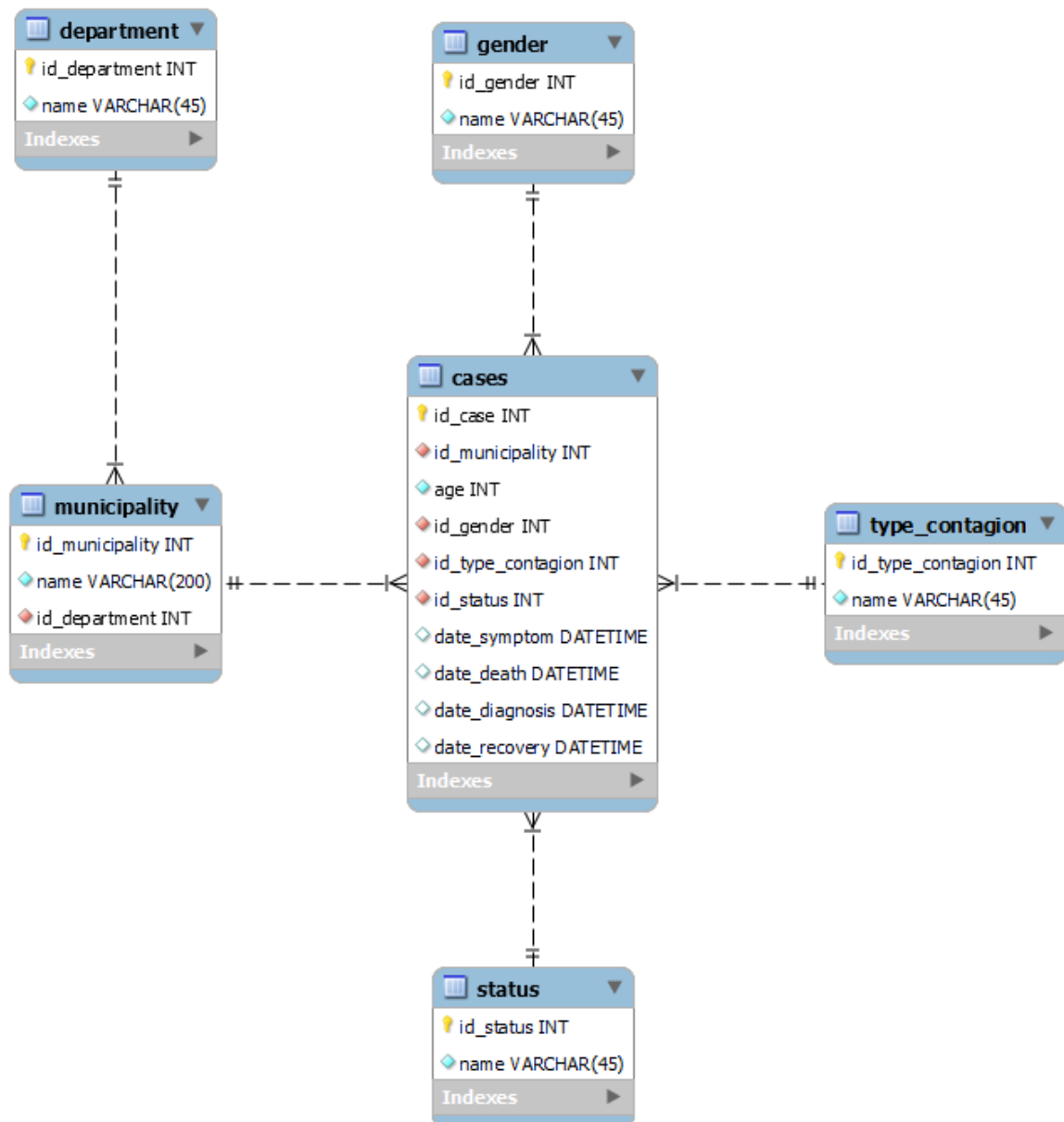


## PRUEBA TÉCNICA – RECURSO BI SABRINA BONILL

### 1. Crear una base de datos en MySQL.

- Con base en el modelo de datos presentado a continuación, construir una base de datos en el motor MySQL, con las tablas, atributos y relaciones indicadas.



## SCRIP CREACIÓN DE TABLAS MYSQL

-- La tabla principal Cases Fue alimentada de la ETL Python, posteriormente con esta creada se procedio a cerar tabalas de dimensiones del moelelo.

-- se Crea Base de Datos

```
CREATE DATABASE prueba_practica;
```

-- Cambio tipo de datos en cases

```
USE prueba_practica ;
```

```
ALTER TABLE cases MODIFY COLUMN id_case INT;
ALTER TABLE cases MODIFY COLUMN id_municipality INT;
ALTER TABLE cases MODIFY COLUMN name_municipality VARCHAR(200);
ALTER TABLE cases MODIFY COLUMN id_department INT;
ALTER TABLE cases MODIFY COLUMN name_department VARCHAR(45);
ALTER TABLE cases MODIFY COLUMN age INT;
ALTER TABLE cases MODIFY COLUMN gender VARCHAR(45);
ALTER TABLE cases MODIFY COLUMN id_gender INT;
ALTER TABLE cases MODIFY COLUMN type_contagion VARCHAR(45);
ALTER TABLE cases MODIFY COLUMN id_type_contagion INT;
ALTER TABLE cases MODIFY COLUMN status VARCHAR(45);
ALTER TABLE cases MODIFY COLUMN id_status INT;
ALTER TABLE cases MODIFY COLUMN date_symptom DATETIME;
ALTER TABLE cases MODIFY COLUMN date_death DATETIME;
ALTER TABLE cases MODIFY COLUMN date_diagnosis DATETIME;
ALTER TABLE cases MODIFY COLUMN date_recovery DATETIME;
```

-- Crear Tabla department

```
CREATE TABLE department
SELECT DISTINCT id_department, name_department
FROM cases
ORDER BY name_department;
```

```
ALTER TABLE department ADD PRIMARY KEY (id_department);
```

```
-- Crear Tabla municipality
```

```
CREATE TABLE municipality  
SELECT DISTINCT id_municipality, name_municipality, id_department  
FROM cases  
ORDER BY name_municipality;
```

```
-- Crear Tabla gender
```

```
CREATE TABLE gender  
SELECT DISTINCT id_gender, gender  
FROM cases ;
```

```
ALTER TABLE gender ADD PRIMARY KEY ( id_gender);
```

```
-- Crear Tabla type_contagion
```

```
CREATE TABLE type_contagion  
SELECT DISTINCT id_type_contagion, type_contagion  
FROM cases  
ORDER BY type_contagion;
```

```
ALTER TABLE type_contagion ADD PRIMARY KEY ( id_type_contagion);
```

```
-- Crear Tabla status
```

```
CREATE TABLE status  
SELECT DISTINCT id_status, status  
FROM cases  
ORDER BY status;
```

```
ALTER TABLE status ADD PRIMARY KEY (id_status);
```

```
-- Crear relaciones
```

```
-- fk_departament_municipality
```

```
ALTER TABLE municipality  
ADD CONSTRAINT fk_department_municipality  
FOREIGN KEY (id_department) REFERENCES department(id_department);
```

```
-- fk_municipality_cases PENDIENTE
```

```
DESCRIBE cases;  
DESCRIBE municipality;
```

```
ALTER TABLE cases  
ADD CONSTRAINT fk_municipality_cases  
FOREIGN KEY (id_municipality) REFERENCES municipality(id_municipality);
```

```
-- fk_municipality_state
```

```
DESCRIBE cases;  
DESCRIBE status;
```

```
ALTER TABLE cases  
ADD CONSTRAINT fk_status_cases  
FOREIGN KEY (id_status) REFERENCES status(id_status);
```

```
-- fk_type_contagion_cases
```

```
DESCRIBE cases;  
DESCRIBE type_contagion;
```

```
ALTER TABLE cases  
ADD CONSTRAINT fk_type_contagion_cases  
FOREIGN KEY (id_type_contagion) REFERENCES  
type_contagion(id_type_contagion);
```

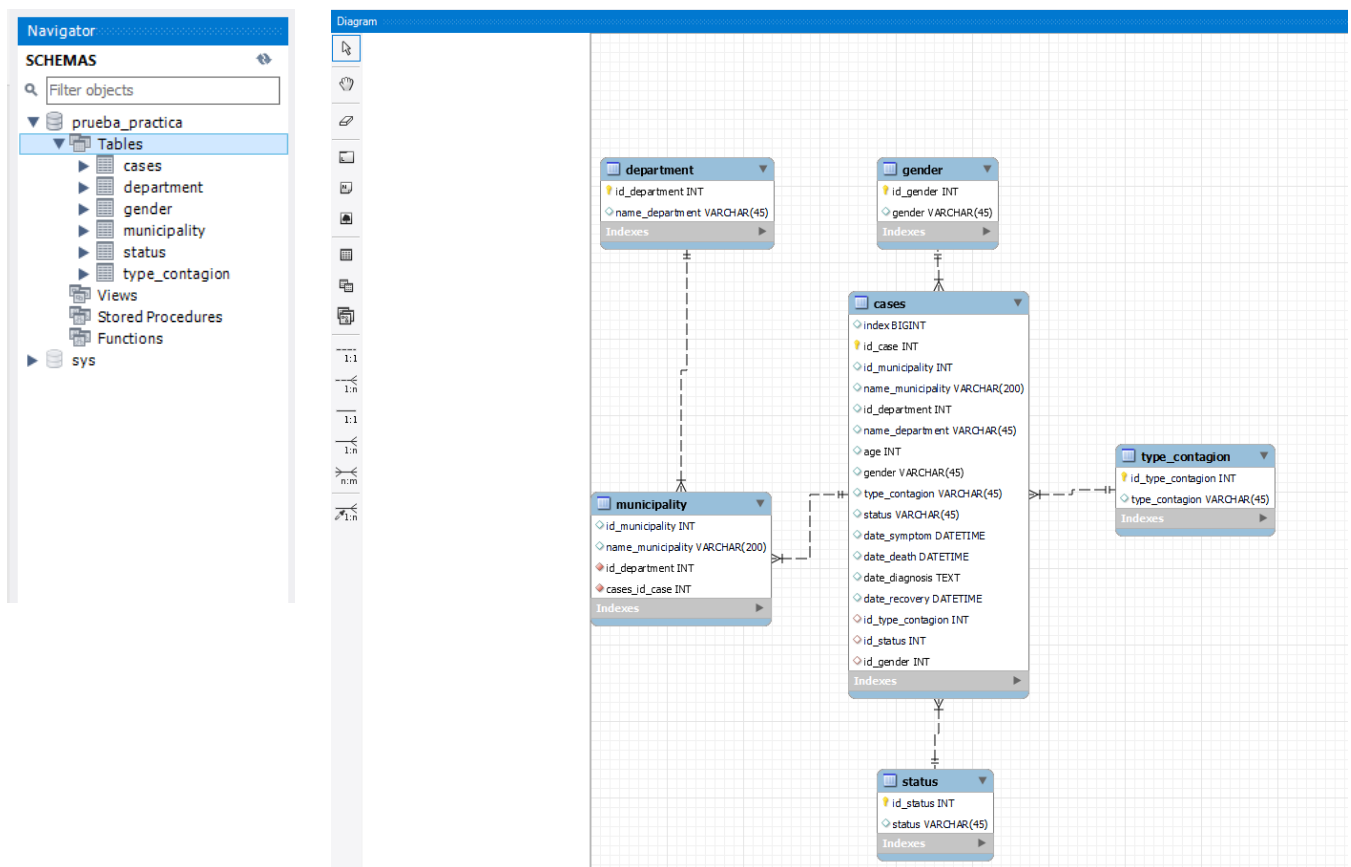
```
-- fk_gender_cases
```

```
DESCRIBE cases;  
DESCRIBE gender;
```

```
ALTER TABLE cases  
ADD CONSTRAINT fk_gender_cases  
FOREIGN KEY (id_gender) REFERENCES gender(id_gender);
```

Se presentaron algunos inconvenientes en la relación de la tabla municipalidades ya que esta relación es de muchos a uno puesto que el ID de municipalidades este duplicado, se procedió a generar una tabla intermedia para evitarse este conflicto, sin embargo, para el modelo en visualización se organizo una solución diferente.

Finalmente, el esquema quedo así:



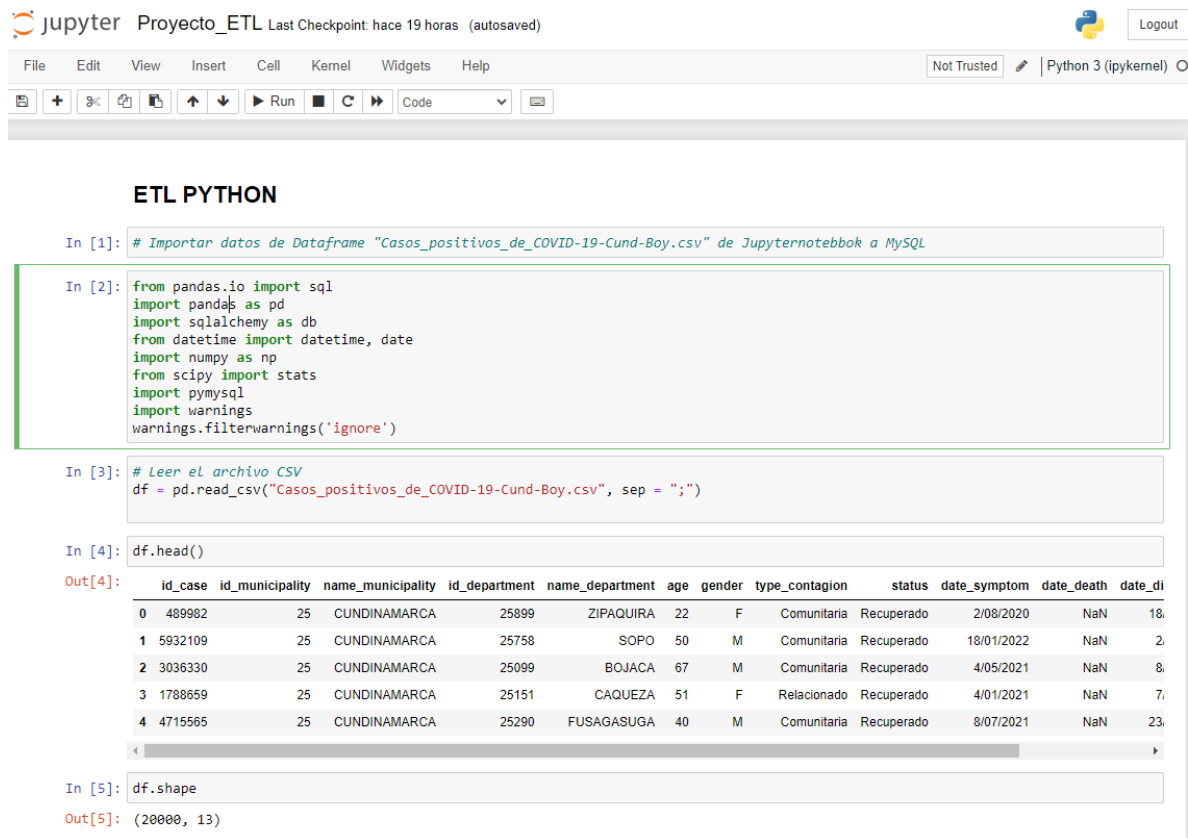
## 2. Desarrollar una ETL en Python.

Crear un proyecto en Python que permita realizar la lectura de los registros compartidos en el archivo CSV anexo a esta prueba “Casos\_positivos\_de\_COVID-19-Cund-Boy.csv”.

- Transformar los datos para realizar el cargue en las tablas previamente creadas en MySQL.
- Cargar los datos en las tablas destinadas.
- Considerar que si se vuelve a ejecutar la ETL no se deben duplicar los registros en la base de datos.

Para el desarrollo de la ETL en python se trabajó en Jupyter Notebook, generando la extracción de información de archivo csv Casos\_positivos\_de\_COVID-19-Cund-Boy.

Después de la importación de datos, se realizó transformación y finalmente carga de datos a MYSQL.



The screenshot shows a Jupyter Notebook titled "Proyecto\_ETL" with a toolbar at the top. The notebook contains five code cells. The first cell is a comment. The second cell imports various Python libraries including pandas, sqlalchemy, datetime, numpy, scipy, stats, pymysql, and warnings. The third cell reads a CSV file named "Casos\_positivos\_de\_COVID-19-Cund-Boy.csv". The fourth cell displays the first five rows of the data frame. The fifth cell shows the shape of the data frame.

```
In [1]: # Importar datos de Dataframe "Casos_positivos_de_COVID-19-Cund-Boy.csv" de Jupyternotebbok a MySQL
```

```
In [2]: from pandas.io import sql
import pandas as pd
import sqlalchemy as db
from datetime import datetime, date
import numpy as np
from scipy import stats
import pymysql
import warnings
warnings.filterwarnings('ignore')
```

```
In [3]: # Leer el archivo CSV
df = pd.read_csv("Casos_positivos_de_COVID-19-Cund-Boy.csv", sep = ";")
```

```
In [4]: df.head()
```

	id_case	id_municipality	name_municipality	id_department	name_department	age	gender	type_contagion	status	date_symptom	date_death	date_di
0	489982	25	CUNDINAMARCA	25899	ZIPAQUIRA	22	F	Comunitaria	Recuperado	2/08/2020	NaN	18.
1	5932109	25	CUNDINAMARCA	25758	SOPO	50	M	Comunitaria	Recuperado	18/01/2022	NaN	2.
2	3036330	25	CUNDINAMARCA	25099	BOJACA	67	M	Comunitaria	Recuperado	4/05/2021	NaN	8.
3	1788659	25	CUNDINAMARCA	25151	CAQUEZA	51	F	Relacionado	Recuperado	4/01/2021	NaN	7.
4	4715565	25	CUNDINAMARCA	25290	FUSAGASUGA	40	M	Comunitaria	Recuperado	8/07/2021	NaN	23.

```
In [5]: df.shape
```

```
Out[5]: (20000, 13)
```

```
In [6]: ## Exploración y transformación de datos de datos
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 13 columns):
#   Column                Non-Null Count  Dtype
---  -
0   id_case                20000 non-null  int64
1   id_municipality        20000 non-null  int64
2   name_municipality      20000 non-null  object
3   id_department          20000 non-null  int64
4   name_department        20000 non-null  object
5   age                   20000 non-null  int64
6   gender                 20000 non-null  object
7   type_contagion         20000 non-null  object
8   status                 19857 non-null  object
9   date_symptom           19873 non-null  object
10  date_death             589 non-null    object
11  date_diagnosis         19996 non-null  object
12  date_recovery          19422 non-null  object
dtypes: int64(4), object(9)
memory usage: 2.0+ MB
```

```
In [7]: #Quitar duplicados
df.drop_duplicates(subset='id_case',inplace=True)
```

```
In [8]: # Verificación si habian duplicados
df.shape
```

```
Out[8]: (20000, 13)
```

```
In [9]: # cambiar tipo de datos date

df['date_symptom'] = pd.to_datetime(df['date_symptom'])
df['date_death'] = pd.to_datetime(df['date_death'])
df['date_death'] = pd.to_datetime(df['date_diagnosis'])
df['date_recovery'] = pd.to_datetime(df['date_recovery'])
```

```
In [10]: # Crear campo id_type_contagion

condiciones = [
    (df['type_contagion'] == 'Comunitaria'),
    (df['type_contagion'] == 'Relacionado'),
    (df['type_contagion'] == 'Importado')
]
```

```

In [11]: resultados = [1,2,3]

In [12]: df['id_type_contagion'] = np.select(condiciones, resultados)

In [13]: # Crear campo id_status
condicionessta = [
    (df['status'] == 'Activo'),
    (df['status'] == 'Fallecido'),
    (df['status'] == 'Recuperado'),
    (df['status'] == 'Sin_Informacion')
]

In [14]: resultadossta = [1,2,3,4]

In [15]: df['id_status'] = np.select(condicionessta, resultadossta)

In [16]: # Crear campo id_gender
condicionesgen = [
    (df['gender'] == 'F'),
    (df['gender'] == 'M')
]

In [17]: resultadosgen = [1,2]

In [18]: df['id_gender'] = np.select(condicionesgen, resultadosgen)

In [19]: df
Out[19]:
   id_case  id_municipality  name_municipality  id_department  name_department  age  gender  type_contagion  status  date_symptom  date_death  da
0      489982           25  CUNDINAMARCA           25899      ZIQUAIRA      22    F  Comunitaria  Recuperado  2020-02-08  2020-08-18
1      5932109          25  CUNDINAMARCA           25758        SOPO      50    M  Comunitaria  Recuperado  2022-01-18  2022-02-02
2      3036330          25  CUNDINAMARCA           25099        BOJACA      67    M  Comunitaria  Recuperado  2021-04-05  2021-08-05
3      1788659          25  CUNDINAMARCA           25151        CAQUEZA      51    F  Relacionado  Recuperado  2021-04-01  2021-07-01
4      4715565          25  CUNDINAMARCA           25290      FUSAGASUGA      40    M  Comunitaria  Recuperado  2021-08-07  2021-07-23
...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...
19995  598195           15    BOYACA           15238        DUITAMA      46    F  Comunitaria  Recuperado  2020-12-08  2020-08-26
19996  4186035          15    BOYACA           15001        TUNJA      39    F  Relacionado  Recuperado  2021-12-06  2021-06-26

```

```

In [20]: # Conectar con MYSQL y Cargar datos

In [21]: database_username = 'root'
database_password = '1111'
database_ip = 'localhost'
database_name = 'prueba_practica'
database_connection = db.create_engine('mysql+pymysql://{0}:{1}@{2}/{3}'.
                                       format(database_username,database_password,
                                             database_ip, database_name))

connection = database_connection.connect()
metadata = db.MetaData()

In [22]: df.to_sql('cases', connection)
Out[22]: 20000

In [ ]:

```



## Se realiza verificación de creación en Mysql

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays the 'prueba\_practica' schema. The main area shows the 'cases' table with 1000 rows. The table has columns: index, id\_case, id\_municipality, name\_municipality, id\_department, name\_department, age, gender, type\_contagion, status, date\_symptom, date\_death, date\_diagnosis, and date\_recovery. The data includes various municipalities like CUNDINAMARCA, BOYACA, and SOACHA, with different statuses like 'Fallecido', 'Recuperado', and 'Comunitaria'.

index	id_case	id_municipality	name_municipality	id_department	name_department	age	gender	type_contagion	status	date_symptom	date_death	date_diagnosis	date_recovery
4915	587	25	CUNDINAMARCA	25873	VILLAPINZON	67	F	Relacionado	Fallecido	2020-03-15 00:00:00	2020-03-28 00:00:00	28/03/2020	2020-12-04 00:00:00
313	840	25	CUNDINAMARCA	25214	COTA	57	M	Importado	Recuperado	2020-03-24 00:00:00	2020-03-31 00:00:00	31/03/2020	2020-12-04 00:00:00
8648	1604	25	CUNDINAMARCA	25430	MADRID	35	M	Comunitaria	Recuperado	2020-03-28 00:00:00	2020-07-04 00:00:00	7/04/2020	2020-04-21 00:00:00
3696	1605	25	CUNDINAMARCA	25899	ZIPAQUIRA	20	F	Relacionado	Recuperado	2020-03-29 00:00:00	2020-07-04 00:00:00	7/04/2020	2020-04-15 00:00:00
13951	2007	15	BOYACA	15798	TENZA	58	F	Relacionado	Recuperado	2020-03-29 00:00:00	2020-08-04 00:00:00	8/04/2020	2020-04-19 00:00:00
11921	2010	15	BOYACA	15886	SANTANA	42	M	Relacionado	Recuperado	2020-03-29 00:00:00	2020-08-04 00:00:00	8/04/2020	2020-05-16 00:00:00
15728	2135	15	BOYACA	15816	TOGLI	19	M	Relacionado	Recuperado	2020-09-04 00:00:00	2020-09-04 00:00:00	9/04/2020	2020-05-05 00:00:00
17756	2137	15	BOYACA	15696	SANTA SOFIA	64	M	Relacionado	Recuperado	2020-03-28 00:00:00	2020-09-04 00:00:00	9/04/2020	2020-05-24 00:00:00
10548	2366	15	BOYACA	15001	TUNJA	24	F	Relacionado	Recuperado	2020-03-30 00:00:00	2020-10-04 00:00:00	10/04/2020	2020-05-25 00:00:00
5500	3486	25	CUNDINAMARCA	25430	MADRID	27	F	Relacionado	Recuperado	2020-02-04 00:00:00	2020-04-18 00:00:00	18/04/2020	2020-05-31 00:00:00
3630	4098	25	CUNDINAMARCA	25754	SOACHA	28	F	Relacionado	Recuperado	2020-03-28 00:00:00	2020-04-21 00:00:00	21/04/2020	2020-05-31 00:00:00

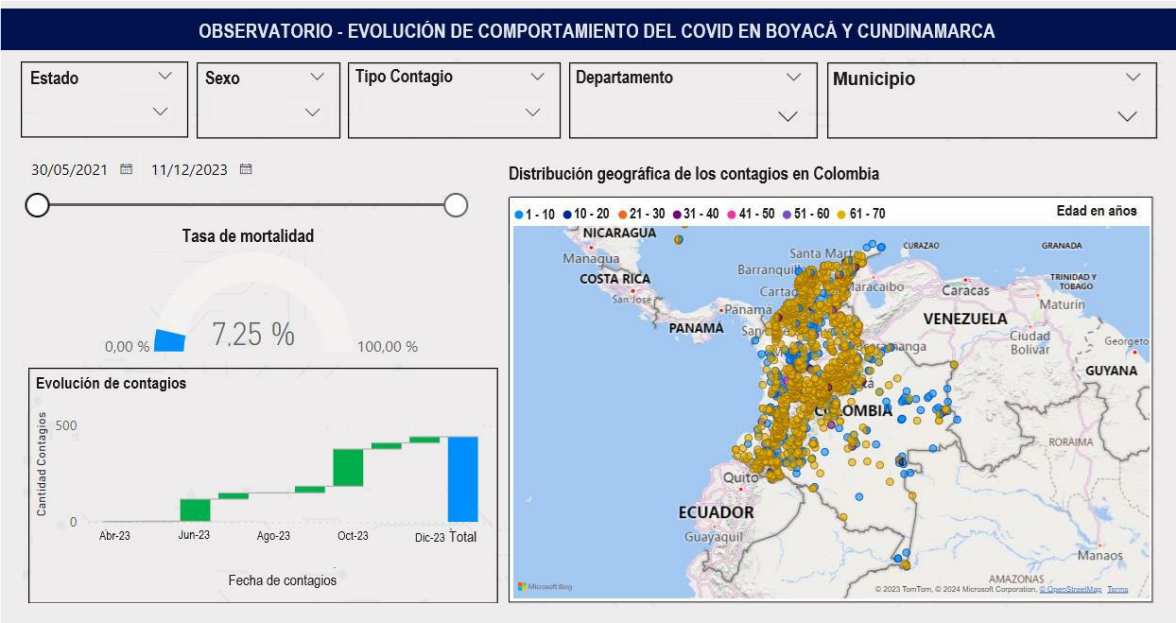
The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays the 'prueba\_practica' schema. The main area shows the 'cases' table with 20000 rows. The table has columns: index, id\_case, id\_municipality, name\_municipality, id\_department, name\_department, age, gender, type\_contagion, status, date\_symptom, date\_death, date\_diagnosis, and date\_recovery. The data includes various municipalities like CUNDINAMARCA, BOYACA, and SOACHA, with different statuses like 'Fallecido', 'Recuperado', and 'Comunitaria'.

index	id_case	id_municipality	name_municipality	id_department	name_department	age	gender	type_contagion	status	date_symptom	date_death	date_diagnosis	date_recovery
4915	587	25	CUNDINAMARCA	25873	VILLAPINZON	67	F	Relacionado	Fallecido	2020-03-15 00:00:00	2020-03-28 00:00:00	28/03/2020	2020-12-04 00:00:00
313	840	25	CUNDINAMARCA	25214	COTA	57	M	Importado	Recuperado	2020-03-24 00:00:00	2020-03-31 00:00:00	31/03/2020	2020-12-04 00:00:00
8648	1604	25	CUNDINAMARCA	25430	MADRID	35	M	Comunitaria	Recuperado	2020-03-28 00:00:00	2020-07-04 00:00:00	7/04/2020	2020-04-21 00:00:00
3696	1605	25	CUNDINAMARCA	25899	ZIPAQUIRA	20	F	Relacionado	Recuperado	2020-03-29 00:00:00	2020-07-04 00:00:00	7/04/2020	2020-04-15 00:00:00
13951	2007	15	BOYACA	15798	TENZA	58	F	Relacionado	Recuperado	2020-03-29 00:00:00	2020-08-04 00:00:00	8/04/2020	2020-04-19 00:00:00
11921	2010	15	BOYACA	15886	SANTANA	42	M	Relacionado	Recuperado	2020-03-29 00:00:00	2020-08-04 00:00:00	8/04/2020	2020-05-16 00:00:00
15728	2135	15	BOYACA	15816	TOGLI	19	M	Relacionado	Recuperado	2020-09-04 00:00:00	2020-09-04 00:00:00	9/04/2020	2020-05-05 00:00:00
17756	2137	15	BOYACA	15696	SANTA SOFIA	64	M	Relacionado	Recuperado	2020-03-28 00:00:00	2020-09-04 00:00:00	9/04/2020	2020-05-24 00:00:00
10548	2366	15	BOYACA	15001	TUNJA	24	F	Relacionado	Recuperado	2020-03-30 00:00:00	2020-10-04 00:00:00	10/04/2020	2020-05-25 00:00:00
5500	3486	25	CUNDINAMARCA	25430	MADRID	27	F	Relacionado	Recuperado	2020-02-04 00:00:00	2020-04-18 00:00:00	18/04/2020	2020-05-31 00:00:00
3630	4098	25	CUNDINAMARCA	25754	SOACHA	28	F	Relacionado	Recuperado	2020-03-28 00:00:00	2020-04-21 00:00:00	21/04/2020	2020-05-31 00:00:00

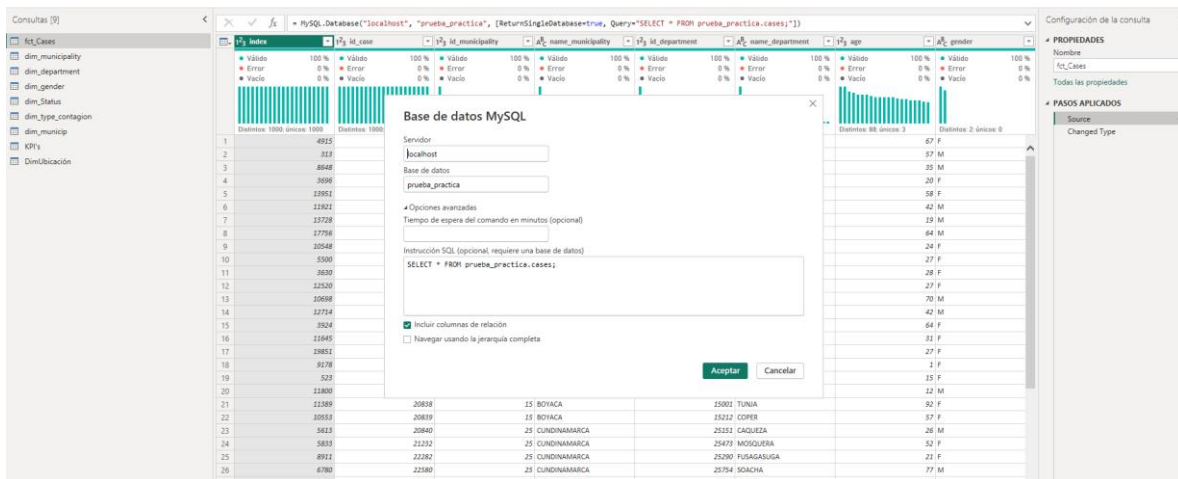
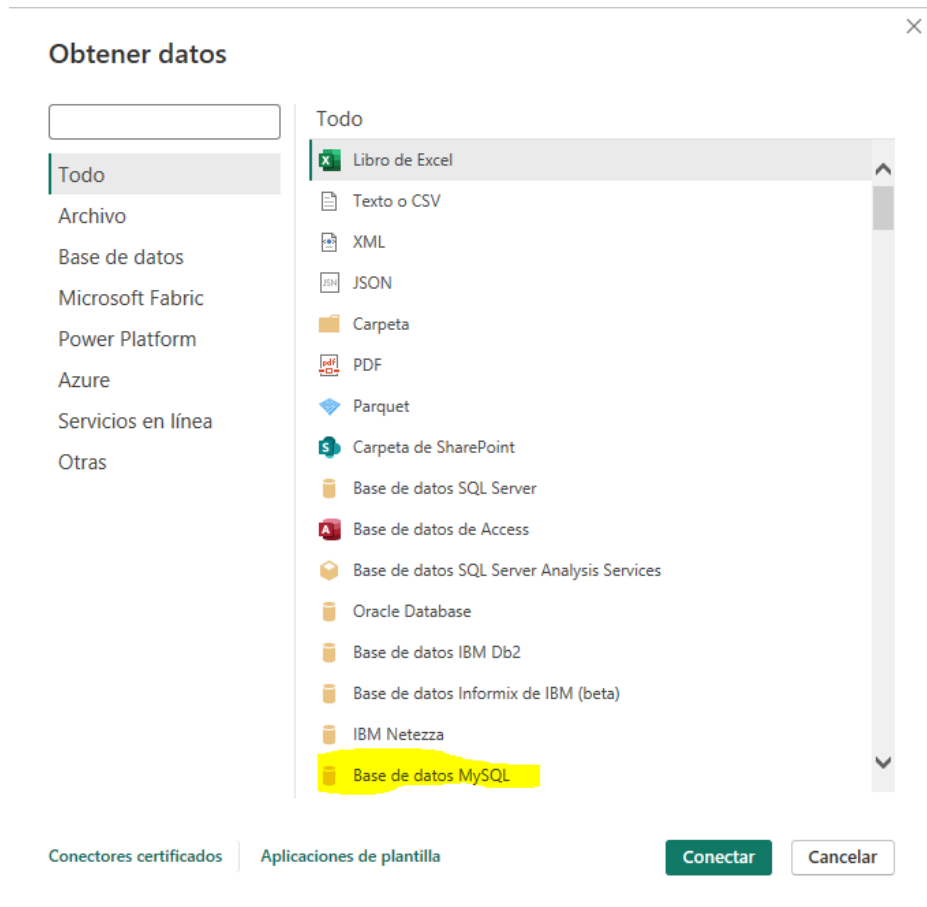
20000 registros cargados, a partir de esta tabla se crearon las tablas de dimensiones.

3. Construir un tablero de control en POWERBI.

Construir un tablero en POWERBI que permita realizar los filtros, el cálculo de los indicadores y la visualización geográfica de los registros cargados en la base de datos MySQL. A continuación, se podrá evidenciar un Mockup con la estructura deseada del tablero de control.

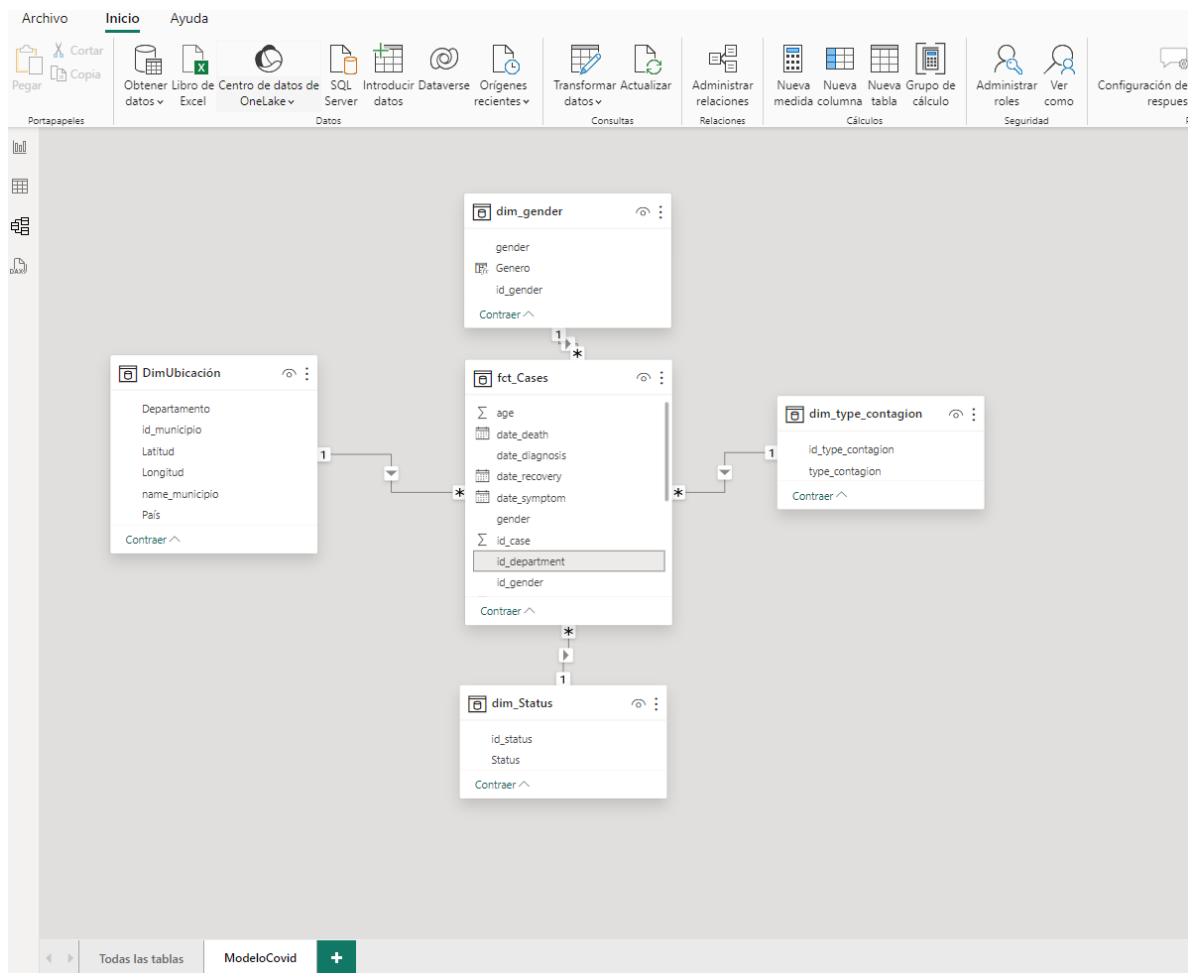


Para la creación de tablero se procedió a realizar conexión por medio de Base de datos MySQL



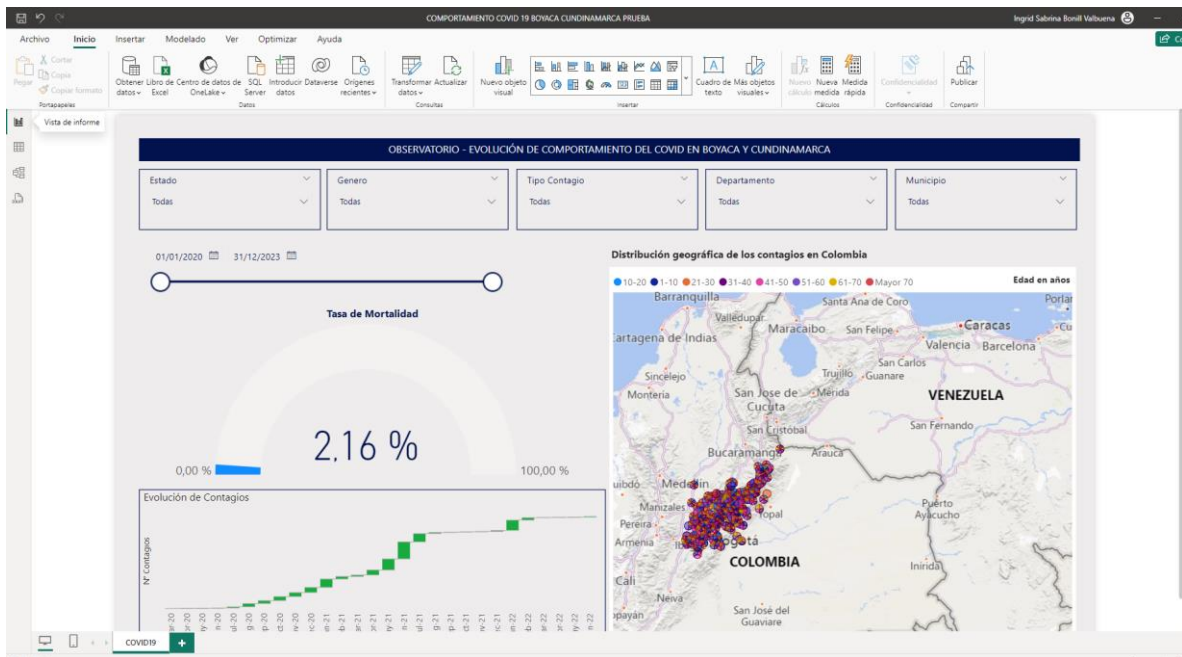
También se realizó la creación de tabla calendario y medidas para posteriormente realizar la creación del modelo.

## Modelo PBI:



Debido a que la relación existente entre base Municipalidades y Cases era de una muchos generando algunos conflictos se procedió a crear tabla Dimubicaciones en la cual se concentro toda la información para relaciones geográficas lo cual facilito la visualización.

Finalmente, este es el resultado del tablero.



Todos los archivos fueron enviados por correo y se subieron a repositorio GITHUB.