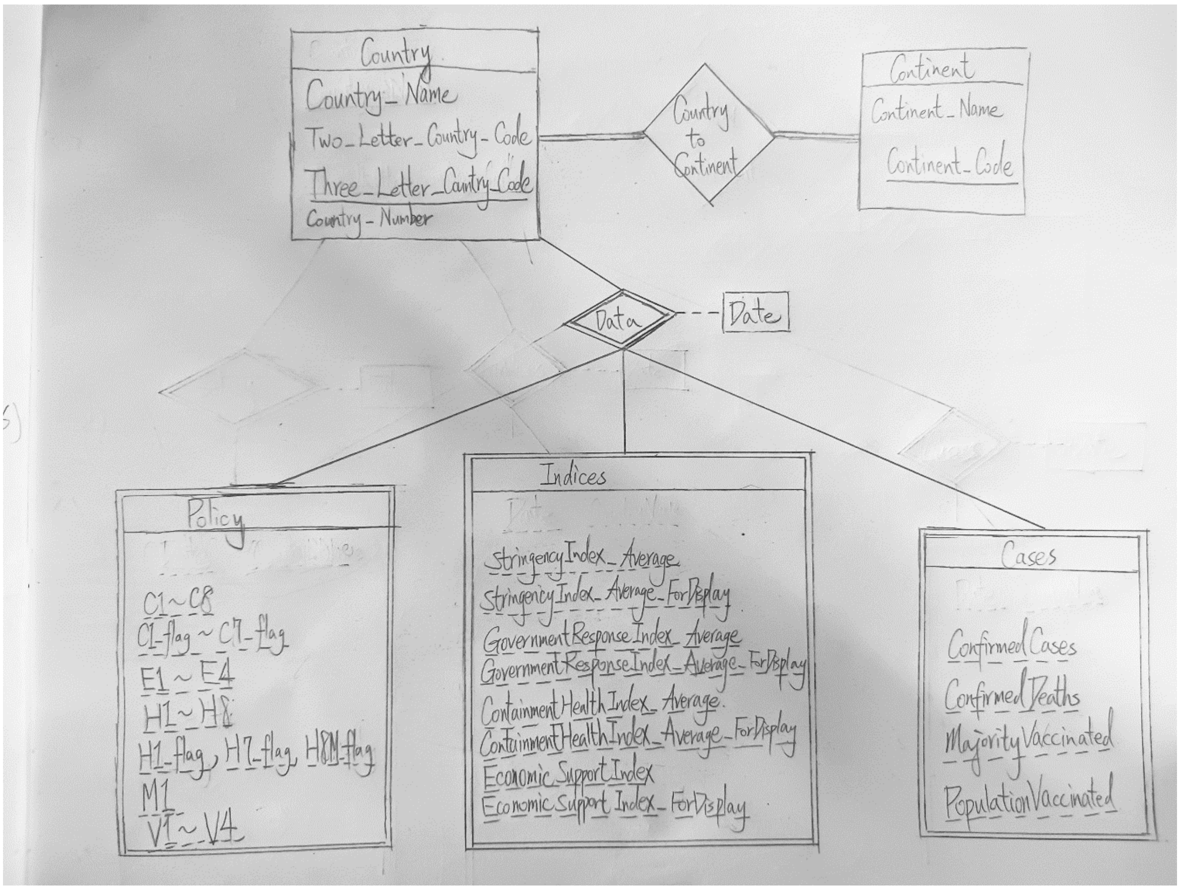
Database HW2

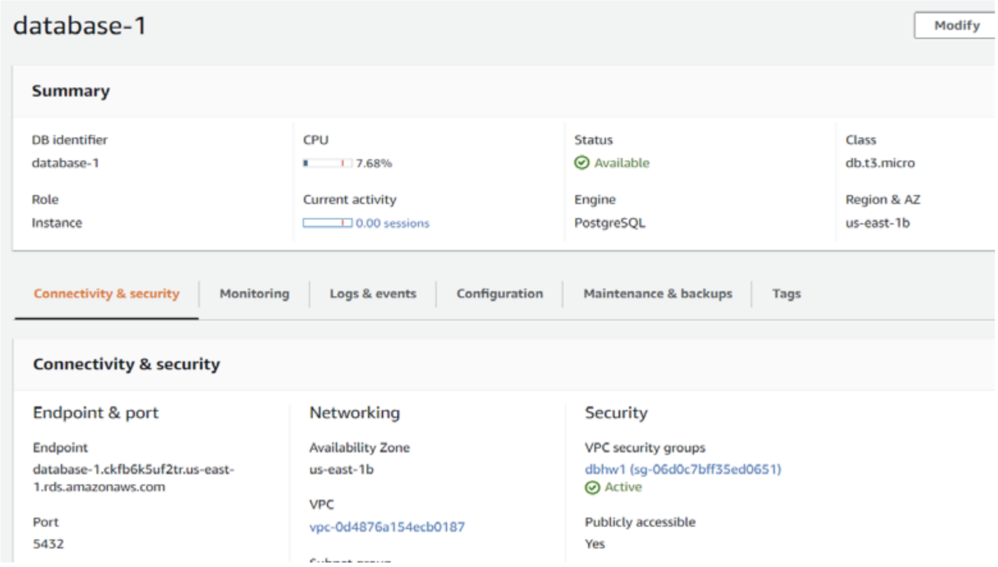
蔡師睿 110550093

1. ER diagram with entity sets and relationship sets, with or without attributes. Add constraints if needed.

I remove the columns, *"RegionName"*, *"RegionCode"*, and *"Jurisdiction,"* in *"OxCGRT_nat_latest.csv."* Because *"RegionName"* and *"RegionCode"* are all null in the data, and *"Jurisdiction"* are all NAT_TOTAL in the data. In *"country-and-continent-codes-list-csv.csv,"* I use *"Three_Letter_Country_Code"* to be the primary key, but some countries' *"Three_Letter_Country_Code"* are null; thus, I replace them with their *"Two_Letter_Country_Code."* Moreover, I add the country, Kosovo, in the csv file manually, for Kosovo isn't in *"country-and-continent-codes-list-csv.csv."*



2. **Provide print screens of the 1) AWS RDS lunch page, and 2) the way you connect to the AWS RDS.**



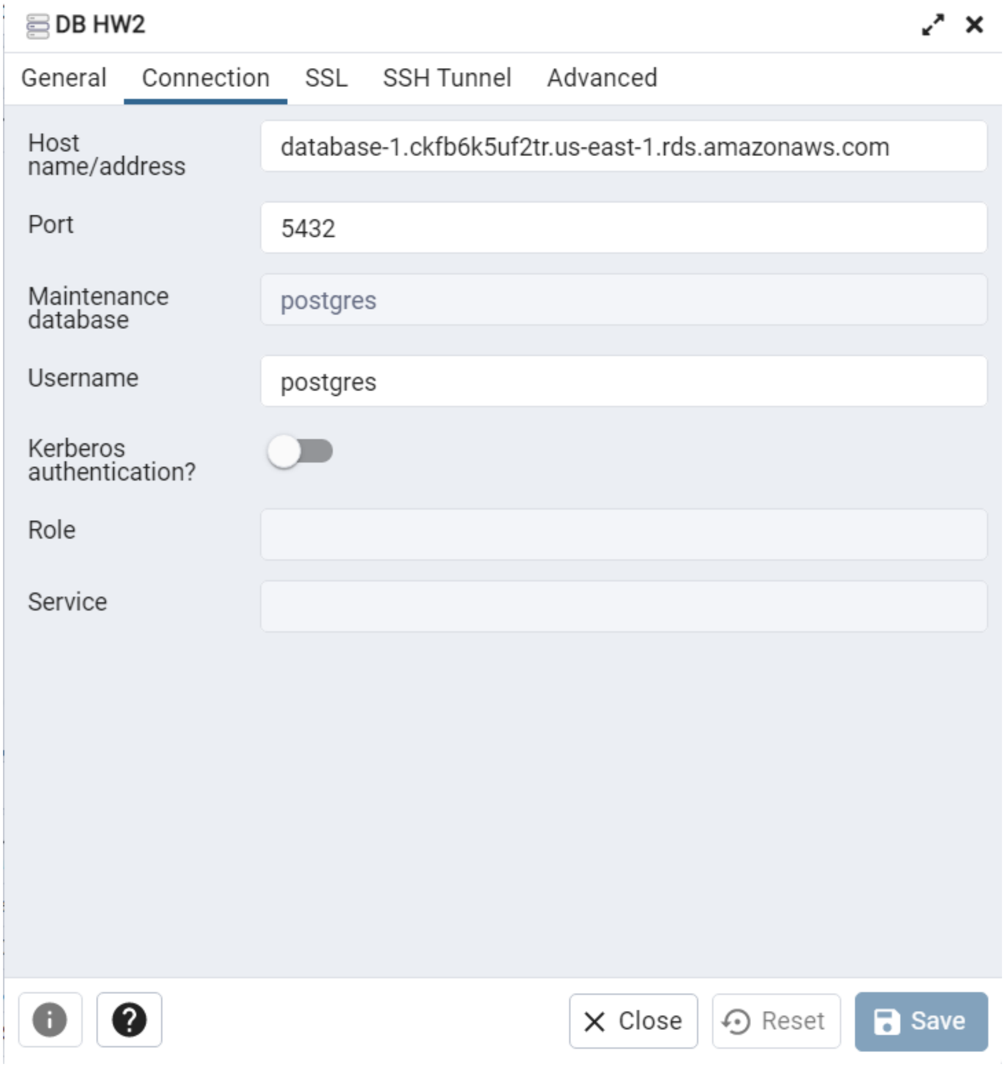
database-1

Summary

DB identifier database-1	CPU 7.68%	Status Available	Class db.t3.micro
Role Instance	Current activity 0.00 sessions	Engine PostgreSQL	Region & AZ us-east-1b

Connectivity & security

Endpoint & port	Networking	Security
Endpoint database-1.ckfb6k5uf2tr.us-east-1.rds.amazonaws.com	Availability Zone us-east-1b	VPC security groups dbhw1 (sg-06d0c7bff35ed0651) Active
Port 5432	VPC vpc-0d4876a154ecb0187	Publicly accessible Yes

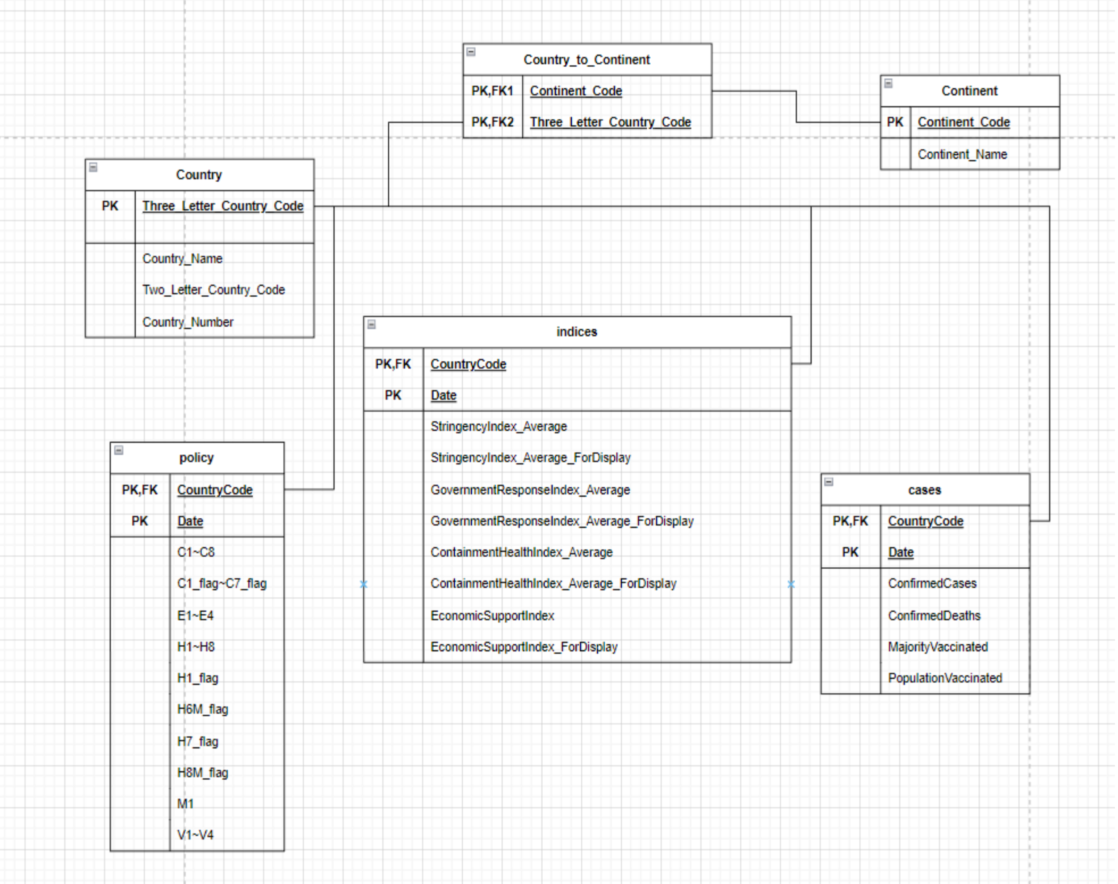


DB HW2

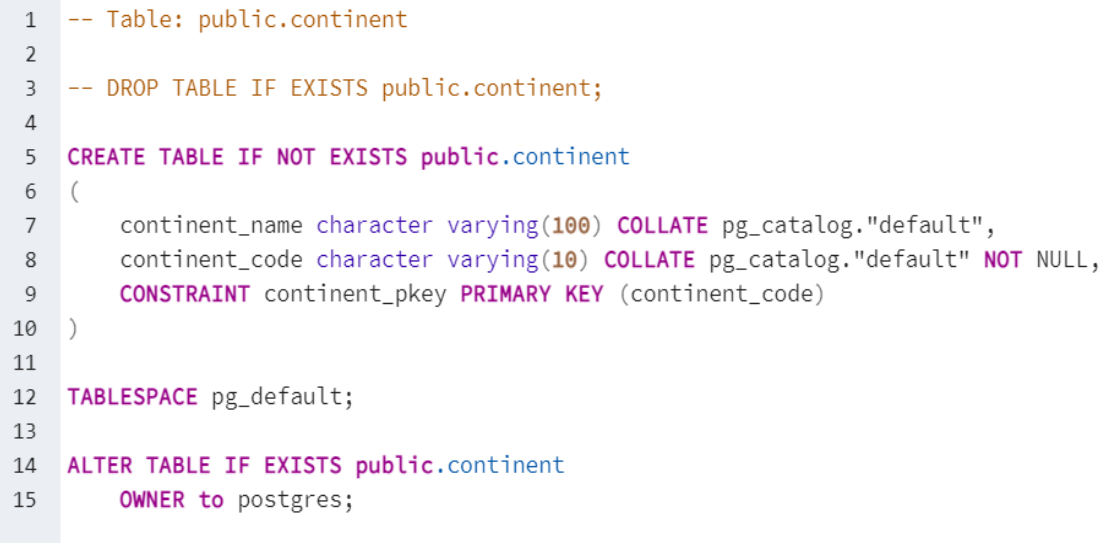
Host name/address	database-1.ckfb6k5uf2tr.us-east-1.rds.amazonaws.com
Port	5432
Maintenance database	postgres
Username	postgres
Kerberos authentication?	○
Role	
Service	

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.

Schema

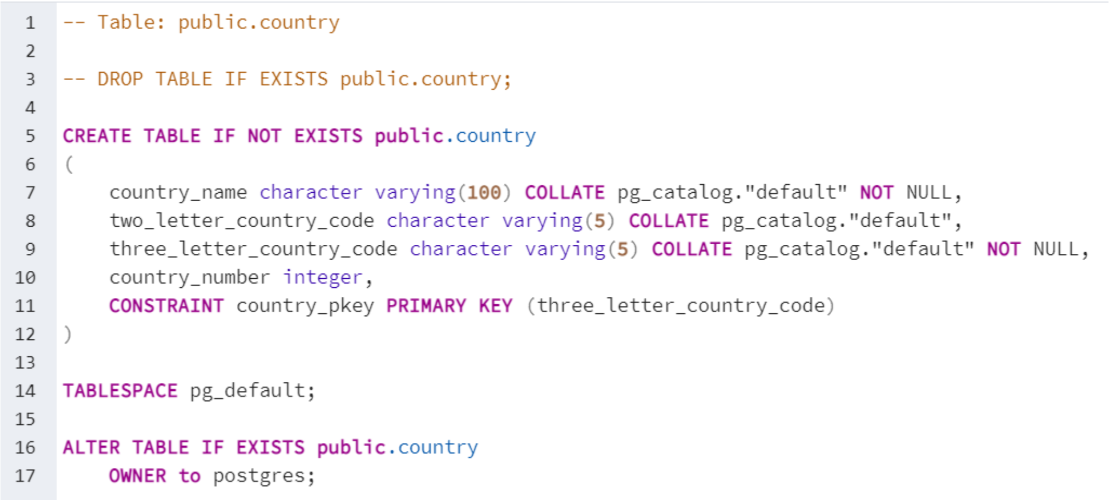


Continent



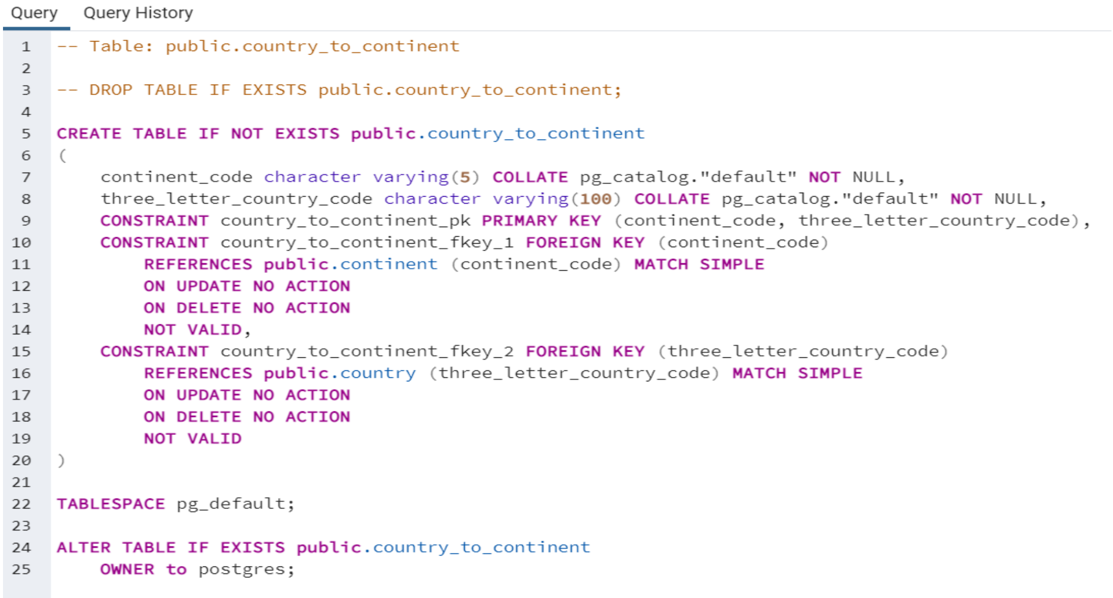
```sql
-- Table: public.continent

-- DROP TABLE IF EXISTS public.continent;

CREATE TABLE IF NOT EXISTS public.continent
(
    continent_name character varying(100) COLLATE pg_catalog."default",
    continent_code character varying(10) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT continent_pkey PRIMARY KEY (continent_code)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.continent
    OWNER to postgres;
```

Country



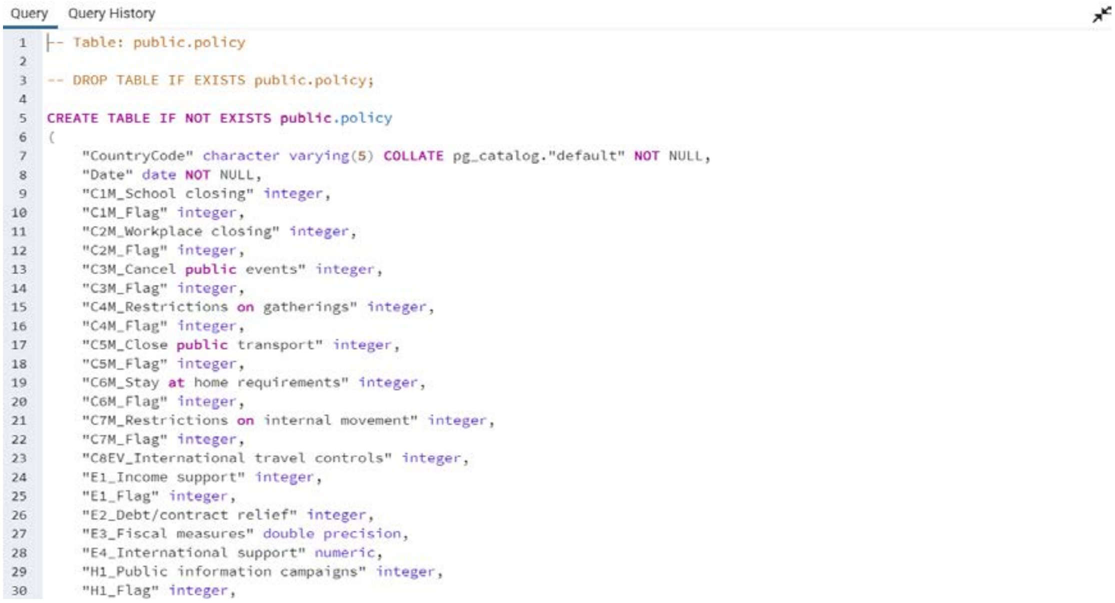
```sql
-- Table: public.country

-- DROP TABLE IF EXISTS public.country;

CREATE TABLE IF NOT EXISTS public.country
(
    country_name character varying(100) COLLATE pg_catalog."default" NOT NULL,
    two_letter_country_code character varying(5) COLLATE pg_catalog."default",
    three_letter_country_code character varying(5) COLLATE pg_catalog."default" NOT NULL,
    country_number integer,
    CONSTRAINT country_pkey PRIMARY KEY (three_letter_country_code)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.country
    OWNER to postgres;
```

Country_to_continent

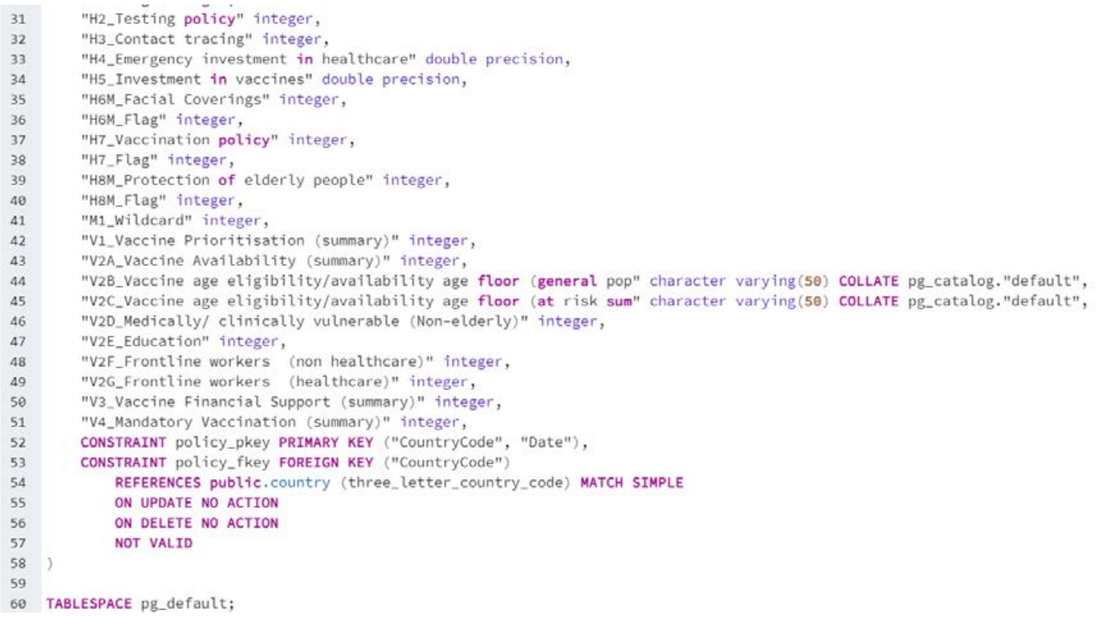


```sql
-- Table: public.country_to_continent

-- DROP TABLE IF EXISTS public.country_to_continent;

CREATE TABLE IF NOT EXISTS public.country_to_continent
(
    continent_code character varying(5) COLLATE pg_catalog."default" NOT NULL,
    three_letter_country_code character varying(100) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT country_to_continent_pk PRIMARY KEY (continent_code, three_letter_country_code),
    CONSTRAINT country_to_continent_fkey_1 FOREIGN KEY (continent_code)
        REFERENCES public.continent (continent_code) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT country_to_continent_fkey_2 FOREIGN KEY (three_letter_country_code)
        REFERENCES public.country (three_letter_country_code) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.country_to_continent
    OWNER to postgres;
```

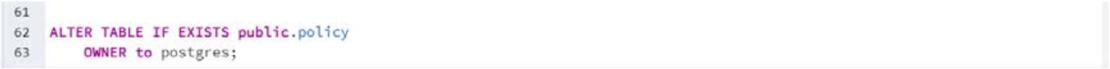
Policy



```sql
-- Table: public.policy

-- DROP TABLE IF EXISTS public.policy;

CREATE TABLE IF NOT EXISTS public.policy
(
    "CountryCode" character varying(5) COLLATE pg_catalog."default" NOT NULL,
    "Date" date NOT NULL,
    "C1M_School closing" integer,
    "C1M_Flag" integer,
    "C2M_Workplace closing" integer,
    "C2M_Flag" integer,
    "C3M_Cancel public events" integer,
    "C3M_Flag" integer,
    "C4M_Restrictions on gatherings" integer,
    "C4M_Flag" integer,
    "C5M_Close public transport" integer,
    "C5M_Flag" integer,
    "C6M_Stay at home requirements" integer,
    "C6M_Flag" integer,
    "C7M_Restrictions on internal movement" integer,
    "C7M_Flag" integer,
    "C8EV_International travel controls" integer,
    "E1_Income support" integer,
    "E1_Flag" integer,
    "E2_Debt/contract relief" integer,
    "E3_Fiscal measures" double precision,
    "E4_International support" numeric,
    "H1_Public information campaigns" integer,
    "H1_Flag" integer,
```

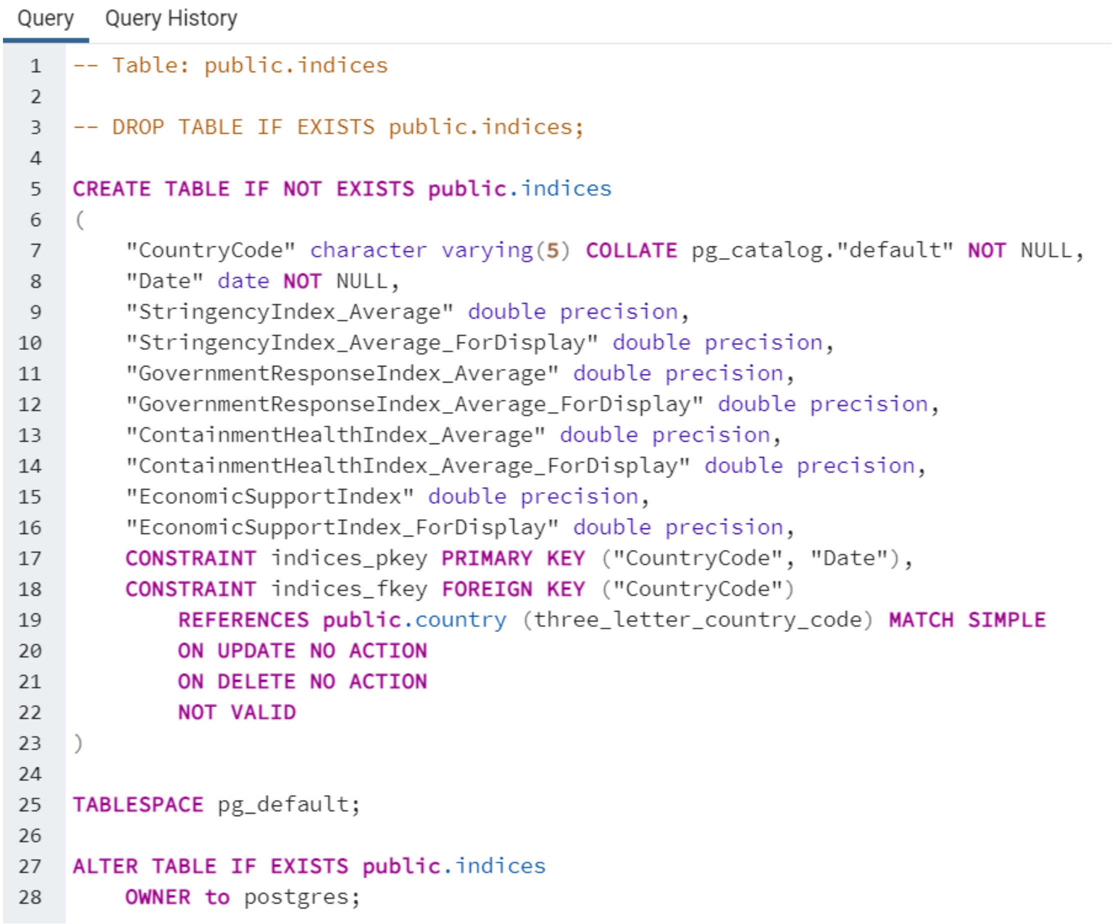


```
    "H2_Testing policy" integer,
    "H3_Contact tracing" integer,
    "H4_Emergency investment in healthcare" double precision,
    "H5_Investment in vaccines" double precision,
    "H6M_Facial Coverings" integer,
    "H6M_Flag" integer,
    "H7_Vaccination policy" integer,
    "H7_Flag" integer,
    "H8M_Protection of elderly people" integer,
    "H8M_Flag" integer,
    "M1_Wildcard" integer,
    "V1_Vaccine Prioritisation (summary)" integer,
    "V2A_Vaccine Availability (summary)" integer,
    "V2B_Vaccine age eligibility/availability age floor (general pop" character varying(50) COLLATE pg_catalog."default",
    "V2C_Vaccine age eligibility/availability age floor (at risk sum" character varying(50) COLLATE pg_catalog."default",
    "V2D_Medically/ clinically vulnerable (Non-elderly)" integer,
    "V2E_Education" integer,
    "V2F_Frontline workers  (non healthcare)" integer,
    "V2G_Frontline workers  (healthcare)" integer,
    "V3_Vaccine Financial Support (summary)" integer,
    "V4_Mandatory Vaccination (summary)" integer,
    CONSTRAINT policy_pkey PRIMARY KEY ("CountryCode", "Date"),
    CONSTRAINT policy_fkey FOREIGN KEY ("CountryCode")
        REFERENCES public.country (three_letter_country_code) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;
```



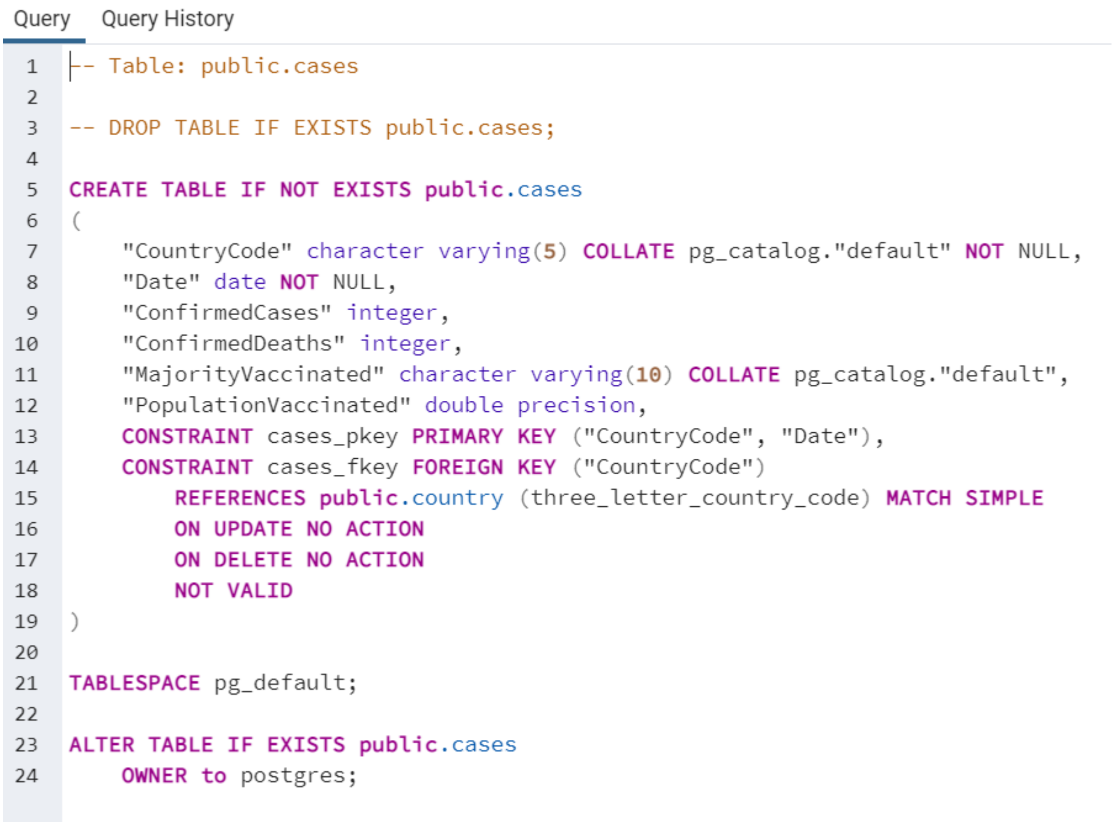
```

ALTER TABLE IF EXISTS public.policy
    OWNER to postgres;
```

Indices



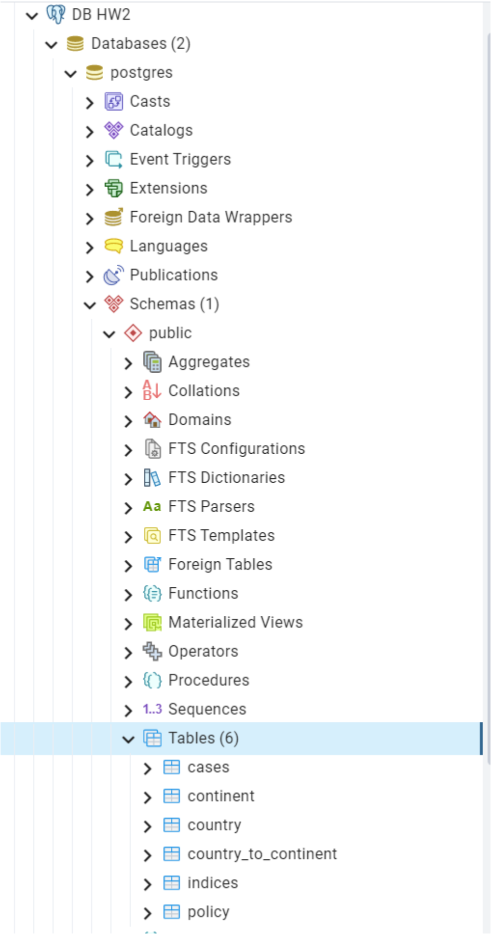
```
-- Table: public.indices

-- DROP TABLE IF EXISTS public.indices;

CREATE TABLE IF NOT EXISTS public.indices
(
    "CountryCode" character varying(5) COLLATE pg_catalog."default" NOT NULL,
    "Date" date NOT NULL,
    "StringencyIndex_Average" double precision,
    "StringencyIndex_Average_ForDisplay" double precision,
    "GovernmentResponseIndex_Average" double precision,
    "GovernmentResponseIndex_Average_ForDisplay" double precision,
    "ContainmentHealthIndex_Average" double precision,
    "ContainmentHealthIndex_Average_ForDisplay" double precision,
    "EconomicSupportIndex" double precision,
    "EconomicSupportIndex_ForDisplay" double precision,
    CONSTRAINT indices_pkey PRIMARY KEY ("CountryCode", "Date"),
    CONSTRAINT indices_fkey FOREIGN KEY ("CountryCode")
        REFERENCES public.country (three_letter_country_code) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.indices
    OWNER to postgres;
```

Cases



```sql
-- Table: public.cases

-- DROP TABLE IF EXISTS public.cases;

CREATE TABLE IF NOT EXISTS public.cases
(
    "CountryCode" character varying(5) COLLATE pg_catalog."default" NOT NULL,
    "Date" date NOT NULL,
    "ConfirmedCases" integer,
    "ConfirmedDeaths" integer,
    "MajorityVaccinated" character varying(10) COLLATE pg_catalog."default",
    "PopulationVaccinated" double precision,
    CONSTRAINT cases_pkey PRIMARY KEY ("CountryCode", "Date"),
    CONSTRAINT cases_fkey FOREIGN KEY ("CountryCode")
        REFERENCES public.country (three_letter_country_code) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.cases
    OWNER to postgres;
```

Created Tables



4. Clearly indicate the level of normal form, test the level of normal form for each table.

Country(*Three_Letter_Country_Code, Country_Name, Two_Letter_Coutnry_Code, Country_Number*)

Normal Form : BCNF

Test :

The primary key of this table is *Three_Letter_Country_Code*.

I. {*Three_Letter_Country_Code*} is super key.

II. {*Three_Letter_Country_Code*} isn't included in { *Country_Name, Two_Letter_Coutnry_Code, Country_Number*}.

Continent(*Continent_Code, Continent_Name*)

Normal Form : BCNF

Test :

The primary key of this table is *Continent_Code*.

I. Both {*Continent_Code*} and {*Continent_Name*} are superkeys.

Country_to_Continent(*Continent_Code, Three_Letter_Coutnry_Code*)

Normal Form : BCNF

Test :

The primary key of this table is {*Continent_Code, Three_Letter_Country_Code*}

I. The table only contains primary key, so it corresponds to BCNF.

Policy(*CountryCode, Date, C1_School closing, C1_flag, …*)

Normal Form : BCNF

Test :

The primary key of this table is {*CountryCode, Date*}.

I. {*CountryCode, Date*} is super key.

II. {*CountryCode, Date*} isn't include in {*C1_School closing, C1_flag, …*}.

III. {*CountryCode*} and {*Date*} both don't have functional dependency with the other attributes.

Indices(*CountryCode, Date, StringencyIndex_Average, …, EconomicSupportIndex_ForDisplay*)

Normal Form : BCNF

Test :

The primary key is {*CountryCode, Date*}.

I. {*CountryCode, Date*} is super key.

II. {*CountryCode, Date*} isn't include in { *StringencyIndex_Average, …, EconomicSupportIndex_ForDisplay*}.

III. {*CountryCode*} and {*Date*} both don't have functional dependency with the other attributes.

Cases(*CountryCode, Date, ConfirmedCases, …, PopulationVaccinated*)

Normal Form : BCNF

Test :

The primary key is {*CountryCode, Date*}

I. {*CountryCode, Date*} is super key.

II. {*CountryCode, Date*} isn't include in { *ConfirmedCases, …, PopulationVaccinated*}.

III. {*CountryCode*} and {*Date*} both don't have functional dependency with the other attributes.

5. List the functional dependency of each table.

Country

- *Three_Letter_Country_Code*→{*Country_Name, Two_Letter_Country_Code, Country_Number*}
- {*Country_Name, Two_Letter_Country_Code, Country_Number*}→ *Three_Letter_Country_Code*

Continent

- *Continent_Code*→*Continent_Name*
- *Continent_Name*→*Continent_Code*

Country_to_Continent

- No functional dependency, because the table only contains primary key.

Policy

- *{CountryCode, Date}*→*{C1_School closing, C1_flag,…}*

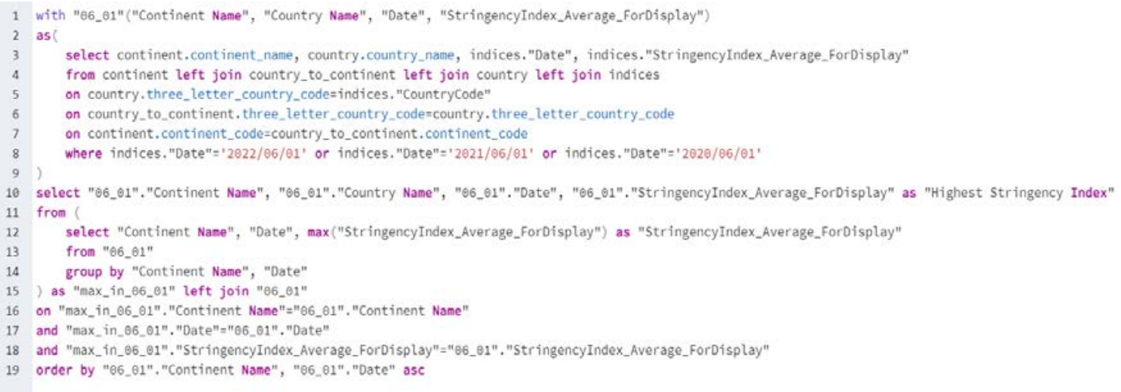
Indices

- *{CountryCode, Date}*→*{StringencyIndex_Average, …, EconomicSupportIndex_ForDisplay}*

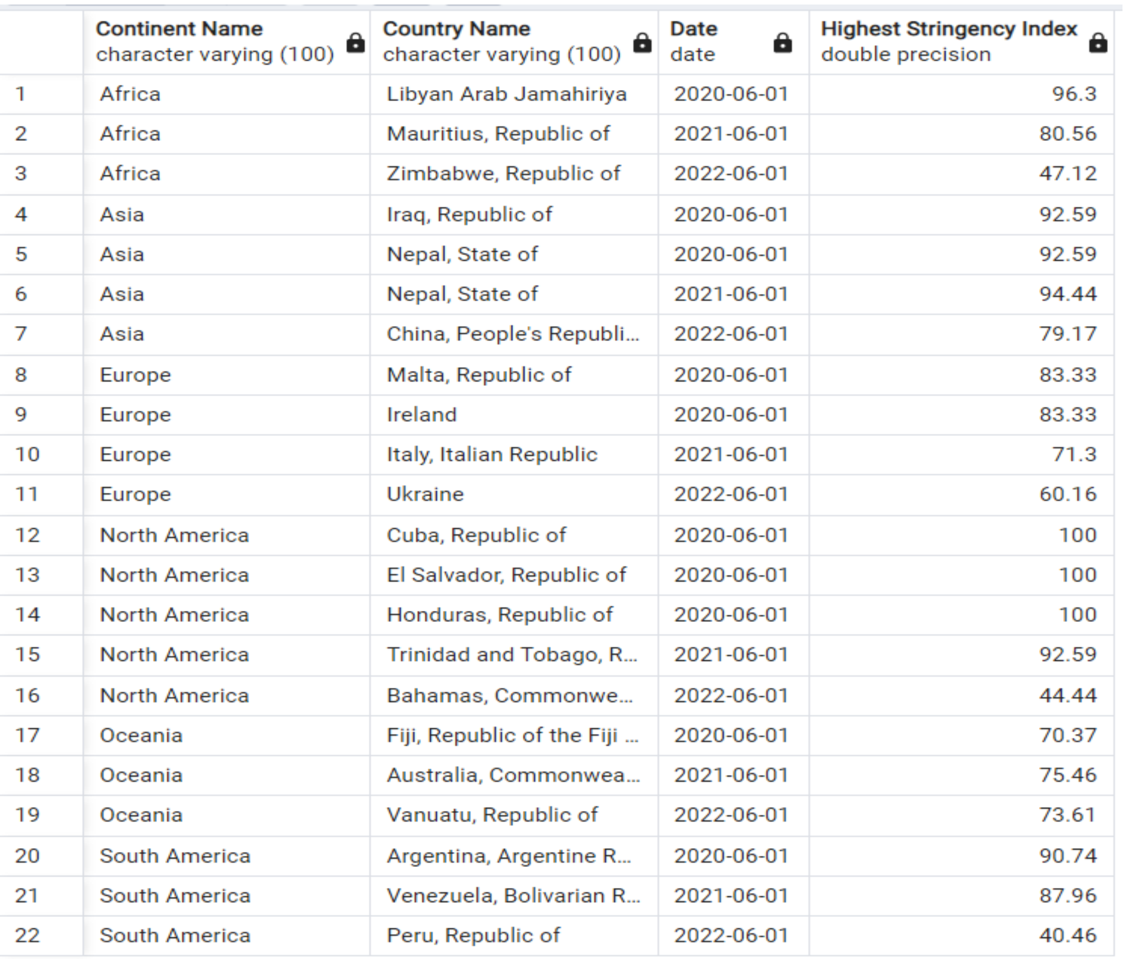
Cases

- *{CountryCode, Date}*→*{ConfirmedCases, …, PopulationVaccinated}*

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



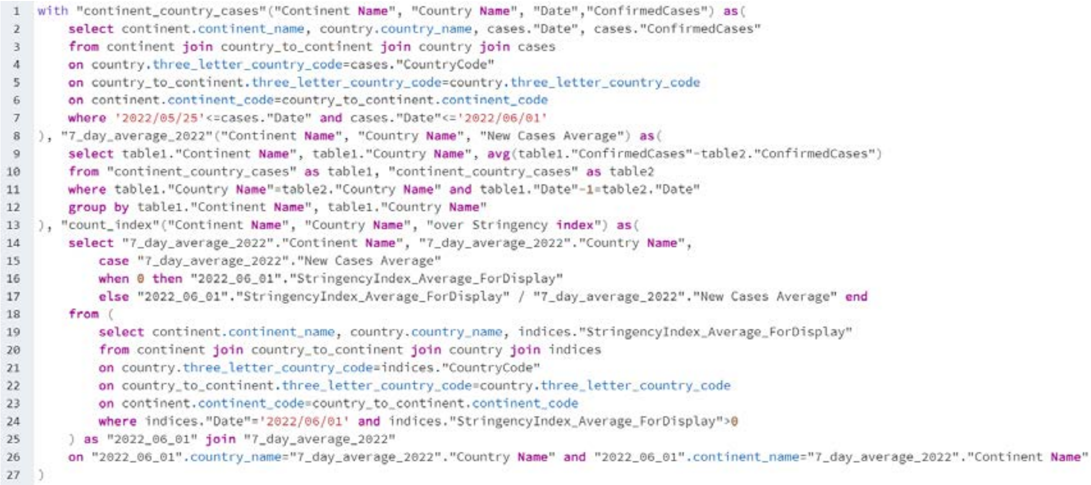
```sql
with "06_01"("Continent Name", "Country Name", "Date", "StringencyIndex_Average_ForDisplay")
as(
    select continent.continent_name, country.country_name, indices."Date", indices."StringencyIndex_Average_ForDisplay"
    from continent left join country_to_continent left join country left join indices
    on country.three_letter_country_code=indices."CountryCode"
    on country_to_continent.three_letter_country_code=country.three_letter_country_code
    on continent.continent_code=country_to_continent.continent_code
    where indices."Date"='2022/06/01' or indices."Date"='2021/06/01' or indices."Date"='2020/06/01'
)
select "06_01"."Continent Name", "06_01"."Country Name", "06_01"."Date", "06_01"."StringencyIndex_Average_ForDisplay" as "Highest Stringency Index"
from (
    select "Continent Name", "Date", max("StringencyIndex_Average_ForDisplay") as "StringencyIndex_Average_ForDisplay"
    from "06_01"
    group by "Continent Name", "Date"
) as "max_in_06_01" left join "06_01"
on "max_in_06_01"."Continent Name"="06_01"."Continent Name"
and "max_in_06_01"."Date"="06_01"."Date"
and "max_in_06_01"."StringencyIndex_Average_ForDisplay"="06_01"."StringencyIndex_Average_ForDisplay"
order by "06_01"."Continent Name", "06_01"."Date" asc
```



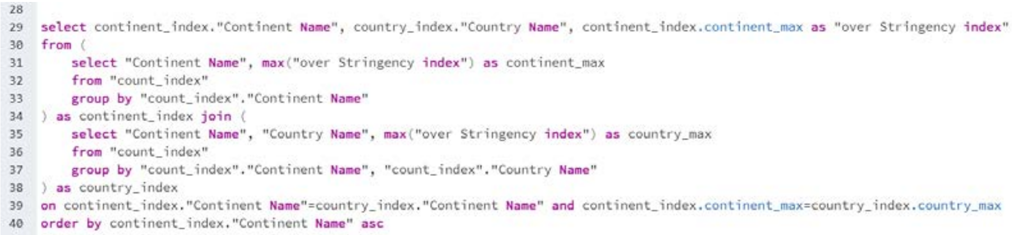
	Continent Name character varying (100)	Country Name character varying (100)	Date date	Highest Stringency Index double precision
1	Africa	Libyan Arab Jamahiriya	2020-06-01	96.3
2	Africa	Mauritius, Republic of	2021-06-01	80.56
3	Africa	Zimbabwe, Republic of	2022-06-01	47.12
4	Asia	Iraq, Republic of	2020-06-01	92.59
5	Asia	Nepal, State of	2020-06-01	92.59
6	Asia	Nepal, State of	2021-06-01	94.44
7	Asia	China, People's Republi...	2022-06-01	79.17
8	Europe	Malta, Republic of	2020-06-01	83.33
9	Europe	Ireland	2020-06-01	83.33
10	Europe	Italy, Italian Republic	2021-06-01	71.3
11	Europe	Ukraine	2022-06-01	60.16
12	North America	Cuba, Republic of	2020-06-01	100
13	North America	El Salvador, Republic of	2020-06-01	100
14	North America	Honduras, Republic of	2020-06-01	100
15	North America	Trinidad and Tobago, R...	2021-06-01	92.59
16	North America	Bahamas, Commonwe...	2022-06-01	44.44
17	Oceania	Fiji, Republic of the Fiji ...	2020-06-01	70.37
18	Oceania	Australia, Commonwea...	2021-06-01	75.46
19	Oceania	Vanuatu, Republic of	2022-06-01	73.61
20	South America	Argentina, Argentine R...	2020-06-01	90.74
21	South America	Venezuela, Bolivarian R...	2021-06-01	87.96
22	South America	Peru, Republic of	2022-06-01	40.46

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

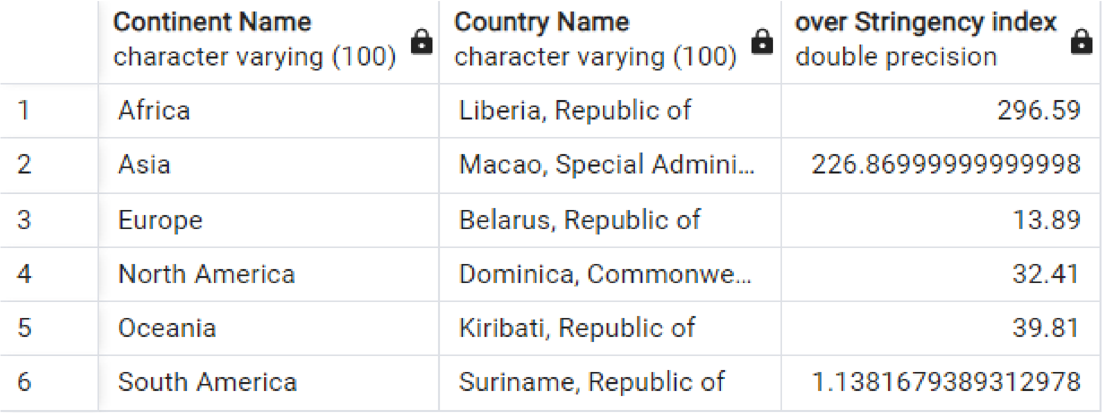
2022/06/01



```sql
with "continent_country_cases"("Continent Name", "Country Name", "Date","ConfirmedCases") as(
    select continent.continent_name, country.country_name, cases."Date", cases."ConfirmedCases"
    from continent join country_to_continent join country join cases
    on country.three_letter_country_code=cases."CountryCode"
    on country_to_continent.three_letter_country_code=country.three_letter_country_code
    on continent.continent_code=country_to_continent.continent_code
    where '2022/05/25'<=cases."Date" and cases."Date"<='2022/06/01'
), "7_day_average_2022"("Continent Name", "Country Name", "New Cases Average") as(
    select table1."Continent Name", table1."Country Name", avg(table1."ConfirmedCases"-table2."ConfirmedCases")
    from "continent_country_cases" as table1, "continent_country_cases" as table2
    where table1."Country Name"=table2."Country Name" and table1."Date"-1=table2."Date"
    group by table1."Continent Name", table1."Country Name"
), "count_index"("Continent Name", "Country Name", "over Stringency index") as(
    select "7_day_average_2022"."Continent Name", "7_day_average_2022"."Country Name",
        case "7_day_average_2022"."New Cases Average"
        when 0 then "2022_06_01"."StringencyIndex_Average_ForDisplay"
        else "2022_06_01"."StringencyIndex_Average_ForDisplay" / "7_day_average_2022"."New Cases Average" end
    from (
        select continent.continent_name, country.country_name, indices."StringencyIndex_Average_ForDisplay"
        from continent join country_to_continent join country join indices
        on country.three_letter_country_code=indices."CountryCode"
        on country_to_continent.three_letter_country_code=country.three_letter_country_code
        on continent.continent_code=country_to_continent.continent_code
        where indices."Date"='2022/06/01' and indices."StringencyIndex_Average_ForDisplay">0
    ) as "2022_06_01" join "7_day_average_2022"
    on "2022_06_01".country_name="7_day_average_2022"."Country Name" and "2022_06_01".continent_name="7_day_average_2022"."Continent Name"
)
```

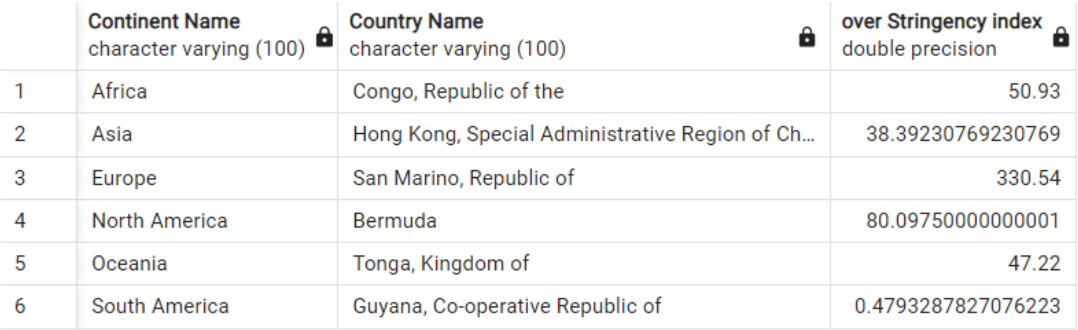


```sql

select continent_index."Continent Name", country_index."Country Name", continent_index.continent_max as "over Stringency index"
from (
    select "Continent Name", max("over Stringency index") as continent_max
    from "count_index"
    group by "count_index"."Continent Name"
) as continent_index join (
    select "Continent Name", "Country Name", max("over Stringency index") as country_max
    from "count_index"
    group by "count_index"."Continent Name", "count_index"."Country Name"
) as country_index
on continent_index."Continent Name"=country_index."Continent Name" and continent_index.continent_max=country_index.country_max
order by continent_index."Continent Name" asc
```

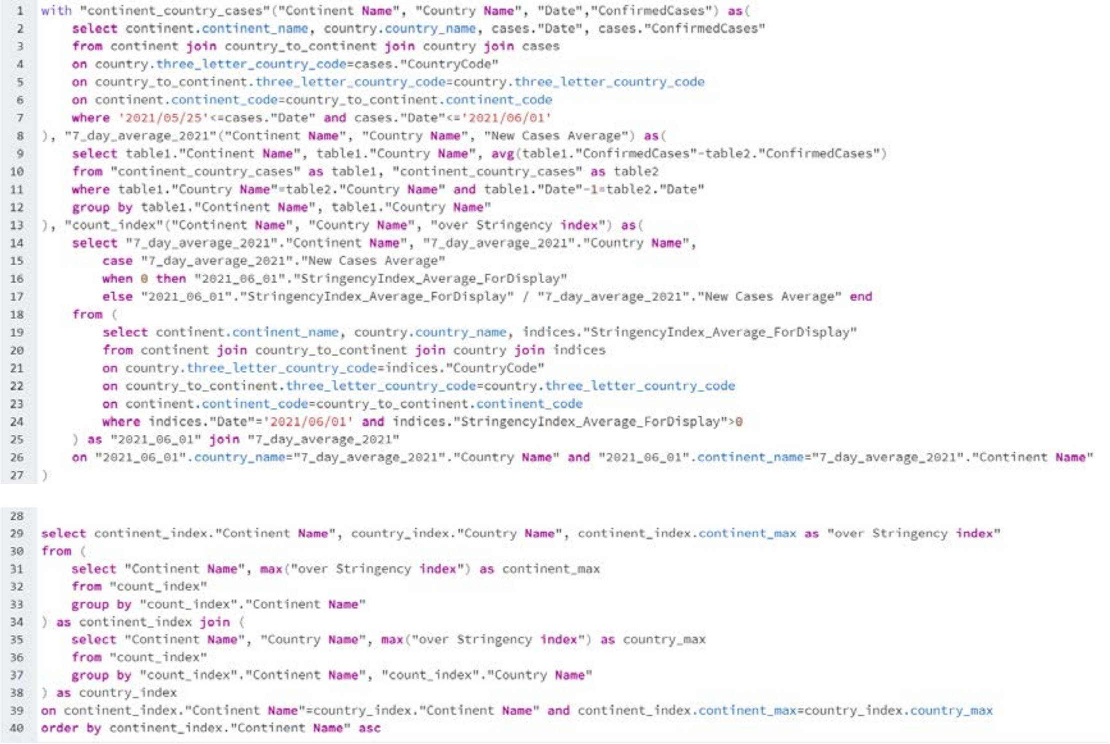


	Continent Name character varying (100)	Country Name character varying (100)	over Stringency index double precision
1	Africa	Liberia, Republic of	296.59
2	Asia	Macao, Special Admini...	226.86999999999998
3	Europe	Belarus, Republic of	13.89
4	North America	Dominica, Commonwe...	32.41
5	Oceania	Kiribati, Republic of	39.81
6	South America	Suriname, Republic of	1.1381679389312978

2021/06/01

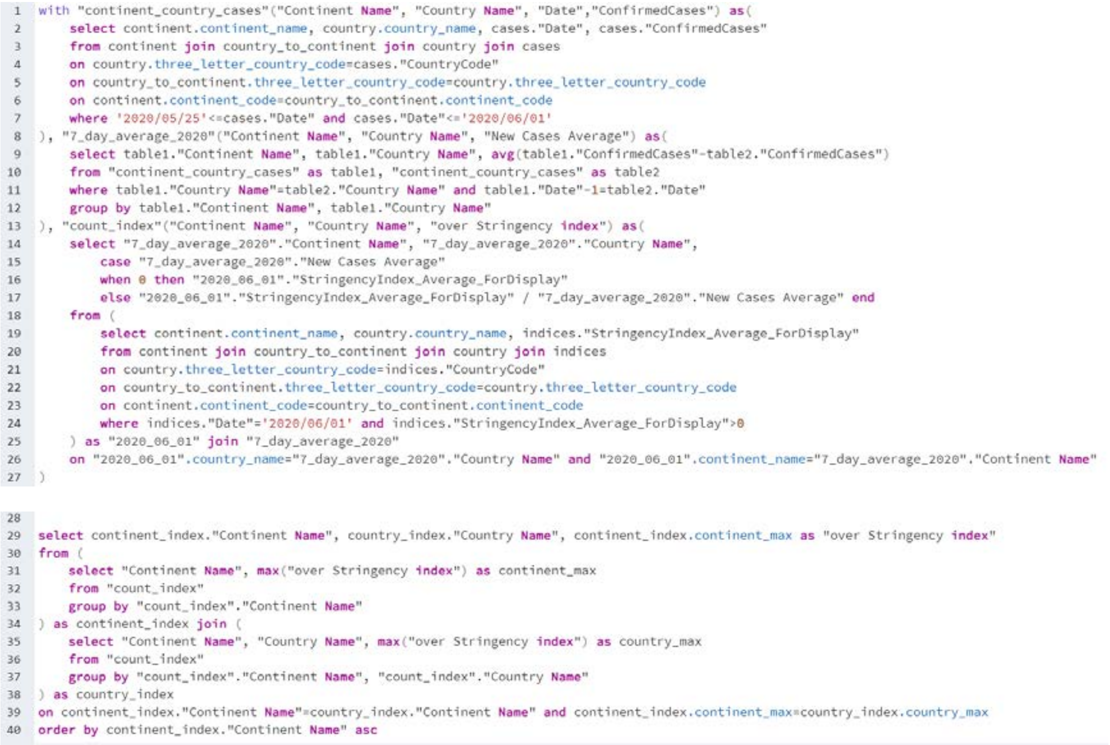


	Continent Name character varying (100)	Country Name character varying (100)	over Stringency index double precision
1	Africa	Congo, Republic of the	50.93
2	Asia	Hong Kong, Special Administrative Region of Ch...	38.39230769230769
3	Europe	San Marino, Republic of	330.54
4	North America	Bermuda	80.09750000000001
5	Oceania	Tonga, Kingdom of	47.22
6	South America	Guyana, Co-operative Republic of	0.4793287827076223

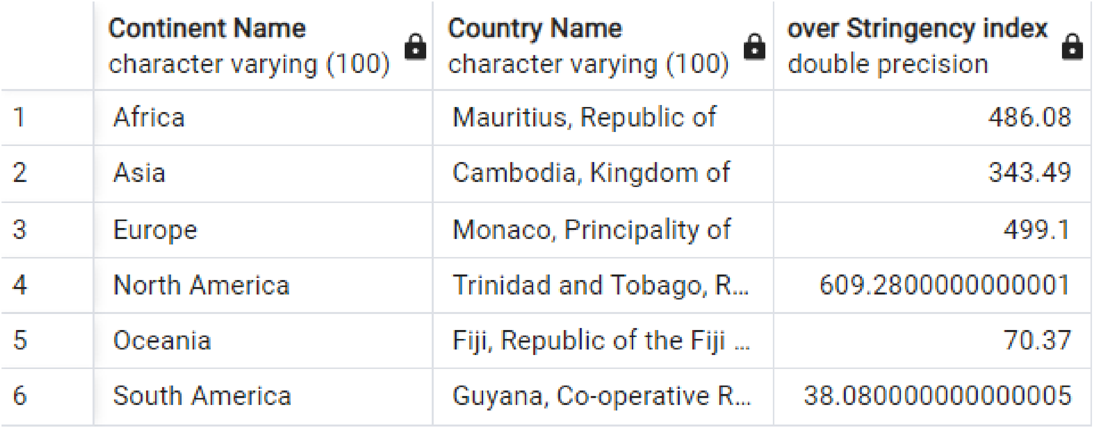


```sql
with "continent_country_cases"("Continent Name", "Country Name", "Date","ConfirmedCases") as(
    select continent.continent_name, country.country_name, cases."Date", cases."ConfirmedCases"
    from continent join country_to_continent join country join cases
    on country.three_letter_country_code=cases."CountryCode"
    on country_to_continent.three_letter_country_code=country.three_letter_country_code
    on continent.continent_code=country_to_continent.continent_code
    where '2021/05/25'<=cases."Date" and cases."Date"<='2021/06/01'
), "7_day_average_2021"("Continent Name", "Country Name", "New Cases Average") as(
    select table1."Continent Name", table1."Country Name", avg(table1."ConfirmedCases"-table2."ConfirmedCases")
    from "continent_country_cases" as table1, "continent_country_cases" as table2
    where table1."Country Name"=table2."Country Name" and table1."Date"-1=table2."Date"
    group by table1."Continent Name", table1."Country Name"
), "count_index"("Continent Name", "Country Name", "over Stringency index") as(
    select "7_day_average_2021"."Continent Name", "7_day_average_2021"."Country Name",
        case "7_day_average_2021"."New Cases Average"
        when 0 then "2021_06_01"."StringencyIndex_Average_ForDisplay"
        else "2021_06_01"."StringencyIndex_Average_ForDisplay" / "7_day_average_2021"."New Cases Average" end
    from (
        select continent.continent_name, country.country_name, indices."StringencyIndex_Average_ForDisplay"
        from continent join country_to_continent join country join indices
        on country.three_letter_country_code=indices."CountryCode"
        on country_to_continent.three_letter_country_code=country.three_letter_country_code
        on continent.continent_code=country_to_continent.continent_code
        where indices."Date"='2021/06/01' and indices."StringencyIndex_Average_ForDisplay">0
    ) as "2021_06_01" join "7_day_average_2021"
    on "2021_06_01".country_name="7_day_average_2021"."Country Name" and "2021_06_01".continent_name="7_day_average_2021"."Continent Name"
)

select continent_index."Continent Name", country_index."Country Name", continent_index.continent_max as "over Stringency index"
from (
    select "Continent Name", max("over Stringency index") as continent_max
    from "count_index"
    group by "count_index"."Continent Name"
) as continent_index join (
    select "Continent Name", "Country Name", max("over Stringency index") as country_max
    from "count_index"
    group by "count_index"."Continent Name", "count_index"."Country Name"
) as country_index
on continent_index."Continent Name"=country_index."Continent Name" and continent_index.continent_max=country_index.country_max
order by continent_index."Continent Name" asc
```

2020/06/01



```sql
with "continent_country_cases"("Continent Name", "Country Name", "Date","ConfirmedCases") as(
    select continent.continent_name, country.country_name, cases."Date", cases."ConfirmedCases"
    from continent join country_to_continent join country join cases
    on country.three_letter_country_code=cases."CountryCode"
    on country_to_continent.three_letter_country_code=country.three_letter_country_code
    on continent.continent_code=country_to_continent.continent_code
    where '2020/05/25'<=cases."Date" and cases."Date"<='2020/06/01'
), "7_day_average_2020"("Continent Name", "Country Name", "New Cases Average") as(
    select table1."Continent Name", table1."Country Name", avg(table1."ConfirmedCases"-table2."ConfirmedCases")
    from "continent_country_cases" as table1, "continent_country_cases" as table2
    where table1."Country Name"=table2."Country Name" and table1."Date"-1=table2."Date"
    group by table1."Continent Name", table1."Country Name"
), "count_index"("Continent Name", "Country Name", "over Stringency index") as(
    select "7_day_average_2020"."Continent Name", "7_day_average_2020"."Country Name",
        case "7_day_average_2020"."New Cases Average"
        when 0 then "2020_06_01"."StringencyIndex_Average_ForDisplay"
        else "2020_06_01"."StringencyIndex_Average_ForDisplay" / "7_day_average_2020"."New Cases Average" end
    from (
        select continent.continent_name, country.country_name, indices."StringencyIndex_Average_ForDisplay"
        from continent join country_to_continent join country join indices
        on country.three_letter_country_code=indices."CountryCode"
        on country_to_continent.three_letter_country_code=country.three_letter_country_code
        on continent.continent_code=country_to_continent.continent_code
        where indices."Date"='2020/06/01' and indices."StringencyIndex_Average_ForDisplay">0
    ) as "2020_06_01" join "7_day_average_2020"
    on "2020_06_01".country_name="7_day_average_2020"."Country Name" and "2020_06_01".continent_name="7_day_average_2020"."Continent Name"
)

select continent_index."Continent Name", country_index."Country Name", continent_index.continent_max as "over Stringency index"
from (
    select "Continent Name", max("over Stringency index") as continent_max
    from "count_index"
    group by "count_index"."Continent Name"
) as continent_index join (
    select "Continent Name", "Country Name", max("over Stringency index") as country_max
    from "count_index"
    group by "count_index"."Continent Name", "count_index"."Country Name"
) as country_index
on continent_index."Continent Name"=country_index."Continent Name" and continent_index.continent_max=country_index.country_max
order by continent_index."Continent Name" asc
```



	Continent Name character varying (100)	Country Name character varying (100)	over Stringency index double precision
1	Africa	Mauritius, Republic of	486.08
2	Asia	Cambodia, Kingdom of	343.49
3	Europe	Monaco, Principality of	499.1
4	North America	Trinidad and Tobago, R...	609.2800000000001
5	Oceania	Fiji, Republic of the Fiji ...	70.37
6	South America	Guyana, Co-operative R...	38.080000000000005