Universidad de Sonora

Facultad Interdisciplinaria de Ciencias Exactas y Naturales

Curso propedéutico de Base de Datos

Base de Datos del COVID

Realizado por:

Merino Cedeño Ángel Alberto

Hermosillo, Sonora

11 de junio de 2023

Contenido

Tabla de ilustraciones

Introducción

En diciembre de 2019, se inició un brote de una neumonía grave en la ciudad de Wuhan, China. La enfermedad se propagó rápidamente y afectó principalmente a adultos de entre 30 y 79 años, con una tasa de letalidad global del 2.3%. Los primeros casos estaban relacionados con un mercado de mariscos llamado Huanan Seafood Wholesale Market, donde también se vendía carne de animales silvestres consumidos localmente. Los estudios iniciales para identificar la causa de la enfermedad dieron resultados negativos para agentes comunes de infección respiratoria. Sin embargo, mediante el uso de secuenciación genética, se descubrió que se trataba de un nuevo tipo de coronavirus, denominado inicialmente como 2019-nCoV. El brote se extendió rápidamente en China y luego se propagó a otros países y continentes, lo que llevó a la Organización Mundial de la Salud (OMS) a declarar la pandemia el 11 de marzo de 2020. Esta situación ha sido descrita como la mayor emergencia en la salud pública mundial de los tiempos modernos (Javier Díaz-Castrillón & Toro-Montoya, 2020).

En un inicio, la enfermedad tenía tasas de letalidad estimadas entre el 1% y el 3% y afectaba principalmente a adultos mayores y personas con enfermedades preexistentes. Su comportamiento hoy en día sigue siendo el mismo. El período de incubación promedio es de 5 días, pero puede durar hasta 14 días. También, cuando el COVID, fue un problema de grado mayor, existieron infectados que eran asintomáticos, pero que seguían siendo altamente contagiosos debido a su alta carga viral. Problema que dificultó el control de la propagación y que colapsó los sistemas de salud en áreas afectadas. Se analizaron aspectos como el patógeno, la epidemiología, las manifestaciones clínicas, el diagnóstico y el tratamiento (Javier Díaz-Castrillón & Toro-Montoya, 2020).

Este proyecto tiene fines meramente académicos. Dado que el COVID fue un problema a nivel mundial, hubo una gran cantidad de registros de personas que sufrieron de síntomas y querían asegurarse de su estado. En un principio fueron pocas las personas, sin embargo, al poco tiempo pasó de ser una enfermedad que podía tener cualquiera y que afectó, e impactó a un gran número de personas. El objetivo de este trabajo es hacer un análisis con una base de datos de México sobre este tema en cuestión. Análisis basado en la creación y manipulación de datos, interacción con la misma base y manejo del modelo entidad-relación, tablas, *queries*, vistas, etc. El resultado de este trabajo se

puede observar en el repositorio de GitHub, donde vienen los archivos *sql*. Los cuales tienen los siguientes significados:

- Queries.sql: Es el archivo que demuestra las queries que se hicieron para crear tablas, views, funciones, etc.
- Modelo_ER_covid.mwb: Es el archivo que demuestra el modelo Entidad – Relación creado en *mysql workbench*.
- Covid.sql: Es el archivo que permite importar toda la base de datos creada. Al importar este archivo podrá verse todos los datos con los que se trabajaron, las tablas creadas, vistas, etc. El modelo Entidad – Relación se genera automáticamente con Reverse Engineer.

Metodología

La base de datos se recopiló de los datos abiertos proporcionados por el Gobierno de México, y se utilizaron alrededor de 120 mil registros, dado que era para fines académicos y demostrativos; si fuera con un fin profesional, o de investigación, se trabajaría con una base de datos lo más completa posible. Para este proyecto se hizo uso del programa *mysql workbench*, el cual es muy útil y sencillo de utilizar. El análisis de la metodología será paso a paso para que sea comprensible lo que se hizo y lo que se obtuvo.

Empezamos por la parte de la importación de datos. Primero, creamos una base de datos (o *schema*) llamada *covid*. Seleccionamos este *schema* y empezamos a trabajar. Dado que los datos vienen en formato *csv*, este se puede leer sin problemas en el *workbench*. Hacemos la importación.

FECHA_ACTUALIZACION	ID_REGISTRO	ORIGEN	SECTOR	ENTIDAD_UM	SEXO	ENTIDAD_NAC	ENTIDAD_RES	MUNICIPIO_RES	TIPO_PACIENTE	FECHA_INGRESO	FECHA_SINTOMAS	FECH
2023-02-07	013a0e	2	12	9	2	9	9	9	1	2022-01-17	2022-01-14	9999-
2023-02-07	014651	2	12	28	1	28	28	22	1	2022-10-23	2022-10-10	9999-
2023-02-07	01e27d	2	9	25	2	25	25	1	1	2022-02-14	2022-02-14	9999-
2023-02-07	02380f	1	12	7	2	9	7	19	1	2022-08-23	2022-08-21	9999-
2023-02-07	03a438	2	12	1	1	1	1	1	1	2022-02-17	2022-02-16	9999-
2023-02-07	03b5db	2	12	8	2	8	8	37	1	2022-01-10	2022-01-06	9999-
2023-02-07	03d72b	2	6	28	2	28	28	22	1	2022-09-27	2022-09-23	9999-
2023-02-07	03d8b5	2	9	9	2	9	9	3	1	2022-06-19	2022-06-18	9999-
2023-02-07	043fb6	2	12	15	1	20	15	58	1	2022-01-24	2022-01-24	9999-
2023-02-07	044350	2	6	9	1	15	9	2	2	2022-03-06	2022-03-03	9999-
2023-02-07	045408	1	12	22	1	22	22	4	1	2022-01-28	2022-01-24	9999-
2023-02-07	045795	1	12	9	2	9	9	10	1	2022-08-30	2022-08-30	9999-
2023-02-07	04583b	1	6	5	1	5	5	28	1	2022-06-20	2022-06-17	9999-
2023-02-07	04af5f	2	12	9	1	9	9	7	1	2022-07-06	2022-07-01	9999-
2023-02-07	04c140	1	12	23	2	8	23	8	1	2022-07-22	2022-07-20	9999-
2023-02-07	04f190	2	6	9	1	9	9	15	1	2022-02-18	2022-02-18	9999-
2023-02-07	04f613	2	12	27	1	27	27	4	1	2022-07-06	2022-07-03	9999-
2023-02-07	04fc18	2	10	9	2	9	15	12	1	2022-05-19	2022-05-19	9999-

Figura 1 Visualización de una parte de los datos importados

Para saber si hay algún problema, se hizo una *query* que nos enseñara los primeros 500 registros de toda la tabla de registros. Lo primero que vemos es que hay un error de acentuación. Para resolver este problema de los datos, hacemos una *query* que arregle los valores de la siguiente manera.

```
UPDATE registros
SET pais_nacionalidad =
    CASE
        WHEN pais_nacionalidad LIKE '%MÃ©xico%' THEN REPLACE(pais_nacionalidad, 'MÃ©xico', 'México')
        WHEN pais_nacionalidad LIKE '%AmÃ©rica%' THEN REPLACE(pais_nacionalidad, 'AmÃ©rica', 'América')
        WHEN pais_nacionalidad LIKE '%TaiwÃ¡n%' THEN REPLACE(pais_nacionalidad, 'TaiwÃ¡n', 'Taiwán')
        WHEN pais_nacionalidad LIKE '%EspaÃ±a%' THEN REPLACE(pais_nacionalidad, 'EspaÃ±a', 'España')
        WHEN pais_nacionalidad LIKE '%JapÃ³n%' THEN REPLACE(pais_nacionalidad, 'JapÃ³n', 'Japón')
        WHEN pais_nacionalidad LIKE '%RepÃºblica%' THEN REPLACE(pais_nacionalidad, 'RepÃºblica', 'República')
        WHEN pais_nacionalidad LIKE '%CanadÃ¡%' THEN REPLACE(pais_nacionalidad, 'CanadÃ¡', 'Canadá')
        WHEN pais_nacionalidad LIKE '%GabÃ³n%' THEN REPLACE(pais_nacionalidad, 'GabÃ³n', 'Gabón')
        WHEN pais_nacionalidad LIKE '%HaitÃ-%' THEN REPLACE(pais_nacionalidad, 'HaitÃ-', 'Haití')
        WHEN pais_nacionalidad LIKE '%BretaÃ±a%' THEN REPLACE(pais_nacionalidad, 'BretaÃ±a', 'Bretaña')
        WHEN pais_nacionalidad LIKE '%IrÃ¡n%' THEN REPLACE(pais_nacionalidad, 'IrÃ¡n', 'Irán')
        WHEN pais_nacionalidad LIKE '%PerÃº%' THEN REPLACE(pais_nacionalidad, 'PerÃº', 'Perú')
        WHEN pais_nacionalidad LIKE '%PanamÃ¡%' THEN REPLACE(pais_nacionalidad, 'PanamÃ¡', 'Panamá')
        ELSE pais_nacionalidad
    END;
```

Figura 2 Query para arreglar los datos con errores

Con esto resolvemos el problema que traían los datos. Es importante hacer correcciones de este tipo porque si se requiere de algún análisis con estos datos, es necesario tenerlo bien y no con errores.

El *dataset* viene sólo con la tabla de registros, y no deja claro el significado de algunos valores dentro de cada registro. El significado de estos valores, y de las columnas, de la tabla importada, viene dado en el diccionario y catálogos dado por el Gobierno de México. El archivo de mayor relevancia para hacer *queries* y más, es el de catálogos, donde vienen las claves de las columnas y demás. Siendo que este archivo es un Excel de varias pestañas y que *mysql* no puede leer este tipo de archivos, se crearon las tablas a partir de *queries*, se pudieron haber importado con la opción *Table Data Import Wizard*, pero la mayoría de las tablas eran de pocos datos, entonces no había mucho problema en hacerlo de la manera tradicional. Sólo la tabla de los municipios fue importada de esa manera dado que hay aproximadamente 2400 municipios en México y esa sí valía la pena importarla en vez de hacer la *query*.

```
CREATE TABLE origen (
    CLAVE INT PRIMARY KEY NOT NULL,
    DESCRIPCION VARCHAR(255) NOT NULL,
    UNIQUE (Clave));
INSERT INTO origen (CLAVE, DESCRIPCION)
VALUES (1, 'USMER'),
       (2, 'FUERA DE USMER'),
       (99, 'NO ESPECIFICADO');
```

```
drop table if exists tipo_paciente;
CREATE TABLE tipo_paciente (
    CLAVE INT PRIMARY KEY NOT NULL,
    DESCRIPCION VARCHAR(255) NOT NULL,
    UNIQUE (Clave));
INSERT INTO tipo_paciente (CLAVE, DESCRIPCION)
VALUES (1, 'AMBULATORIO'),
       (2, 'HOSPITALIZADO'),
       (99, 'NO ESPECIFICADO');
```

Figura 3 Ejemplo de queries para crear las tablas

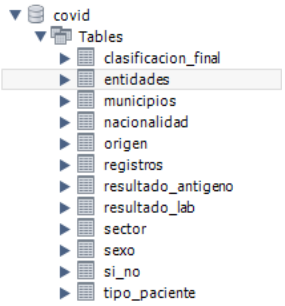

Figura 4 Resultado de las tablas a partir de las queries e import wizard

Esto se hizo para generar varias tablas necesarias para la descripción e interpretación de los datos a la hora de hacer *queries.* También para la construcción del modelo Entidad – Relación. Dado que ya tenemos lo necesario, construimos dicho modelo, indicando las llaves primarias y foráneas de la tabla *registros*.

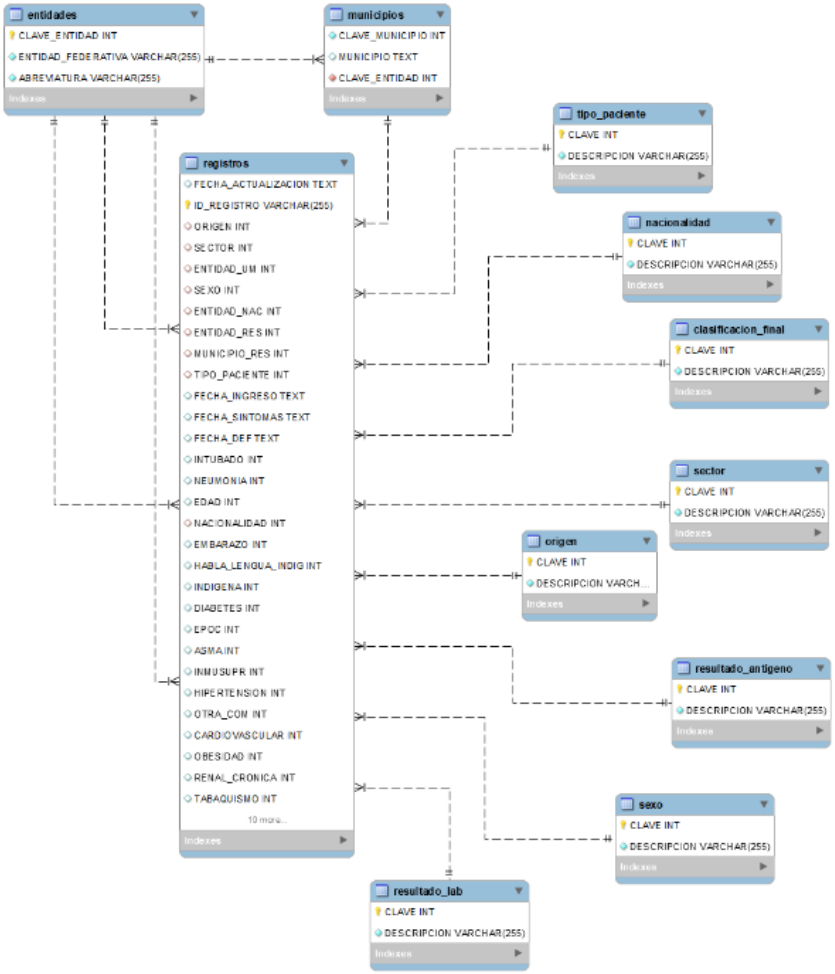

Figura 5 Modelo Entidad - Relación

Siendo que no podemos permitir que cualquiera manipule la base de datos interna, si queremos que alguien vea y trabaje con ella, hay columnas que no son de su interés y sólo se le deberían permitir lo que se considere como accesible para esa persona. Bajo este contexto se crea la siguiente vista, junto con la siguiente *query*.

```
CREATE VIEW registros_vista AS
SELECT
FECHA_INGRESO, SEXO, EDAD, ENTIDAD_NAC, ENTIDAD_RES, MUNICIPIO_RES, PAIS_NACIONALIDAD, FECHA_DEF, SECTOR,
TIPO_PACIENTE, INTUBADO, NEUMONIA, EMBARAZO, HABLA_LENGUA_INDIG, INDIGENA, DIABETES, EPOC, ASMA, INMUSUPR,
 HIPERTENSION, OTRA_COM, CARDIOVASCULAR, OBESIDAD, RENAL_CRONICA, TABAQUISMO, OTRO_CASO, TOMA_MUESTRA_LAB,
 RESULTADO_LAB, TOMA_MUESTRA_ANTIGENO, RESULTADO_ANTIGENO, CLASIFICACION_FINAL, MIGRANTE, UCI
FROM registros; //
```

Figura 6 Vista de los registros permitidos

FECHA_INGRESO	SEXO	EDAD	ENTIDAD_NAC	ENTIDAD_RES	MUNICIPIO_RES	PAIS_NACIONALIDAD	FECHA_DEF	SECTOR	TIPO_PACIENTE	INTUBADO	NEUMONIA	EMBARAZO	HABLA_L
2022-01-17	2	23	9	9	9	México	9999-99-99	12	1	97	2	97	2
2022-10-23	1	36	28	28	22	México	9999-99-99	12	1	97	2	1	2
2022-02-14	2	81	25	25	1	México	9999-99-99	9	1	97	2	97	2
2022-08-23	2	62	9	7	19	México	9999-99-99	12	1	97	2	97	2
2022-02-17	1	35	1	1	1	México	9999-99-99	12	1	97	2	2	2
2022-01-10	2	34	8	8	37	México	9999-99-99	12	1	97	1	97	2
2022-09-27	2	34	28	28	22	México	9999-99-99	6	1	97	2	97	2
2022-06-19	2	62	9	9	3	México	9999-99-99	9	1	97	2	97	2
2022-01-24	1	40	20	15	58	México	9999-99-99	12	1	97	2	2	2
2022-03-06	1	28	15	9	2	México	9999-99-99	6	2	2	2	2	2
2022-01-28	1	34	22	22	4	México	9999-99-99	12	1	97	2	2	2
2022-08-30	2	57	9	9	10	México	9999-99-99	12	1	97	2	97	2
2022-06-20	1	42	5	5	28	México	9999-99-99	6	1	97	2	2	99
2022-07-06	1	28	9	9	7	México	9999-99-99	12	1	97	2	2	2
2022-07-22	2	23	8	23	8	México	9999-99-99	12	1	97	2	97	2
2022-02-18	1	52	9	9	15	México	9999-99-99	6	1	97	2	2	2
2022-07-06	1	39	27	27	4	México	9999-99-99	12	1	97	2	2	2
2022-05-19	2	40	9	15	12	México	9999-99-99	10	1	97	2	97	2

Figura 7 Tabla generada a partir de la vista

Esta vista sólo incluye las columnas que creemos pertinentes para la persona que quiera trabajar con estos datos, o la manipulación que desee, sin miedo de que los datos de la tabla original se vean afectados. Ya con esta vista, se crearon los siguientes procedimientos almacenados. Procedimientos sencillos que pueden servir como ejemplo de lo que se puede generar con esta vista, y del potencial que se tienen.

```
CREATE PROCEDURE indigenas_por_estado()
BEGIN
    SELECT en.ENTIDAD_FEDERATIVA as Entidad,
    SUM(case when re.sexo = 1 then 1 else 0 end) as Mujeres,
    SUM(case when re.sexo = 2 then 1 else 0 end) as Hombres,
    SUM(case when re.indigena = 1 and sexo = 1 then 1 else 0 end) as 'Mujeres Indigenas',
    SUM(case when re.indigena = 1 and sexo = 2 then 1 else 0 end) as 'Hombres Indigenas',
    count(*) as 'Total por Estado'
    FROM registros_vista re, entidades en
    WHERE re.ENTIDAD_RES = en.CLAVE_ENTIDAD
    GROUP BY ENTIDAD_FEDERATIVA
    ORDER BY Entidad;
END //
```

Figura 8 Query para crear el procedimiento almacenado "indígenas por estado"

Entidad	Mujeres	Hombres	Mujeres Indigenas	Hombres Indigenas	Total por Estado
AGUASCALIENTES	828	624	2	2	1452
BAJA CALIFORNIA	2083	1475	7	3	3558
BAJA CALIFORNIA SUR	2294	1911	1	3	4205
CAMPECHE	273	262	3	1	535
CHIAPAS	285	219	4	1	504
CHIHUAHUA	1543	970	5	5	2513
CIUDAD DE MÉXICO	21909	16648	58	39	38557
COAHUILA DE ZARAGOZA	2393	1789	5	3	4182
COLIMA	732	582	0	0	1314
DURANGO	1210	912	4	1	2122
GUANAJUATO	3125	2373	8	3	5498
GUERRERO	492	386	8	6	878
HIDALGO	834	653	12	4	1487
JALISCO	2543	1881	6	3	4424

Figura 9 Tabla parcial generada por el procedimiento almacenado "indígenas por estado"

En México, uno de los grupos vulnerables es el de los grupos indígenas. Este grupo va disminuyendo cada vez más al paso del tiempo, por esto es importante ver cuántos afectados han sido por culpa de este virus, que, en su momento fue mortal. Este procedimiento genera una tabla que provee la siguiente información. Indica el número de casos, separado por mujeres y hombres, y dentro este número, indica cuántos fueron casos donde la mujer, o el hombre, eran indígenas; al final se muestra el total de casos, o sea, la suma de los casos de mujeres y hombres, independientemente de si eran indígenas o no.

El siguiente procedimiento almacenado indica el tipo de paciente al que pertenece cada registro.

```
CREATE PROCEDURE casos_tipo_paciente()
BEGIN
    SELECT en.ENTIDAD_FEDERATIVA AS 'ESTADO', ti.DESCRIPCION AS 'TIPO DE PACIENTE', COUNT(*) AS 'TOTAL DE CASOS'
    FROM registros_vista re, tipo_paciente ti, entidades en
    WHERE re.ENTIDAD_RES = en.CLAVE_ENTIDAD
    AND re.TIPO_PACIENTE = ti.CLAVE
    GROUP BY en.ENTIDAD_FEDERATIVA , ti.DESCRIPCION
    ORDER BY en.ENTIDAD_FEDERATIVA;
END //
```

Figura 10 Query para crear el procedimiento almacenado "casos, tipo de paciente"

ESTADO	TIPO DE PACIENTE	TOTAL DE CASOS
AGUASCALIENTES	AMBULATORIO	1386
AGUASCALIENTES	HOSPITALIZADO	66
BAJA CALIFORNIA	AMBULATORIO	3418
BAJA CALIFORNIA	HOSPITALIZADO	140
BAJA CALIFORNIA SUR	AMBULATORIO	4163
BAJA CALIFORNIA SUR	HOSPITALIZADO	42
CAMPECHE	AMBULATORIO	519
CAMPECHE	HOSPITALIZADO	16
CHIAPAS	AMBULATORIO	492
CHIAPAS	HOSPITALIZADO	12
CHIHUAHUA	AMBULATORIO	2428
CHIHUAHUA	HOSPITALIZADO	85
CIUDAD DE MÉXICO	AMBULATORIO	38142

Figura 11 Tabla parcial generada por el procedimiento almacenado "casos, tipo de paciente"

Esto nos indica el número de casos por estado, donde el tipo de paciente fue ambulatorio u hospitalizado. Esta información puede ser importante para alguien como el director de un hospital

que quiera tomar alguna decisión para ampliar una zona del hospital dado el número de casos de gente hospitalizada.

Los siguientes dos procedimientos almacenados demuestran la cantidad de casos en los que fueron gente de la tercera edad (mayores de edad), y personas menores de edad. Para esto, se le requiere al usuario el ID correspondiente al estado.

```
CREATE PROCEDURE PacientesMenores_x_Estado (IN idEstado INT)
READS SQL DATA
BEGIN
    SELECT mu.MUNICIPIO AS 'Municipio de Residencia',
    COUNT(re.MUNICIPIO_RES) AS Casos
    FROM registros_vista re, municipios mu
    WHERE mu.CLAVE_MUNICIPIO = re.MUNICIPIO_RES
    AND mu.CLAVE_ENTIDAD = re.ENTIDAD_RES
    AND re.ENTIDAD_RES = idEstado
    AND re.EDAD < 18
    GROUP BY mu.MUNICIPIO
    ORDER BY mu.MUNICIPIO;
END //
```

Figura 12 Query para crear el procedimiento almacenado "pacientes menores de edad"

```
CREATE PROCEDURE PacientesMayores_x_Estado (IN idEstado INT)
READS SQL DATA
BEGIN
    SELECT mu.MUNICIPIO AS 'Municipio de Residencia',
    COUNT(re.MUNICIPIO_RES) AS Casos
    FROM registros_vista re, municipios mu
    WHERE mu.CLAVE_MUNICIPIO = re.MUNICIPIO_RES
    AND mu.CLAVE_ENTIDAD = re.ENTIDAD_RES
    AND re.ENTIDAD_RES = idEstado
    AND re.EDAD >= 60
    GROUP BY mu.MUNICIPIO
    ORDER BY mu.MUNICIPIO;
END //
```

Figura 13 Query para crear el procedimiento almacenado "pacientes mayores de edad"

Municipio de Residencia	Casos
AGUA PRIETA	1
ALAMOS	2
ARIZPE	1
CABORCA	18
CAJEME	25
CANANEA	7
EMPALME	7

Figura 14 Tabla parcial generada por el procedimiento almacenado "pacientes mayores de edad"

Municipio de Residencia	Casos
AGUA PRIETA	2
BACANORA	1
CABORCA	42
CAJEME	30
CANANEA	6
EMPALME	2
GENERAL PLUTARCO ELÍAS CALLES	2
GUAYMAS	6
HERMOSILLO	34

Figura 15 Tabla parcial generada por el procedimiento almacenado "pacientes menores de edad"

Esto es otro procedimiento bastante relevante dado que en un inicio era bien sabido que los pacientes que más se veían afectados por el virus, eran los pacientes mayores de edad. También, se sabía que los menos afectados eran los niños, o gente menor de edad, sin embargo, siempre es importante hacer un estudio, o análisis, de los datos para poder saber si dicha hipótesis es cierta o nula.

El último procedimiento indica el número de defunciones por estado. Recordemos que se trabajó con una base de datos parcial, aproximadamente 120 mil datos, esto dado que es una actividad con fines prácticos y académicos, si se quisiera saber un valor más acertado del número de defunciones, se deberá descargar del mismo lugar, la base de datos más completa. Este trabajo se hizo con la finalidad de que sea reproducible, por lo que sólo se debería importar dicha base de datos completa, correr el archivo *queries.sql*, archivo en el cual, al correrlo, crearía el mismo procedimiento permitiendo conocer los valores más acertados de dicho procedimiento almacenado. Pero para este trabajo, con los 120 mil datos, se obtuvo lo siguiente.

```
CREATE PROCEDURE defunciones_x_edad()
BEGIN
    SELECT en.ENTIDAD_FEDERATIVA as 'ESTADO',
    SUM(case when re.edad <18 then 1 else 0 end) as 'Menores de edad',
    SUM(case when re.edad between 18 and 59 then 1 else 0 end) as 'Mayores de edad',
    SUM(case when re.edad >= 60 then 1 else 0 end) as 'Adultos mayores',
    count(*) as 'Total de defunciones'
    FROM registros_vista re, entidades en
    WHERE re.FECHA_DEF not like '%9999-99-99%'
    AND re.ENTIDAD_RES = en.CLAVE_ENTIDAD
    GROUP BY en.ENTIDAD_FEDERATIVA
    ORDER BY en.ENTIDAD_FEDERATIVA;
END //
```

Figura 16 Query para crear el procedimiento almacenado "defunciones por edad"

ESTADO	Menores de edad	Mayores de edad	Adultos mayores	Total de defunciones
AGUASCALIENTES	1	7	7	15
BAJA CALIFORNIA	0	10	17	27
BAJA CALIFORNIA SUR	0	2	7	9
CAMPECHE	0	0	1	1
CHIAPAS	0	2	0	2
CHIHUAHUA	1	10	22	33
CIUDAD DE MÉXICO	2	12	45	59
COAHUILA DE ZARAGOZA	0	2	27	29
COLIMA	0	0	1	1
DURANGO	0	3	14	17
GUANAJUATO	0	12	26	38
GUERRERO	0	1	3	4

Figura 17 Tabla parcial generada por el procedimiento almacenado "defunciones por edad"

Esto último nos permite ver la cantidad de defunciones por estado, dividido por las edades. De nuevo, trabajamos con aproximadamente 120 mil datos, sin embargo, podemos ver que el número que más predomina sigue siendo el de adultos mayores.

Finalmente, dentro de nuestro proyecto, tenemos dos funciones sencillas.

```
CREATE FUNCTION casos_sexo (sexo_id int) RETURNS int
DETERMINISTIC
begin
    DECLARE sexo_count int;
    SELECT count(*) as 'Numero de casos' into sexo_count from registros_vista where sexo = sexo_id;
    RETURN sexo_count;
end//
```

Figura 18 Query que genera la función "número de casos dependiendo el sexo"

```
CREATE FUNCTION id_entidad(nombre_entidad varchar(255)) returns int
deterministic
begin
    DECLARE id_ent int;
    SELECT CLAVE_ENTIDAD into id_ent from registros_vista re, entidades en where en.ENTIDAD_FEDERATIVA = nombre_entidad
    GROUP BY CLAVE_ENTIDAD;
    RETURN id_ent ;
end//
```

Figura 19 Query que genera la función "ID de la entidad"

La primera función sirve para saber el número de casos, en total, dependiendo si era hombre o mujer. La segunda función permite saber el ID de la entidad (estado); siendo que normalmente no sabemos el ID que le pertenece a nuestro estado, esta función permite lo siguiente, uno mete el nombre de su entidad, y la función le devuelve el ID del mismo.

Recordemos que las funciones pueden ser utilizadas dentro de las queries ya que deben devolver un único valor. Imaginemos que queremos saber las personas que fallecieron en Sonora, pero no sabemos el ID de Sonora, además, no queremos hacer que nuestra *query* sea tan larga. Es aquí donde hacemos una *query* más corta, usando esta función en la sección correspondiente, y listo.

Conclusiones

El objetivo del trabajo, o proyecto, era poner en practica los conocimientos sobre manipulación de una base de datos con SQL, independientemente si era *mysql*, otros. Dado los resultados, las tablas, las *views*, los procedimientos, etc., obtenidos, podemos decir que el objetivo se cumplió. Esto se logró debido a que SQL es un lenguaje amigable y sencillo de aprender, hasta cierto punto, intuitivo. Recordemos que la base de datos construida, las *queries* y el modelo, se encuentran en el repositorio de GitHub.

Bibliografía

Javier Díaz-Castrillón, F., & Toro-Montoya, A. (2020). *Artículo de revisión SARS-CoV-2/COVID-19: el virus, la enfermedad y la pandemia SARS-CoV-2/COVID-19: The virus, the disease and the pandemic*. https://docs.bvsalud.org/biblioref/2020/05/1096519/covid-19.pdf

W3Schools. (2023). W3schools.com. https://profile.w3schools.com/log-in?redirect_url=https%3A%2F%2Fwww.w3schools.com%2Fmysql%2Fmysql_delete.asp

Secretaría de Salud. (2023). *Datos Abiertos Dirección General de Epidemiología*. Gob.mx. https://www.gob.mx/salud/documentos/datos-abiertos-152127