-

- It counts the number of rows present in the corona_virus_dataset using the COUNT function that is a logical operator in SQL.
- It clears show that there are 78,386 number of rows present in the corona_virus_dataset table.

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

SELECT min(DATE) as start_date ,MAX(DATE) AS end_date FROM corona_virus_dataset

```

corona_virus_dataset (1r × 2c)

start_date	end_date
2020-01-22	2021-06-13

Analysis:-

- It will extract the start_date and end_date of the date column in the corona_virus_dataset table using the MIN() and MAX() functions. Now, it will showcase the start_date as 2020-01-22 and end_date as 2021-06-13 as the record of the coronavirus cases occurred in the period of time.

```sql
-- Q5. Number of month present in dataset

SELECT count(distinct DATE_FORMAT(DATE,"%Y-%m")) AS No_of_Month
FROM corona_virus_dataset

```

corona_virus_dataset (1r × 1c)

No_of_Month
18

Analysis:-

- The months are extracted from the date column in the table using the DATE_FORMAT method , the COUNT function,and the DISTINCT function in the SQL query. DISTINCT will extract only the unique values from the date column .The number of months included in the corona_virus_dataset will be retrieved.

```sql
-- Q6. Find monthly average for confirmed, deaths, recovered
SELECT DATE_FORMAT(DATE,"%Y-%m") AS Month_Year,
       ROUND(AVG(confirmed),2) AS Avg_Confirmed,
       ROUND(AVG(deaths),2) AS Avg_Deaths,
       ROUND(AVG(recovered),2) AS Avg_Recovered
FROM corona_virus_dataset
GROUP BY DATE_FORMAT(DATE,"%Y-%m")

```

corona_virus_dataset (18r × 4c)

Month_Year	Avg_Confirmed	Avg_Deaths	Avg_Recovered
2020-01	4.15	0.12	0.09
2020-02	15.3	0.59	7.03
2020-03	161.13	8.66	27.87
2020-04	505.8	41.52	171.64
2020-05	574.85	30.28	318.3
2020-06	859.23	29.82	548.79
2020-07	1,432.36	35.11	983.06
2020-08	1,611.84	37.54	1,299.29
2020-09	1,784.59	34.78	1,438.91
2020-10	2,412.2	36.76	1,420.64
2020-11	3,592.19	56.76	1,985.34
2020-12	4,050.44	71.22	2,497.89
2021-01	3,911.23	84.18	1,919.64
2021-02	2,433.36	69.16	1,558.39
2021-03	2,916.8	59.2	1,652.29
2021-04	4,699.36	78.44	3,074.79
2021-05	4,005.25	76.78	4,007.51
2021-06	2,508.63	66.26	2,769.45

Analysis:-

- By using average function it will calculate average confirmed ,recovered and deaths cases in the corona_virus_dataset.
- The results are grouped by month, and the avg_confirmed , avg_fatalities and avg_recovered monthly are filtered out. This inquiry facilitates comprehension of the monthly cases average across a 18 months sample period.

```sql
-- Q7. Find most frequent value for confirmed, deaths, recovered each month
SELECT
     Date_Format(DATE,"%Y-%m") AS month,
     COUNT(*) AS Frequency,
     MAX(confirmed) AS Most_Frequent_Confirmed,
     MAX(deaths) AS Most_Frequent_Deaths,
     MAX(recovered) AS Most_Frequent_Recovered
FROM Corona_virus_dataset
GROUP BY DATE_FORMAT(DATE,"%Y-%m");
```

Corona_virus_dataset (18r × 5c)

month	Frequency	Most_Frequent_Confirmed	Most_Frequent_Deaths	Most_Frequent_Recovered
2020-01	1,540	2,131	49	51
2020-02	4,466	14,840	242	3,418
2020-03	4,774	26,314	1,085	4,289
2020-04	4,620	50,740	2,607	33,227
2020-05	4,774	34,907	2,309	51,717
2020-06	4,620	54,771	2,003	94,305
2020-07	4,774	75,866	1,595	140,050
2020-08	4,774	85,687	1,505	95,881
2020-09	4,620	97,894	1,703	101,468
2020-10	4,774	99,264	3,351	388,340
2020-11	4,620	207,933	2,259	139,292
2020-12	4,774	823,225	3,752	1,123,456
2021-01	4,774	300,462	4,475	87,090
2021-02	4,312	134,975	3,907	98,389
2021-03	4,774	100,158	3,869	102,138
2021-04	4,620	401,993	4,249	299,988
2021-05	4,774	414,188	4,529	422,436
2021-06	2,002	134,154	7,374	231,456

Analysis:-

- The goal of the SQL query is to retrieve the most frequent values of corona_virus_dataset table cases that are confirmed,recovered and deaths.
- The result are grouped by month, and the most_frequent_confirmed,most_frequent_Deaths and most_frequent_recovered are filtered out. This inquiry facilitates comprehension of the monthly cases that are most frequent across a 18 months sample period.

```sql
-- Q8. Find minimum values for confirmed, deaths, recovered per year
SELECT
    YEAR(date) AS year,
    COUNT(*) AS Frequency,
    MIN(confirmed) AS Min_Values_Confirmed,
    MIN(deaths) AS Min_Values_Deaths,
    MIN(recovered) AS Min_Values_Recovered
FROM
    corona_virus_dataset
GROUP BY
    YEAR(date);
```

corona_virus_dataset (2r × 5c)

year	frequency	min_values_confirmed	min_values_deaths	min_values_recovered
2,020	53,130	0	0	0
2,021	25,256	0	0	0

Analysis:-

- By using min() function we will retrieve the minimum values for confirmed, deaths and recovered cases present in the corona_virus_dataset table.
- The min_values_confirmed,min_values_deaths and min_values_recovered are filtered out after the results are aggregated by year.This inquiry makes the annual cases with minimal values easier to understand.

```sql
-- Q9. Find maximum values of confirmed, deaths, recovered per year
  SELECT
     YEAR(DATE) AS Year,
     MAX(confirmed) AS Max_Confirmed,
     MAX(deaths) AS Max_Deaths,
     MAX(recovered) AS Max_Recovered
FROM
     corona_virus_dataset
GROUP BY
     YEAR(date);

```

corona_virus_dataset (2r × 4c)

Year	Max_Confirmed	Max_Deaths	Max_Recovered
2,020	823,225	3,752	1,123,456
2,021	414,188	7,374	422,436

Analysis:-

- In this SQL query, maximum values of confirmed , recovered and deaths cases are filtered out from the corona_virus_dataset table.
- The outcome are grouped by year, that is extracted from the Date column and then the Max_confirmed , Max_Deaths and Max_Recovered are showcase. This SQL query facilitated the understanding of the annual situations with maximum values.

```sql
-- Q10. The total number of case of confirmed, deaths, recovered each month
 SELECT
     DATE_FORMAT(DATE,"%Y-%m") AS Month,
     SUM(confirmed) AS Total_Confirmed,
     SUM(deaths) AS Total_Deaths,
     SUM(recovered) AS Total_Recovered
FROM corona_virus_dataset
GROUP BY
     DATE_FORMAT(DATE,"%Y-%m");
```

corona_virus_dataset (18r × 4c)

Month	Total_Confirmed	Total_Deaths	Total_Recovered
2020-01	6,384	190	143
2020-02	68,312	2,651	31,405
2020-03	769,236	41,346	133,070
2020-04	2,336,798	191,833	792,987
2020-05	2,744,333	144,561	1,519,547
2020-06	3,969,634	137,757	2,535,417
2020-07	6,838,092	167,613	4,693,120
2020-08	7,694,938	179,200	6,202,833
2020-09	8,244,794	160,671	6,647,749
2020-10	11,515,841	175,484	6,782,150
2020-11	16,595,938	262,247	9,172,292
2020-12	19,336,799	339,996	11,924,903
2021-01	18,672,205	401,893	9,164,347
2021-02	10,492,664	298,239	6,719,785
2021-03	13,924,790	282,620	7,888,013
2021-04	21,711,021	362,387	14,205,507
2021-05	19,121,083	366,549	19,131,842
2021-06	5,022,282	132,657	5,544,438

Analysis:-

- It utilizes the sum function on confirmed , deaths and recovered and it gives a total number of cases which is a good measure to compare the data.
- It then group the calculation by month that is extracted from date column, then it filter out as the Total_Confirmed , Total_Deaths and Total_Recovered cases . This query helps to understand the total number of cases that are present in the 18 months of sample period one-by-one.

```sql
-- 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 Average_Confirmed_Cases,
    ROUND(VARIANCE(confirmed),2) AS Confirmed_Cases_Variance,
    ROUND(SQRT(VARIANCE(confirmed)),2) AS Confirmed_Cases_Stdev
FROM
    corona_virus_dataset;
```

corona_virus_dataset (1r × 4c)

Total_Confirmed_Cases	Average_Confirmed_Cases	Confirmed_Cases_Variance	Confirmed_Cases_Stdev
169,065,144	2,156.83	157,288,925.08	12,541.49

Analysis:-

- The Total_confirmed_cases result from the SQL query , which also does statistical calculations such as average , variance and standard deviation . The standard deviation is computed using the SQRT function . We remove the Avg_confirmed_cases , Confirmed_cases_variance and confirmed_cases_stdev from the corona_virus_dataset as a result of this function. This given a clear understanding about the confirmed cases.

```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-%m") AS month,
    ROUND(SUM(deaths),2) AS Total_Confirmed_Deaths_cases,
    ROUND(AVG(deaths),2) AS Average_Deaths_Cases,
    ROUND(VARIANCE(deaths),2) AS Deaths_Cases_Variance,
    ROUND(SQRT(VARIANCE(deaths)),2) AS Deaths_Cases_Stdev
FROM corona_virus_dataset
GROUP BY
    DATE_FORMAT(DATE,"%Y-%m");
```

corona_virus_dataset (18r × 5c)

month	Total_Confirmed_Deaths_cases	Average_Deaths_Cases	Deaths_Cases_Variance	Deaths_Cases_Stdev
2020-01	190.0	0.12	4.25	2.06
2020-02	2,651.0	0.59	68.32	8.27
2020-03	41,346.0	8.66	3,900.79	62.46
2020-04	191,833.0	41.52	40,504.27	201.26
2020-05	144,561.0	30.28	20,684.91	143.82
2020-06	137,757.0	29.82	16,929.45	130.11
2020-07	167,613.0	35.11	21,140.15	145.4
2020-08	179,200.0	37.54	23,273.0	152.55
2020-09	160,671.0	34.78	20,102.77	141.78
2020-10	175,484.0	36.76	17,580.07	132.59
2020-11	262,247.0	56.76	27,773.79	166.65
2020-12	339,996.0	71.22	65,345.37	255.63
2021-01	401,893.0	84.18	102,758.43	320.56
2021-02	298,239.0	69.16	68,478.87	261.68
2021-03	282,620.0	59.2	54,385.97	233.21
2021-04	362,387.0	78.44	94,611.47	307.59
2021-05	366,549.0	76.78	131,769.47	363.0

Analysis:-

- By using the SUM() , AVG() , VARAINCE() and SQRT() function in the SQL query it will generate a quick overview of the deaths cases in the corona_virus_dataset table.
- The results are group by month , then Total_confirmed_death_cases , avg_deaths_cases , Deaths_cases_variance and the deaths_cases_Stdev will be filtered out. This SQL query will help to understand the deaths cases that are occurred in the 18 months of sample period.

```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_Confirmed_Recovered_Cases,
    ROUND(AVG(recovered),2) AS Average_Recovered_Cases,
    ROUND(VARIANCE(recovered),2) AS Recovered_Cases_Variance,
    ROUND(SQRT(VARIANCE(recovered)),2) AS Recovered_Cases_Stdev
FROM
    corona_virus_dataset;

```

corona_virus_dataset (1r × 4c)

Total_Confirmed_Recovered_Cases	Average_Recovered_Cases	Recovered_Cases_Variance	Recovered_Cases_Stdev
113,089,548	1,442.73	107,029,523.26	10,345.51

Analysis:-

- The Total_confirmed_Recovered_cases result from the SQL query , which also does statistical calculations such as average , variance and standard deviation . The standard deviation is computed using the SQRT function . We remove the Avg_Recovered_cases , Recovered_cases_variance and Recovered_cases_stdev from the corona_virus_dataset as a result of this function. This given a clear understanding about the recovered cases.

```sql
-- Q14. Find Country having highest number of the Confirmed case
SELECT province,country_region, confirmed
FROM corona_virus_dataset
ORDER BY confirmed DESC
LIMIT 1;
```

corona_virus_dataset (1r × 3c)

province	country_region	confirmed
Turkey	Turkey	823,225

Analysis:-

- It displays the country with highest number of confirmed cases in the corona_virus_dataset table. The table has has been arranged in descending order by confirmed column , and a limit of one has been applied to view only the first record of each verified column . Now, it shows Turkey as the country having the most confirmed cases of coronavirus.

```sql
-- Q15. Find Country having lowest number of the death case
SELECT province,country_region, deaths
FROM corona_virus_dataset
ORDER BY deaths ASC
LIMIT 1;
```

corona_virus_dataset (1r × 3c)

province	country_region	deaths
Afghanistan	Afghanistan	0

Analysis:-

- In the corona_virus_dataset table , it will diplay the country with the lowest number of deaths cases . The table has been arranged in descending order by the death column , and the limit of one has been applied to view only the first verified record in the death column . Now , it shows Afghanistan as the country that have lowest number of death cases.

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

SELECT province, country_region,recovered
FROM corona_virus_dataset
ORDER BY recovered DESC
LIMIT 5;
```

corona_virus_dataset (5r × 3c)

province	country_region	recovered
Turkey	Turkey	1,123,456
India	India	422,436
India	India	389,851
Brazil	Brazil	388,340
India	India	386,404

Analysis:-

- In the corona_virus_dataset table , it displays the Top 5 countries with the most recovered cases . The table has been arranged in descending order by recovered column , and a 5 limit has been applied to display only the first five records for each recovered column . The result shows the top 5 countries having the most recovered cases . Countries such as India , Brazil and Turkey have the most recovered cases.

SUMMARY

- In our coronavirus analysis project, we utilized SQL to examine spread of coronavirus all over the world. The project involved querying a database containing table for coronavirus cases and among other relevant data. To gather comprehensive insights, we employed SQL functions to collect information from the table, such as coronavirus cases with there respective records. These SQL queries enable a comprehensive perspective on cases that exist in different part of the world.
- We made use of sql functions and features to enhance our analysis. Aggregate functions like SUM(), AVG(), COUNT() helped us calculate statistics such as average confirmed & recovered cases, total number of cases present & the total count of observations.
- Additionally, we made use of other sql functions to enhance our analysis. Functions like MIN(),MAX() to get the country having maximum and minimum number of recovered and death cases. We also use function like VARIANCE(),SQRT() to help us calculate statistics such as variance of death and confirmed cases & standard deviation of deaths and recovered cases.
- Overall, our coronavirus analysis project leveraged sql's powerful capabilities including functions, constraint and other sql features to conduct a comprehensive evaluation of affected and recovered cases. Through careful analysis of the data and strategic application of SQL functions , we were able to derive valuable insights to inform decision-making and optimization strategies within the global environment.


Thank You!
alt