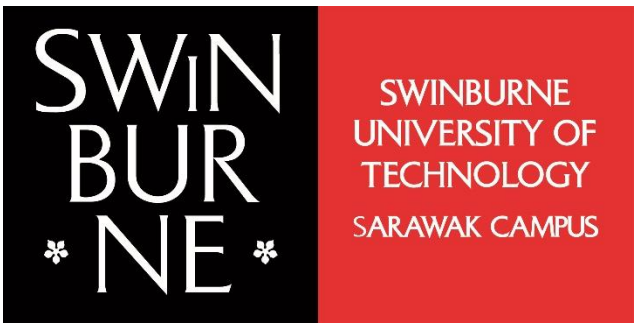
Distinction Project

Virus Information Database



Prepared By : Bernard Joshua Raja Rajan

Student ID : **103365867**

Overview

Over the recent years there has been an influx of viruses that have evolved and caused diseases throughout the planet. Therefore, there is a need to develop methods or ways to track these diseases and get information from them. In order to this I have implemented a database using SQL. This database keeps a record of 27 different types of viruses and all their related information. Through queries, the researcher/user would be able to find information on particular viruses such as where those viruses originated from, how many vaccines are there for each virus, what is the virus parent virus, what is the virus's length and so on. the user will also be able to delete less important viruses, or update information relating to the viruses too. Below are the system details of the database.

Entities, Fields, and Data Types

Variant Entity

Attributes	**Data Type**	**Explanation**
VariantID	INT(7)	The VariantID acts as the primary key. It is better for primary keys to be numeric because it takes longer for the database to compare string values than numeric values. Hence if the PK is a numeric value, it can be matched and found faster. The Length of the Primary key is kept at 7 for identification purposes. For example if it is a primary key for a Variant it starts with 101 but if it a primary key for a transmission it starts with 202.
VariantName	VARCHAR(40)	There might be some variants with long names and some with short names. Hence, it's better to use a flexible data type like VARCHAR as it will save space.
VirusID	INT(5)	Same as in Virus Table
GeneID	INT(7)	Same as in Gene Table
Status	VARCHAR(10)	The status will be inserted as either "active" or "not-active". If "active" is used, then only 6 characters are needed if "not-active" is used then 10 characters are used. Since we do not know how many characters will be used at which instance its better to use VARCHAR as it is more flexible that CHAR and can save space.
Origin	VARCHAR(40)	Some origin locations may have long names, and some have short so it better to use VARCHAR as it is more flexible.

Primary Key: VariantID

Foreign Key's: VirusID, GeneID

Gene Entity

Attributes	Data Type	Explanation
GeneID	INT(7)	GeneID is a primary key. It is better for primary keys to be numeric as it is faster to match as compared to text. Length explanation is the same with VariantID
GeneType	VARCHAR(10)	A virus has either DNA or RNA genes and is called a DNA virus or an RNA virus. These genes are further classified as positive single strain or double strain for RNA and DNA. Example of value: “ssRNA(-) “, “dsDNA”. As you can see the length differs for each value therefore it is better to use VARCHAR in this case as it is more flexible and saves space.
GeneLength	INT(11)	Gene lengths can go as high as 11 digits so the best data type to use is INT as the value will be in integers and set it to 11 digits

Primary Key: GeneID

Virus Entity

Attributes	Data Type	Explanation
VirusID	INT(5)	VirusID is a primary key. It is better for primary keys to be numeric as it is faster to match as compared to text. VirusID pk lengths are shorter, this is also used to identify tables.
VirusName	VARCHAR(40)	Some viruses have really long names, and some have short names, therefore it is better to use VARCHAR as it is more flexible than CHAR.
Year_Discovered	INT(4)	Year contains only 4 digits so its better to use INT.
First_Origin	VARCHAR(40)	Some Origin locations may have very long names and some would have short names so its better to use VARCHAR over CHAR as it is flexible and will save space. Example of long name : "Democratic Republic Of Congo"

Primary Key: VirusID

Variant_Location Entity

Attributes	Data Type	Explanation
VariantID	INT(7)	Same as in Variant Table
LocationID	INT(3)	Same as in Locations table
Climate	VARCHAR(10)	Climate length is not fixed hence it is better to use VARCHAR as it is more flexible than CHAR.

Primary Key: VariantID, LocationID

Foreign Key: VariantID, LocationID

Location Entity

Attributes	Data Type	Explanation
LocationID	INT(3)	LocationsID is a primary key. It is better for primary keys to be numeric as it is faster to match as compared to text.
CountryName	VARCHAR(30)	CountryName value length is not fixed as some countries have long names and some countries have short names. Therefore, it is better to use VARCHAR as it is more flexible compared to CHAR.
Population	INT(11)	Population of countries are numeric as they can be counted. The length is put as 11 because there could be up to a billion people in a country. Example: India has more than a billion citizens.

Primary-Key: LocationID

Variant_Host Entity

Attributes	Data Type	Explanation
VariantID	INT(7)	Same as in Variant Table
HostID	INT(5)	Same as in Host Table

Primary-Key: VariantID, HostID

Foreign-Key: VariantID, HostID

Host Entity

Attributes	Data Type	Explanation
HostID	INT(5)	HostID is a primary key. It is better for primary keys to be numeric as it is faster to match as compared to text.
Host_Name	VARCHAR(15)	The value length of this attribute is not fixed as some values could be long and some short. Therefore, it is better to use VARCHAR instead of CHAR as it is more flexible.
Host_Type	VARCHAR(15)	Same as Host_Name

Primary-Key: HostID

Transmission-Mode Entity

Attributes	Data Type	Explanation
VariantD	INT(7)	Same as in Variant Table
TransmissionID	INT(6)	Same as in Transmission Table
Risk	VARCHAR(9)	Risk levels have four type “Very High”, “High”, “Moderate”, “Low”. Since each of these types are not fixed in length it is better to use varchar for this as it is more flexible than char and would save up more space.

Primary-Key: VariantID, TransmissionID

Foreign-Key: VariantID, TransmissionID

Transmission Entity

Attributes	Data Type	Explanation
TransmissionID	INT(6)	TransmissionID is a primary key. It is better for primary keys to be numeric as it is faster to match as compared to text.
TransmissionType	VARCHAR(20)	Transmission type value would be a text value and could have various lengths hence the data type used should be flexible. That is why VARCHAR is used.

Primary-Key: TransmissionID

Variant-Vaccine Entity

Attributes	Data Type	Explanation
VariantID	INT(7)	Same as Variant Table
VaccineID	INT(6)	Same as Vaccine Table
Vaccine_Status	VARCHAR(3)	The status will be inserted as “A” for available, “N-A” for not available or “P” for pending . Therefore, the length of it can be either 1 or 3. Since the length of the value is not fixed it is better to use VARCHAR instead or CHAR as VARCHAR is more flexible.

Primary-Key: VariantID, VaccineID

Foreign-Key: VariantID, VaccineID

Vaccine Entity

Attributes	**Data Type**	**Explanation**
VaccineID	INT(6)	VaccineID is a primary key. It is better for primary keys to be numeric as it is faster to match as compared to text.
VaccineName	VARCHAR(10)	Vaccines can have either long or short names therefore VARCHAR is used as it is more flexible.

Primary-Key: VaccineID

Researchers Entity

Attributes	Data Type	Explanation
VariantID	INT(7)	Same as Variant Table
ResearcherID	VARCHAR(7)	Same as Researcher Table
Status_of_Research	VARCHAR(15)	States either if the research is complete or ongoing therefore the length is not set for the value so its better to use VARCHAR for this as it is more flexible and saves up more space when compared to CHAR.

Primary Key : VariantID, ResearcherID

Foreign Key: VariantD, ResearcherID

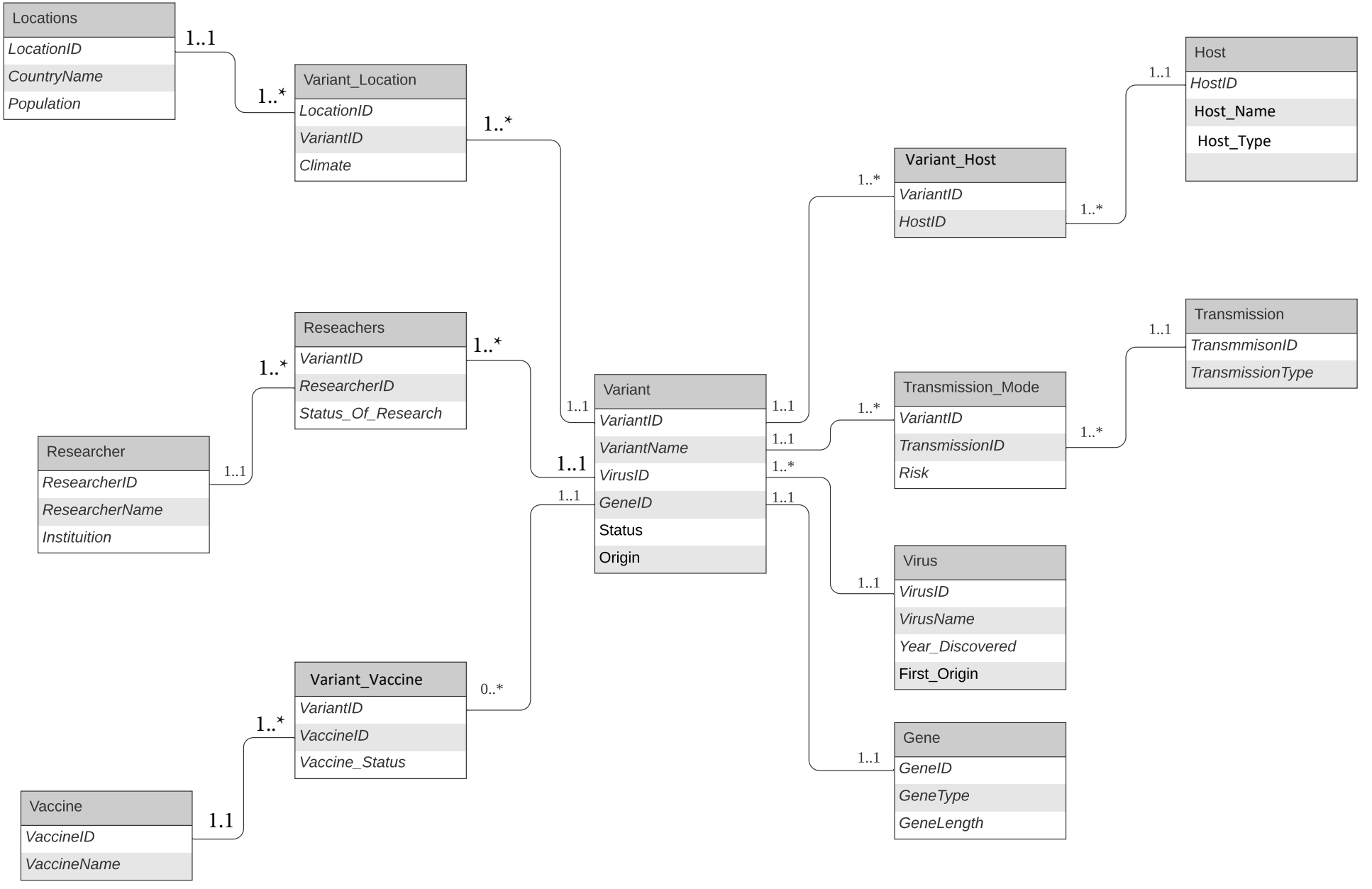
Researcher Entity

Attributes	Data Type	Explanation
ResearcherID	INT(7)	ResearcherID is a primary key. It is better for primary keys to be numeric as it is faster to match as compared to text.
ResearcherName	VARCHAR(40)	Australian Government sets about 100 characters including spaces, apostrophes, and hyphens for names. (Source: Australian Home Affairs Ministry). And since names can be any length as its not fixed its better to use VARCHAR.
Institution	VARCHAR(40)	Some institution names are very long such as Western Sydney University which has 25 characters there are some which are even longer especially if it is in another language so the data type should be flexible to cater to this. Hence, we should use varchar for this data type as it is more flexible and saves up more space.

Primary Key: ResearcherID

Virus Information Database

Bernard Joshua | COS20015



Relationships

1. One **Variant** has one **Parent Virus**, but one **Parent Virus** has many **Variants**.
2. One **Variant** has one type of **Gene**, and one type of **Gene** has one **Variant**.
3. One **Variant** has many types of **Hosts**, and one **Host** has many types of **Variants**
4. One **Variant** has many types of **Transmissions**, and one **Transmission** has many types of **Variants**
5. One **Variant** is active in **many Locations** and one **Location** may have one or **many Variants**.
6. One **Variant** may have **many Vaccines** , One **Vaccine** has only one or **more Variants**
7. Each **Variant** has **one or many Researchers** and each **Researcher** has **one or many Variants**.

Justification of entities.

As you can see this table has a few many to many relationships. Therefore, to implement a better design many weak entities were used. Now the question is why this database does have so many tables. Well based on my research all these tables are needed as they contain information that is used to track down and get information about the virus. For proper, virus tracing the user should be able to know the gene type as well as the gene length (gene table) this can be used by researchers to do gene sequencing a key important aspect in creating vaccines for viruses. On that note, knowing which parent virus (virus table) the current virus/variant that they are researching id from will help them to understand the current structure of the variant that they are studying. Besides that, researchers should know what host type (host table) and transmission type (transmission table) each variant is using as to know how much the virus has evolved hand how contagious it has become. Furthermore, knowing about where the virus has originated from and what locations it has spread will give key insight to how fast it is spreading and how adaptable it is to different geolocations and climates. Knowing which researcher (researcher table) is also working on the virus would help current researchers to collaborate with each other or reference existing work from these researchers. Lastly, knowing what vaccines (vaccine table) are available for the virus could help researchers in two ways one they will know how to treat the diseases caused by the virus and two they will know if they should move on to research another virus as there is already a cure for the one that they're currently tracking/researching.

Script to Create The Database

*Note database must be created in the order of the following statement sequence.

* Another note: after creating the database you have to select the database first before executing the following statements.

```
CREATE DATABASE Virus_Information;


CREATE TABLE Virus (
VirusID INT(5) NOT NULL,
VirusName VARCHAR(40) NOT NULL,
Year_Discovered INT(4) NOT NULL,
First_Origin VARCHAR(40) DEFAULT "Not Available",

PRIMARY KEY(VirusID)
);


CREATE TABLE Gene (
GeneID INT(7) NOT NULL,
GeneType VARCHAR(11) NOT NULL,
GeneLength_nt INT(11) NOT NULL,

PRIMARY KEY(GeneID)
);


CREATE TABLE Variant (
VariantID INT(7) NOT NULL,
VariantName VARCHAR(25) NOT NULL,
VirusID INT(5) NOT NULL,
GeneID INT(7) NOT NULL,
Status VARCHAR(10) NOT NULL,
Origin VARCHAR(40) NOT NULL,

PRIMARY KEY(VariantID),
CHECK(Status in ('Active', 'Not-Active')),
FOREIGN KEY (VirusID) REFERENCES virus(VirusID),
FOREIGN KEY (GeneID) REFERENCES gene(GeneID)
);




CREATE TABLE Host (
HostID INT(5) NOT NULL,
Host_Name VARCHAR(15) NOT NULL,
Host_Type VARCHAR(15) NOT NULL,

PRIMARY KEY(HostID)
);


CREATE TABLE Variant_Host (
```

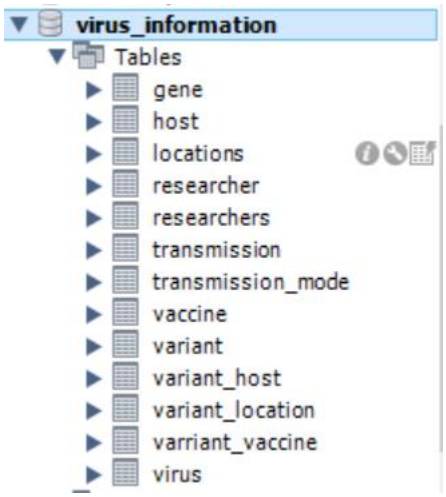
```
VariantID INT(7) NOT NULL,
HostID INT(5) NOT NULL,


PRIMARY KEY(VariantID, HostID),
FOREIGN KEY(VariantID) REFERENCES Variant(VariantID),
FOREIGN KEY(HostID) REFERENCES Host(HostID)
);


CREATE TABLE Locations (
LocationID INT(3) NOT NULL,
CountryName VARCHAR(30) NOT NULL,
Population INT(11) NOT NULL,

PRIMARY KEY(LocationID)
);


CREATE TABLE Variant_Location (
VariantID INT(7) NOT NULL,
LocationID INT(3) NOT NULL,
Climate VARCHAR(15) NOT NULL,

PRIMARY KEY(VariantID, LocationID),
FOREIGN KEY(VariantID) REFERENCES Variant(VariantID),
FOREIGN KEY(LocationID) REFERENCES Locations(LocationID)
);



CREATE TABLE Transmission (
TransmissionID INT(6) NOT NULL,
TransmissionType VARCHAR(20) NOT NULL,

PRIMARY KEY(TransmissionID)
);



CREATE TABLE Transmission_Mode (
VariantID INT(7) NOT NULL,
TransmissionID INT(6) NOT NULL,
Risk VARCHAR(9) NOT NULL,

PRIMARY KEY(VariantID, TransmissionID),
FOREIGN KEY(VariantID) REFERENCES Variant(VariantID),
FOREIGN KEY(TransmissionID) REFERENCES Transmission(TransmissionID)
);
```

```
CREATE TABLE Vaccine (
VaccineID INT(6) NOT NULL,
VaccineName VARCHAR(15) NOT NULL,

PRIMARY KEY(VaccineID)
);

CREATE TABLE Varriant_Vaccine (
VariantID INT(7) NOT NULL,
VaccineID INT(6) NOT NULL,
Vaccine_Status VARCHAR(3) NOT NULL,

PRIMARY KEY(VariantID, VaccineID),
CHECK(Vaccine_Status in ('A', 'N-A', 'P')),
FOREIGN KEY(VariantID) REFERENCES Variant(VariantID),
FOREIGN KEY(VaccineID) REFERENCES Vaccine(VaccineID)
);


CREATE TABLE Researcher (
ResearcherID INT(7) NOT NULL,
ResearcherName VARCHAR(40) NOT NULL,
Instituition VARCHAR(40) NOT NULL,

PRIMARY KEY(ResearcherID)
);


CREATE TABLE Researchers (
VariantID INT(7) NOT NULL,
ResearcherID INT(7) NOT NULL,
Status_Of_Research VARCHAR(15) NOT NULL,

PRIMARY KEY(VariantID, ResearcherID),
FOREIGN KEY(VariantID) REFERENCES Variant(VariantID),
FOREIGN KEY(ResearcherID) REFERENCES Researcher(ResearcherID)
);
```

Verification Of Built Database

Name	Engine	Version	Row Format	Rows
gene	InnoDB	10	Dynamic	27
host	InnoDB	10	Dynamic	10
locations	InnoDB	10	Dynamic	196
researcher	InnoDB	10	Dynamic	29
researchers	InnoDB	10	Dynamic	49
transmission	InnoDB	10	Dynamic	6
transmission_mode	InnoDB	10	Dynamic	52
vaccine	InnoDB	10	Dynamic	13
variant	InnoDB	11	Dynamic	27
variant_host	InnoDB	10	Dynamic	44
variant_location	InnoDB	10	Dynamic	1091
varriant_vaccine	InnoDB	11	Dynamic	33
virus	InnoDB	10	Dynamic	10

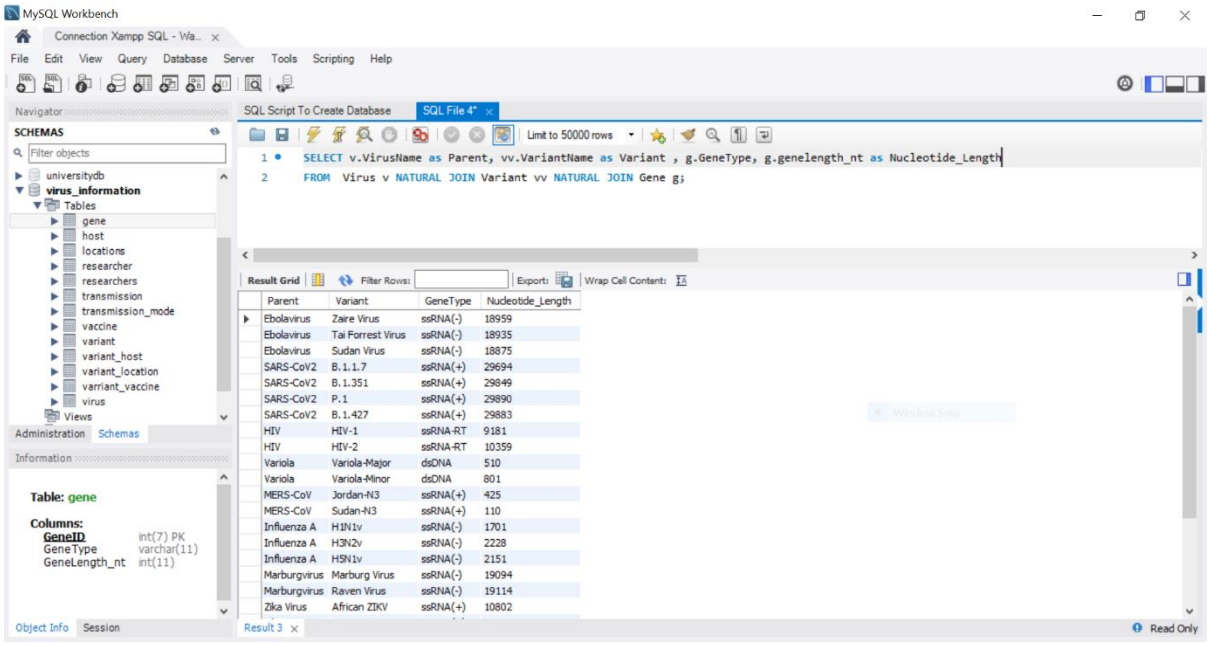
This database has more than 25 records in majority of its tables.



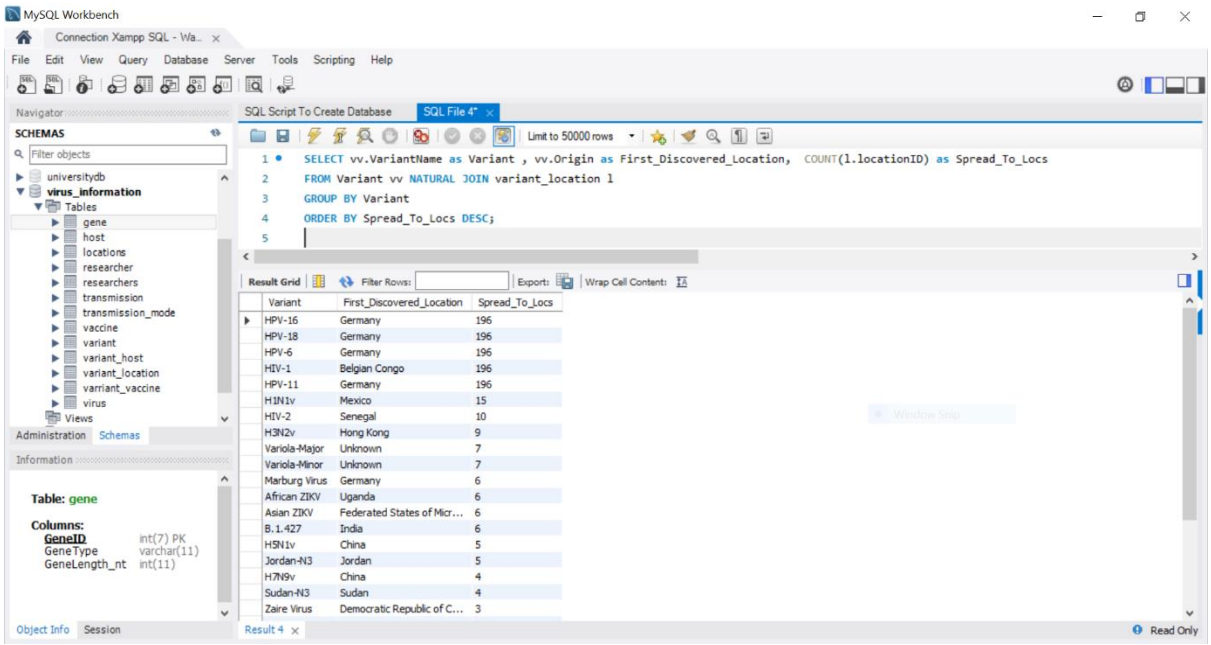
Proof that database has been built and implemented on Local Server and MySQL Workbench.

Typical Use of the Database using JOIN's Queries

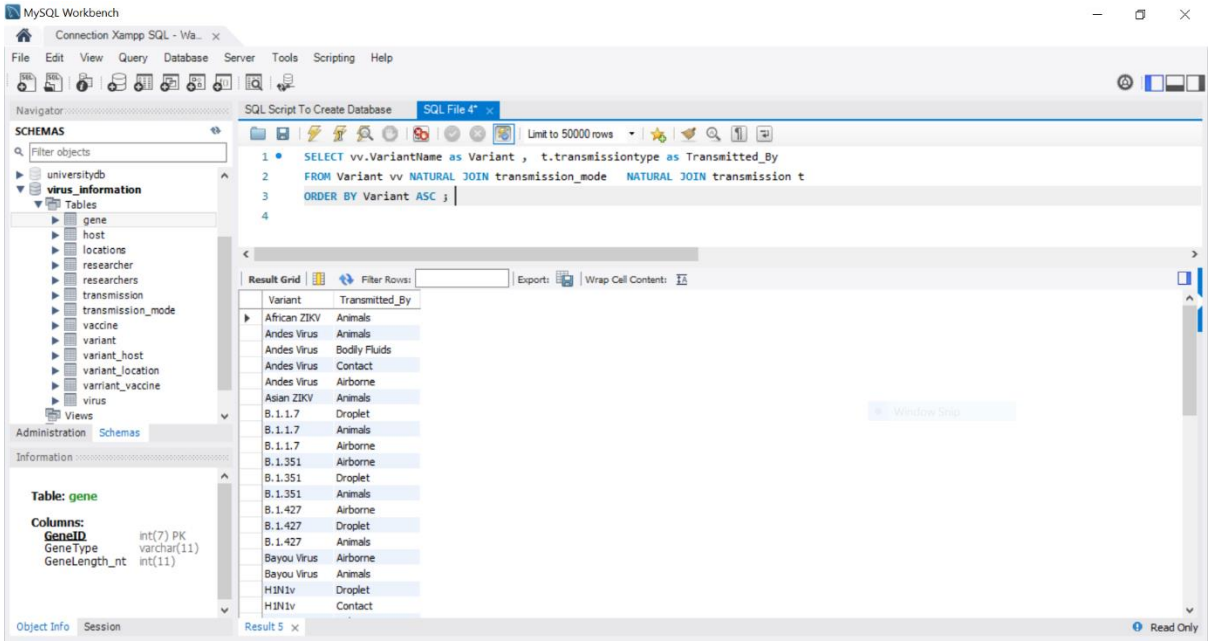
Query 1: Get the full details of the Variants Parent and the Variants Individual Gene information.



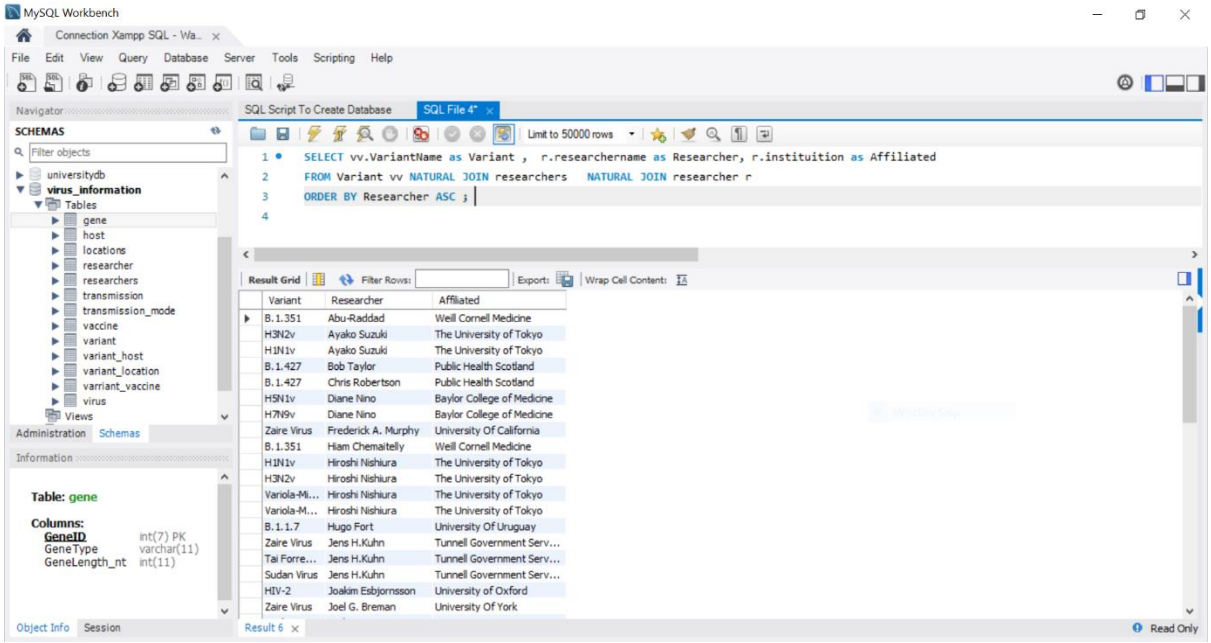
Query 2: See where the Variant originated from and where it has spread to.



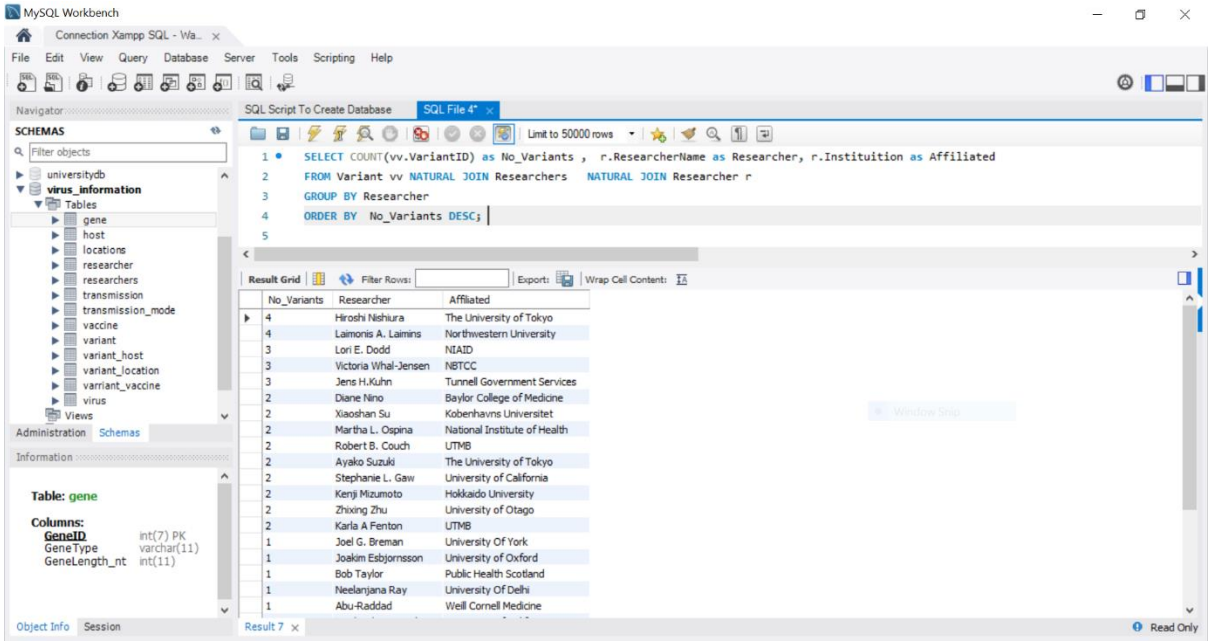
Query 3: See what type of transmission modes each Variant uses.



Query 4: See which Researchers conduct research on which type of Variant.

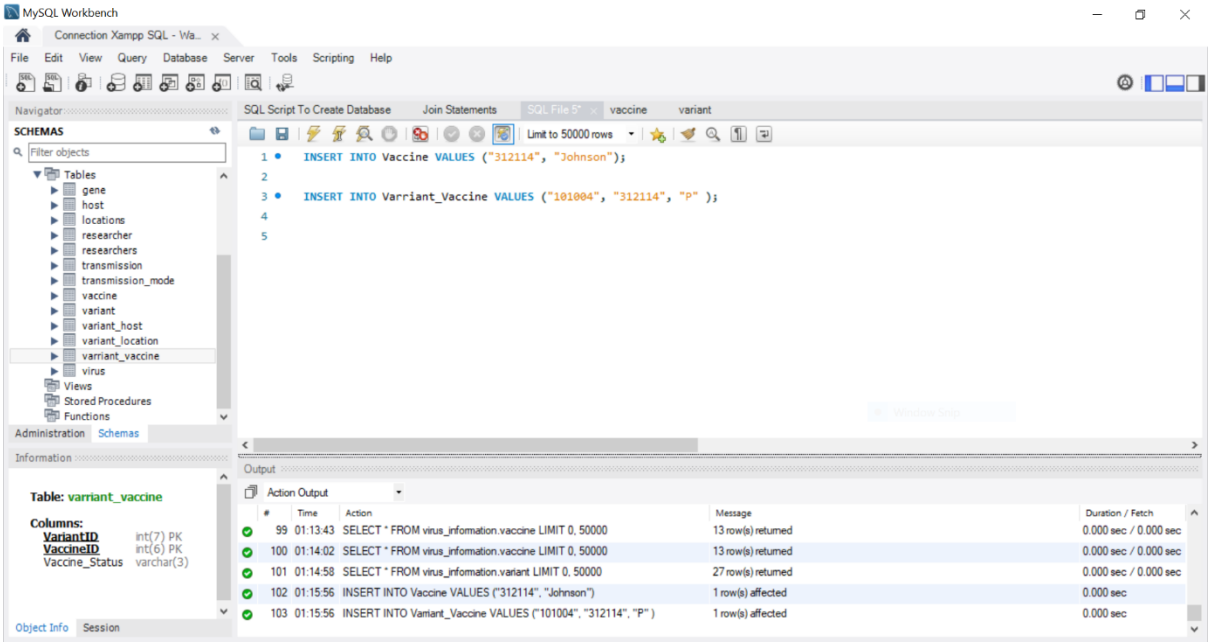


Query 5: See how many Variants Each researcher researches.



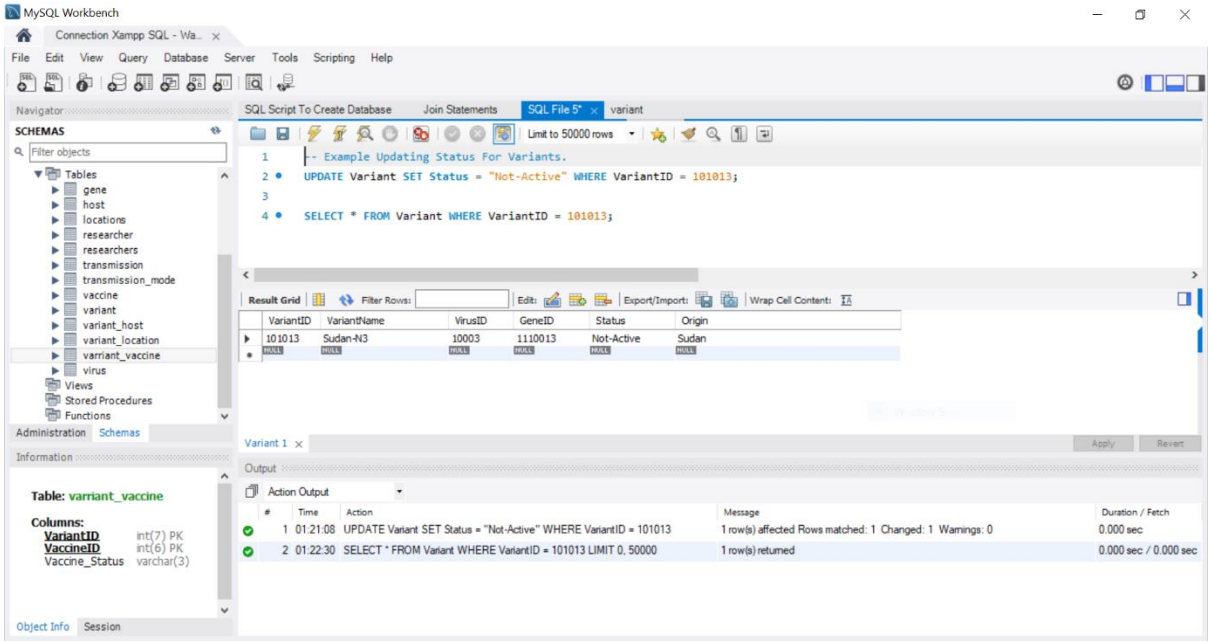
Other Typical Use Cases

First USE CASE: Updating Important Information Related To The Virus, such as new Vaccines.



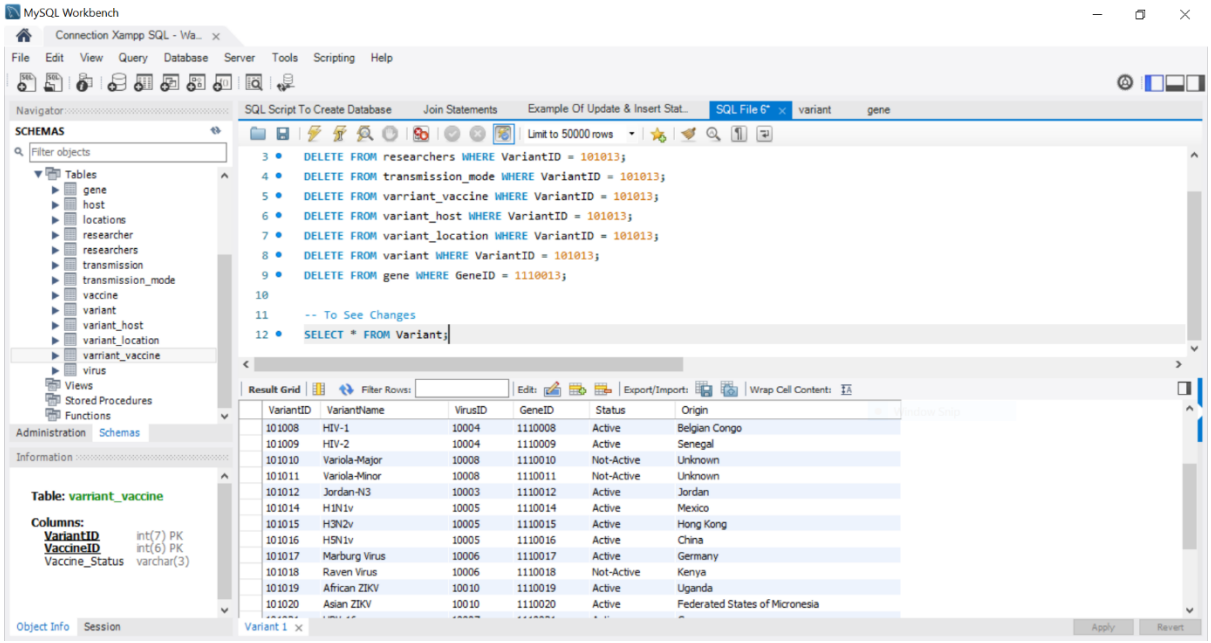
To insert values in to tables. Note for this database since everything is relate if you want to insert a value into a table you should also insert into the related tables.

Second Use Case: Updating Tables.



Updating a value in a table. For example, let's say a virus is not active anymore. Then it should be possible to update the virus variants status to not active as above.

Third Use Case: Deleting Values From Tables



Let us say in an event that a virus become unimportant or no longer neede to keep it details then, it can be deleted from the database. In order to do so one would have to delete it from all its relating tables, as above. (We have successfully deleted the Sudan-N3 Virus .Hence, there is no more Sudan-N3 virus variant in the table).

NOTE: All SQL Scripts are available in the ZIP files attached with this assignment.

Declaration Of Work:

I hereby declare that this work is full mine and not from other sources:

Recoverable Signature

X Bernard Joshua Raja Rajan

Bernard Joshua Raja Rajan

Author

Signed by: afd2a90f-f27d-4eee-9c06-d88ea6b51e55