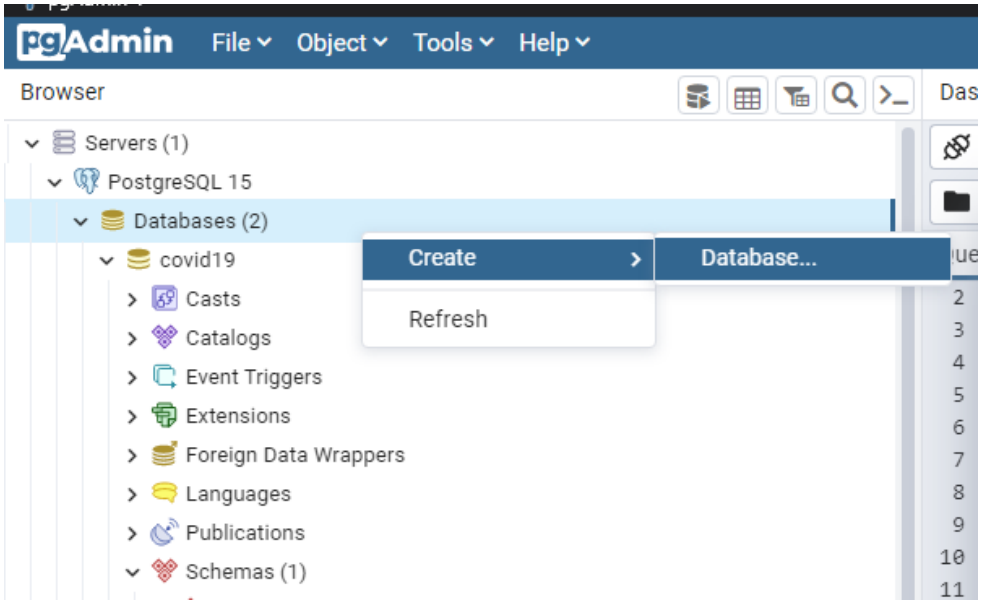
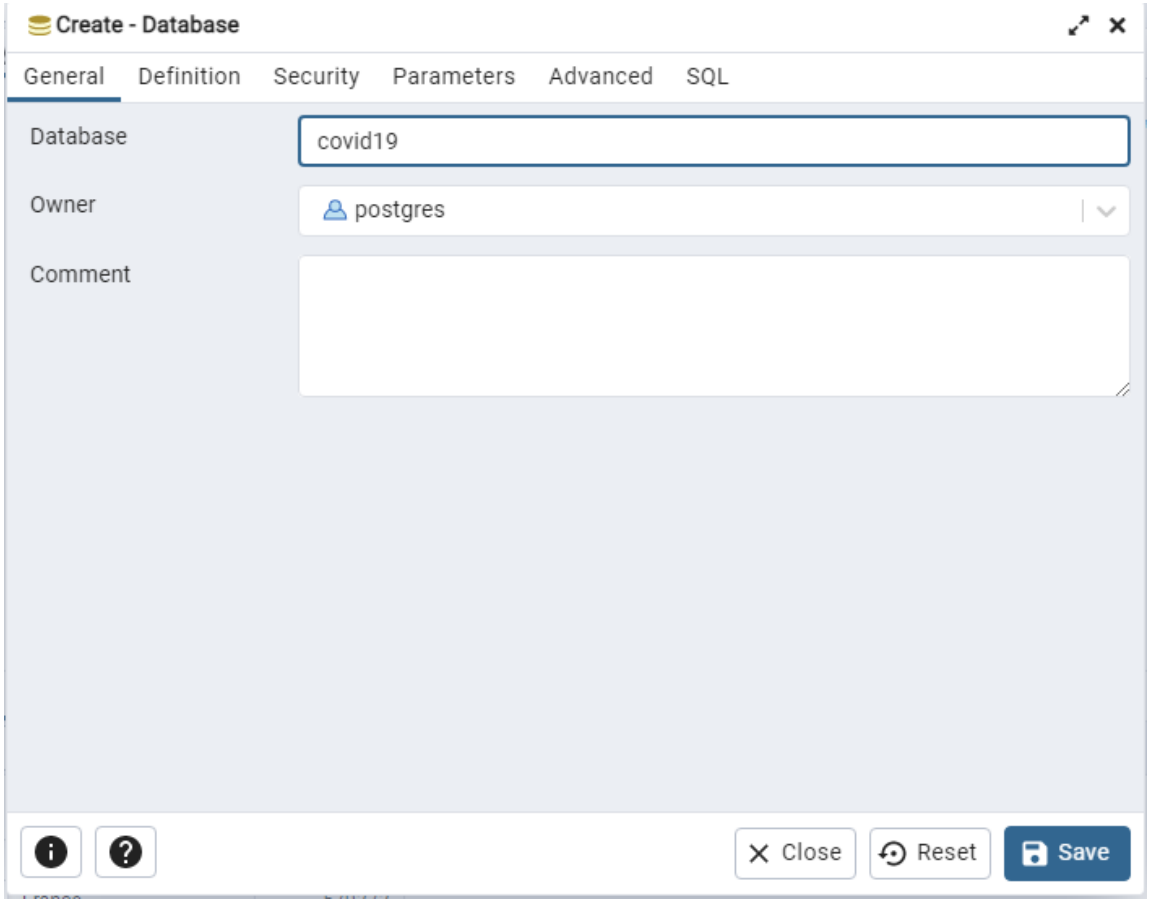
Database HW1

李杰穎 110550088

1. The process of creating the “covid19” databases (can be screenshot and/or SQL/non-SQL statements with text explanation)

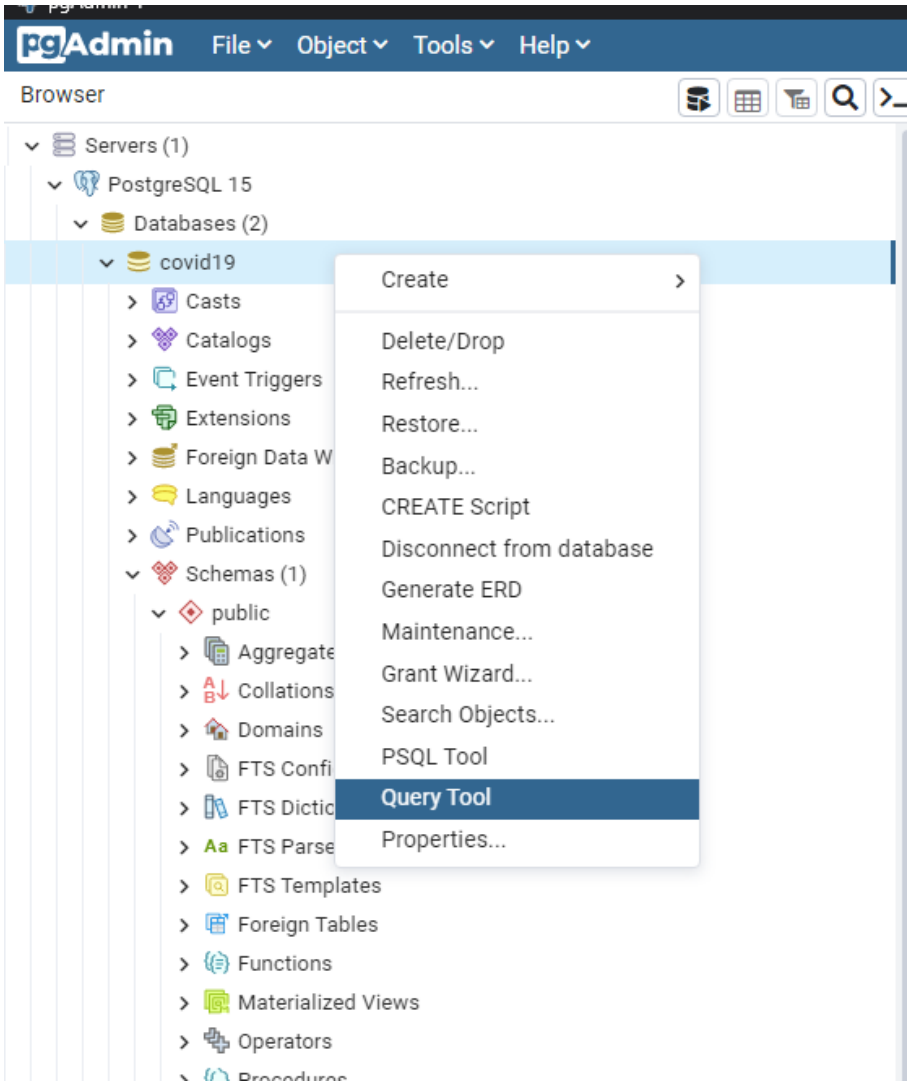
Open pgAdmin 4, right click on `Databases` section, and select `Create > Database`. Next, in the newly opened window, fill in the name of database. In this homework, the name of database is `covid19`. I keep other configurations as default. Finally, click `Save` to create database.





2. The process of importing three required .csv files into covid19 database (can be screenshot and/or SQL/non-SQL statements with text explanation). Please included/described the data type and keys of the imported table in your screenshot, SQL statements, and explanations

The following processes are done in Query Tool in pgAdmin 4. To open Query Tool, right click on the `covid19` database, and select Query Tool.



Create table for data in 2022/10/01

Before importing the data from the given csv file, we need to create schema for the table. The SQL for creating this table is as below.

`COLLATE` in character varying means that those columns will use the default collation.

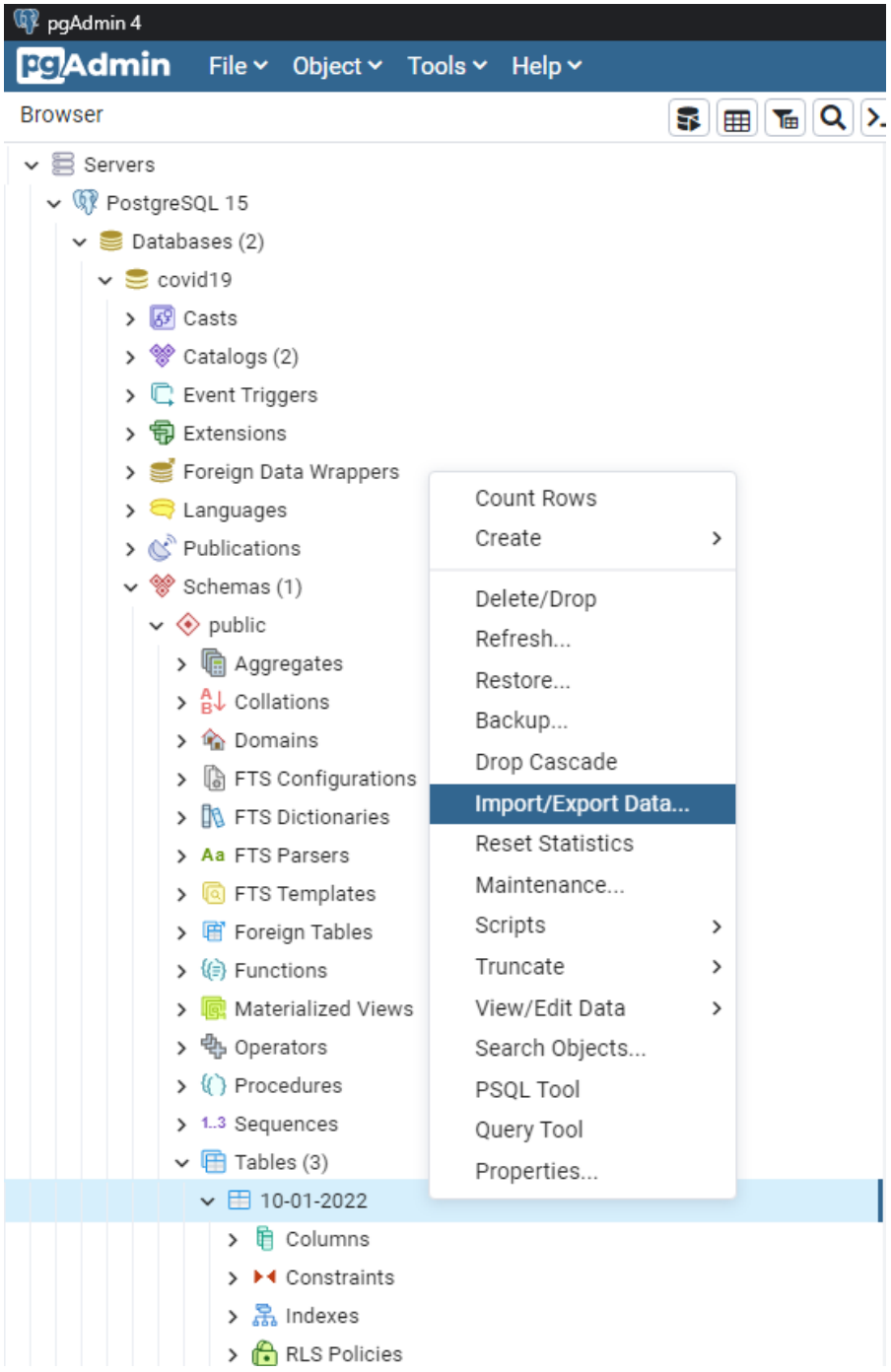
I set `Combined_Key` to `NOT NULL`, because `Combined_Key` is primary key. And the reason why I set `Combined_Key` as primary key is because I found that `Combined_Key` is unique for all rows in the csv files.

```
CREATE TABLE IF NOT EXISTS public."10-01-2022"
(
    "FIPS" character varying(5) COLLATE pg_catalog."default",
    "Admin2" character varying(100) COLLATE pg_catalog."default",
    "Province_State" character varying(100) COLLATE pg_catalog."default",
    "Country_Region" character varying(100) COLLATE pg_catalog."default",
    "Last_Update" timestamp without time zone,
    "Lat" double precision,
    "Long_" double precision,
    "Confirmed" integer,
    "Deaths" integer,
```

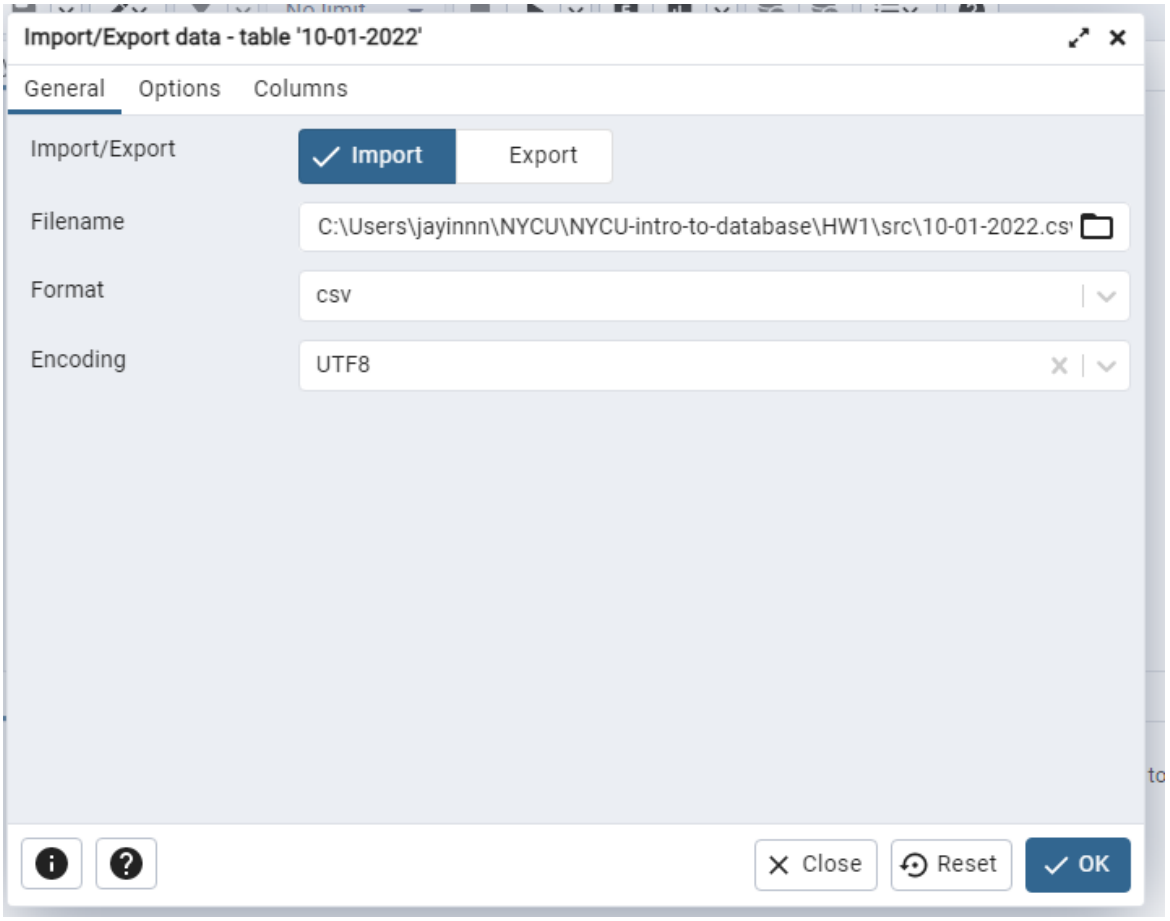
```
    "Recovered" integer,
    "Active" integer,
    "Combined_Key" character varying(100) COLLATE pg_catalog."default" NOT NULL,
    "Incident_Rate" double precision,
    "Case_Fatality_Ratio" double precision,
    CONSTRAINT "10-01-2022_pkey" PRIMARY KEY ("Combined_Key")
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."10-01-2022"
    OWNER to postgres;
```

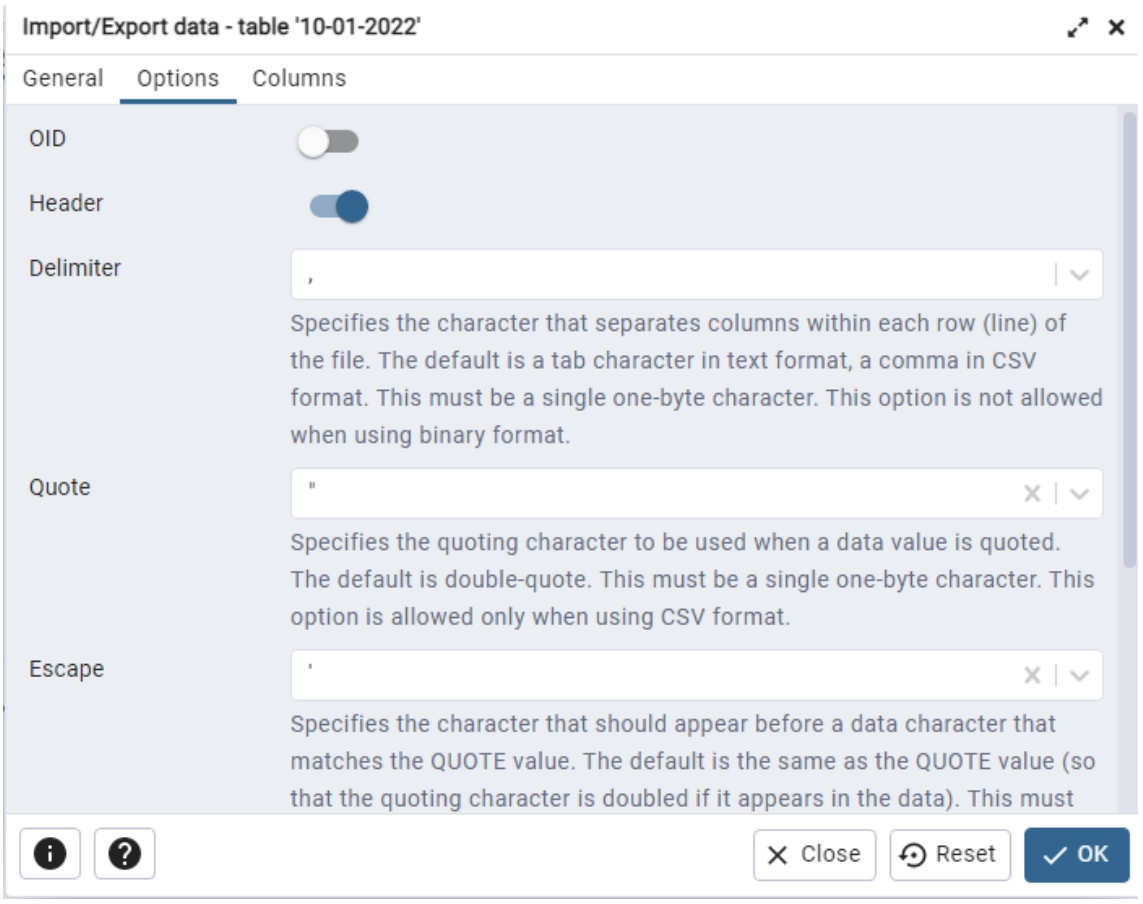
After sending this query to PostgreSQL server, the database server will create a table with the schema.

Finally, we need to import data from csv file. For convenience, I use pgAdmin 4 to import csv file. To import data, we first need find the newly created table. We can find the table in `Databases > covid19 > Schemas > public > 10-01-2022`. After finding the table, we can right click on in, and select `Import/Export Data`.



In the opened windows, select correct csv files, and also, in `Options`, turn on the header option to import header. Finally, click `OK` to import the data from csv file. After processing, we will see the hint that tell us we've already imported the data to table.





Create table for data in 2022/10/11

For creating table for 2022/10/11, the step is almost identical to the step that creates 2022/10/01. The only difference is the schema. The schema for 2022/10/11 is as below.

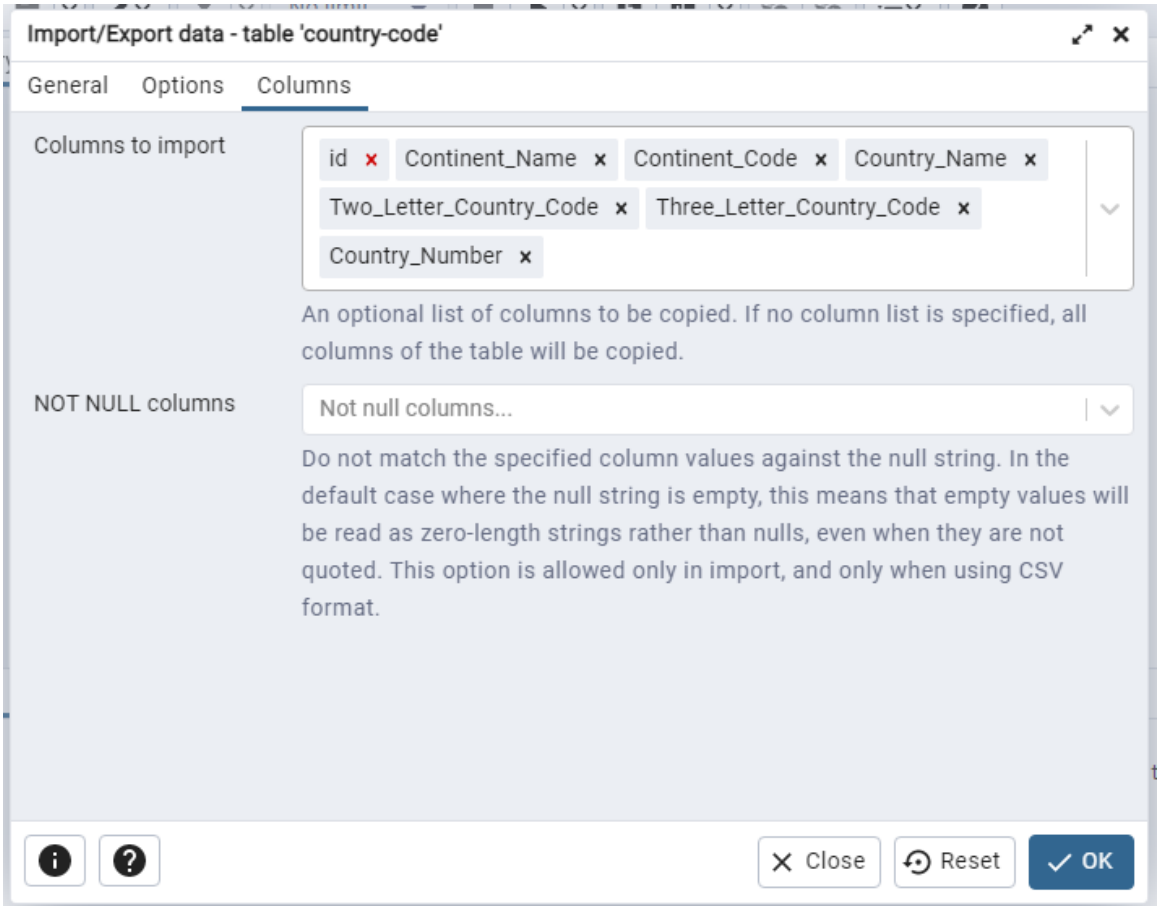
```
CREATE TABLE IF NOT EXISTS public."10-11-2022"
(
    "FIPS" character varying(5) COLLATE pg_catalog."default",
    "Admin2" character varying(100) COLLATE pg_catalog."default",
    "Province_State" character varying(100) COLLATE pg_catalog."default",
    "Country_Region" character varying(100) COLLATE pg_catalog."default",
    "Last_Update" timestamp without time zone,
    "Lat" double precision,
    "Long_" double precision,
    "Confirmed" integer,
    "Deaths" integer,
    "Recovered" integer,
    "Active" integer,
    "Combined_Key" character varying(100) COLLATE pg_catalog."default" NOT NULL,
    "Incident_Rate" double precision,
    "Case_Fatality_Ratio" double precision,
    CONSTRAINT "10-11-2022_pkey" PRIMARY KEY ("Combined_Key")
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."10-11-2022"
    OWNER to postgres;
```

Create table for country code

The step for creating table for country code is also almost identical to the step for creating previous table, but because in the csv file, there doesn't exist a unique column for all row. Therefore, we need to assign a auto-increasing id for every rows (tuples), and use this id as primary key. To create a auto-increasing id, declare `id` as `SERIAL` type. By this way, when importing data, `id` will automatically generate and assign to the imported rows.

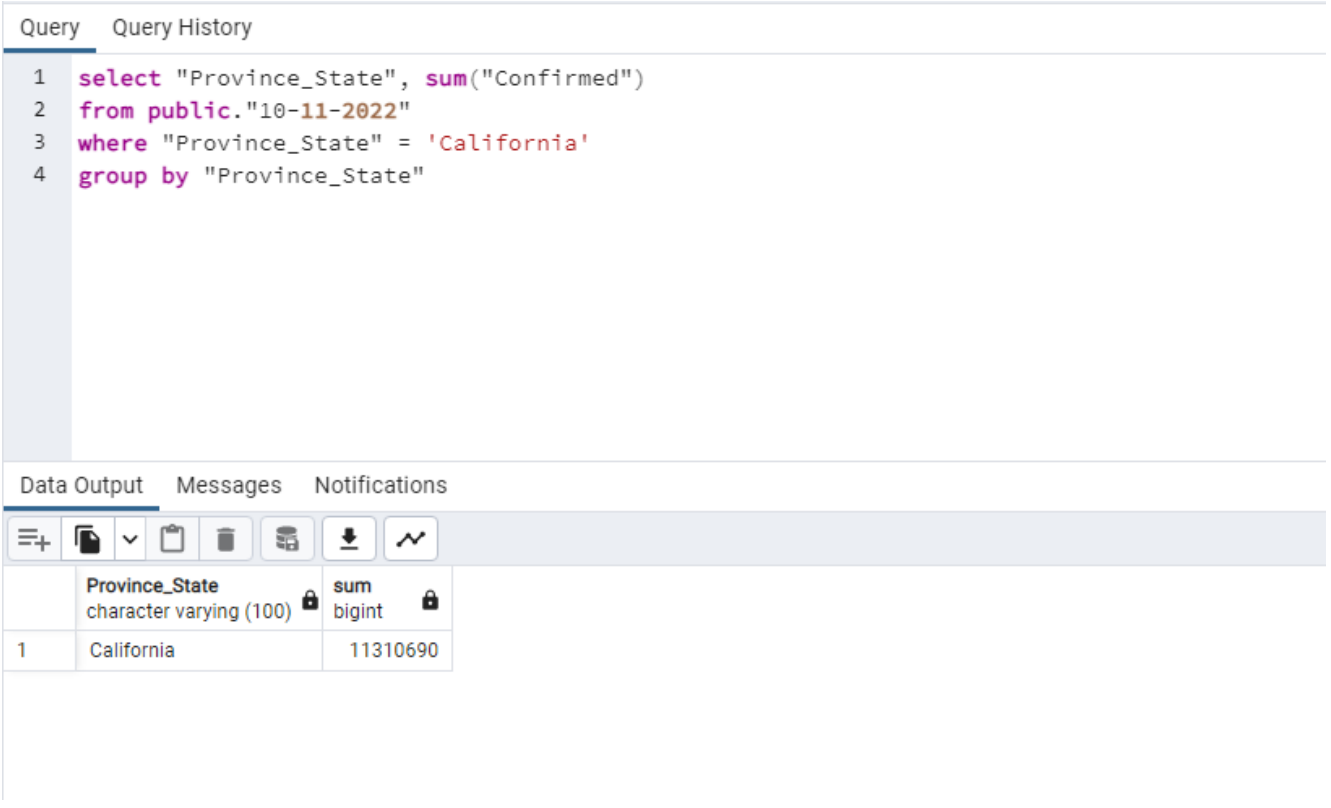
Another change is when importing data, we can't import `id` from csv file. Therefore, make sure to unselect `id` in `Columns to import`.

```
CREATE TABLE IF NOT EXISTS public."country-code"
(
    id SERIAL NOT NULL,
    "Continent_Name" character varying(100) COLLATE pg_catalog."default",
    "Continent_Code" character(2) COLLATE pg_catalog."default",
    "Country_Name" character varying(100) COLLATE pg_catalog."default" NOT NULL,
    "Two_Letter_Country_Code" character(2) COLLATE pg_catalog."default",
    "Three_Letter_Country_Code" character(3) COLLATE pg_catalog."default",
    "Country_Number" integer,
    CONSTRAINT "country-code_pkey" PRIMARY KEY (id)
)

TABLESPACE pg_default;

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



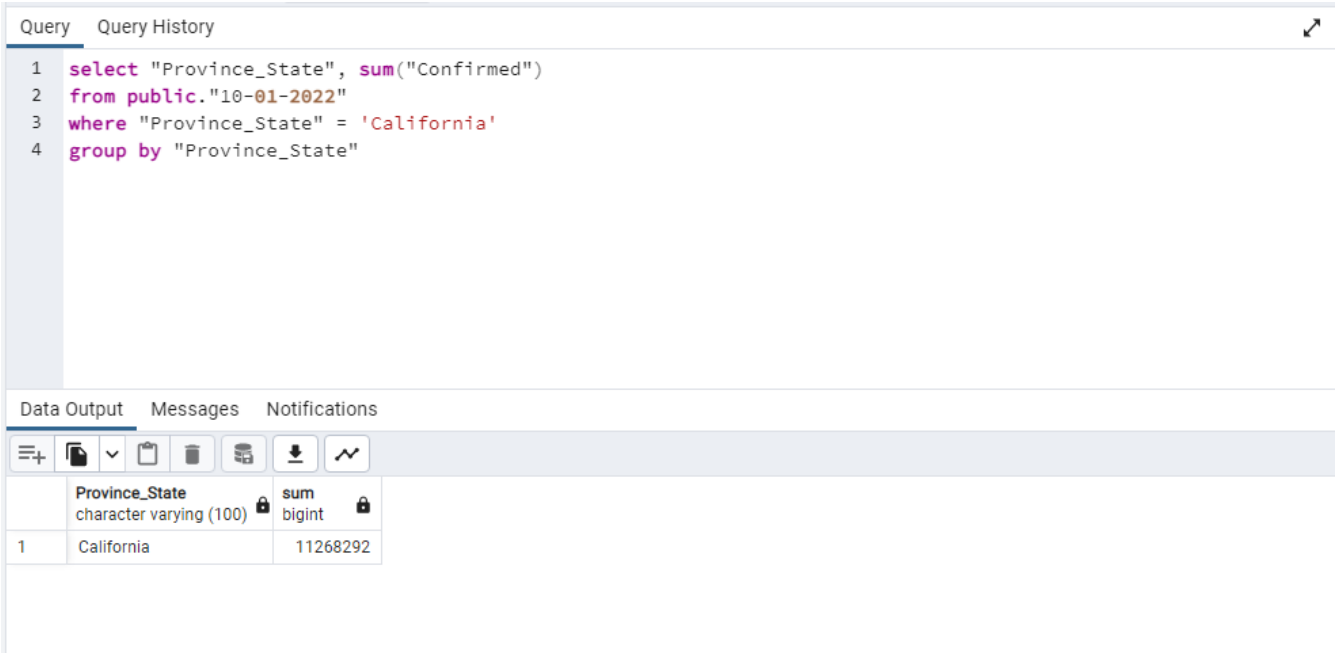
3. The SQL statements and output results of 4a

```
select "Province_State", sum("Confirmed")
from public."10-11-2022"
where "Province_State" = 'California'
group by "Province_State"
```



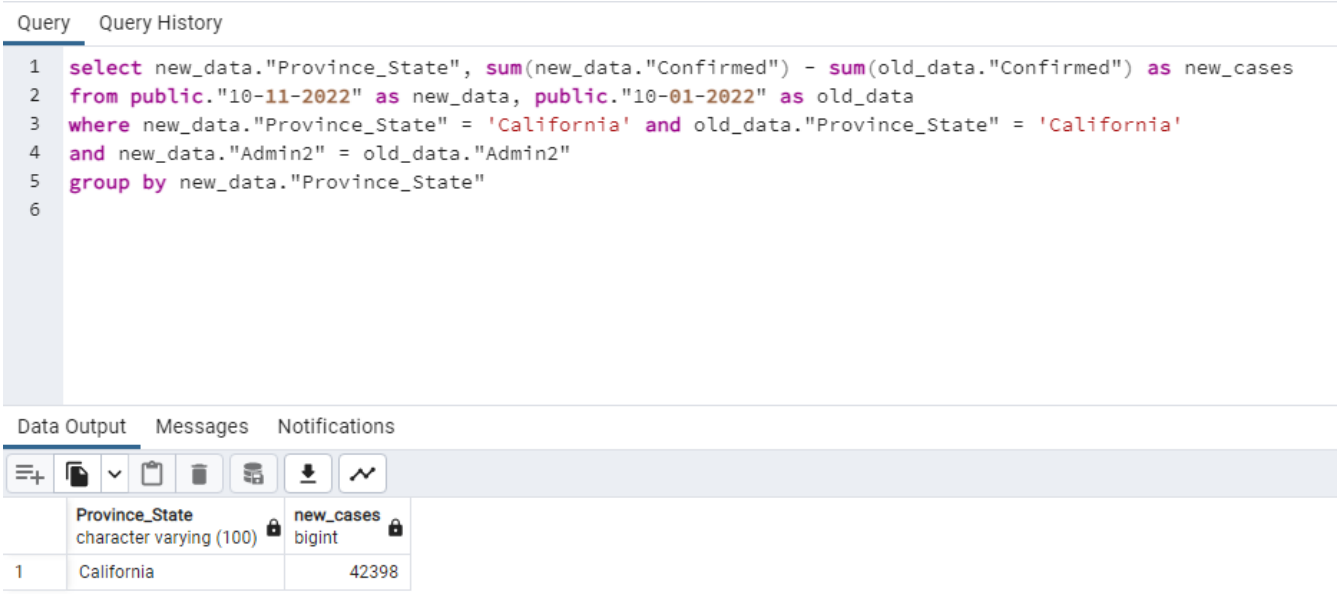
4. The SQL statements and output results of 4b

```
select "Province_State", sum("Confirmed")
from public."10-01-2022"
where "Province_State" = 'California'
group by "Province_State"
```



5. The SQL statements and output results of 4c

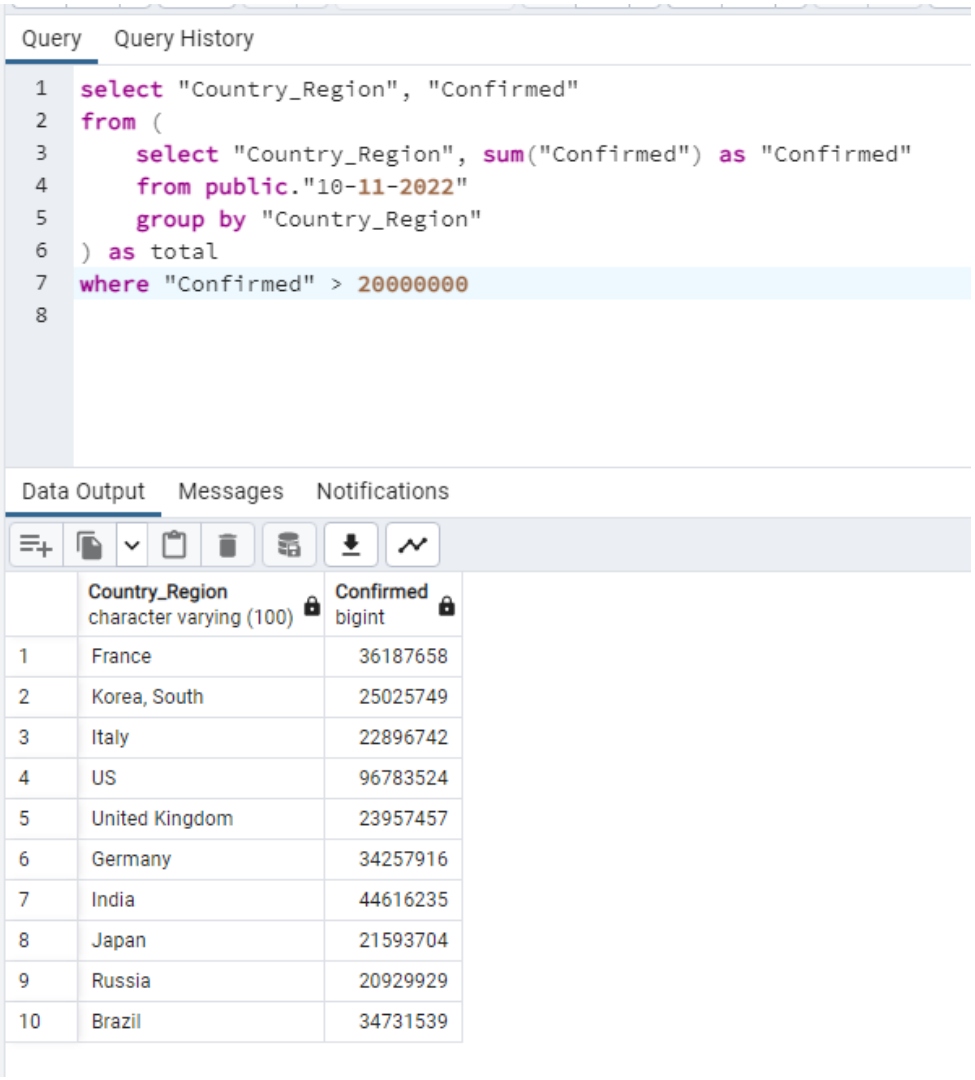
```
select new_data."Province_State", sum(new_data."Confirmed") - sum(old_data."Confirmed") as
new_cases
from public."10-11-2022" as new_data, public."10-01-2022" as old_data
where new_data."Province_State" = 'California' and old_data."Province_State" = 'California'
and new_data."Admin2" = old_data."Admin2"
group by new_data."Province_State"
```



6. The SQL statements and output results of 4d

```
select "Country_Region", "Confirmed"
from (
```

```
        select "Country_Region", sum("Confirmed") as "Confirmed"
        from public."10-11-2022"
        group by "Country_Region"
) as total
where "Confirmed" > 20000000
```



Query | Query History

```
1  select "Country_Region", "Confirmed"
2  from (
3      select "Country_Region", sum("Confirmed") as "Confirmed"
4      from public."10-11-2022"
5      group by "Country_Region"
6  ) as total
7  where "Confirmed" > 20000000
8
```

Data Output | Messages | Notifications

	Country_Region character varying (100)	Confirmed bigint
1	France	36187658
2	Korea, South	25025749
3	Italy	22896742
4	US	96783524
5	United Kingdom	23957457
6	Germany	34257916
7	India	44616235
8	Japan	21593704
9	Russia	20929929
10	Brazil	34731539

7. The SQL statements and output results of 4e

```
select distinct code."Continent_Name", "Country_Region", "Confirmed"
from (
        select "Country_Region", sum("Confirmed") as "Confirmed"
        from public."10-11-2022"
        group by "Country_Region"
) as total
inner join public."country-code" as code
on '%' || code."Country_Name" || '%' like '%' || split_part(total."Country_Region", ',', 1) ||
'%'
where "Confirmed" > 20000000 and "Continent_Name" = 'Asia'
```

Query | Query History

```
select distinct code."Continent_Name", "Country_Region", "Confirmed"
from (
    select "Country_Region", sum("Confirmed") as "Confirmed"
    from public."10-11-2022"
    group by "Country_Region"
) as total
inner join public."country-code" as code
on '%' || code."Country_Name" || '%' like '%' || split_part(total."Country_Region", ',', 1) || '%'
where "Confirmed" > 20000000 and "Continent_Name" = 'Asia'
```

Data Output | Messages | Notifications

	Continent_Name character varying (100)	Country_Region character varying (100)	Confirmed bigint
1	Asia	India	44616235
2	Asia	Japan	21593704
3	Asia	Korea, South	25025749
4	Asia	Russia	20929929

8. The SQL statements and output results of 4f

```
select "Country_Region", "New_Cases"
from (
        select old_data."Country_Region", (new_data."Confirmed" - old_data."Confirmed") as
"New_Cases"
        from (
                select "Country_Region", sum("Confirmed") as "Confirmed"
                from public."10-11-2022"
                group by "Country_Region"
        ) as new_data,
        (
                select "Country_Region", sum("Confirmed") as "Confirmed"
                from public."10-01-2022"
                group by "Country_Region"
        ) as old_data
        where old_data."Country_Region" = new_data."Country_Region"
) as res
where "New_Cases" > 100000
order by "New_Cases" desc
```

Query | Query History

```sql
select "Country_Region", "New_Cases"
from (
    select old_data."Country_Region", (new_data."Confirmed" - old_data."Confirmed") as "New_Cases"
    from (
        select "Country_Region", sum("Confirmed") as "Confirmed"
        from public."10-11-2022"
        group by "Country_Region"
    ) as new_data,
    (
        select "Country_Region", sum("Confirmed") as "Confirmed"
        from public."10-01-2022"
        group by "Country_Region"
    ) as old_data
    where old_data."Country_Region" = new_data."Country_Region"
) as res
where "New_Cases" > 100000
order by "New_Cases" desc
```

Data Output | Messages | Notifications

	Country_Region character varying (100)	New_Cases bigint
1	Germany	871687
2	France	579373
3	Taiwan*	440596
4	Italy	396396
5	US	386493
6	Japan	264185
7	Russia	212106
8	Korea, South	206138
9	Austria	129544
10	Greece	106302