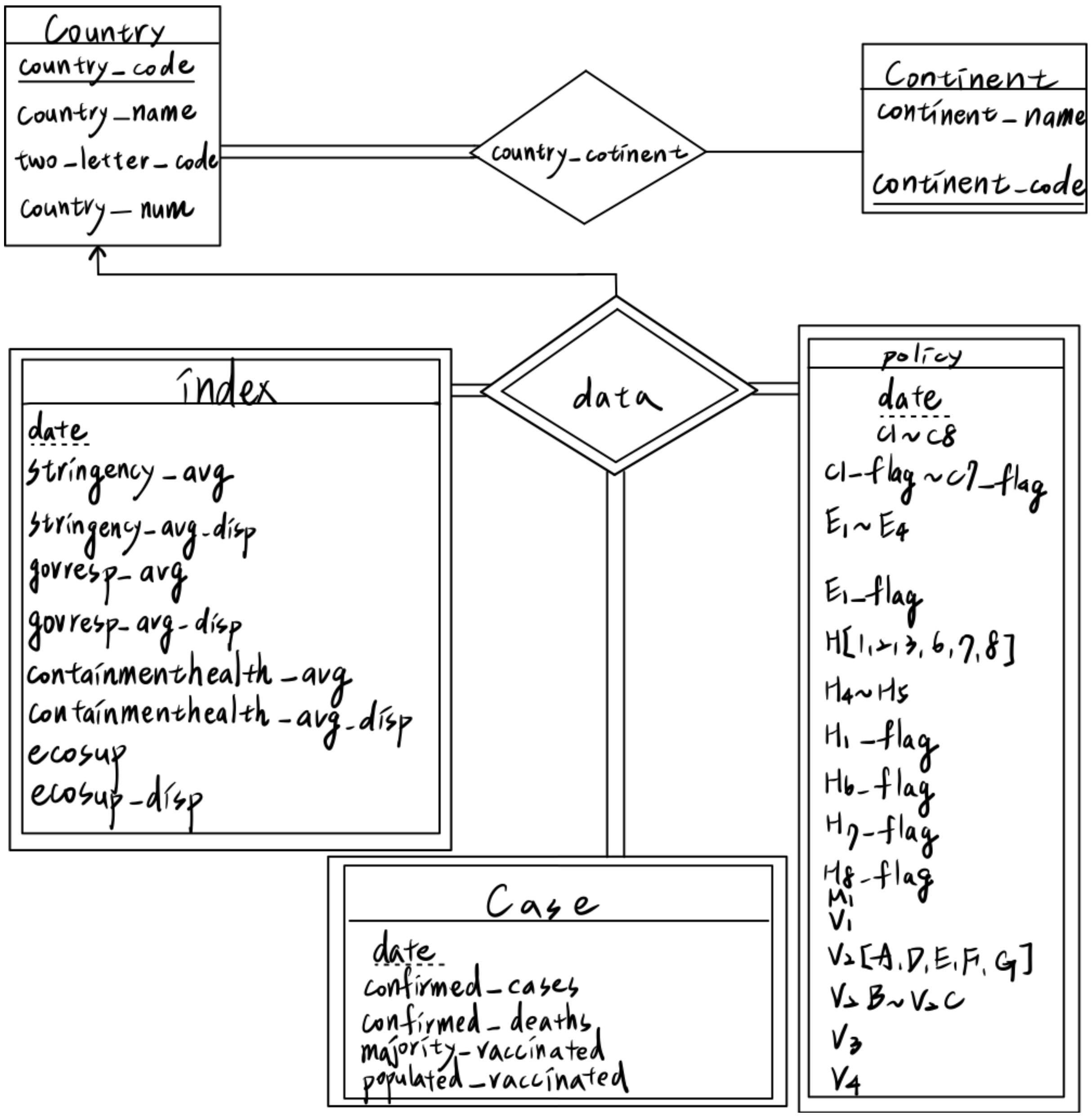
Database HW2

111550100 邱振源

1. 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)



ER Diagram

💡 The meanings represented by H[1, 2, 3, 6, 7, 8] and V2[A, D, E, F, G] are respectively H1, H2, H3, H6, H7, H8, and V2A, V2D, V2E, V2F, V2G. The reason for writing them in the previous format is to save space.

Redundancy-avoiding

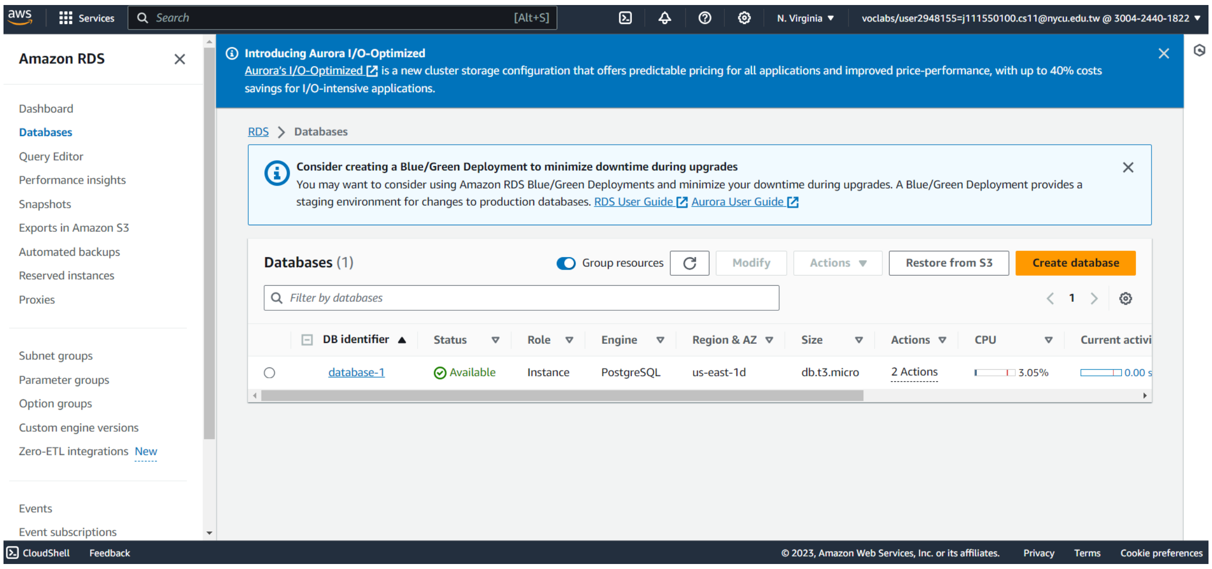
- Since RegionName, RegionCode are all null in the file, and Jurisdiction is always NAT_TOTAL, I have removed these three attributes from my ER diagram.
- In addition, in the country-and-continent-codes-list-csv, the last 4 records in the original file do not have a Three_Letter_Country_Code. Therefore, I have deleted the last 4 records to maintain data consistency.
- Furthermore, I believe that placing Two_Letter_Country_Code and Country_Number in the Country Table would result in many functional dependency issues. Also, these two attributes are not likely to be used in this assignment. Therefore, I have separated these two tables to reduce redundancy issues.

Add Constraints

- I choose Many-to-Many cardinality constraint for “Country_Continent” relationship set, since there are some countries in the data that cross two continents.
- “Countries” entity set has total participation in the relationship of “Country_Continent”, since there should be a continent correspond to every country in “Countries” entity set.
- As there is no country in the 'Countries' entity set located in Antarctica, I have opted for partial participation in “Countries”.

2. Provide print screens of the 1) AWS RDS launch 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)

AWS RDS launch page



Introducing Aurora I/O-Optimized
Aurora's I/O-Optimized is a new cluster storage configuration that offers predictable pricing for all applications and improved price-performance, with up to 40% costs savings for I/O-intensive applications.

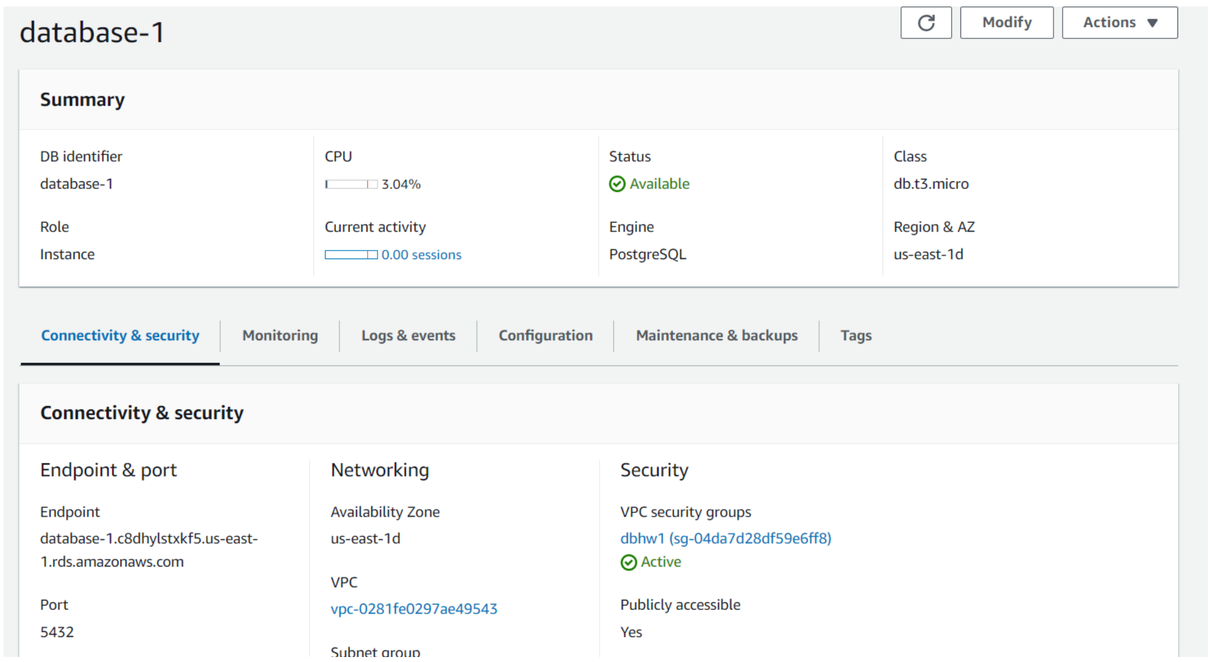
RDS > Databases

Consider creating a Blue/Green Deployment to minimize downtime during upgrades
You may want to consider using Amazon RDS Blue/Green Deployments and minimize your downtime during upgrades. A Blue/Green Deployment provides a staging environment for changes to production databases. RDS User Guide Aurora User Guide

Databases (1) — Group resources | Modify | Actions | Restore from S3 | Create database

Filter by databases

DB identifier	Status	Role	Engine	Region & AZ	Size	Actions	CPU	Current activi
database-1	Available	Instance	PostgreSQL	us-east-1d	db.t3.micro	2 Actions	3.05%	0.00 s



database-1

Summary

DB identifier	CPU	Status	Class
database-1	3.04%	Available	db.t3.micro
Role	Current activity	Engine	Region & AZ
Instance	0.00 sessions	PostgreSQL	us-east-1d

Connectivity & security

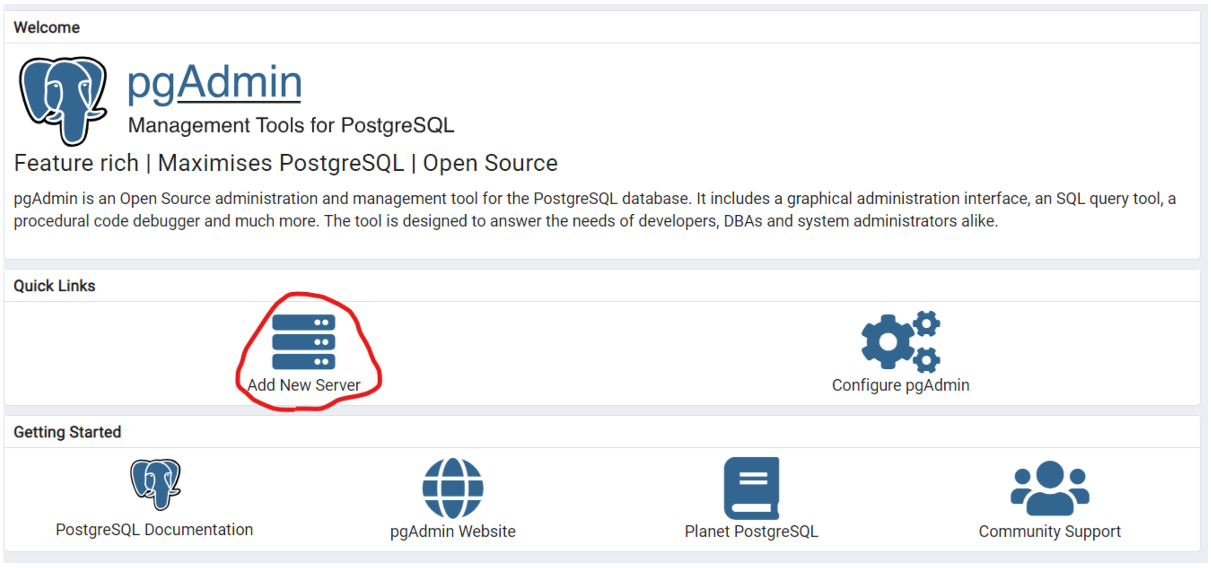
Endpoint & port
Endpoint
database-1.c8dhylstxkf5.us-east-1.rds.amazonaws.com
Port
5432

Networking
Availability Zone
us-east-1d
VPC
vpc-0281fe0297ae49543
Subnet group

Security
VPC security groups
dbhw1 (sg-04da7d28df59e6ff8)
Active
Publicly accessible
Yes

Connect to the AWS RDS

- I use pgAdmin to connect AWS RDS.
- URL is database-1.c8dhylstxkf5.us-east-1.rds.amazonaws.com


Welcome

pgAdmin
Management Tools for PostgreSQL

Feature rich | Maximises PostgreSQL | Open Source

pgAdmin is an Open Source administration and management tool for the PostgreSQL database. It includes a graphical administration interface, an SQL query tool, a procedural code debugger and much more. The tool is designed to answer the needs of developers, DBAs and system administrators alike.

Quick Links

Add New Server

Configure pgAdmin

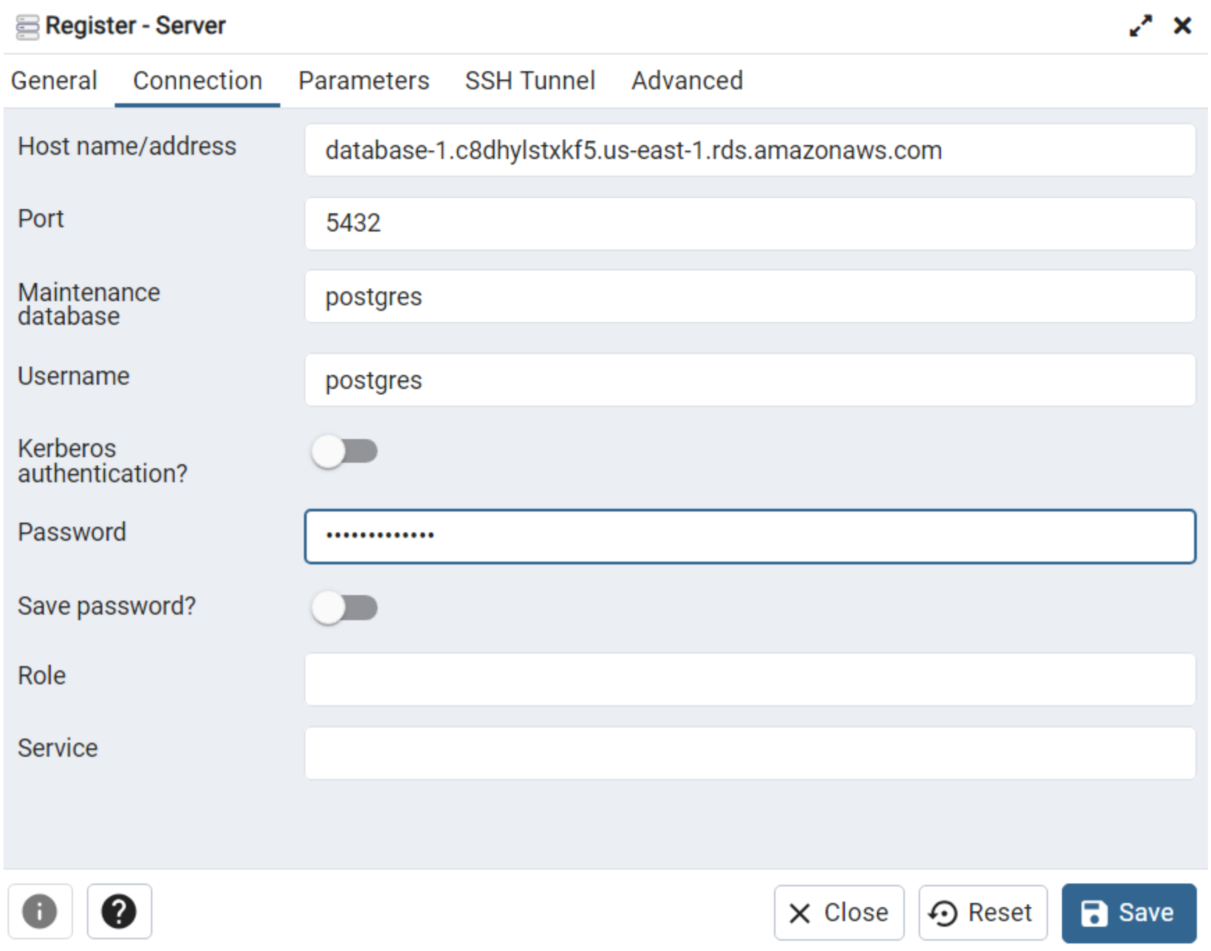
Getting Started

PostgreSQL Documentation

pgAdmin Website

Planet PostgreSQL

Community Support


Register - Server

General | Connection | Parameters | SSH Tunnel | Advanced

Host name/address	database-1.c8dhylstxkf5.us-east-1.rds.amazonaws.com
Port	5432
Maintenance database	postgres
Username	postgres
Kerberos authentication?	
Password	•••••••••••••
Save password?	
Role	
Service	

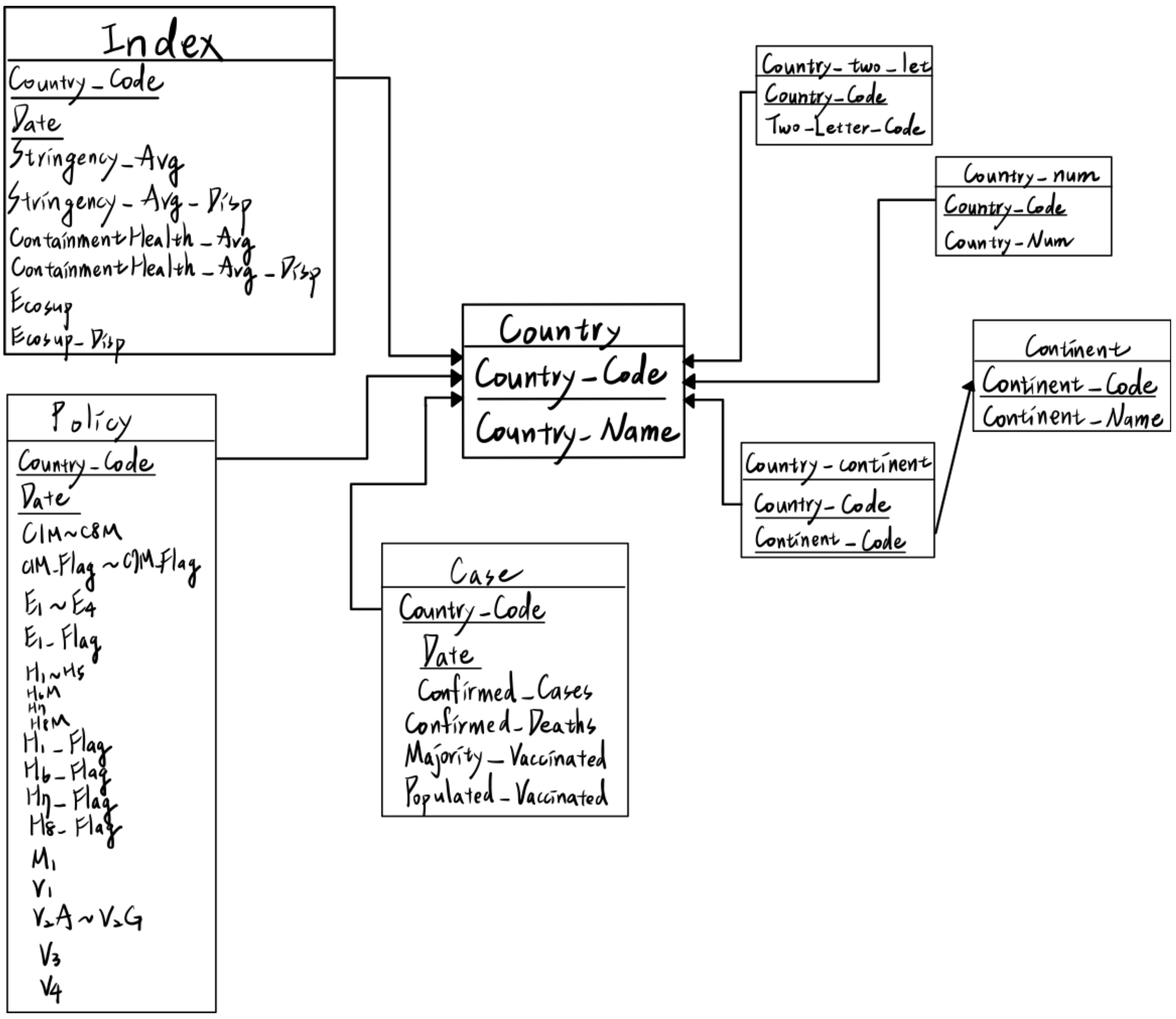
Close | Reset | Save

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)

My schema

I decomposed the “Country” entity into “Country,” “Country_two_let,” and “Country_num.” The rest remains the same as in the ER diagram, except that I pulled the weak entity back to become part of the schema.

Additionally, during this step, I noticed that 'RKS' (Kosovo, Republic of) was not present in the continent-to-country mapping table. Therefore, I added ('Europe', 'EU', 'Kosovo, Republic of', 'KS', 'RKS', 383) to the 'country_and_continent_codes_list.'



Schema

SQL queries to create table

***“Country”* relation**

***“Continent”* relation**

```
CREATE TABLE public.Country
(
    Country_Code   CHAR(3),
    Country_Name   VARCHAR(100),
    primary key   (Country_Code)
);
```

```
CREATE TABLE public.Continent
(
    Continent_Code    CHAR(2),
    Continent_Name    VARCHAR(100),
    primary key (Continent_Code)
);
```

"Country_two_let" relation

```
CREATE TABLE public.Country_two_let
(
    Country_Code      CHAR(3),
    Two_Letter_Code   CHAR(2),
    primary key   (Country_Code),
    foreign key (Country_Code)
      references Country(Country_Code)
);
```

"Country_num" relation

```
CREATE TABLE public.Country_num
(
    Country_Code  CHAR(3),
    Country_Num   INT,
    primary key   (Country_Code),
    foreign key (Country_Code)
      references Country(Country_Code)
);
```

"Country_continent" relation

```
CREATE TABLE public.Country_continent
(
    Country_Code     CHAR(3),
    Continent_Code   CHAR(2),
    primary key
      (Country_Code, Continent_code),
    foreign key (Country_Code)
      references
          Country (Country_Code),
    foreign key (Continent_Code)
      references
          Continent (Continent_Code)
);
```

"Index" relation

```
CREATE TABLE public.Index
(
    Country_Code            CHAR(3),
    Date                    INT,
    Stringency_Avg          NUMERIC(10,2),
    Stringency_Avg_Disp     NUMERIC(10,2),
    GovResp_Avg             NUMERIC(10,2),
    GovResp_Avg_Disp        NUMERIC(10,2),
    ContainmentHealth_Avg NUMERIC(10,2),
    ContainmentHealth_Avg_Disp
                            NUMERIC(10,2),
    EcoSup                  NUMERIC(10,2),
    EcoSup_Disp             NUMERIC(10,2),
    primary key (Country_Code, Date),
    foreign key (Country_Code)
      references Country(Country_Code)
);
```

"Policy" relation

```
CREATE TABLE public.Policy
(
    Country_Code        CHAR(3),
    Date                INT,

    C1M                 NUMERIC(3,2),
    C2M                 NUMERIC(3,2),
    C3M                 NUMERIC(3,2),
    C4M                 NUMERIC(3,2),
    C5M                 NUMERIC(3,2),
    C6M                 NUMERIC(3,2),
    C7M                 NUMERIC(3,2),
    C8EV                NUMERIC(3,2),
    C1M_Flag            BOOLEAN,
    C2M_Flag            BOOLEAN,
    C3M_Flag            BOOLEAN,
    C4M_Flag            BOOLEAN,
    C5M_Flag            BOOLEAN,
    C6M_Flag            BOOLEAN,
    C7M_Flag            BOOLEAN,

    E1                  NUMERIC(3,2),
    E2                  NUMERIC(3,2),
    E3                  FLOAT,
    E4                  FLOAT,
    E1_Flag             BOOLEAN,

    H1                  NUMERIC(3,2),
    H2                  NUMERIC(3,2),
    H3                  NUMERIC(3,2),
    H4                  FLOAT,
    H5                  FLOAT,
    H6M                 NUMERIC(3,2),
    H7                  NUMERIC(3,2),
    H8M                 NUMERIC(3,2),
    H1_Flag             BOOLEAN,
    H6M_Flag            BOOLEAN,
    H7_Flag             BOOLEAN,
    H8M_Flag            BOOLEAN,
    M1                  NUMERIC(3,2),

    V1                  INT,
    V2A                 INT,
    V2B                 VARCHAR(20),
    V2C                 VARCHAR(20),
    V2D                 INT,
    V2E                 INT,
    V2F                 INT,
    V2G                 INT,
    V3                  INT,
    V4                  INT,

    primary key (Country_Code, Date),
    foreign key (Country_Code) references Country(Country_Code)
);
```

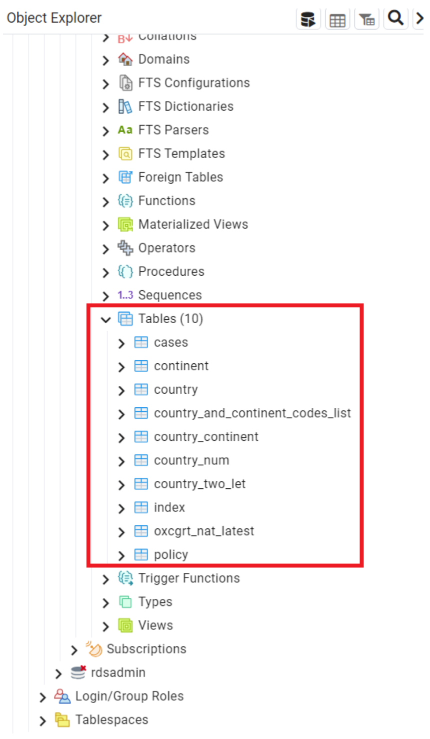
***"Cases"* relation**

```
CREATE TABLE public.Cases
(
    Country_Code                CHAR(3),
    Date                        INT,
    Confirmed_Cases             FLOAT,
    Confirmed_Deaths            FLOAT,
    Majority_Vaccinated         VARCHAR(5),
    Population_Vaccinated       NUMERIC(5,2),

    primary key (Country_Code, Date),
    foreign key (Country_Code) references Country(Country_Code)
  );
```

ScreenShot

- The picture on the right_hand_side shows that the tables have been created in your database on AWS RDS
- "oxcgrt_nat_latest" and "country_and_continent_codes_list" are tables of original csv files.



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

Original table

- Test 1NF
  Sinve, all datas from original csv file have no repeat group and all attributes are single value and atomic, the original table are already 1NF.

> Since we need to test normal form of each tables, we need to check functional dependency sets which is shown on the "5. *List the functional dependency of each table*".

Country

- Normal Form: BCNF
- Test
  - Country_Code is a candidate key for R.
  - Country_Name is a candidate key for R.
  - For all non-trivial functional dependencies in $\alpha \rightarrow \beta$ in F^+, α are super key.
- So, it's BCNF.

Continent

- Normal Form: BCNF
- Test
  - Continent_Code is a candidate key for R.
  - Continent_Name is a candidate key for R.
  - For all non-trivial functional dependencies in $\alpha \rightarrow \beta$ in F^+, α are super key.
- So, it's BCNF.

Country_two_let

- Normal Form: BCNF
- Test
  - Country_Code is a candidate key for R.

    - Two_Letter_Code is a candidate key for R.
    - So for all non-trivial functional dependencies in $\alpha \rightarrow \beta$ in F^+, α are super key.
- So, it's BCNF.

Country_num

- Normal Form: BCNF
- Test
    - Continent_Code is a candidate key for R.
    - Country_Num is a candidate key for R.
    - So for all non-trivial functional dependencies in $\alpha \rightarrow \beta$ in F^+, α are super key.
- So, it's BCNF.

Country_continent

- Normal Form: BCNF
- Test
    - {Continent_Code, Continent_Code} is a candidate key for R.
    - So for all non-trivial functional dependencies in $\alpha \rightarrow \beta$ in F^+, α are super key.
- So, it's BCNF.

Index

- Normal Form: BCNF
- Test
    - {Date, Continent_Code} is a candidate key for R.
    - So for all non-trivial functional dependencies in $\alpha \rightarrow \beta$ in F^+, α are super key.
- So, it's BCNF.

Policy

- Normal Form: BCNF

- Test
    - {Date, Continent_Code} is a candidate key for R.
    - So for all non-trivial functional dependencies in $\alpha \rightarrow \beta$ in F^+, α are super key.
- So, it's BCNF.

Cases

- Normal Form: BCNF
- Test
    - {Date, Continent_Code} is a candidate key for R.
    - So for all non-trivial functional dependencies in $\alpha \rightarrow \beta$ in F^+, α are super key.
- So, it's BCNF.

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

Country

- (Country_Code) → (Country_Name)
- (Country_Name) → (Country_Code)

Continent

- (Continent_Code) → (Contient_Name)
- (Continent_Name) → (Continent_Code)

Country_two_let

- (Country_Code) → (Two_Letter_Code)
- (Two_Letter_Code) → (Country_Code)

Country_num

- (Country_Code) → (Country_Num)
- (Country_Num)→ (Country_Code)

Country_continent

- (Country_Code, Continent_Code) → (Country_Code)

- (Country_Code, Continent_Code) → (Continent_Code)

Index

- (Date, Country_Code) → (*all attributes in *Index*)

Policy

- (Date, Country_Code) → (*all attributes in *Policy*)

Cases

- (Date, Country_Code) → (*all attributes in *Cases*)

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

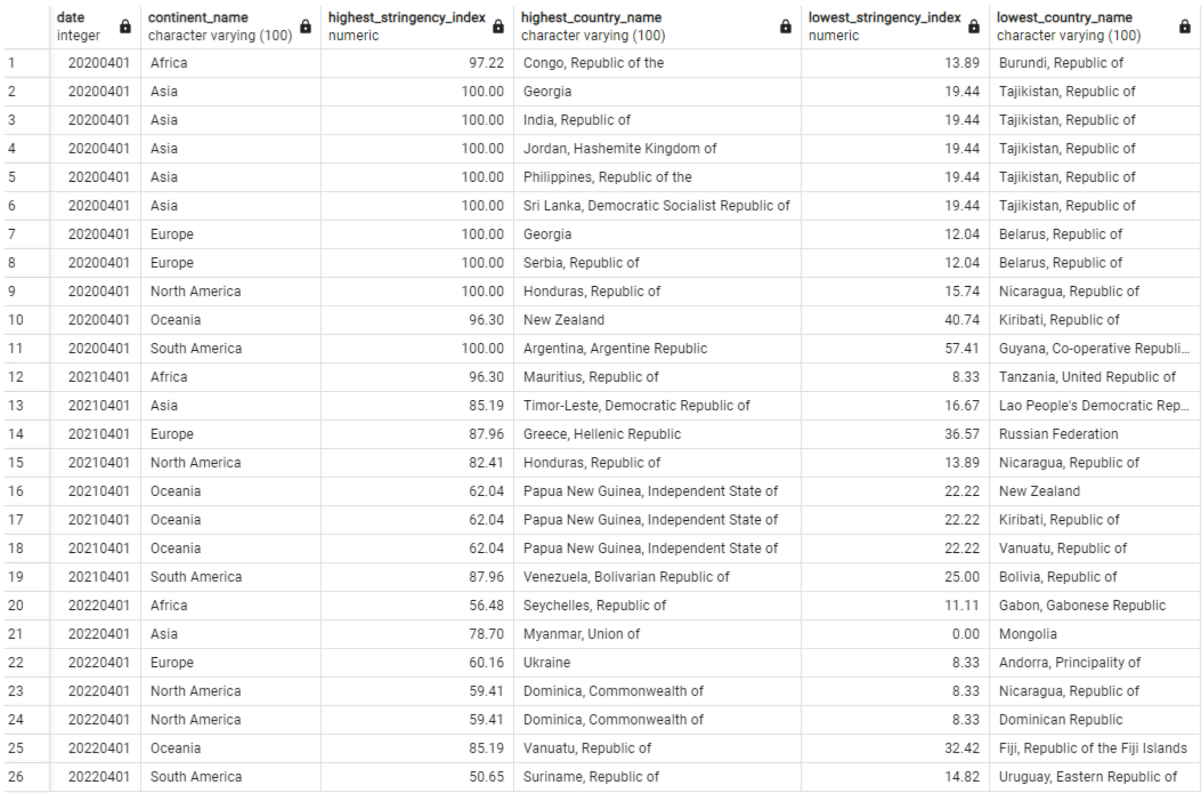
SQL Statements

```
WITH Country_To_Continent AS(
  SELECT
    Country.Country_Code,
    Country.Country_Name,
    Continent.Continent_Code,
    Continent.Continent_Name
  FROM
    Country Natural JOIN Country_Continent
        Natural JOIN Continent
  ),
  Country_Code_Name AS(
    SELECT
      Country.Country_Code,
      Country_two_let.Two_Letter_Code,
      Country_num.Country_Num
    FROM
      Country Natural JOIN Country_two_let
          Natural JOIN Country_num
  ),
  StringencyId_2020(Date, Continent_Code, Maximum, Minimum) AS(
  SELECT Date, Continent_Code, MAX(Stringency_Avg_Disp), Min(Stringency_Avg_Disp)
  FROM Index Natural JOIN Country_Continent
  WHERE Date = 20200401
  GROUP BY Date, Continent_Code
  ),
  StringencyId_2021(Date, Continent_Code, Maximum, Minimum) AS(
  SELECT Date, Continent_Code, MAX(Stringency_Avg_Disp), Min(Stringency_Avg_Disp)
  FROM Index Natural JOIN Country_Continent
  WHERE Date = 20210401
  GROUP BY Date, Continent_Code
```

```sql
),
StringencyId_2022(Date, Continent_Code, Maximum, Minimum) AS(
SELECT Date, Continent_Code, MAX(Stringency_Avg_Disp), Min(Stringency_Avg_Disp)
FROM Index Natural JOIN Country_Continent
WHERE Date = 20220401
GROUP BY Date, Continent_Code
),
StringencyId_2022_12(Date, Continent_Code, Maximum, Minimum) AS(
SELECT Date, Continent_Code, MAX(Stringency_Avg_Disp), Min(Stringency_Avg_Disp)
FROM Index Natural JOIN Country_Continent
WHERE Date = 20221201
GROUP BY Date, Continent_Code
),
CountryCode_Id_2020 AS(
SELECT
  S.Date, S.Continent_Code, Maximum,
  MaxCC.Country_Code AS Max_CountryCode,
  Minimum, MinCC.Country_Code AS Min_CountryCode
FROM
  StringencyId_2020 AS S
  LEFT JOIN (Index Natural JOIN Country_Continent) AS MaxCC
    ON (S.Maximum = MaxCC.Stringency_Avg_Disp
      AND S.Date = MaxCC.Date
        AND S.Continent_Code = MaxCC.Continent_Code)
  LEFT JOIN (Index Natural JOIN Country_Continent) AS MinCC
    ON (S.Minimum = MinCC.Stringency_Avg_Disp
      AND S.Date = MinCC.Date
        AND S.Continent_Code = MinCC.Continent_Code)
ORDER BY S.Continent_Code
),
CountryCode_Id_2021 AS(
SELECT
  S.Date, S.Continent_Code, Maximum,
  MaxCC.Country_Code AS Max_CountryCode,
  Minimum, MinCC.Country_Code AS Min_CountryCode
FROM
  StringencyId_2021 AS S
  LEFT JOIN (Index Natural JOIN Country_Continent) AS MaxCC
    ON (S.Maximum = MaxCC.Stringency_Avg_Disp
      AND S.Date = MaxCC.Date
        AND S.Continent_Code = MaxCC.Continent_Code)
  LEFT JOIN (Index Natural JOIN Country_Continent) AS MinCC
    ON (S.Minimum = MinCC.Stringency_Avg_Disp
      AND S.Date = MinCC.Date
        AND S.Continent_Code = MinCC.Continent_Code)
ORDER BY S.Continent_Code
),
CountryCode_Id_2022 AS(
SELECT
  S.Date, S.Continent_Code, Maximum,
  MaxCC.Country_Code AS Max_CountryCode,
    Minimum, MinCC.Country_Code AS Min_CountryCode
  FROM
    StringencyId_2022 AS S
    LEFT JOIN (Index Natural JOIN Country_Continent) AS MaxCC
      ON (S.Maximum = MaxCC.Stringency_Avg_Disp
        AND S.Date = MaxCC.Date
          AND S.Continent_Code = MaxCC.Continent_Code)
    LEFT JOIN (Index Natural JOIN Country_Continent) AS MinCC
      ON (S.Minimum = MinCC.Stringency_Avg_Disp
        AND S.Date = MinCC.Date
          AND S.Continent_Code = MinCC.Continent_Code)
  ORDER BY S.Continent_Code
  ),
  CountryCode_Id_2022_12 AS(
  SELECT
    S.Date, S.Continent_Code, Maximum,
    MaxCC.Country_Code AS Max_CountryCode,
    Minimum, MinCC.Country_Code AS Min_CountryCode
  FROM
    StringencyId_2022_12 AS S
    LEFT JOIN (Index Natural JOIN Country_Continent) AS MaxCC
      ON (S.Maximum = MaxCC.Stringency_Avg_Disp
        AND S.Date = MaxCC.Date
          AND S.Continent_Code = MaxCC.Continent_Code)
    LEFT JOIN (Index Natural JOIN Country_Continent) AS MinCC
      ON (S.Minimum = MinCC.Stringency_Avg_Disp
        AND S.Date = MinCC.Date
          AND S.Continent_Code = MinCC.Continent_Code)
  ORDER BY S.Continent_Code
  )

SELECT
  Date,
  Ma.Continent_Name,
  Maximum AS Highest_Stringency_Index,
  Ma.Country_Name AS Highest_Country_Name,
  Minimum AS Lowest_Stringency_Index,
  Mi.Country_Name AS Lowest_Country_Name
FROM
  CountryCode_Id_2020
  LEFT JOIN Country_To_Continent AS Ma
    ON (CountryCode_Id_2020.Max_CountryCode = Ma.Country_Code
      AND CountryCode_Id_2020.Continent_Code = Ma.Continent_Code)
  LEFT JOIN Country_To_Continent AS Mi
    ON (CountryCode_Id_2020.Min_CountryCode = Mi.Country_Code
      AND CountryCode_Id_2020.Continent_Code = Mi.Continent_Code)
UNION
SELECT
  Date,
  Ma.Continent_Name,
  Maximum AS Highest_Stringency_Index,
  Ma.Country_Name AS Highest_Country_Name,
  Minimum AS Lowest_Stringency_Index,
  Mi.Country_Name AS Lowest_Country_Name
FROM
  CountryCode_Id_2021
  LEFT JOIN Country_To_Continent AS Ma
    ON (CountryCode_Id_2021.Max_CountryCode = Ma.Country_Code
      AND CountryCode_Id_2021.Continent_Code = Ma.Continent_Code)
  LEFT JOIN Country_To_Continent AS Mi
    ON (CountryCode_Id_2021.Min_CountryCode = Mi.Country_Code
      AND CountryCode_Id_2021.Continent_Code = Mi.Continent_Code)
UNION
SELECT
  Date,
  Ma.Continent_Name,
  Maximum AS Highest_Stringency_Index,
  Ma.Country_Name AS Highest_Country_Name,
  Minimum AS Lowest_Stringency_Index,
  Mi.Country_Name AS Lowest_Country_Name
FROM
  CountryCode_Id_2022
  LEFT JOIN Country_To_Continent AS Ma
    ON (CountryCode_Id_2022.Max_CountryCode = Ma.Country_Code
      AND CountryCode_Id_2022.Continent_Code = Ma.Continent_Code)
  LEFT JOIN Country_To_Continent AS Mi
    ON (CountryCode_Id_2022.Min_CountryCode = Mi.Country_Code
      AND CountryCode_Id_2022.Continent_Code = Mi.Continent_Code)
UNION
SELECT
  Date,
  Ma.Continent_Name,
  Maximum AS Highest_Stringency_Index,
  Ma.Country_Name AS Highest_Country_Name,
  Minimum AS Lowest_Stringency_Index,
  Mi.Country_Name AS Lowest_Country_Name
FROM
  CountryCode_Id_2022_12
  LEFT JOIN Country_To_Continent AS Ma
    ON (CountryCode_Id_2022_12.Max_CountryCode = Ma.Country_Code
      AND CountryCode_Id_2022_12.Continent_Code = Ma.Continent_Code)
  LEFT JOIN Country_To_Continent AS Mi
    ON (CountryCode_Id_2022_12.Min_CountryCode = Mi.Country_Code
      AND CountryCode_Id_2022_12.Continent_Code = Mi.Continent_Code)
ORDER BY
  Date, Continent_Name, Highest_Country_Name
```

Output Result



	date integer	continent_name character varying (100)	highest_stringency_index numeric	highest_country_name character varying (100)	lowest_stringency_index numeric	lowest_country_name character varying (100)
1	20200401	Africa	97.22	Congo, Republic of the	13.89	Burundi, Republic of
2	20200401	Asia	100.00	Georgia	19.44	Tajikistan, Republic of
3	20200401	Asia	100.00	India, Republic of	19.44	Tajikistan, Republic of
4	20200401	Asia	100.00	Jordan, Hashemite Kingdom of	19.44	Tajikistan, Republic of
5	20200401	Asia	100.00	Philippines, Republic of the	19.44	Tajikistan, Republic of
6	20200401	Asia	100.00	Sri Lanka, Democratic Socialist Republic of	19.44	Tajikistan, Republic of
7	20200401	Europe	100.00	Georgia	12.04	Belarus, Republic of
8	20200401	Europe	100.00	Serbia, Republic of	12.04	Belarus, Republic of
9	20200401	North America	100.00	Honduras, Republic of	15.74	Nicaragua, Republic of
10	20200401	Oceania	96.30	New Zealand	40.74	Kiribati, Republic of
11	20200401	South America	100.00	Argentina, Argentine Republic	57.41	Guyana, Co-operative Republi...
12	20210401	Africa	96.30	Mauritius, Republic of	8.33	Tanzania, United Republic of
13	20210401	Asia	85.19	Timor-Leste, Democratic Republic of	16.67	Lao People's Democratic Rep...
14	20210401	Europe	87.96	Greece, Hellenic Republic	36.57	Russian Federation
15	20210401	North America	82.41	Honduras, Republic of	13.89	Nicaragua, Republic of
16	20210401	Oceania	62.04	Papua New Guinea, Independent State of	22.22	New Zealand
17	20210401	Oceania	62.04	Papua New Guinea, Independent State of	22.22	Kiribati, Republic of
18	20210401	Oceania	62.04	Papua New Guinea, Independent State of	22.22	Vanuatu, Republic of
19	20210401	South America	87.96	Venezuela, Bolivarian Republic of	25.00	Bolivia, Republic of
20	20220401	Africa	56.48	Seychelles, Republic of	11.11	Gabon, Gabonese Republic
21	20220401	Asia	78.70	Myanmar, Union of	0.00	Mongolia
22	20220401	Europe	60.16	Ukraine	8.33	Andorra, Principality of
23	20220401	North America	59.41	Dominica, Commonwealth of	8.33	Nicaragua, Republic of
24	20220401	North America	59.41	Dominica, Commonwealth of	8.33	Dominican Republic
25	20220401	Oceania	85.19	Vanuatu, Republic of	32.42	Fiji, Republic of the Fiji Islands
26	20220401	South America	50.65	Suriname, Republic of	14.82	Uruguay, Eastern Republic of

Worth noting is that although I considered Date = 20221201, there is no data for Date = 20221201. Therefore, there is no output result for that date.

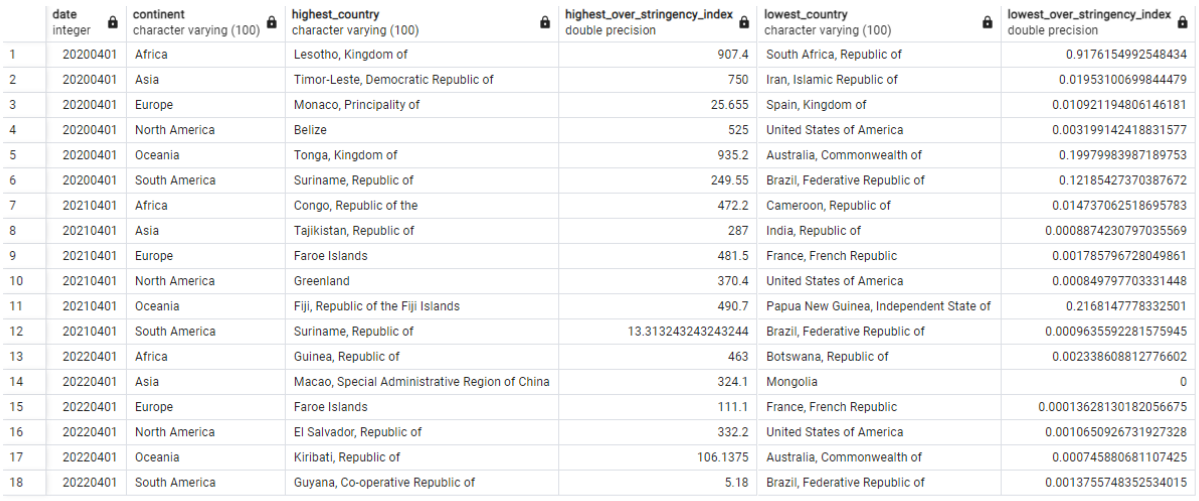
7. The SQL statements (in .sql file) and output results of 4b (10pts)

SQL Statements

```
WITH Country_To_Continent AS (
    SELECT
        Country.Country_Code,
        Country.Country_Name,
        Continent.Continent_Code,
        Continent.Continent_Name
    FROM
        Country
        NATURAL JOIN Country_Continent
        NATURAL JOIN Continent
),
Country_Code_Name AS (
    SELECT
        Country.Country_Code,
        Country_two_let.Two_Letter_Code,
        Country_num.Country_Num
    FROM
        Country
        NATURAL JOIN Country_two_let
        NATURAL JOIN Country_num
),
tem AS (
    SELECT
        Country.Country_Code,
        Country_Continent.Continent_Code,
        Cases.date,
        Cases.Confirmed_Cases
    FROM
        Cases,
        Country,
        Country_Continent
    WHERE
        Country.Country_Code=Cases.Country_Code AND
        (Cases.Date = 20221126
        OR Cases.Date = 20220325
        OR Cases.Date = 20210325
        OR Cases.Date = 20200325)
),
tem_temp AS (
    SELECT
        tem.Country_Code,
        tem.Continent_Code,
        tem.Date,
        Cases.Confirmed_Cases
    FROM
        cases
        JOIN tem ON cases.country_Code = tem.Country_Code
    WHERE
        (cases.date=20221201 AND tem.date=20221126)
        OR (cases.date=20220401 AND tem.date=20220325)
        OR (cases.date=20210401 AND tem.date=20210325)
        OR (cases.date=20200401 AND tem.date=20200325)
),
Mov_Avg AS (
    SELECT
        Date,
        Country_Code,
        Confirmed_Cases,
        (Confirmed_Cases - LAG(Confirmed_Cases, 7) OVER (
            PARTITION BY Country_Code ORDER BY Date
        )) / 7 AS Moving_Average
    FROM
        Cases
),
OvStId AS (
    SELECT
        Index.Date,
        Index.Country_Code,
        Country_Continent.Continent_Code,
        CASE
            WHEN Mov_Avg.Moving_Average = 0 THEN Index.Stringency_Avg_Disp / 0.1
            ELSE Index.Stringency_Avg_Disp / Mov_Avg.Moving_Average
        END AS OveStId
    FROM
        Mov_Avg
        NATURAL JOIN Index
        NATURAL JOIN Country_Continent
),
MinMax_2020 AS (
    SELECT
        Date, Continent_Code, MAX(OveStId) AS Maximum, MIN(OveStId) AS Minimum
    FROM OvStId
    WHERE Date = 20200401
    GROUP BY Date, Continent_Code
),
MinMax_2021 AS (
    SELECT
        Date, Continent_Code, MAX(OveStId) AS Maximum, MIN(OveStId) AS Minimum
    FROM OvStId
    WHERE Date = 20210401
    GROUP BY Date, Continent_Code
),
MinMax_2022 AS (
    SELECT
        Date, Continent_Code, MAX(OveStId) AS Maximum, MIN(OveStId) AS Minimum
    FROM OvStId
    WHERE Date = 20220401
    GROUP BY Date, Continent_Code
),
MinMax_2022_12 AS (
    SELECT
        Date, Continent_Code, MAX(OveStId) AS Maximum, MIN(OveStId) AS Minimum
    FROM OvStId
    WHERE Date = 20221201
    GROUP BY Date, Continent_Code
),
Country_Code_Id_2020 AS (
    SELECT
        MM.Date, MM.Continent_Code, MM.Maximum, MM.Minimum,
        Highest.Country_Code AS Highest_CountryCode,
        Lowest.Country_Code AS Lowest_CountryCode
    FROM
        MinMax_2020 AS MM
        LEFT JOIN OvStId AS Highest
            ON (MM.Maximum = Highest.OveStId
                AND MM.Continent_Code = Highest.Continent_Code
                AND MM.Date = Highest.Date)
        LEFT JOIN OvStId AS Lowest
            ON (MM.Minimum = Lowest.OveStId
                AND MM.Continent_Code = Lowest.Continent_Code
                AND MM.Date = Lowest.Date)
),
Country_Code_Id_2021 AS (
    SELECT
        MM.Date, MM.Continent_Code, MM.Maximum, MM.Minimum,
        Highest.Country_Code AS Highest_CountryCode,
        Lowest.Country_Code AS Lowest_CountryCode
    FROM
        MinMax_2021 AS MM
        LEFT JOIN OvStId AS Highest
            ON (MM.Maximum = Highest.OveStId
                AND MM.Continent_Code = Highest.Continent_Code
                AND MM.Date = Highest.Date)
        LEFT JOIN OvStId AS Lowest
            ON (MM.Minimum = Lowest.OveStId
                AND MM.Continent_Code = Lowest.Continent_Code
                AND MM.Date = Lowest.Date)
),
Country_Code_Id_2022 AS (
    SELECT
        MM.Date, MM.Continent_Code, MM.Maximum, MM.Minimum,
        Highest.Country_Code AS Highest_CountryCode,
        Lowest.Country_Code AS Lowest_CountryCode
    FROM
        MinMax_2022 AS MM
        LEFT JOIN OvStId AS Highest
            ON (MM.Maximum = Highest.OveStId
                AND MM.Continent_Code = Highest.Continent_Code
                AND MM.Date = Highest.Date)
        LEFT JOIN OvStId AS Lowest
            ON (MM.Minimum = Lowest.OveStId
                AND MM.Continent_Code = Lowest.Continent_Code
                AND MM.Date = Lowest.Date)
),
Country_Code_Id_2022_12 AS (
    SELECT
        MM.Date, MM.Continent_Code, MM.Maximum, MM.Minimum,
        Highest.Country_Code AS Highest_CountryCode,
        Lowest.Country_Code AS Lowest_CountryCode
    FROM
        MinMax_2022_12  AS MM
        LEFT JOIN OvStId AS Highest
            ON (MM.Maximum = Highest.OveStId
                AND MM.Continent_Code = Highest.Continent_Code
                AND MM.Date = Highest.Date)
        LEFT JOIN OvStId AS Lowest
            ON (MM.Minimum = Lowest.OveStId
                AND MM.Continent_Code = Lowest.Continent_Code
                AND MM.Date = Lowest.Date)
)

SELECT
    CCI.Date,
    Highest.Continent_Name AS Continent,
    Highest.Country_Name AS Highest_Country,
    CCI.Maximum AS Highest_Over_Stringency_Index,
    Lowest.Country_Name AS Lowest_Country,
    CCI.Minimum AS Lowest_Over_Stringency_Index
FROM
    Country_Code_Id_2020 AS CCI
    LEFT JOIN Country_To_Continent AS Highest
        ON CCI.Highest_CountryCode = Highest.Country_Code
    LEFT JOIN Country_To_Continent AS Lowest
        ON CCI.Lowest_CountryCode = Lowest.Country_Code
UNION
SELECT
    CCI.Date,
    Highest.Continent_Name AS Continent,
    Highest.Country_Name AS Highest_Country,
    CCI.Maximum AS Highest_Over_Stringency_Index,
    Lowest.Country_Name AS Lowest_Country,
    CCI.Minimum AS Lowest_Over_Stringency_Index
FROM
    Country_Code_Id_2021 AS CCI
    LEFT JOIN Country_To_Continent AS Highest
        ON CCI.Highest_CountryCode = Highest.Country_Code
    LEFT JOIN Country_To_Continent AS Lowest
        ON CCI.Lowest_CountryCode = Lowest.Country_Code
UNION
SELECT
    CCI.Date,
    Highest.Continent_Name AS Continent,
    Highest.Country_Name AS Highest_Country,
    CCI.Maximum AS Highest_Over_Stringency_Index,
    Lowest.Country_Name AS Lowest_Country,
    CCI.Minimum AS Lowest_Over_Stringency_Index
FROM
    Country_Code_Id_2022 AS CCI
    LEFT JOIN Country_To_Continent AS Highest
        ON CCI.Highest_CountryCode = Highest.Country_Code
    LEFT JOIN Country_To_Continent AS Lowest
        ON CCI.Lowest_CountryCode = Lowest.Country_Code
UNION
SELECT
    CCI.Date,
    Highest.Continent_Name AS Continent,
    Highest.Country_Name AS Highest_Country,
    CCI.Maximum AS Highest_Over_Stringency_Index,
    Lowest.Country_Name AS Lowest_Country,
    CCI.Minimum AS Lowest_Over_Stringency_Index
FROM
    Country_Code_Id_2022_12 AS CCI
    LEFT JOIN Country_To_Continent AS Highest
        ON CCI.Highest_CountryCode = Highest.Country_Code
    LEFT JOIN Country_To_Continent AS Lowest
        ON CCI.Lowest_CountryCode = Lowest.Country_Code
ORDER BY
    Date, Continent;
```

Output Result



	date integer	continent character varying (100)	highest_country character varying (100)	highest_over_stringency_index double precision	lowest_country character varying (100)	lowest_over_stringency_index double precision
1	20200401	Africa	Lesotho, Kingdom of	907.4	South Africa, Republic of	0.9176154992548434
2	20200401	Asia	Timor-Leste, Democratic Republic of	750	Iran, Islamic Republic of	0.01953100699844479
3	20200401	Europe	Monaco, Principality of	25.655	Spain, Kingdom of	0.010921194806146181
4	20200401	North America	Belize	525	United States of America	0.003199142418831577
5	20200401	Oceania	Tonga, Kingdom of	935.2	Australia, Commonwealth of	0.19979983987189753
6	20200401	South America	Suriname, Republic of	249.55	Brazil, Federative Republic of	0.12185427370387672
7	20210401	Africa	Congo, Republic of the	472.2	Cameroon, Republic of	0.014737062518695783
8	20210401	Asia	Tajikistan, Republic of	287	India, Republic of	0.0008874230797035569
9	20210401	Europe	Faroe Islands	481.5	France, French Republic	0.001785796728049861
10	20210401	North America	Greenland	370.4	United States of America	0.0008497977033331448
11	20210401	Oceania	Fiji, Republic of the Fiji Islands	490.7	Papua New Guinea, Independent State of	0.2168147778332501
12	20210401	South America	Suriname, Republic of	13.313243243243244	Brazil, Federative Republic of	0.0009635592281575945
13	20220401	Africa	Guinea, Republic of	463	Botswana, Republic of	0.002338608812776602
14	20220401	Asia	Macao, Special Administrative Region of China	324.1	Mongolia	0
15	20220401	Europe	Faroe Islands	111.1	France, French Republic	0.00013628130182056675
16	20220401	North America	El Salvador, Republic of	332.2	United States of America	0.0010650926731927328
17	20220401	Oceania	Kiribati, Republic of	106.1375	Australia, Commonwealth of	0.000745880681107425
18	20220401	South America	Guyana, Co-operative Republic of	5.18	Brazil, Federative Republic of	0.0013755748352534015

💡 Worth noting is that although I considered Date = 20221201, there is no data for Date = 20221201. Therefore, there is no output result for that date.