COMP421 Database System
Group Project 3

Question 1.

a. Previous price of vaccination

```
SELECT price, vname
FROM vaccination
ORDER BY vname asc;
```

	vname character varying (255)	price double precision
1	COVID-19	29
2	COVID-19	50
3	COVID-19	68
4	MERS-CoV	7
5	MERS-CoV	7
6	MERS-CoV	18
7	MERS-CoV	18
8	MERS-CoV	73
9	MERS-CoV	400
10	SARS-CoV	10
11	SARS-CoV	10
12	SARS-CoV	15
13	SARS-CoV	15
14	SARS-CoV	20
15	SARS-CoV	27
16	SARS-CoV	120
17	Zika virus	20
18	Zika virus	20
19	Zika virus	23
20	Zika virus	28
21	Zika virus	35
22	Zika virus	57

Figure 1. Screenshot of before

b. Use a stored procedure to update the price with respect to a given ratio for each vaccination of a certain virus according to the total patient number (aggregated count) infected with that virus.

Idea: more expensive with more people infected

```
create FUNCTION update_vaccination_price(base int)
returns void AS $$
declare
   curs1 cursor for select vname, count(*) as total from
infect
   group by vname;
   curs2 cursor for select vname, price from vaccination;
   begin
   <<outerloop>>
   for r1 in curs1 LOOP
      <<innerloop>>
      for r2 in curs2 LOOP
         if r1.vname = r2.vname then
             update vaccination
         set price = (r1.total*1. / base*1.) * price + price
                     where current of curs2;
         END IF;
      end loop innerloop;
   end loop outerloop;
   end;
   $$ language 'plpgsql'
```

c. After update (base = 20)

```
SELECT v1.vname , price
FROM vaccination v1
left outer JOIN infect_count ON
v1.vname = infect_count.vname
ORDER BY vname ASC;
```

	vname character varying (255)	price double precision
1	COVID-19	46.4
2	COVID-19	80
3	COVID-19	108.8
4	MERS-CoV	9.45
5	MERS-CoV	9.45
6	MERS-CoV	24.3
7	MERS-CoV	24.3
8	MERS-CoV	98.55
9	MERS-CoV	540
10	SARS-CoV	12.5
11	SARS-CoV	12.5
12	SARS-CoV	18.75
13	SARS-CoV	18.75
14	SARS-CoV	25
15	SARS-CoV	33.75
16	SARS-CoV	150
17	Zika virus	30
18	Zika virus	30
19	Zika virus	34.5
20	Zika virus	42
21	Zika virus	52.5
22	Zika virus	85.5

Figure 2. Screenshot after update

Question 2.

The programme (Q2.java and Helpers.java) and a script showing the program running are in folder ‘Q2’.

There are Six options in the main menu (one is Quit). By inputting the number before the name of the options, the user can enter the sub-menu and fulfil the requirements by following instructions.

0. Quit

1. Query : P (Virus | Symptom)

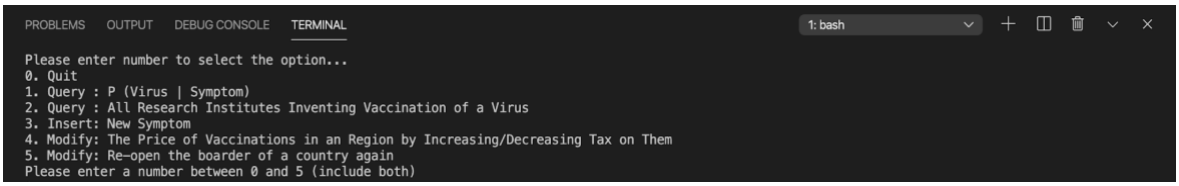
2. Query : All Research Institutes Inventing Vaccination of a Virus

3. Insert : New Symptom

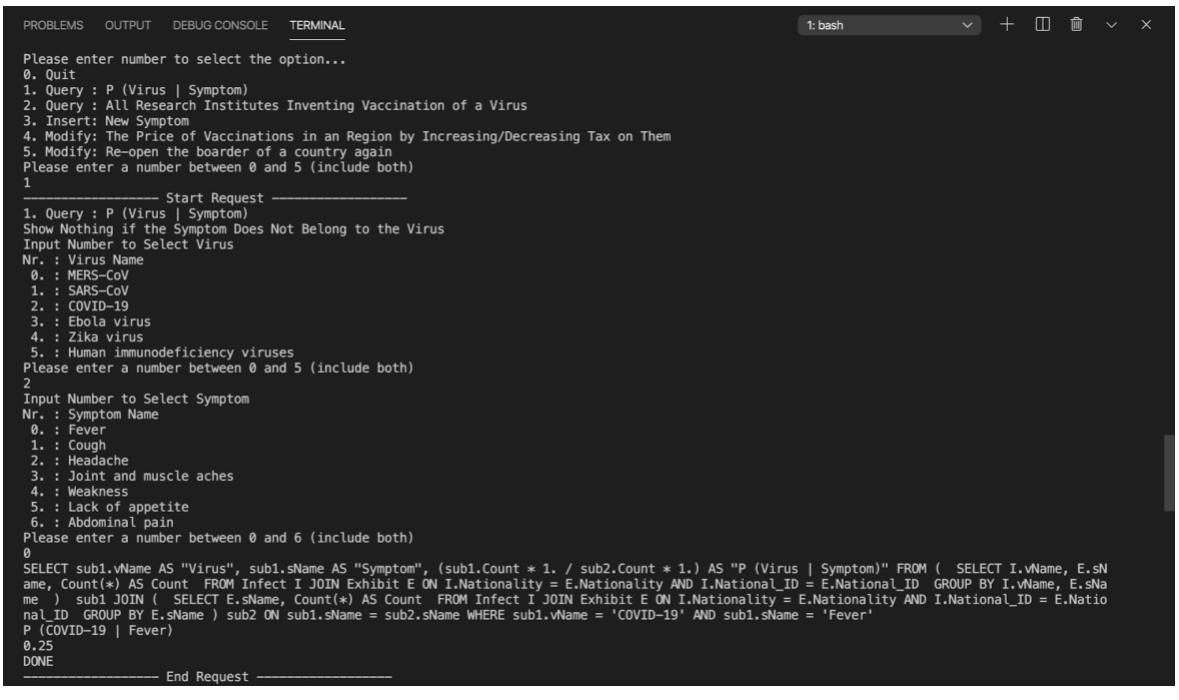
4. Modify: The Price of Vaccinations in an Region by Increasing/Decreasing Tax on Them

5. Modify: Re-open the border of a country again

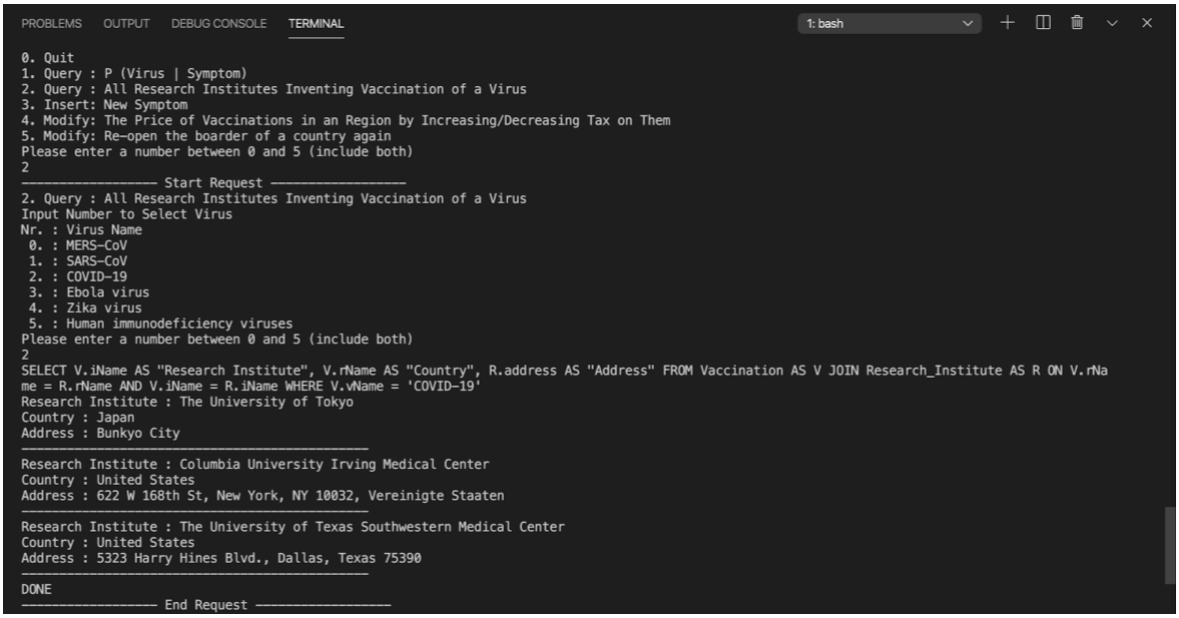
Below are screenshots of the programme executing every option



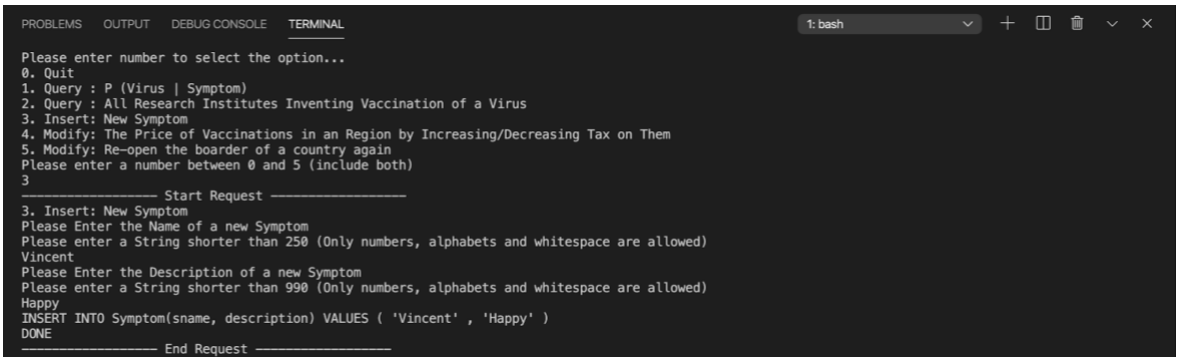
Main Menu



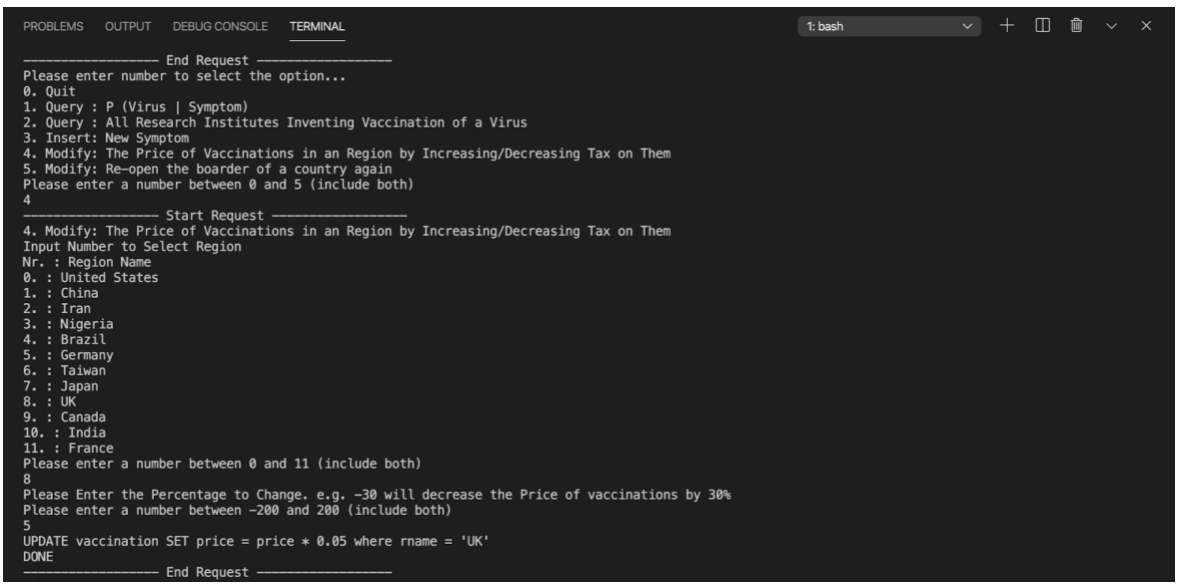
Query - P(Virus | Symptom)



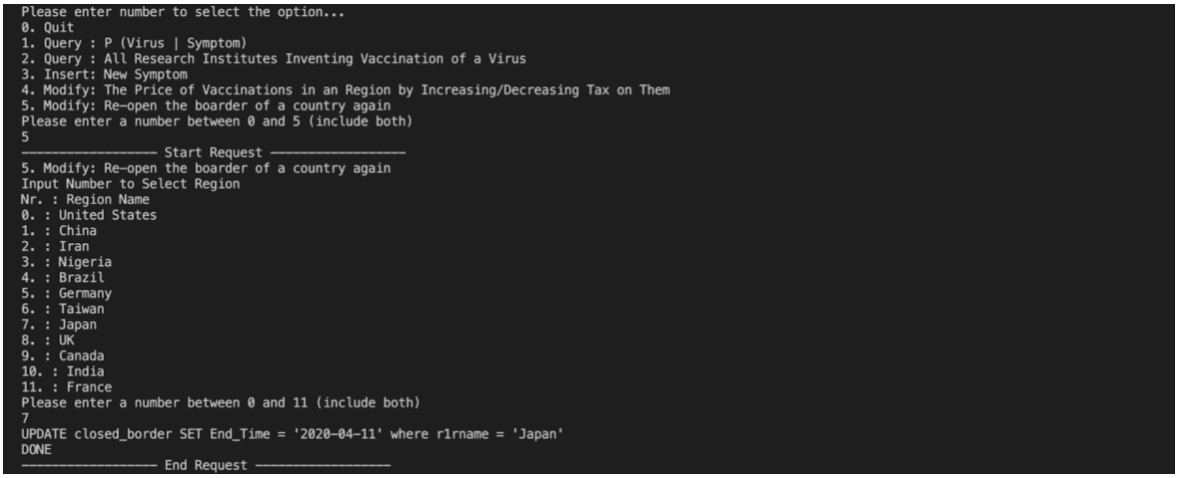
Query - List Inventers of a Vaccination



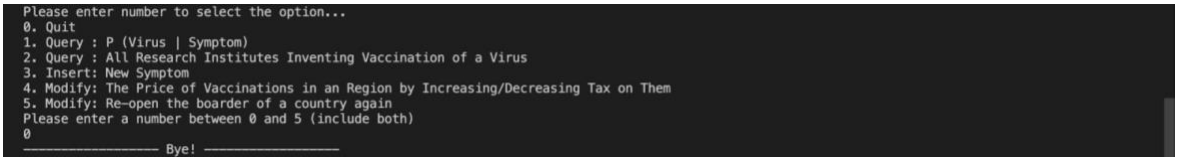
Insert - New Symptom



Modify - Change the Price of Vaccinations



Modify - Open Boarder



Quit

Question 3.

1. **CREATE INDEX idx_end_time ON Closed_Border (End_Time);**

   When the situation of virus is under controlled in some region, it's border will probably be opened. The end time of closing border can reflect the control process of the virus. The index idx_end_time can improve the query performance when the query has constraints about End_Time.

   To get the top five date when most borders were reopened:

   ```
   SELECT C.End_Time, COUNT(*) AS Count
   FROM Closed_Border C
   GROUP BY C.End_Time
   ORDER BY Count DSEC
   LIMIT 5;
   ```

   By maintaining a sorted order of End_Time, the above query would execute quicker.

2. **CREATE INDEX idx_price ON Vaccination (Currency, Price);**

When epidemics or pandemics outbreaks, many institutions will do research and development of the vaccine. The price of vaccine will affect people's choice of vaccine. To get the rNami, iName, Code, Currency, Price of the cheapest index for each currency, the following query will be used.

```
SELECT Va.rName, Va.iName, Va.Code, Va.Currency, Va.Price
From Vaccination Va JOIN
(
   SELECT V.Currency, MIN (V.Price) AS minPrice
   FROM Vaccination V
   GROUP BY V.Currency
)
WHERE Va.Currency = V.currency AND Va.Price = minPrice;
```

By maintaining a sorted order of (Currency, Price), the above query would execute quicker.

Question 4.

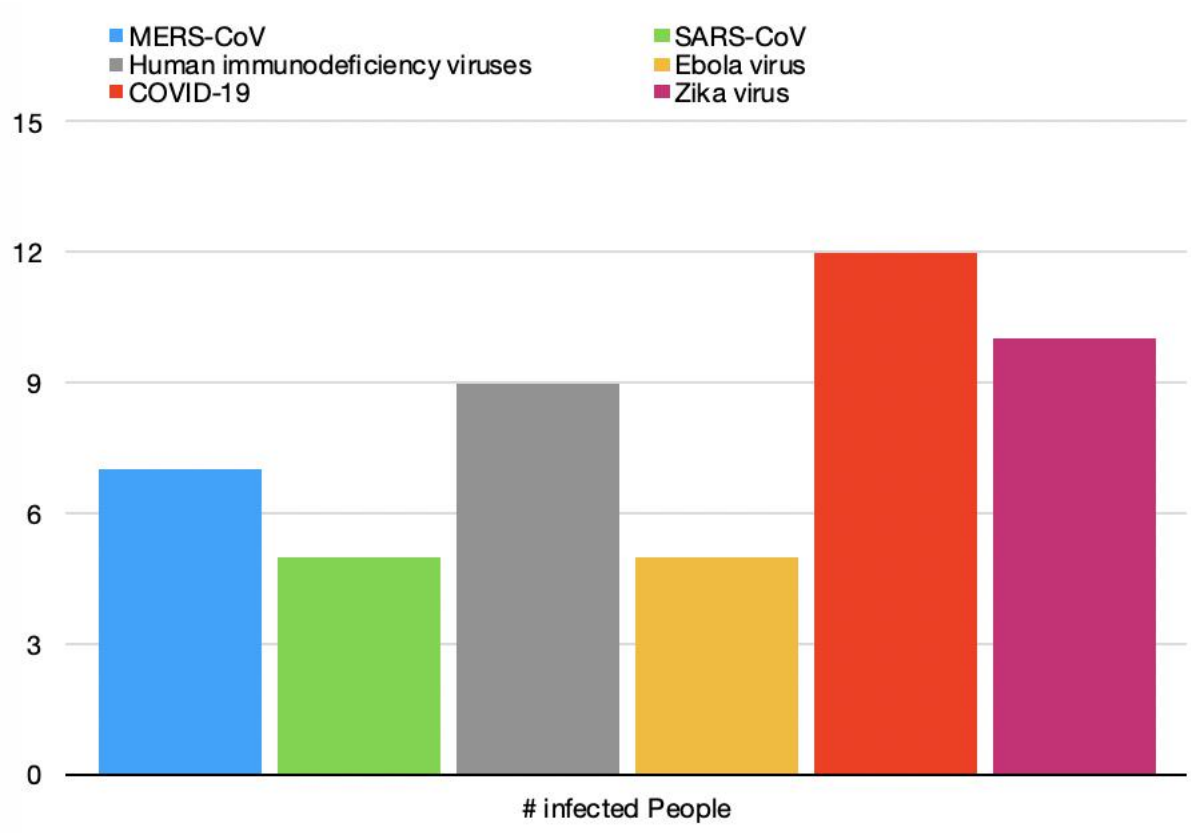


Figure 3. The number of infected people per virus

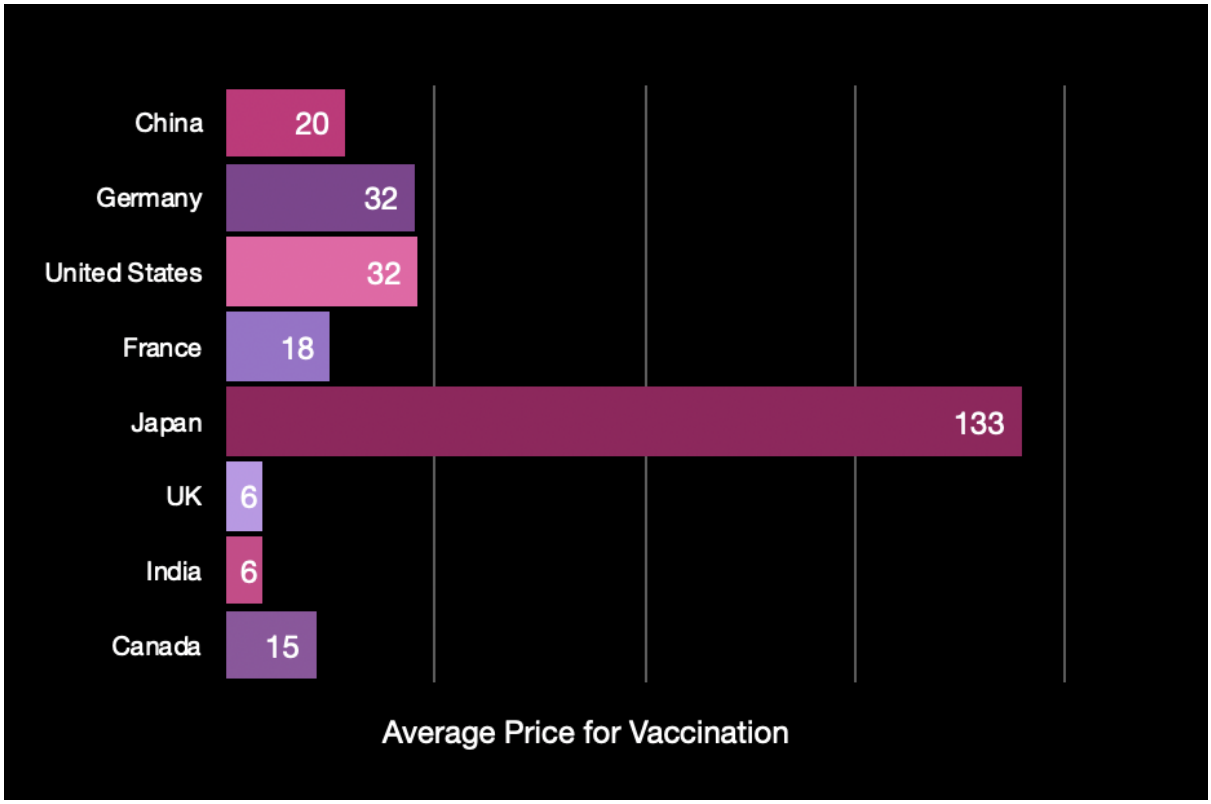


Figure 4. The average price for vaccination for different countries

The (i) SQL used to generate the data, (ii) The Excel / Google spreadsheet you did the work on are in the folder Q4

Question 5.

SQL Stored Procedure

```
CREATE OR REPLACE FUNCTION hospital_analysis(
  virusNm VARCHAR,
  hosNm VARCHAR,
  OUT total_patient INT,
  OUT recovered_p INT,
  OUT avg_time_month REAL,
  OUT not_recovered_P INT  ) AS $$
DECLARE
  rec         RECORD;
  ctrnotHeal  INT := 0;
  ctrAll      INT := 0;
  ctrHeal     INT := 0;
  avg_time    REAL:= 0.;
  cursor1     CURSOR(virusNm VARCHAR, hosNm VARCHAR)
              FOR SELECT acc.begin_time AS bt, acc.end_time AS et
                  FROM accommodation acc, infect i
                  WHERE i.vname = virusNm
                  AND i.nationality = acc.nationality
                  AND i.national_id = acc.national_id
                  AND acc.iname = hosNm;
BEGIN
  OPEN cursor1(virusNm, hosNm);
```

```
  LOOP
  FETCH cursor1 INTO rec;
  EXIT WHEN NOT FOUND;
  IF rec.et IS NULL THEN
      ctrnotHeal := ctrnotHeal + 1;
  END IF;
  IF rec.et IS NOT NULL THEN
     ctrHeal := ctrHeal + 1;
     avg_time := avg_time + (DATE_PART('year', rec.et::date) -
                  DATE_PART('year', rec.bt::date)) * 12 +
                  (DATE_PART('month', rec.et::date) -
                  DATE_PART('month', rec.bt::date));
  END IF;
  ctrAll := ctrAll + 1;
  END LOOP;
  CLOSE cursor1;
  IF ctrHeal <> 0 THEN
     avg_time := (avg_time * 1.0) / (ctrAll * 1.0);
     END IF;
     total_patient := ctrAll;
     recovered_p := ctrHeal;
     avg_time_month := avg_time;
     not_recovered_P := ctrnotHeal;
END;
$$ LANGUAGE 'plpgsql';
```

Output	Running Query
hospital_analysis record (23,11,1.43478,11)	`SELECT hospital_analysis('COVID-19', 'Winsen Hospital Germany');`
hospital_analysis record (18,13,2.16667,13)	`SELECT hospital_analysis('SARS-CoV', 'Winsen Hospital Germany');`
hospital_analysis record (1,1,16,1)	`SELECT hospital_analysis('COVID-19', 'Florida Hospital Orlando');`
hospital_analysis record (9,2,3.88889,2)	`SELECT hospital_analysis('COVID-19', 'Ruijin Hospital');`