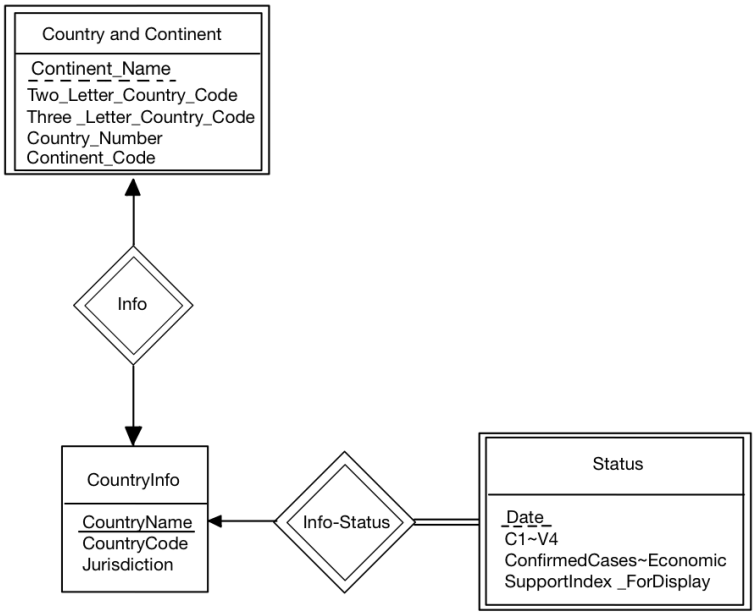
HW2_109550017_黃品云

Q1. ER diagram with entity sets and relationship sets, with or without attributes. Add constraints if needed. (30pts) (if it is hard to include your ER diagram in the .pdf file, you can submit the diagram separately)

Ans:



Including entity sets and relationship sets, with or without attributes

- **Country and Continent** entity comes from country-and-continent-codes-list-csv.csv. Containing the information of each country and corresponding continent, Country and Continent is a weak entity set and is linked to CountryInfo by a identifying relationship set called **Info.**
- **CountryInfo** and **Status** entities comes from the OxCGRT_nat_latest.csv. CountryInfo cantains the basic information about a country, like countryName and countryCode. Status is a weak entities contains the information of various aspects during Covid, and they’re linked by a identifying relationship set called **Info-status.**

Add constraints for the model

- Country — info — CountryInfo: info is a one-to-one relationship.

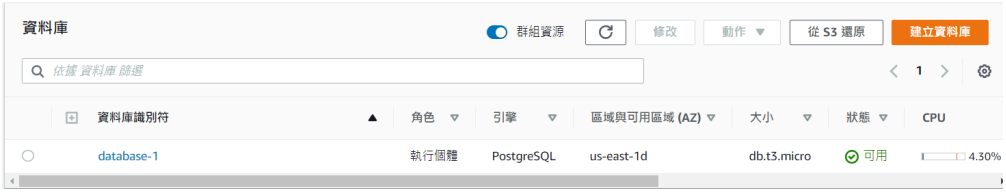
- CountryInfo — info-status — status: info-status is a one-to-many and total on the many side relationship.
- Primary keys can not be NULL.

> 📌 To keep the name consistent throughout the databases, I modified the country name in **“country-and-continent-codes-list-csv.csv”.** I also deleted **RegionName, RegionCode** and **M1_Wildcard** because they only have NULL value throughout the whole table.

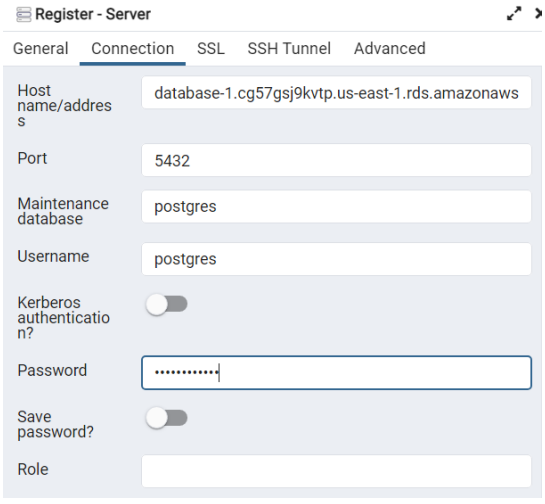
Q2. Provide print screens of the 1) AWS RDS lunch page, and 2) the way you connect to the AWS RDS (PostgreSQL console tool, pgAdmin, or other IDE’s connection page, with the same IP or URL with your AWS RDS) (10pts)

Ans:

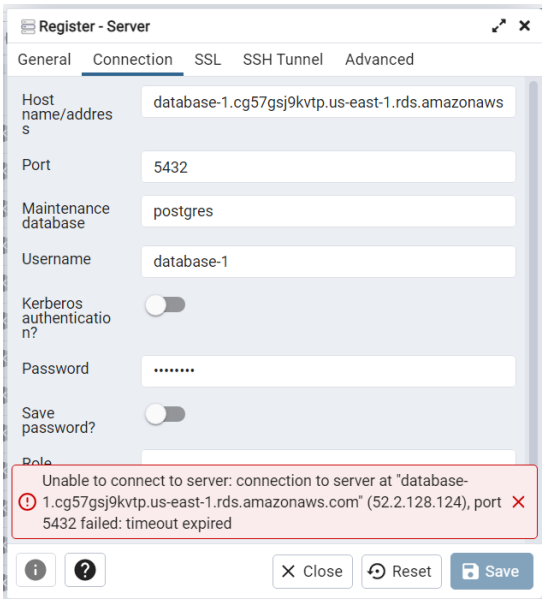
I followed the steps in AWS DB and successfully created the following database.



I entered my DB instance endpoint and password in pgAdmin to register the server on AWS.



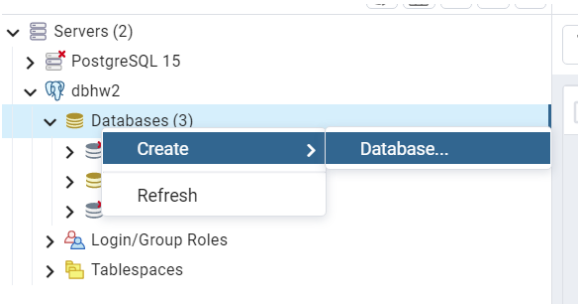
However, when I was trying to save the settings, the following error keeps showing up. To solve the problem, I checked QAs on AWS DB, and found that there were exactly the same problem that I were having, and it turned out I need to modify the ip configuration in inbound.

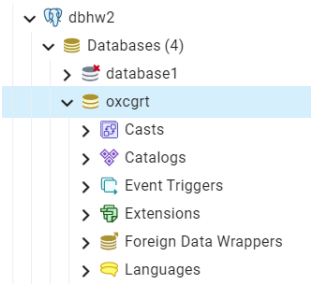


I followed the instructions on stackoverflow, and changed the source to “my IP”.

After editting the IP configuration, it worked! I successfully connected pgAdmin to AWS.

Finally, I created a database called “oxcgrt”.



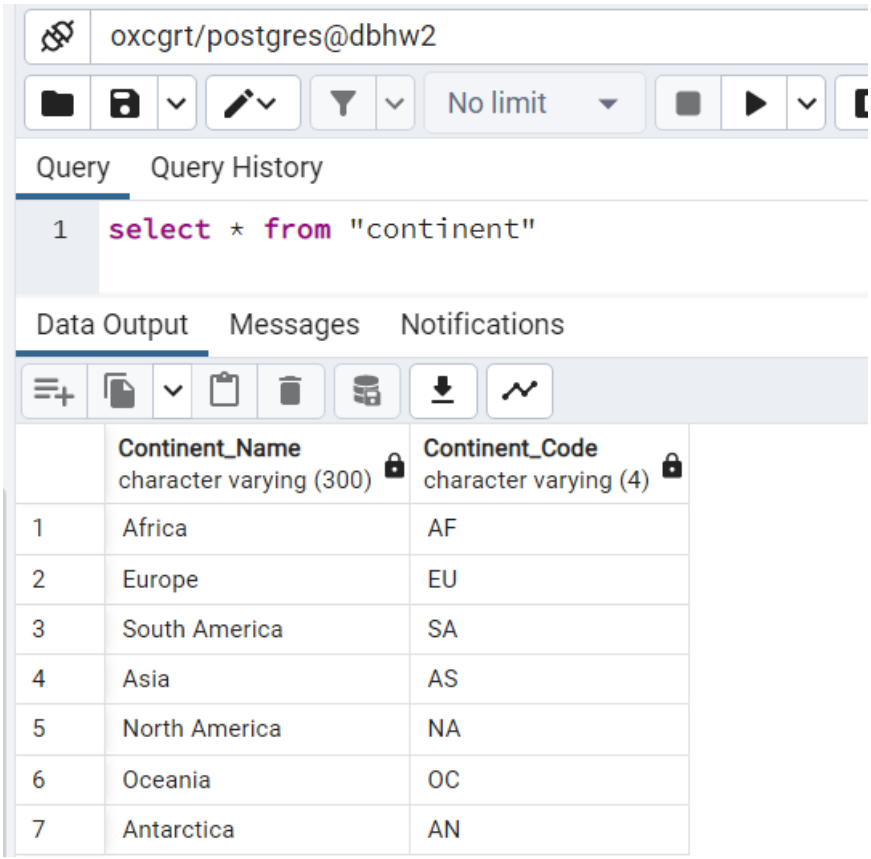


3. Please provide the schema after decomposition, of each table, and a print screen to show that the tables have been created in your database on AWS RDS. (10+10pts)

Ans:

1. continent(Continent_Name, Continent_Code)
   a. table is created due to normalization(remove partial dependencies)
   b. Continent_Name is primary key.

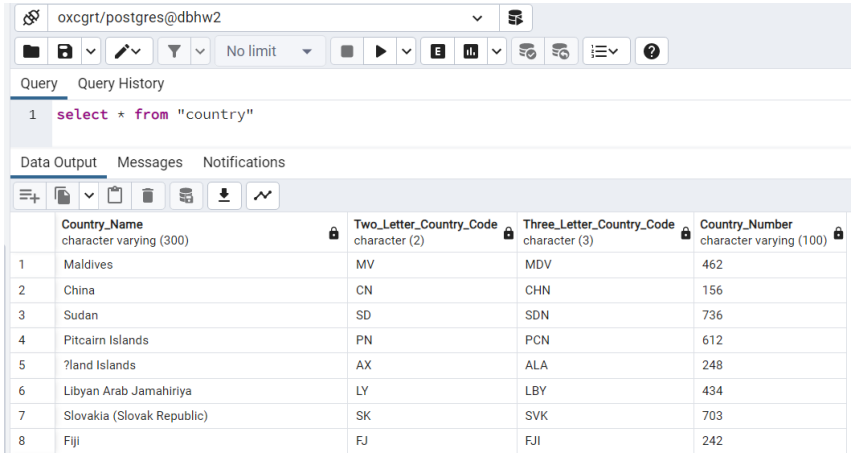
```
1  create table continent as (
2      select distinct "Continent_Name", "Continent_Code"
3      from "country-and-continent-codes-list-csv"
4  );
```



	Continent_Name character varying (300)	Continent_Code character varying (4)
1	Africa	AF
2	Europe	EU
3	South America	SA
4	Asia	AS
5	North America	NA
6	Oceania	OC
7	Antarctica	AN

2. country(CountryName, Two_Letter_Country_Code, Three_Letter_Country_Code, Country_Number, Continent_Name).
   a. info is an identifying relationship set, so the schema corresponding to it is redundant.
   b. CountryName and Continent_Name are primary key.

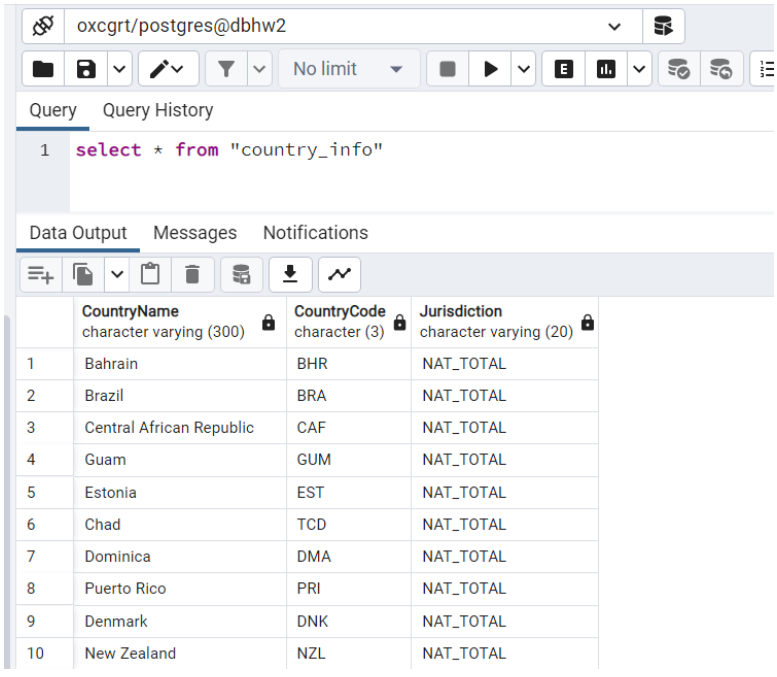
```sql
create table country as(
    select distinct "CountryName", "Two_Letter_Country_Code",
                    "Three_Letter_Country_Code", "Country_Number"
    from "country-and-continent-codes-list-csv"
)
```



	Country_Name character varying (300)	Two_Letter_Country_Code character (2)	Three_Letter_Country_Code character (3)	Country_Number character varying (100)
1	Maldives	MV	MDV	462
2	China	CN	CHN	156
3	Sudan	SD	SDN	736
4	Pitcairn Islands	PN	PCN	612
5	?land Islands	AX	ALA	248
6	Libyan Arab Jamahiriya	LY	LBY	434
7	Slovakia (Slovak Republic)	SK	SVK	703
8	Fiji	FJ	FJI	242

3. country_info(CountryName, CountryCode, Jurisdiction)
    a. CountryName is primary key.

```sql
create table country_info as (
    select distinct "CountryName", "CountryCode", "Jurisdiction"
    from "OxCGRT_nat_latest"
);
```



	CountryName character varying (300)	CountryCode character (3)	Jurisdiction character varying (20)
1	Bahrain	BHR	NAT_TOTAL
2	Brazil	BRA	NAT_TOTAL
3	Central African Republic	CAF	NAT_TOTAL
4	Guam	GUM	NAT_TOTAL
5	Estonia	EST	NAT_TOTAL
6	Chad	TCD	NAT_TOTAL
7	Dominica	DMA	NAT_TOTAL
8	Puerto Rico	PRI	NAT_TOTAL
9	Denmark	DNK	NAT_TOTAL
10	New Zealand	NZL	NAT_TOTAL

4. status(CountryName, Date, C1, …, V4, ConfirmedCases, …, EconomicSupportIndex_ForDisplay)
    a. CountryName, Date are primary keys.
    b. status is **without flags.**

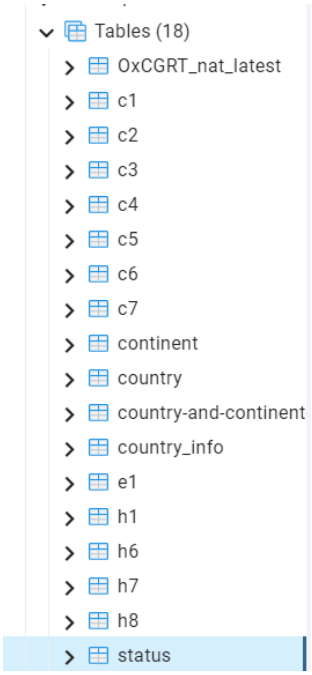
c. CountryName is in the schema because info-status is a one to many relationship, and can be represented by adding an primary key to the “many” side.

```
create table status as (
    select distinct "C1M_School closing", "C2M_Workplace closing",
    "C3M_Cancel public events", "C4M_Restrictions on gatherings",
    "C5M_Close public transport", "C6M_Stay at home requirements",
    "C7M_Restrictions on internal movement", "C8EV_International travel controls",
    "E1_Income support", "E2_Debt/contract relief",
    "E3_Fiscal measures", "E4_International support",
    "H1_Public information campaigns", "H2_Testing policy",
    "H3_Contact tracing", "H4_Emergency investment in healthcare",
    "H5_Investment in vaccines", "H6M_Facial Coverings",
    "H7_Vaccination policy", "H8M_Protection of elderly people",
    "V1_Vaccine Prioritisation (summary)", "V2A_Vaccine Availability (summary)",
    "V2B_Vaccine age eligibility/availability age floor (general population summary)",
    "V2C_Vaccine age eligibility/availability age floor (at risk summary)",
    "V2D_Medically/ clinically vulnerable (Non-elderly)",
    "V2E_Education", "V2F_Frontline workers  (non healthcare)",
    "V2G_Frontline workers  (healthcare)", "V3_Vaccine Financial Support (summary)",
    "V4_Mandatory Vaccination (summary)", "ConfirmedCases",
    "ConfirmedDeaths", "MajorityVaccinated", "PopulationVaccinated",
    "StringencyIndex_Average", "StringencyIndex_Average_ForDisplay",
    "GovernmentResponseIndex_Average", "GovernmentResponseIndex_Average_ForDisplay",
    "ContainmentHealthIndex_Average", "ContainmentHealthIndex_Average_ForDisplay",
    "EconomicSupportIndex", "EconomicSupportIndex_ForDisplay"
    from "OxCGRT_nat_latest"
);
```

5. c1(C1M_School closing, C1M_Flag)
   a. the tables are created due to normalization(remove transitive dependencies)
   b. c2~c7 and e1, h1, h6~h8 is the similar to 5.(c1), having one attribate and flag.

```
create table c1 as (
    select distinct "C1M_School closing", "C1M_Flag"
    from "OxCGRT_nat_latest"
);
```

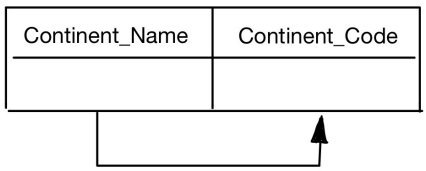
I used “country-and-continent-codes-list-csv” and “OxCGRT_nat_latest” tables to create the tables mentioned above.



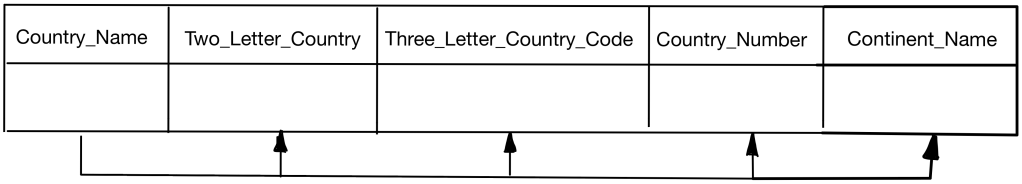
4. Clearly indicate the level of normal form, test the level of normal form for each table (10pts)

Ans:

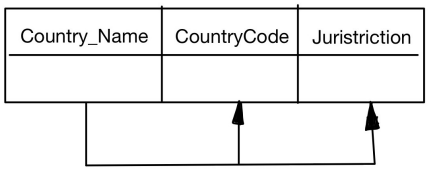
1. continent(Continent_Name, Continent_Code)
   a. satisfies 1NF: no repeating group
   b. satisfies 2NF: no partial dependencies
   c. satisfies 3NF: no transitive dependency
   d. BCNF: Continent_Name is a superkey for continent.



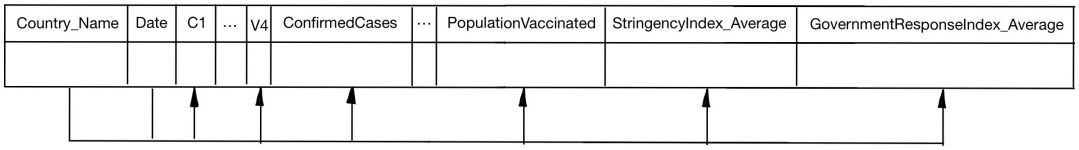
2. country(CountryName, Two_Letter_Country_Code, Three_Letter_Country_Code, Country_Number, Continent_Name).
   a. satisfies 1NF: no repeating group
   b. satisfies 2NF: no partial dependencies
   c. satisfies 3NF: no transitive dependency
   d. BCNF: CountryName is a superkey for country.



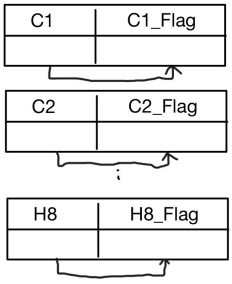
3. country_info(CountryName, CountryCode, Jurisdiction)
    a. satisfies 1NF: no repeating group
    b. satisfies 2NF: no partial dependencies
    c. satisfies 3NF: no transitive dependency
    d. BCNF: Country_Name is a superkey for country_info.



4. status(CountryName, Date, C1, …, V4, ConfirmedCases, …, EconomicSupportIndex_ForDisplay)
    a. satisfies 1NF: no repeating group
    b. satisfies 2NF: no partial dependencies
    c. satisfies 3NF: no transitive dependency
    d. BCNF: (Country_Name, Date) is a superkey for status.



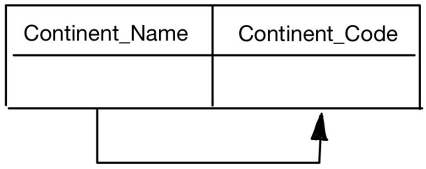
5. c1(C1M_School closing, C1M_Flag). (Plus: c2~c7 and e1, h1, h6~h8 is the similar to 5.(c1), having one attribate and flag.)
    a. satisfies 1NF: no repeating group
    b. satisfies 2NF: no partial dependencies
    c. satisfies 3NF: no transitive dependency
    d. BCNF: C1 is a superkey for c1.



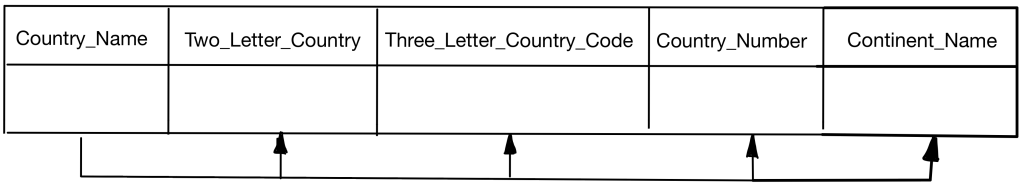
5. List the functional dependency of each table. (10pts)

Ans:

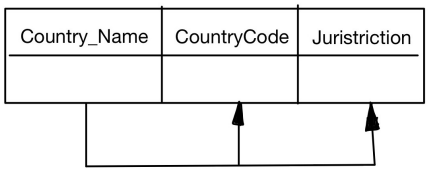
1. continent(Continent_Name, Continent_Code)
   a. F = {Continent_Name → Continent_Code}



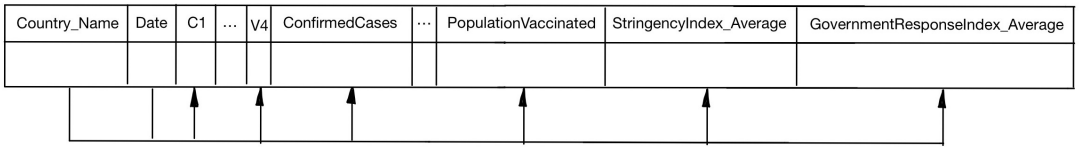
2. country(CountryName, Two_Letter_Country_Code, Three_Letter_Country_Code, Country_Number, Continent_Name).
   a. F = {CountryName → Two_Letter_Country_Code, Country_Name → Three_Letter_Country_Code, Country_Name → Continent_Name}



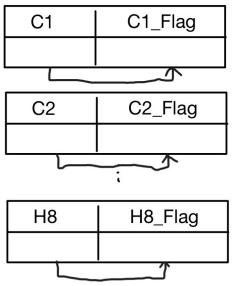
3. country_info(CountryName, CountryCode, Jurisdiction)
   a. F = {CountryName → CountryCode, CountryName → Jurisdiction}



4. status(CountryName, Date, C1, …, V4, ConfirmedCases, …, EconomicSupportIndex_ForDisplay)
    a. F = {CountryName, Date → C1, …, CountryName, Date → V4, CountryName, Date → ConfirmedCases, …, CountryName, Date → EconomicSupportIndex_ForDisplay}



5. c1(C1M_School closing, C1M_Flag). (Plus: c2~c7 and e1, h1, h6~h8 is the similar to 5.(c1), having one attribate and flag.)
    a. F = {C1→C1M_Flag}



6. The SQL statements (in .sql file) and output results of 4a (10pts)

Ans:

```
with "AvgStringencyIndex" as(
    select "CountryName",
           "Date", Avg("StringencyIndex_Average_ForDisplay") as "StringencyIndex_Average_ForDisplay"
    from "status" as s
    natural join "country"
    where "Date" in ('20200601', '20210601', '20220601')
    group by ("CountryName", "Date")
), "MaxStringencyIndex" as (
    select "Continent_Name",
           "Date", MAX("StringencyIndex_Average_ForDisplay") as "MaxStringencyIndex"
    from "AvgStringencyIndex" as asi
    natural join "country"
    group by ("Continent_Name", "Date")
)
select "CountryName", msi."Continent_Name",
           msi."Date", "MaxStringencyIndex"
from "MaxStringencyIndex" as msi
join ("AvgStringencyIndex" natural join "country") as avgc on
avgc."Date" = msi."Date" and avgc."StringencyIndex_Average_ForDisplay" = msi."MaxStringencyIndex"
and avgc."Continent_Name" = msi."Continent_Name"
order by "Continent_Name" asc, "Date" asc
```

	CountryName character varying (300)	Continent_Name character varying (300)	Date date	MaxStringencyIndex double precision
1	Libya	Africa	2020-06-01	96.3
2	Mauritius	Africa	2021-06-01	80.56
3	Zimbabwe	Africa	2022-06-01	47.12
4	Iraq	Asia	2020-06-01	92.59
5	Nepal	Asia	2020-06-01	92.59
6	Nepal	Asia	2021-06-01	94.44
7	China	Asia	2022-06-01	79.17
8	Malta	Europe	2020-06-01	83.33
9	Ireland	Europe	2020-06-01	83.33
10	Italy	Europe	2021-06-01	71.3
11	Ukraine	Europe	2022-06-01	60.16
12	Honduras	North America	2020-06-01	100
13	El Salvador	North America	2020-06-01	100
14	Cuba	North America	2020-06-01	100
15	Trinidad and Tobago	North America	2021-06-01	92.59
16	Bahamas	North America	2022-06-01	44.44
17	Fiji	Oceania	2020-06-01	70.37
18	Australia	Oceania	2021-06-01	75.46
19	Vanuatu	Oceania	2022-06-01	73.61
20	Argentina	South America	2020-06-01	90.74
21	Venezuela	South America	2021-06-01	87.96
22	Peru	South America	2022-06-01	40.46

Stringency index is showing the unreasonable regulation of the government, it's scaled from 0~100. Therefore I average the stringency if there are multiple indexes in one country, and create a table named "**AvgStringencyIndex**" from "**status**" table, which has "**CountryName**", "**Date**", "**StringencyIndex_Average_ForDisplay**" as its columns.

> 📌 I do not check whether there might be multiple Stringency Indexes in one country on the same day since there are too much data. However, averaging should not affect the data without multiple stringency indexes, so I just average all the indexes.

After I get the "**AvgStringencyIndex**" table, now I know the "**StringencyIndex_Average_ForDisplay**" for every country on specific days(2020/06/01, 2021/06/01, 2022/06/01), and I want to find the highest "**StringencyIndex_Average_ForDisplay**" with every continent. Therefore, I join the"**AvgStringencyIndex**" table with "**country**" table and group the data by ("**Continent_Name**", "**Date**"), and get the maximum of "**StringencyIndex_Average_ForDisplay**". In the end, I can get the hightest stringency index in each continent on three separate days, the resulting table is named "**MaxStringencyIndex**".

Now I have the the highest Stringency Index on 2022/06/01, 2021/06/01, and 2020/06/01 in each continent, all I have to do is link the country name with the data. I first use the "**AvgStringencyIndex**" table and "**country**" table to link Stringency Index, date and country name with its continent name, then I combine it with "**MaxStringencyIndex**" table to map the data to correct country names by using specific Stringency index, date and continent name.

7. The SQL statements (in .sql file) and output results of 4b (10pts)

Ans:

```
with "AvgStringencyIndex" as(
        select "CountryName", "Date",
                Avg("StringencyIndex_Average_ForDisplay") as
"StringencyIndex_Average_ForDisplay"
        from "status" as s
        natural join "country"
        where "Date" in ('20200601', '20210601', '20220601')
        group by ("CountryName", "Date")
), "7dMovingAvg" as (
        select t1."CountryName", t1."Date",
        (case when (t1."ConfirmedCases"-t2."ConfirmedCases")/7 = 0 then 0.1
        else (t1."ConfirmedCases"-t2."ConfirmedCases")/7 end) as "MovingAvg"
        from "status" as t1 join "status" as t2
        using ("CountryName")
        where (t1."Date" = '20200601' and t2."Date" = '20200525') or (t1."Date" = '20210601' and
t2."Date" = '20210525') or (t1."Date" = '20220601' and t2."Date" = '20220525')
),
"OverStringencyIndex" as (
        select "CountryName", "Date",
                ("StringencyIndex_Average_ForDisplay"/"MovingAvg") as "osi"
        from "7dMovingAvg" natural join "AvgStringencyIndex"
), "MaxOSI" as (
        select "Continent_Name",
                        "Date", MAX("osi") as "MaxStringencyIndex"
        from "OverStringencyIndex" as osi
        natural join "country"
        group by ("Continent_Name", "Date")
)
select "CountryName", "MaxOSI"."Continent_Name",
                        "MaxOSI"."Date", "MaxStringencyIndex"
from "MaxOSI"
join ("OverStringencyIndex" natural join "country") as osic on
osic."Date" = "MaxOSI"."Date" and osic."osi" = "MaxOSI"."MaxStringencyIndex"
and osic."Continent_Name" = "MaxOSI"."Continent_Name"
order by "Continent_Name" asc, "Date" asc
```

	CountryName character varying (300)	Continent_Name character varying (300)	Date date	MaxStringencyIndex double precision
1	Gambia	Africa	2020-06-01	777.8
2	Congo	Africa	2021-06-01	509.29999999999995
3	Sierra Leone	Africa	2022-06-01	422.09999999999997
4	Brunei	Asia	2020-06-01	527.8
5	Tajikistan	Asia	2021-06-01	324.09999999999997
6	Kyrgyz Republic	Asia	2022-06-01	361.09999999999997
7	Liechtenstein	Europe	2020-06-01	564.8
8	San Marino	Europe	2021-06-01	330.54
9	Belarus	Europe	2022-06-01	138.9
10	Barbados	North America	2020-06-01	787
11	Bermuda	North America	2021-06-01	80.09750000000001
12	Dominica	North America	2022-06-01	324.09999999999997
13	Fiji	Oceania	2020-06-01	703.7
14	Tonga	Oceania	2021-06-01	472.2
15	Kiribati	Oceania	2022-06-01	398.1
16	Guyana	South America	2020-06-01	38.080000000000005
17	Guyana	South America	2021-06-01	0.4793287827076223
18	Suriname	South America	2022-06-01	1.1381679389312978

Stringency index is showing the unreasonable regulation of the government, it's scaled from 0~100. Therefore I average the stringency if there are multiple indexes in one country, and create a table named

“**AvgStringencyIndex**” from “**status**” table, which has “**CountryName**”, “**Date**”, “**StringencyIndex_Average_ForDisplay**” as its columns.

Since “over Stringency index” is composed of “Stringency Index” and “7-day moving average” of daily confirmed new cases, I create another table for the calculation of “over Stringency index”, which is named “**7dMovingAvg**”. It is derived from two “status” table with different dates and has “CountryName”, “Date” and “MovingAvg” as the 7-day moving average of confirmed new cases. I modify the data in “MovingAvg”, if “MovingAvg” is 0, it’ll become 0.1, otherwise it’s the same. I use the join statement to pick up the specific dates and the date that is 7 days earilier, and calculate the moving average.

After I get the “**7dMovingAvg**” table, now all I have to do is calculate the know the “over Stringency index” of each country on the three separate days. I create a table named “**OverStringencyIndex**” from “**AvgStringencyIndex**” and “**7dMovingAvg**”, which has “CountryName”, “Date” and “osi” as its columns, and “osi” is “over Stringency index” that comes from “StringencyIndex_Average_ForDisplay” and “AvgConfirmedCases”.

To find the maximum “over Stringency index” of each continent on three different days, I need to combine the data with continent, so I create a table named “MaxOSI” from “**OverStringencyIndex**” and “**country**”, and group the data by (”Continent_Name”, “Date”), then apply max on “osi” to get the highest “over Stringency index” of each continent.

Now I have the the highest “over Stringency index” on 2022/06/01, 2021/06/01, and 2020/06/01 in each continent, all I have to do is link the country name with the data. I first use the “**OverStringencyIndex**” table and “**country**” table to link over Stringency Index, date and country name with its continent name, then I combine it with “**MaxOSI**” table to map the data to correct country names by using specific Stringency index, date and continent name.