Name: Xinyu Wu

NetID: xwu82

UIN: 661348655

Final Project Report

Summary

More than 100 years ago, in 1918, the Spanish flu swept the world, causing pain that is unforgettable. Today, more than 100 years later, COVID-19 became another challenge for human survival. This application offers several queries that many people are interested in, and the results will be performed as a table.

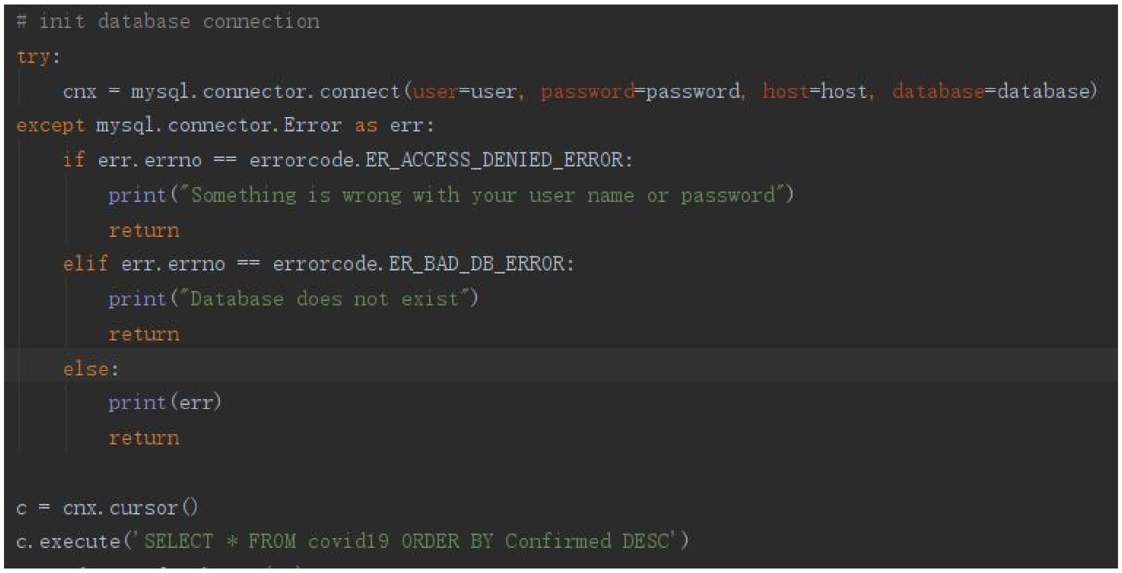
Data Preparation

The original data file can be found here. The original data has columns such as "Active", "New cases", "Confirmed last week", and "New Death" columns, since these columns are about the recent changes, but my queries are concentrate on the overall data all over the world, so I dropped these columns before I imported them to MySQL Workbench.

Materials and Methods

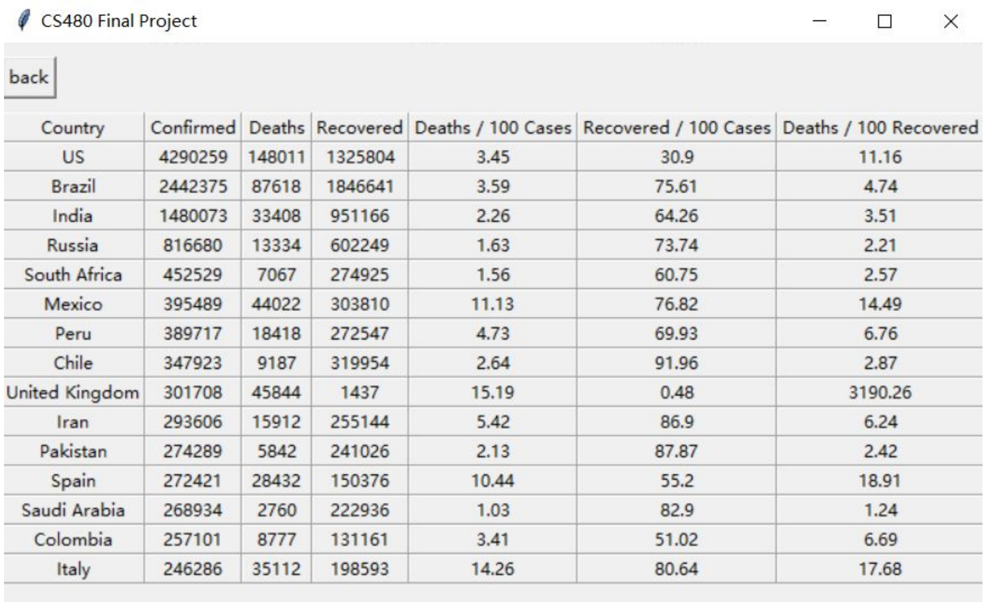
At the main scene, there are numbers of buttons that can be pressed, each button has their function as the description on the button. Once the button is pressed, the second scene is showing the table indicating the result based on the database, and the user can press the back button back to the first scene. Then press any other buttons to see the different result tables.

When I made the connection between application and server, I used the following code, it requires the mysql-connector-python package to make the connection. If the connection cannot be made, the terminal should display the error information.



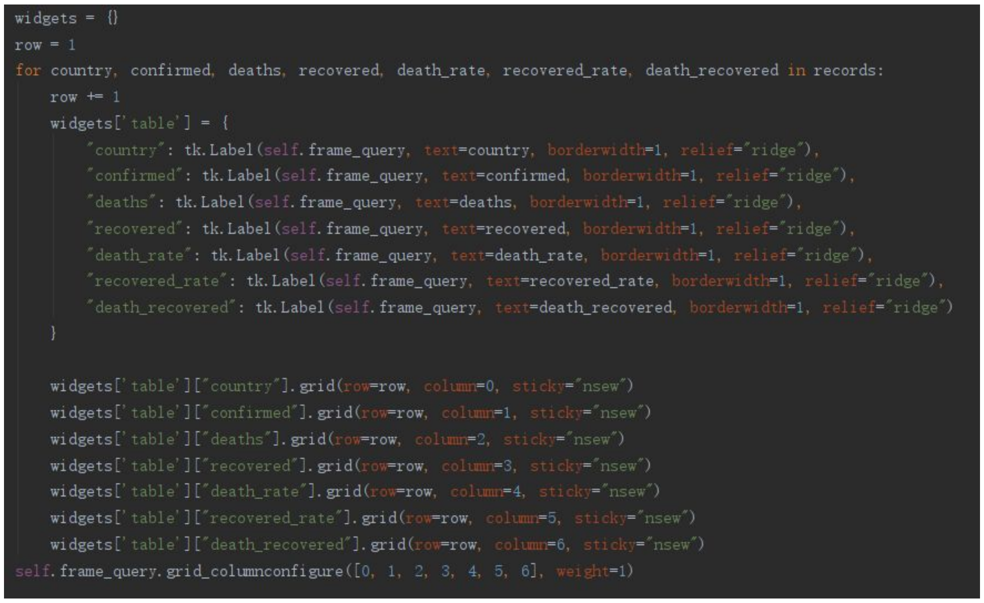
```
# init database connection
try:
    cnx = mysql.connector.connect(user=user, password=password, host=host, database=database)
except mysql.connector.Error as err:
    if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
        print("Something is wrong with your user name or password")
        return
    elif err.errno == errorcode.ER_BAD_DB_ERROR:
        print("Database does not exist")
        return
    else:
        print(err)
        return

c = cnx.cursor()
c.execute('SELECT * FROM covid19 ORDER BY Confirmed DESC')
```

The first button displays the table with all columns and the top 15 rows, and I only display the 15 rows that order by the most confirmed cases.

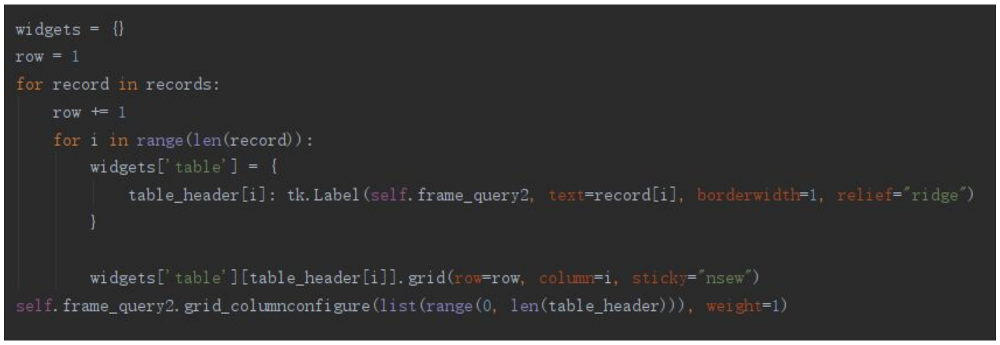


Country	Confirmed	Deaths	Recovered	Deaths / 100 Cases	Recovered / 100 Cases	Deaths / 100 Recovered
US	4290259	148011	1325804	3.45	30.9	11.16
Brazil	2442375	87618	1846641	3.59	75.61	4.74
India	1480073	33408	951166	2.26	64.26	3.51
Russia	816680	13334	602249	1.63	73.74	2.21
South Africa	452529	7067	274925	1.56	60.75	2.57
Mexico	395489	44022	303810	11.13	76.82	14.49
Peru	389717	18418	272547	4.73	69.93	6.76
Chile	347923	9187	319954	2.64	91.96	2.87
United Kingdom	301708	45844	1437	15.19	0.48	3190.26
Iran	293606	15912	255144	5.42	86.9	6.24
Pakistan	274289	5842	241026	2.13	87.87	2.42
Spain	272421	28432	150376	10.44	55.2	18.91
Saudi Arabia	268934	2760	222936	1.03	82.9	1.24
Colombia	257101	8777	131161	3.41	51.02	6.69
Italy	246286	35112	198593	14.26	80.64	17.68

I had never created an application with python before, so the application GUI was very challenging for me to complete this project. At first, I hard coded the table UI for the display overall database part, then I realized I need to hardcode every table for every different query, I could just copy paste and change a few variables every time, but it was so inefficient and low readability for others. Finally I optimized the code and made the table dynamically change by query result. The picture below shows the before optimization and after optimization.



```
widgets = {}
row = 1
for country, confirmed, deaths, recovered, death_rate, recovered_rate, death_recovered in records:
    row += 1
    widgets['table'] = {
        "country": tk.Label(self.frame_query, text=country, borderwidth=1, relief="ridge"),
        "confirmed": tk.Label(self.frame_query, text=confirmed, borderwidth=1, relief="ridge"),
        "deaths": tk.Label(self.frame_query, text=deaths, borderwidth=1, relief="ridge"),
        "recovered": tk.Label(self.frame_query, text=recovered, borderwidth=1, relief="ridge"),
        "death_rate": tk.Label(self.frame_query, text=death_rate, borderwidth=1, relief="ridge"),
        "recovered_rate": tk.Label(self.frame_query, text=recovered_rate, borderwidth=1, relief="ridge"),
        "death_recovered": tk.Label(self.frame_query, text=death_recovered, borderwidth=1, relief="ridge")
    }

    widgets['table']["country"].grid(row=row, column=0, sticky="nsew")
    widgets['table']["confirmed"].grid(row=row, column=1, sticky="nsew")
    widgets['table']["deaths"].grid(row=row, column=2, sticky="nsew")
    widgets['table']["recovered"].grid(row=row, column=3, sticky="nsew")
    widgets['table']["death_rate"].grid(row=row, column=4, sticky="nsew")
    widgets['table']["recovered_rate"].grid(row=row, column=5, sticky="nsew")
    widgets['table']["death_recovered"].grid(row=row, column=6, sticky="nsew")
self.frame_query.grid_columnconfigure([0, 1, 2, 3, 4, 5, 6], weight=1)
```



```
widgets = {}
row = 1
for record in records:
    row += 1
    for i in range(len(record)):
        widgets['table'] = {
            table_header[i]: tk.Label(self.frame_query2, text=record[i], borderwidth=1, relief="ridge")
        }

        widgets['table'][table_header[i]].grid(row=row, column=i, sticky="nsew")
self.frame_query2.grid_columnconfigure(list(range(0, len(table_header))), weight=1)
```