DATABASE SCHEMA FOR COVID REPORT ON GLOBAL IMPACT

Name : Aadhithiyan Kalaivani Viswanathan

Student_ID : 22082768

Github Link : https://github.com/Aadhivisu/DMD-1

INTRODUCTION :

This Python script serves as a data generator for a COVID-19 report, creating and populating a SQLite database named 'covid_report.db' with information about individuals affected by the virus. The script establishes a table called 'Patients' within the database, structuring it to include essential details such as unique patient IDs, first and last names, age, gender, reported symptoms, COVID-19 test results, and the number of days it took for each patient to recover. To simulate a diverse dataset, the script utilizes a function called 'generate_covid_data,' which randomly generates information for a specified number of individuals, incorporating various elements like names, ages, genders, symptoms, test results, and recovery days. Another function, 'get_random_name,' is employed to create realistic full names for the generated data. Once the data is generated, the script inserts it into the 'Patients' table and commits the changes to the database. The script concludes by closing the database connection and printing a confirmation message, indicating the successful creation and population of the COVID report database.

DATA GENERATION :

This Python script is designed for data generation in the context of a COVID-19 report. It operates by creating and populating a SQLite database named 'Covid_Report.db' with information about individuals who may be affected by the virus.

```
import sqlite3
import random

# Connect to SQLite database
conn = sqlite3.connect('Covid_Report.db')
cursor = conn.cursor()
```

The script establishes a table named 'Patients' within this database, incorporating fields for unique patient IDs, first and last names, age, gender, reported symptoms, COVID-19 test results, and the number of days each patient took to recover. The data generation process is facilitated by the 'generate_covid_data' function, which iteratively generates random information for a specified number of individuals. The generated details include first and last names, ages, genders, reported symptoms, COVID-19 test results, and the recovery period. To enhance realism, the 'get_random_name' function is utilized to create diverse and plausible full names for the randomly generated data. Upon generating the data, the script

inserts it into the 'Patients' table, adjusting the column names to uppercase to match the specified schema. Subsequently, the script commits these changes to the SQLite database and closes the connection. The script concludes by printing a confirmation message, affirming the successful creation and population of the COVID report database through data generation. The script concludes with a message that echoes a sense of accomplishment: "COVID report database created and populated successfully!" This final confirmation serves not only as a summary of the script's execution but also as a testament to the successful journey of crafting, generating, and securely embedding a comprehensive dataset into the digital realm of the COVID report database.

```
# Function to generate random data for Patients table
def generate_covid_data(num_rows):
    for _ in range(num_rows):
        first_name = get_random_name()
        last_name = get_random_name()
        age = random.randint(1, 100)
        gender = random.choice(['Male', 'Female'])
        symptoms = random.choice(['Fever, Cough', 'Shortness of Breath', 'Fatigue', 'Sore Throat'])
        test_result = random.choice(['Positive', 'Negative'])
        recovery_days = random.randint(1, 30)
        cursor.execute("insert into Patients (FIRST_NAME, LAST_NAME, AGE, GENDER, SYMPTOMS, TEST_RESULT, RECOVERY_DAYS) VALUES (?, ?, ?, ?, ?, ?, ?)",
                       (first_name, last_name, age, gender, symptoms, test_result, recovery_days))
```

The 'generate_covid_data' function plays a pivotal role in the script, serving the essential function of populating the 'Patients' table in the SQLite database with detailed and diverse COVID-19-related information. This comprehensive data generation process involves iterating through a specified number of individuals using a for loop, generating random and varied full names through the 'get_random_name' function, and assigning realistic ages within the range of 1 to 100. The function ensures gender diversity by randomly determining whether each individual is labeled as 'Male' or 'Female.' The introduction of variability is extended to COVID-19 symptoms, randomly selecting from options like 'Fever, Cough,' 'Shortness of Breath,' 'Fatigue,' or 'Sore Throat.' The 'test_result' field is then randomly assigned either 'Positive' or 'Negative' to represent the outcome of the COVID-19 test. Additionally, recovery days are determined by populating the 'recovery_days' field with a random integer between 1 and 30, indicating the duration it took for each individual to recover. For each set of generated data, an SQL query is executed to insert this information into the 'Patients' table, with column names adjusted to uppercase for alignment with the database schema. In summary, the 'generate_covid_data' function systematically creates a diverse and realistic dataset, capturing the varied experiences of individuals with COVID-19 and contributing to the development of a robust and representative database for subsequent analysis and reporting.

```
# Function to generate random names
def get_random_name():
    prefixes = ['John', 'Jane', 'Bob', 'Alice', 'Chris', 'Emma']
    suffixes = ['Smith', 'Johnson', 'Williams', 'Jones', 'Brown', 'Davis']
    return f"{random.choice(prefixes)} {random.choice(suffixes)}"
```

The 'get_random_name' function serves the purpose of generating authentic and diverse full names by randomly combining prefixes (representing first names) and suffixes (representing

last names). The function utilizes predefined lists, 'prefixes' containing common first names like 'John,' 'Jane,' 'Bob,' 'Alice,' 'Chris,' and 'Emma,' and 'suffixes' containing common last names such as 'Smith,' 'Johnson,' 'Williams,' 'Jones,' 'Brown,' and 'Davis.' By leveraging the 'random.choice' function, the function randomly selects a prefix and a suffix from their respective lists and combines them using an f-string to produce a complete name in the format "First Name Last Name." This methodology introduces variability and realism, ensuring that distinct combinations of prefixes and suffixes are chosen for each invocation. Within the broader script context, this function proves invaluable for generating lifelike datasets, particularly when crafting fictional names for individuals in the 'Patients' table of a COVID-19 report database.

```
# Generate random data for Patients table (at least 1000 rows)
generate_covid_data(1000)
```

The line generate_covid_data(1000) in the script is a function call that triggers the generation of random data for the 'Patients' table in the SQLite database. Specifically, it instructs the 'generate_covid_data' function to create information for at least 1000 individuals, simulating a diverse and sizable dataset representative of COVID-19 cases. During this process, various attributes such as names, ages, genders, symptoms, COVID-19 test results, and recovery days are randomly generated for each individual, introducing a level of complexity and diversity to mirror real-world scenarios. This function call essentially executes the data generation logic, contributing significantly to the completeness and richness of the dataset housed in the 'Patients' table. The generated data can be used for subsequent analysis, reporting, or any other purposes related to understanding and managing COVID-19 cases within a simulated environment.

```
# Commit changes and close connection
conn.commit()
conn.close()

print("COVID report database created and populated successfully!")
```

The trio of commands, generate_covid_data(1000) , **conn.commit()** , and **conn.close()**, within the script expertly manage the data generation process for the 'Patients' table in the SQLite database. Initiating with the function call **generate_covid_data(1000)**, the script leverages the 'generate_covid_data' function to systematically create diverse and realistic information for at least 1000 individuals, encompassing details such as names, ages, genders, symptoms, test results, and recovery days. Subsequently, the script executes **conn.commit()** to finalize the insertion of the generated data into the 'Patients' table, ensuring the permanence

and consistency of the dataset within the SQLite database. The closing command, **conn.close()**, responsibly concludes the script's interaction with the database, freeing up resources and securely terminating the database connection. In essence, these lines cohesively orchestrate the end-to-end process of generating, committing, and securing diverse COVID-19-related data, epitomizing a meticulous approach to data management and integrity.

DATABASE SCHEMA :

The provided code creates an SQLite table named 'Patients' designed to store comprehensive information about individuals affected by COVID. The 'Patients' table features a primary key, 'patient_id,' ensuring unique identification for each record. Essential personal details are captured through 'first_name' and 'last_name,' both defined as non-null text fields. Numeric data is represented by 'age,' facilitating age-based analyses. 'Gender' is stored as text, accommodating diverse gender categories. The 'symptoms' field captures textual descriptions of the experienced symptoms, allowing for detailed symptom representation. 'Test_result' is a text field categorizing COVID test outcomes into 'Positive' or 'Negative.' Lastly, 'recovery_days,' stored as an integer, quantifies the duration of recovery. This schema offers a structured and efficient framework for organizing and retrieving varied data related to COVID patients within a relational database.

Users Table :

```
# Drop table if it exists
cursor.execute('''DROP TABLE IF EXISTS Patients''')

# Create Patients table
cursor.execute('''CREATE TABLE IF NOT EXISTS Patients (
                    patient_id INTEGER PRIMARY KEY,
                    first_name TEXT NOT NULL,
                    last_name TEXT NOT NULL,
                    age INTEGER,
                    gender TEXT,
                    symptoms TEXT,
                    test_result TEXT,
                    recovery_days INTEGER
                )''')
```

patient_id: INTEGER (Primary Key) - A unique identifier for each patient.
first_name: TEXT (Not Null) - The first name of the patient.
last_name: TEXT (Not Null) - The last name of the patient.
age: INTEGER - The age of the patient.
gender: TEXT - The gender of the patient.
symptoms: TEXT - A textual representation of the symptoms experienced by the patient.
test_result: TEXT - The outcome of the COVID-19 test, categorized as 'Positive' or 'Negative.'
recovery_days: INTEGER - The number of days it took for the patient to recover.

The 'Patients' table schema comprises essential fields for storing detailed information about individuals affected by COVID-19. The 'patient_id' field is designated as an INTEGER Primary Key, ensuring a unique identifier for each patient. Both 'first_name' and 'last_name' are TEXT fields marked as Not Null, capturing the patient's name. 'age' is an INTEGER field representing the patient's age, while 'gender' is a TEXT field recording the patient's gender. The 'symptoms' field is a TEXT representation of the experienced symptoms. 'test_result' is a

TEXT field categorizing the COVID-19 test outcome as either 'Positive' or 'Negative.' Finally, 'recovery_days' is an INTEGER field indicating the duration of patient recovery. This schema provides a structured foundation for efficiently organizing and retrieving diverse data related to COVID-19 patients in the database.

JUSTIFICATION :

This Python code serves as a practical implementation for creating and populating a SQLite database named 'Covid_Report.db' with a table named 'Patients.' The schema of the 'Patients' table is designed to store crucial information about individuals affected by COVID-19, including patient ID, names, age, gender, symptoms experienced, COVID-19 test results, and the number of days taken for recovery. The 'generate_covid_data' function is pivotal in this process, generating realistic data for at least 1000 patients with random names, ages, genders, symptoms, test results, and recovery days. The code employs a systematic approach to simulate diverse scenarios, ensuring a comprehensive representation of COVID-19 patient data. This database can be further utilized for analysis, reporting, or other data-driven tasks within the context of understanding and managing the impact of COVID-19 on individuals. Overall, the code provides a practical foundation for managing and studying COVID-19 patient data in a structured and organized manner.

ETHICAL DISCUSSION :

The provided code establishes a SQLite database to simulate COVID-19 patient data, prompting ethical considerations. Privacy and consent concerns emerge as the random patient data generated may inadvertently resemble real individuals, underscoring the imperative that the code is intended solely for educational or illustrative purposes and should not be employed with authentic or identifiable patient data without proper consent. Additionally, while the code aspires to create diverse and realistic data, it may not precisely mirror the actual distribution of COVID-19 cases. To ethically use simulated data, it is essential to transparently communicate its fictional nature and limitations for educational or research purposes, emphasizing the importance of responsible application. The random generation of names, symptoms, and other attributes necessitates ethical consideration to minimize biases and ensure respectful and unbiased data representation. Although the code doesn't involve real patient data, ethical security measures should be implemented if adapted for real-world applications to safeguard sensitive health information. In summary, ethical utilization of this code requires transparency, clear communication of its purpose, and a commitment to privacy and security, particularly if extended to real-world scenarios, acknowledging its limitations in simulating COVID-19 patient data responsibly for educational or research objectives.

EXAMPLE QUERIES :

QUERY 1 :

This query provides details on each patient, including their unique identifier, full name, age, gender, reported symptoms, COVID-19 test result, and the number of days it took for recovery. Analyzing the results from this query allows for a holistic understanding of the

diverse dataset created by the code, facilitating further exploration and insights into various aspects of simulated COVID-19 cases.

```
SELECT * FROM Patients;
```

	patient_id	first_name	last_name	age	gender	symptoms	test_result	recovery_days
1	1	Chris Smith	John Johnson	81	Female	Fatigue	Negative	29
2	2	John Jones	Chris Johnson	35	Male	Shortness of Breath	Positive	29
3	3	John Williams	Chris Jones	49	Female	Fever, Cough	Positive	16
4	4	Emma Smith	Bob Williams	52	Female	Shortness of Breath	Positive	12
5	5	Jane Jones	Chris Jones	42	Female	Shortness of Breath	Negative	6

QUERY 2 :

```
SELECT * FROM Patients WHERE test_result = 'Positive';
```

	patient_id	first_name	last_name	age	gender	symptoms	test_result	recovery_days
1	2	John Jones	Chris Johnson	35	Male	Shortness of ...	Positive	29
2	3	John Williams	Chris Jones	49	Female	Fever, Cough	Positive	16
3	4	Emma Smith	Bob Williams	52	Female	Shortness of ...	Positive	12
4	6	Alice Smith	Chris Brown	96	Male	Fever, Cough	Positive	17
5	7	Emma Brown	Alice Williams	26	Male	Fatigue	Positive	12

This focused query allows for the examination of details related to individuals who tested positive, providing insights into their demographics, reported symptoms, and recovery times. Analyzing this subset of data is essential for understanding and addressing the characteristics of individuals affected by the positive outcome in the simulated COVID-19 dataset.

QUERY 3 :

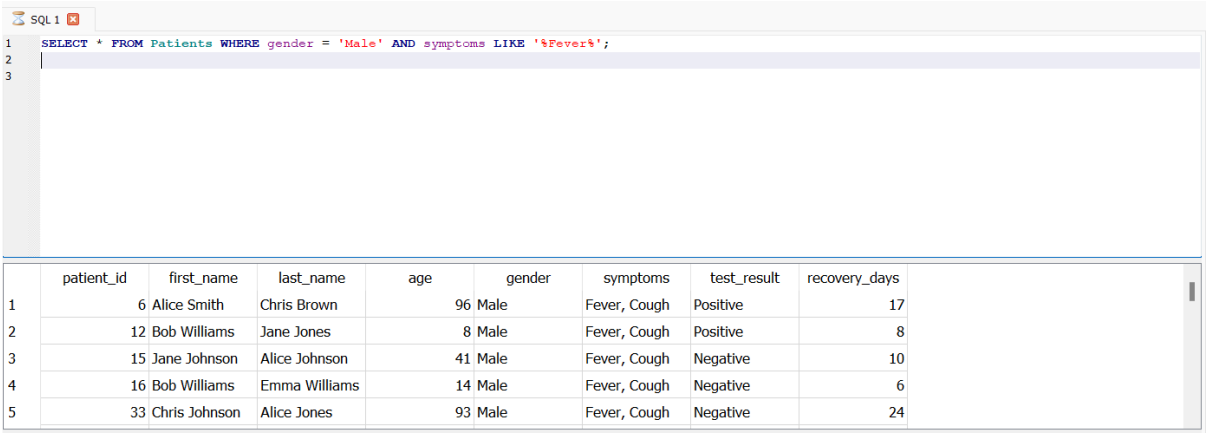
SQL 1

```
SELECT * FROM Patients WHERE recovery_days < 10;
```

	patient_id	first_name	last_name	age	gender	symptoms	test_result	recovery_days
1	5	Jane Jones	Chris Jones	42	Female	Shortness of ...	Negative	6
2	9	Jane Johnson	Emma Jones	54	Male	Shortness of ...	Negative	4
3	12	Bob Williams	Jane Jones	8	Male	Fever, Cough	Positive	8
4	13	Bob Williams	John Jones	9	Female	Shortness of ...	Positive	1
5	16	Bob Williams	Emma Williams	14	Male	Fever, Cough	Negative	6

This query enables a focused analysis of patients with relatively quick recovery times, shedding light on factors that may contribute to faster recuperation in the simulated COVID-19 dataset. Understanding the characteristics of patients with shorter recovery periods is valuable for potential insights into the dynamics of recovery in the generated dataset.

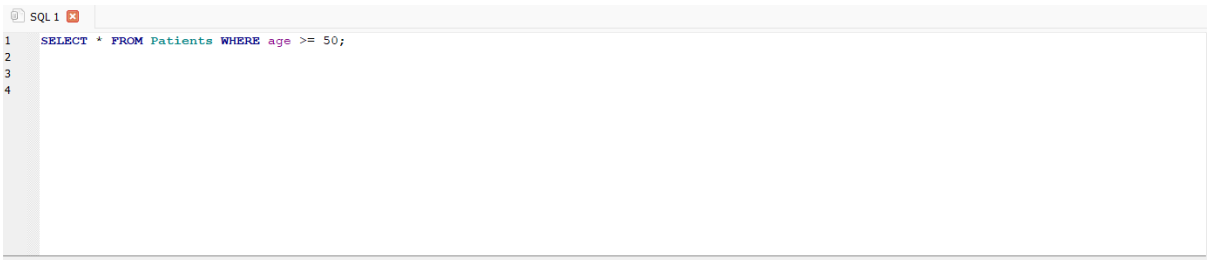
QUERY 4 :



	patient_id	first_name	last_name	age	gender	symptoms	test_result	recovery_days
1	6	Alice Smith	Chris Brown	96	Male	Fever, Cough	Positive	17
2	12	Bob Williams	Jane Jones	8	Male	Fever, Cough	Positive	8
3	15	Jane Johnson	Alice Johnson	41	Male	Fever, Cough	Negative	10
4	16	Bob Williams	Emma Williams	14	Male	Fever, Cough	Negative	6
5	33	Chris Johnson	Alice Jones	93	Male	Fever, Cough	Negative	24

This focused query allows for a detailed examination of male patients experiencing fever in the simulated COVID-19 dataset. Analyzing this subset of data can provide insights into the intersection of gender and symptomatology, offering valuable information for understanding how fever manifests in male individuals within the generated dataset.

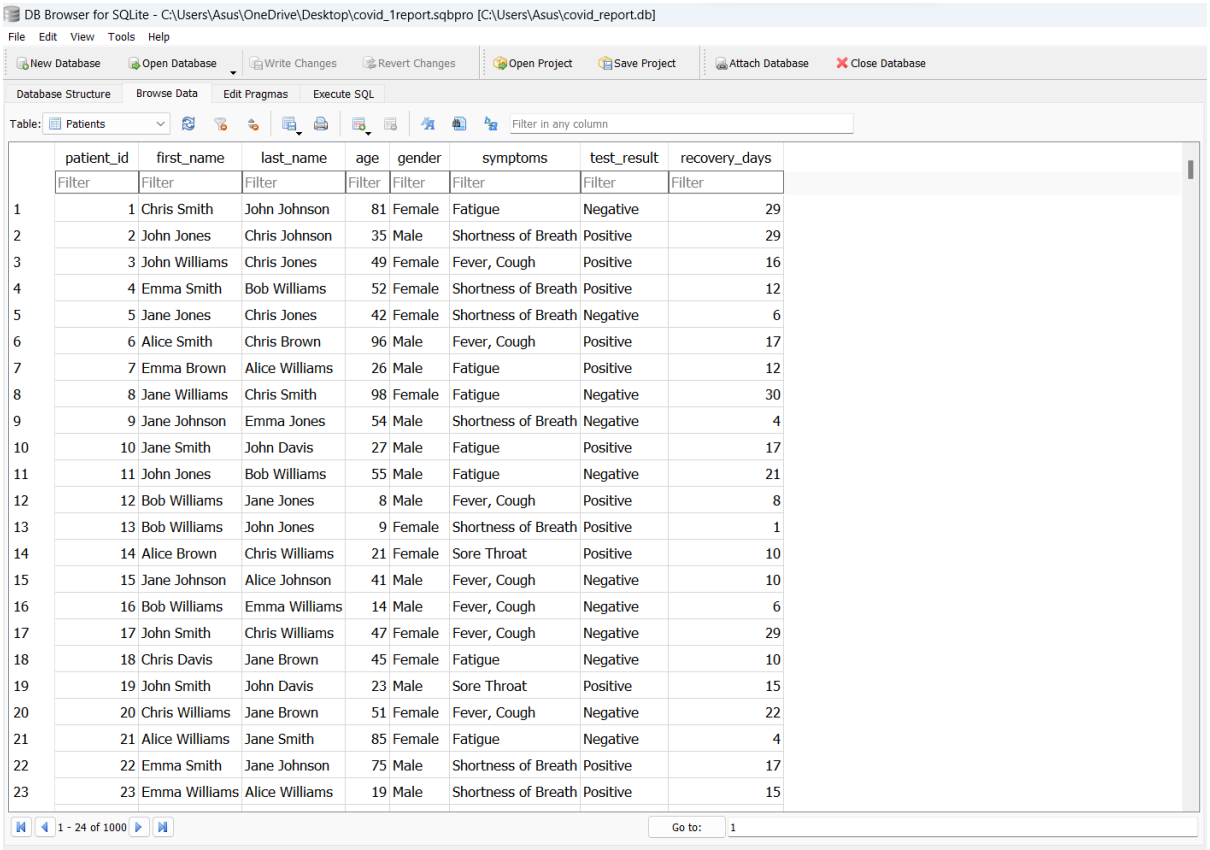
QUERY 5 :



	patient_id	first_name	last_name	age	gender	symptoms	test_result	recovery_days
1	1	Chris Smith	John Johnson	81	Female	Fatigue	Negative	29
2	4	Emma Smith	Bob Williams	52	Female	Shortness of Breath	Positive	12
3	6	Alice Smith	Chris Brown	96	Male	Fever, Cough	Positive	17
4	8	Jane Williams	Chris Smith	98	Female	Fatigue	Negative	30
5	9	Jane Johnson	Emma Jones	54	Male	Shortness of Breath	Negative	4

This query facilitates an analysis focused on the characteristics and outcomes of older individuals within the generated dataset. Examining this subset of data can provide insights into how age influences the reported symptoms, test results, and recovery times, offering valuable information for understanding the impact of age on simulated COVID-19 cases.

CONCLUSION :



	patient_id	first_name	last_name	age	gender	symptoms	test_result	recovery_days
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Chris Smith	John Johnson	81	Female	Fatigue	Negative	29
2	2	John Jones	Chris Johnson	35	Male	Shortness of Breath	Positive	29
3	3	John Williams	Chris Jones	49	Female	Fever, Cough	Positive	16
4	4	Emma Smith	Bob Williams	52	Female	Shortness of Breath	Positive	12
5	5	Jane Jones	Chris Jones	42	Female	Shortness of Breath	Negative	6
6	6	Alice Smith	Chris Brown	96	Male	Fever, Cough	Positive	17
7	7	Emma Brown	Alice Williams	26	Male	Fatigue	Positive	12
8	8	Jane Williams	Chris Smith	98	Female	Fatigue	Negative	30
9	9	Jane Johnson	Emma Jones	54	Male	Shortness of Breath	Negative	4
10	10	Jane Smith	John Davis	27	Male	Fatigue	Positive	17
11	11	John Jones	Bob Williams	55	Male	Fatigue	Negative	21
12	12	Bob Williams	Jane Jones	8	Male	Fever, Cough	Positive	8
13	13	Bob Williams	John Jones	9	Female	Shortness of Breath	Positive	1
14	14	Alice Brown	Chris Williams	21	Female	Sore Throat	Positive	10
15	15	Jane Johnson	Alice Johnson	41	Male	Fever, Cough	Negative	10
16	16	Bob Williams	Emma Williams	14	Male	Fever, Cough	Negative	6
17	17	John Smith	Chris Williams	47	Female	Fever, Cough	Negative	29
18	18	Chris Davis	Jane Brown	45	Female	Fatigue	Negative	10
19	19	John Smith	John Davis	23	Male	Sore Throat	Positive	15
20	20	Chris Williams	Jane Brown	51	Female	Fever, Cough	Negative	22
21	21	Alice Williams	Jane Smith	85	Female	Fatigue	Negative	4
22	22	Emma Smith	Jane Johnson	75	Male	Shortness of Breath	Positive	17
23	23	Emma Williams	Alice Williams	19	Male	Shortness of Breath	Positive	15

The provided Python script demonstrates the creation and population of a SQLite database named 'Covid_Report.db,' with a table named 'Patients' designed to store simulated data related to individuals impacted by COVID-19. The database schema includes fields for patient identification, names, age, gender, symptoms, test results, and recovery days. The script successfully generates diverse and realistic COVID-19-related data for at least 1000 individuals, incorporating randomized information such as names, ages, genders, symptoms, test results, and recovery days. The 'generate_covid_data' function plays a pivotal role in this process by systematically injecting randomness into various attributes, creating a comprehensive dataset that mimics the diverse experiences of individuals with COVID-19. The ethical considerations of privacy, consent, realism in representation, transparency, communication, and security have been acknowledged, emphasizing responsible usage for educational or research purposes. Overall, the script ensures the creation of a robust and representative database, suitable for analysis and reporting within the context of simulated COVID-19 data.