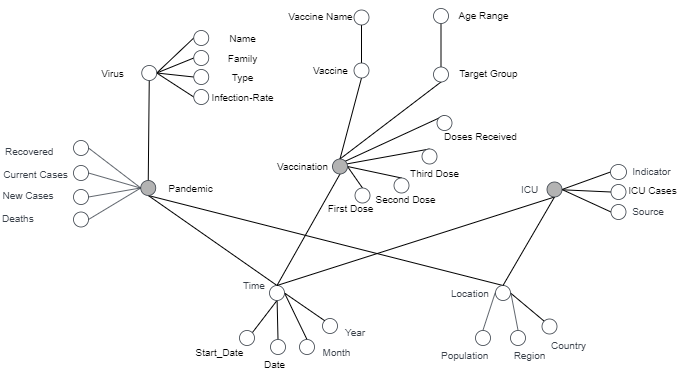
1.i - Functional dependencies:

* V-ID → Name, Family, Type, InfectionRate
* L-ID → Country, Region. Population
* T-ID → Year, Month, Date, Start-Date
* L-ID, T-ID, V-ID → Current-Cases, New-Cases, Recovered, Deaths
* Vac-ID → Name
* Trg-ID → Age-Range
* Vac-ID, T-ID, L-ID, Trg-ID → Doses-Received, First-Dose, Second-Dose, Third-Dose
* L-ID, T-ID→ Indicator, Source, ICU-Cases

1.ii – Considering the provided relational schema and extracted functional dependencies, the initial attribute tree will be as the diagram below.

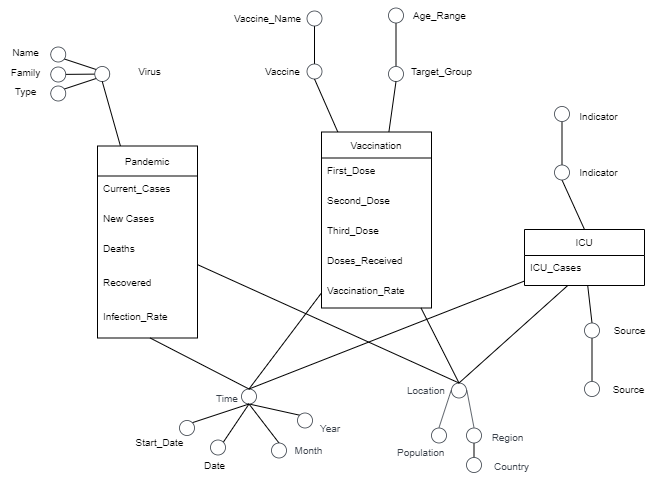


1.iii – For building a fact schema, the initial attribute tree must be pruned and grafted if necessary. Also facts, dimensions and measures must be defined. These are the changes that were applied on the tree for building the fact schema.

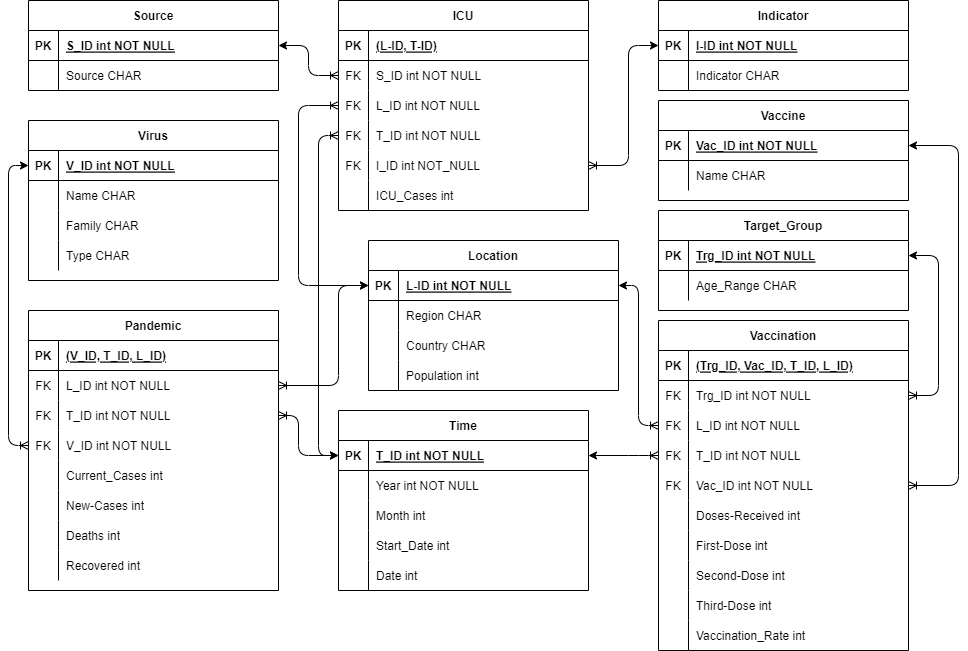
* Country becomes a child of Region.
* Source and Indicator which were child of the ICU node become a dimension because of the count of values they take and therefore are considered dimension rather than measures for the ICU Fact.
* Infection\_Rate was moved to become a child of Pandemic and is considered a measure.
* New node is calculated for the fact table and named Vaccination\_Rate.

The result is a Constellation Fact Schema. Its Facts, Measures and Dimensions are defined below.

* Facts: Pandemic, ICU and Vaccination
* Dimensions: Time, Location, Source, Indicator, Vaccine, Target\_Group, Virus

* Measures: Current Cases, New Cases, Deaths, Recovered, ICU Cases, Doses Received, First Dose, Second Dose, Third Dose

2 – The fact schema turns into the logical model below.



3 – For creating the warehouse schema following DDL statements will be used.

*Location*

CREATE TABLE IF NOT EXISTS Location (

L\_ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

Region TEXT,

Country TEXT,

Population INTEGER

);

*Time*

CREATE TABLE IF NOT EXISTS Time (

T\_ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

Year INTEGER,

Month INTEGER,

Date INTEGER,

Start\_Date INTEGER

);

*Vaccine*

CREATE TABLE IF NOT EXISTS Vaccine (

Vac\_ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

Name TEXT

);

*Source*

CREATE TABLE IF NOT EXISTS Source (

S\_ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

Source TEXT

);

*Indicator*

CREATE TABLE IF NOT EXISTS Indicator (

I\_ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

Indicator TEXT

);

*Taget\_Group*

CREATE TABLE IF NOT EXISTS Target\_Group (

Trg\_ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

Age\_Range TEXT

);

*Virus*

CREATE TABLE IF NOT EXISTS Virus (

V\_ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

Name TEXT,

Family TEXT,

Type TEXT

);

*ICU*

CREATE TABLE IF NOT EXISTS ICU (

L\_ID integer NOT NULL,

T\_ID integer NOT NULL,

I\_ID int NOT NULL,

S\_ID int NOT NULL ,

IcuCases int,

PRIMARY KEY(L\_ID, T\_ID, I\_ID, S\_ID),

FOREIGN KEY (L\_ID) REFERENCES Location (L\_ID),

FOREIGN KEY (T\_ID) REFERENCES Time (T\_ID),

FOREIGN KEY (S\_ID) REFERENCES Source (S\_ID),

FOREIGN KEY (I\_ID) REFERENCES Indicator (I\_ID)

);

*Vaccination*

CREATE TABLE IF NOT EXISTS Vaccination (

L\_ID integer NOT NULL,

T\_ID integer NOT NULL,

Vac\_ID integer NOT NULL,

Trg\_ID integer NOT NULL,

DosesReceived int,

FirstDose int,

SecondDose int,

ThirdDose int,

Vaccination\_Rate int,

PRIMARY KEY(L\_ID, T\_ID, Vac\_ID, Trg\_ID),

FOREIGN KEY (L\_ID) REFERENCES Location (L\_ID),

FOREIGN KEY (T\_ID) REFERENCES Time (T\_ID),

FOREIGN KEY (Vac\_ID) REFERENCES Vaccine (Vac\_ID),

FOREIGN KEY (Trg\_ID) REFERENCES Target\_Group (Trg\_ID)

);

*Pandemic*

CREATE TABLE IF NOT EXISTS Pandemic (

L\_ID integer NOT NULL,

T\_ID integer NOT NULL,

V\_ID integer NOT NULL,

CurrentCases int,

NewCases int,

Deaths int,

Recovered int,

PRIMARY KEY(L\_ID, T\_ID, V\_ID),

FOREIGN KEY (L\_ID) REFERENCES Location (L\_ID),

FOREIGN KEY (T\_ID) REFERENCES Time (T\_ID),

FOREIGN KEY (V\_ID) REFERENCES Virus (V\_ID)

);

4 - The data warehouse design comprises three fact tables - ICU, Vaccination, and Pandemic. To efficiently answer queries that seek information by location and time, we can group the data based on the smallest categories specified in the queries. For instance, we can group the data by region in the location dimension and by month in the time dimension. Additionally, to satisfy the queries, some fact tables may require further summation and grouping, which can be achieved through materialized views.

Materialized View Pandemic:

CREATE VIEW Pandemic\_MV AS

SELECT

SUM(A.NewCases) AS Cases,

SUM(A.Deaths) AS Deaths,

B.Month,

B.Year,

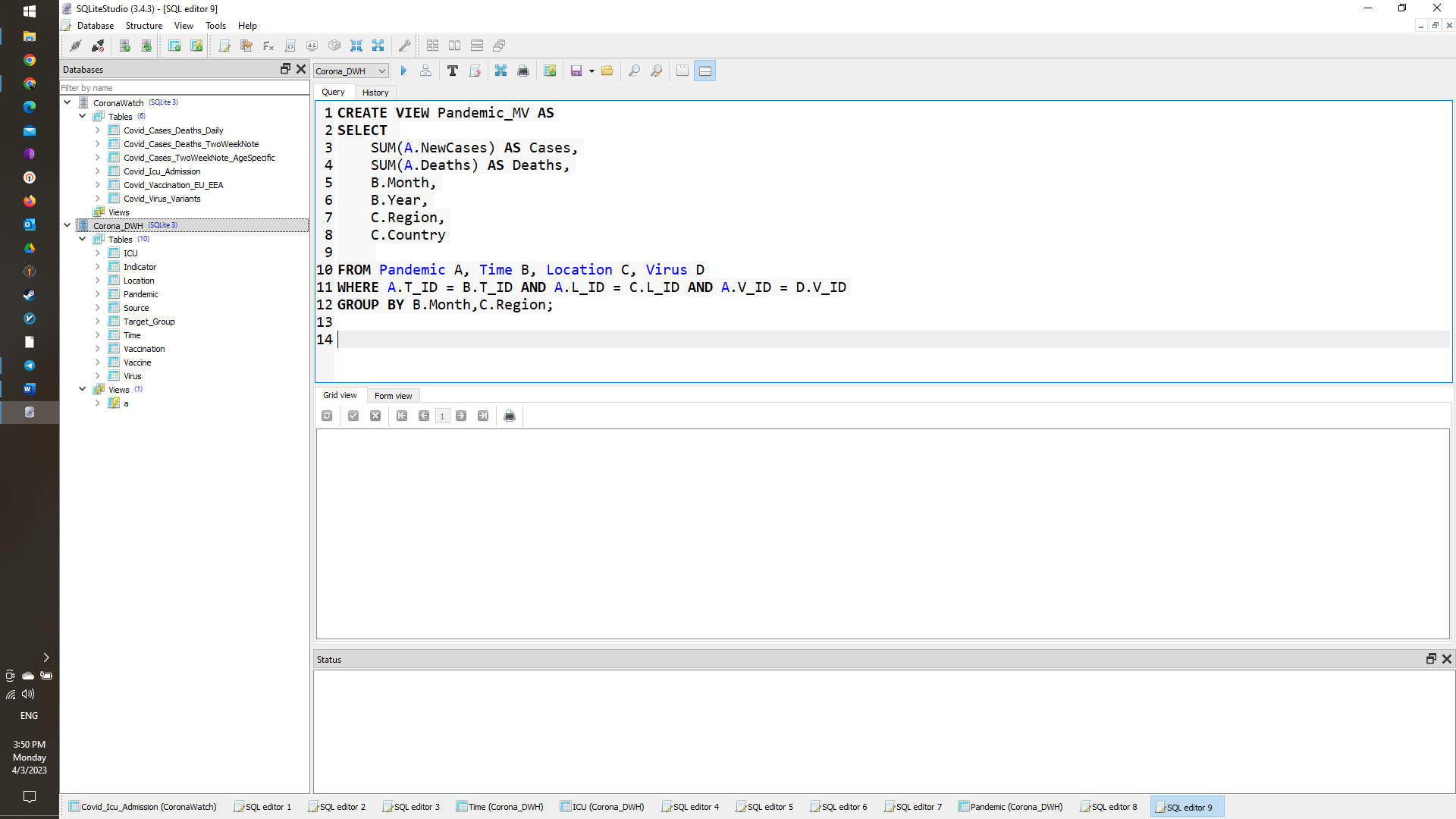
C.Region,

C.Country

FROM Pandemic A, Time B, Location C, Virus D

WHERE A.T\_ID = B.T\_ID AND A.L\_ID = C.L\_ID AND A.V\_ID = D.V\_ID

GROUP BY B.Month,C.Region;



Materialized View ICU:

CREATE VIEW ICU\_VM AS

SELECT

SUM(A.I\_ID) AS Cases,

SUM(A.Deaths) AS Deaths,

B.Month,

B.Year,

C.Region,

C.Country

FROM ICU A, Time B, Location C, Indicator I, Source S

WHERE

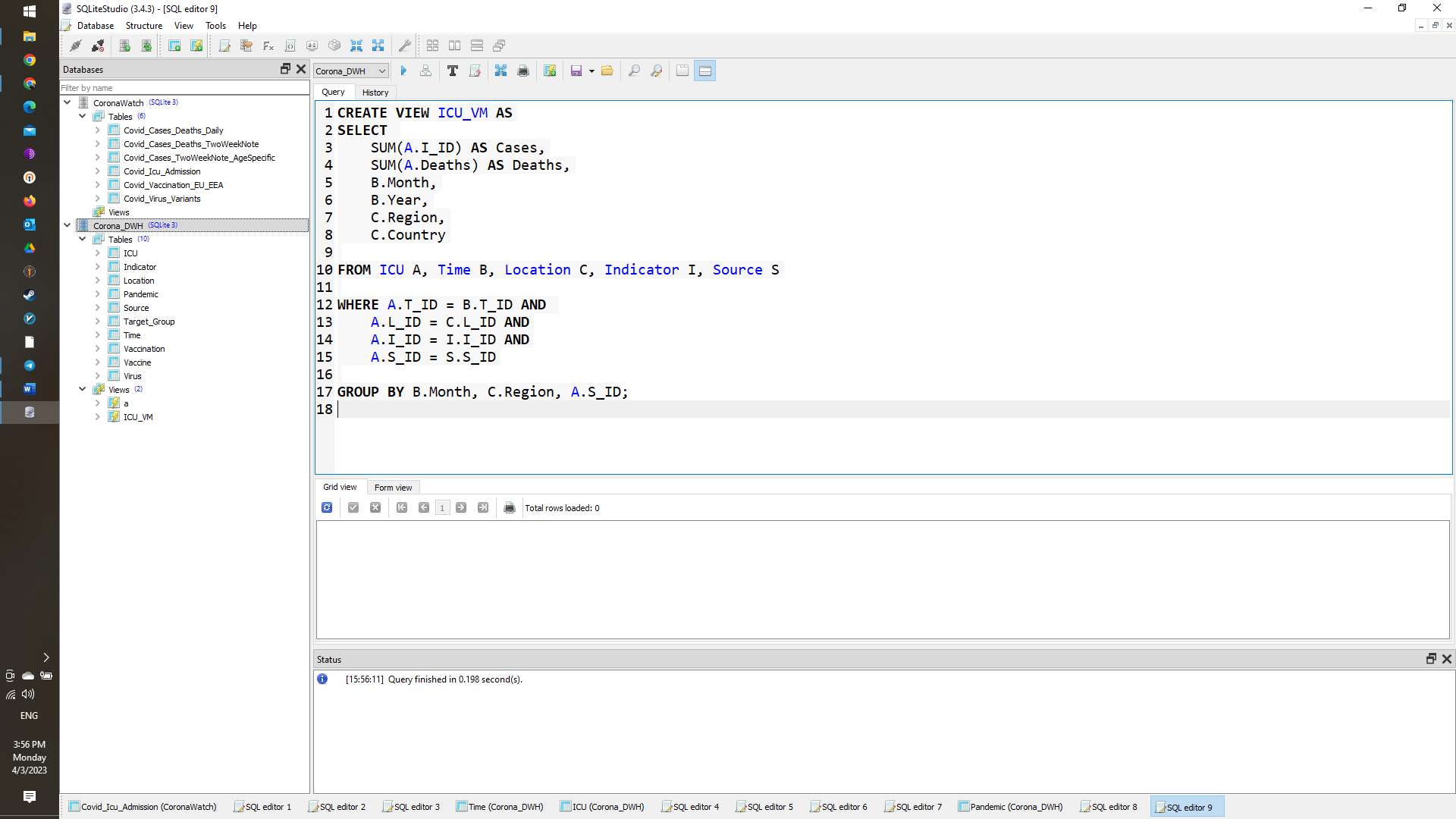
A.T\_ID = B.T\_ID AND

A.L\_ID = C.L\_ID AND

A.I\_ID = I.I\_ID AND

A.S\_ID = S.S\_ID

GROUP BY B.Month,C.Region, A.S\_ID;



Materialized view Vacc:

CREATE VIEW Vacc\_VM AS

SELECT

B.Month,

B.Year,

C.Region,

C.Country,

V.Name,

A.VaccinationRate

FROM Vaccination A, Time B, Location C, Vaccine V, Target\_Group T

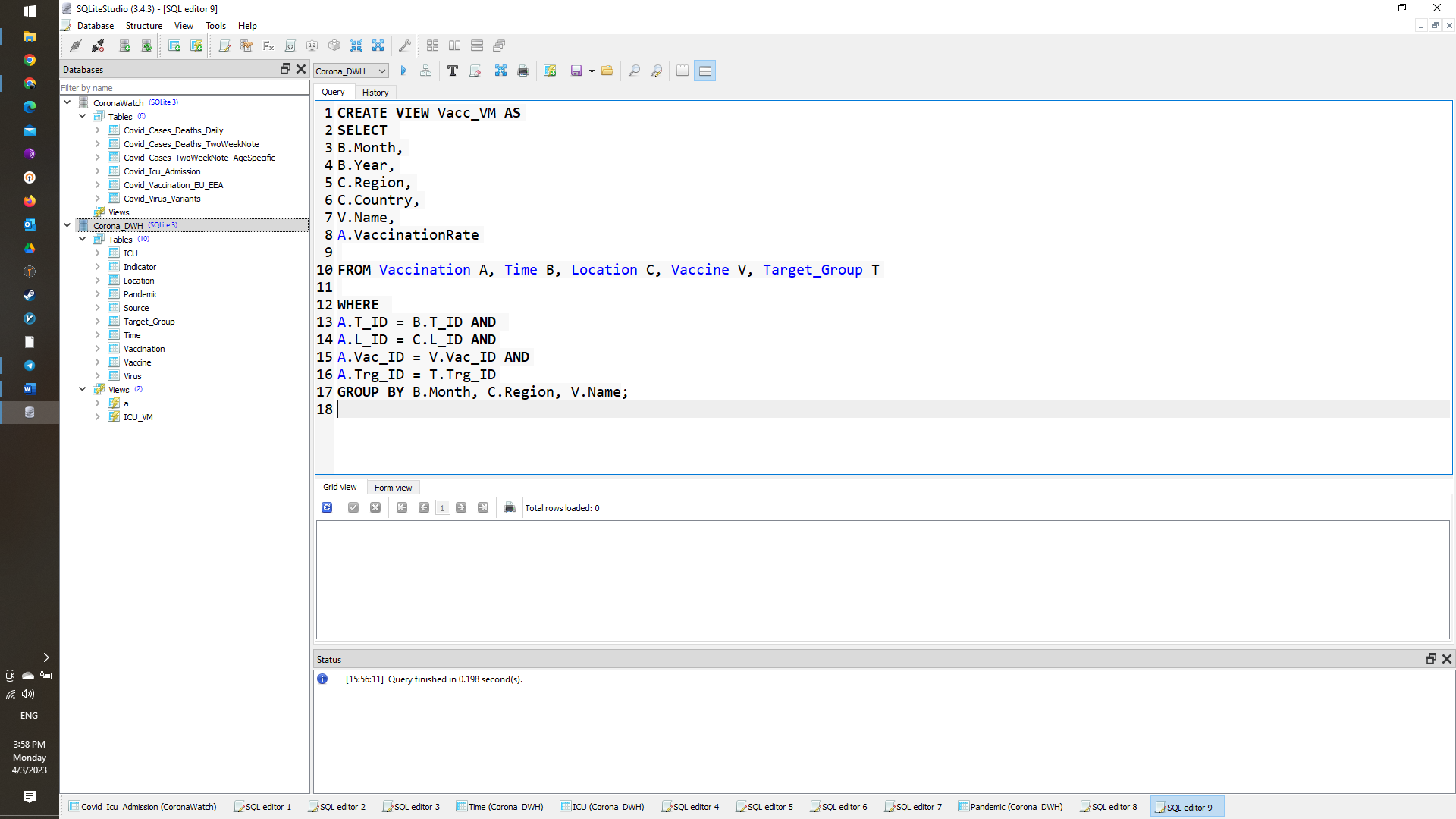
WHERE

A.T\_ID = B.T\_ID AND

A.L\_ID = C.L\_ID AND

A.Vac\_ID = V.Vac\_ID AND

A.Trg\_ID = T.Trg\_ID

GROUP BY B.Month, C.Region, V.Name;

5 -

First Query:

SELECT Region, Month, Cases, Deaths

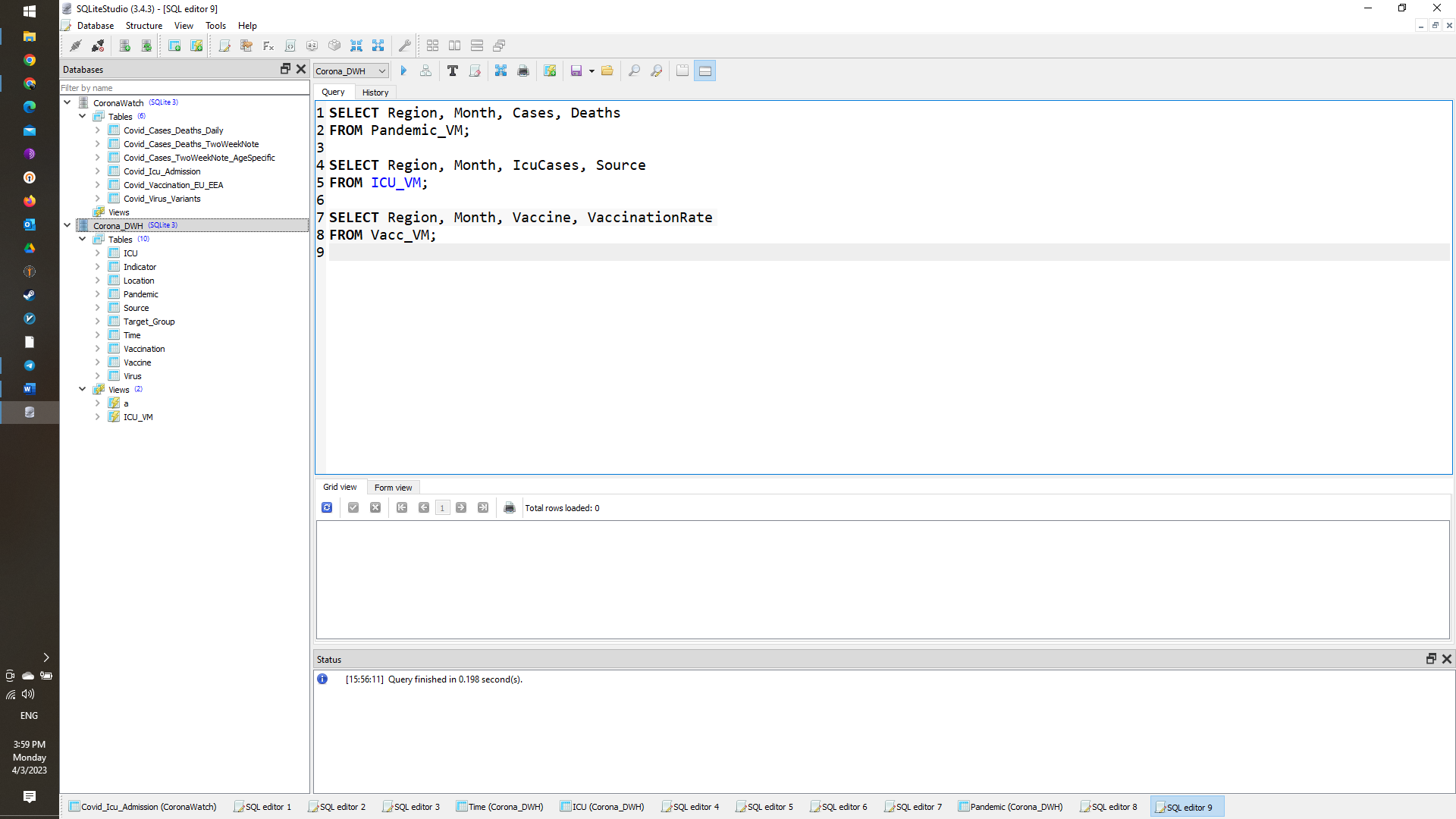
FROM Pandemic\_VM;

SELECT Region, Month, IcuCases, Source

FROM ICU\_VM;

SELECT Region, Month, Vaccine, VaccinationRate

FROM Vacc\_VM;



Second Query:

SELECT Country, Month/3 +1 AS Quarter, Cases, Deaths

FROM Pandemic\_VM

GROUP BY Country, Quarter;

SELECT Country, Month/3 +1 AS Quarter, IcuCases

FROM ICU\_VM

GROUP BY Country, Quarter;

SELECT Country, Month/3 +1 AS Quarter, Vaccine, VaccinationRate

FROM Vacc\_VM

GROUP BY Country, Quarter;

