

# Data Warehouse para COVID-19

Este proyecto utiliza SQL Server para almacenar una gran cantidad de datos relacionados al COVID-19 alrededor del mundo, incluyendo casos confirmados, muertes, recuperados, activos, cantidad de vacunas aplicadas y sintomatologías presentadas. También recopila información específica de México, como el tipo de paciente, resultado de prueba y enfermedades relacionadas. Aquí se hablará sobre la arquitectura del Data Warehouse, su importancia y cómo se almacenan los datos.

La razón por la que decidí presentar un Data Warehouse como proyecto final, es porque engloba y muestra claramente como sería aplicar los conocimientos de base de datos a un proyecto de ciencia de datos a través de la inteligencia de negocios, así dará la oportunidad de observar con más claridad el potencial de una base de datos y la experiencia que se tiene al momento.

**Por Ing. Mario Estrada Ferreira**

**Curso de Base de datos - Docente: Dr. Juan Pablo Soto Barrera**

- ❑ GitHub del proyecto, donde encontrara los archivos del mismo.



"<https://github.com/mariooef/BaseDeDatos>"

# Importancia deBal Data Warehouse

El Data Warehouse es una herramienta crucial para almacenar y analizar grandes cantidades de datos relacionados a COVID-19. En lugar de depender de datos desorganizados y desarticulados, el Data Warehouse permite tener una visión más clara y completa de la pandemia en distintas partes del mundo. Además, permite llevar a cabo análisis detallados del virus y su propagación, particularmente en casos de futuros brotes y en la preparación para futuras pandemias.

## Análisis detallados

El Data Warehouse permite análisis detallados de la pandemia, lo que puede ayudar a identificar patrones en la propagación del virus y en su impacto en la sociedad.

## Mayor eficiencia

El almacenamiento organizado de datos ayuda a las organizaciones a tomar decisiones más rápidas y efectivas, lo que puede salvar vidas en caso de una pandemia.

## Monitorización en tiempo real

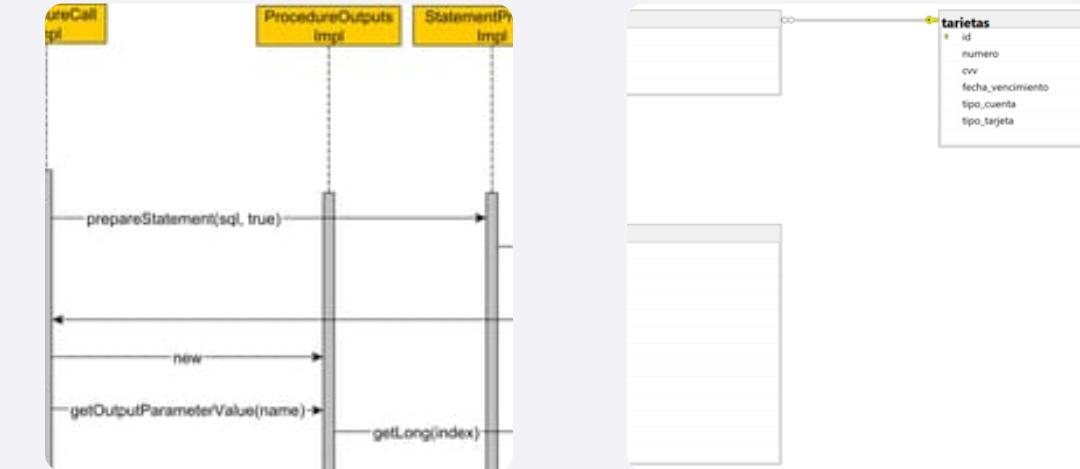
El Data Warehouse permite la monitorización en tiempo real de las tendencias y patrones en la propagación del virus, lo que puede ayudar a predecir futuros brotes y en la preparación para futuras pandemias.

# SQL Server en el Data Warehouse

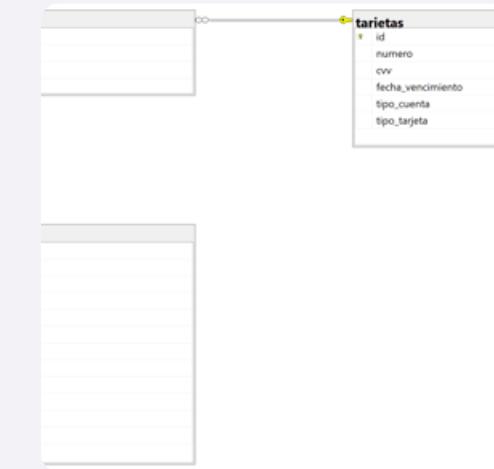
SQL Server es una herramienta esencial en el almacenamiento de datos relacionados al COVID-19. Permite la realización de funciones como Select, Insert, Update y Delete, así como la creación de stored procedures y vistas. Esto permite un manejo más eficiente y efectivo de los datos almacenados en el Data Warehouse, facilitando su análisis y utilización.



SQL Server es una herramienta poderosa para el manejo de grandes cantidades de datos.



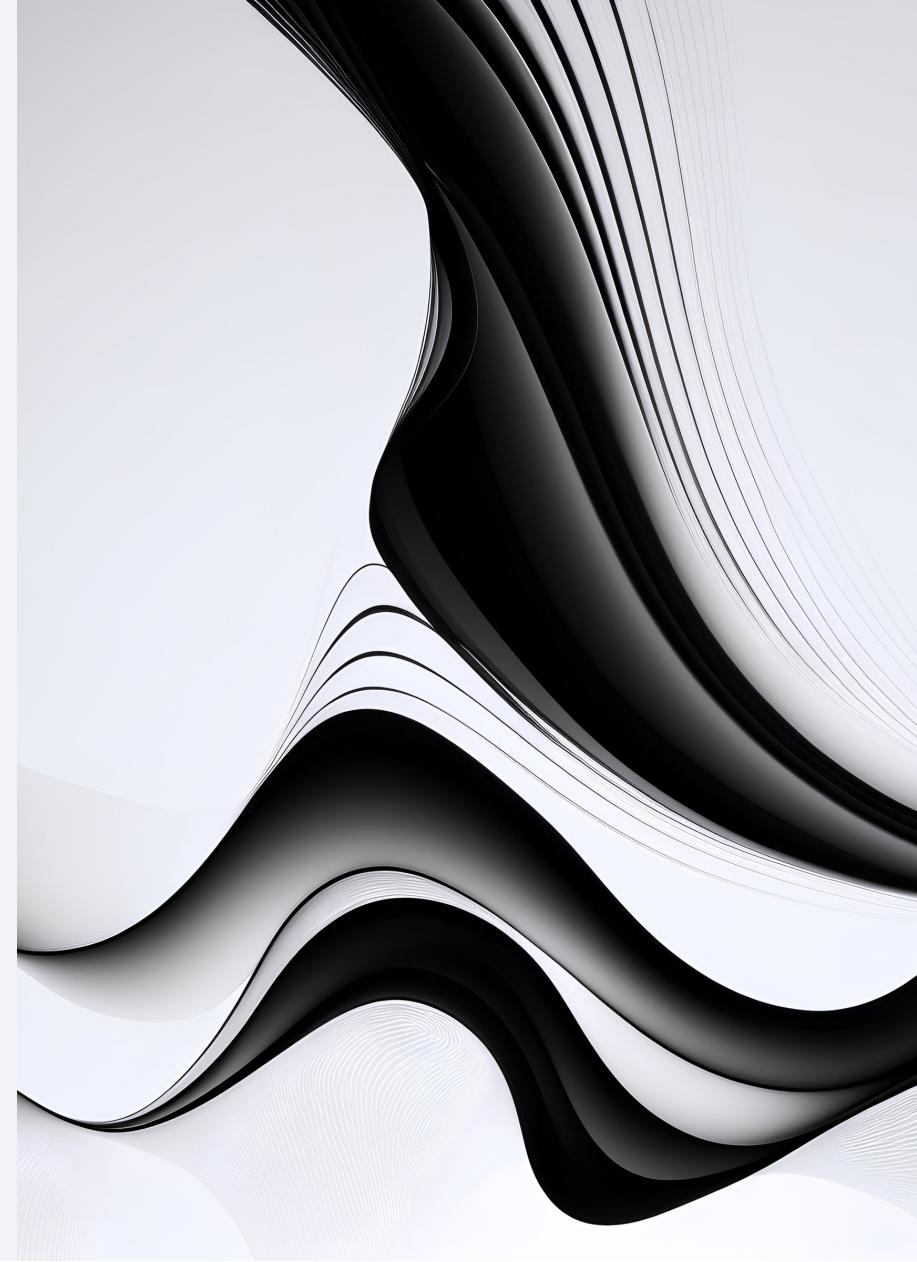
Las stored procedures son una manera eficiente de almacenar y procesar datos relacionados al COVID-19.



Las vistas son una forma útil de visualizar los datos almacenados en el Data Warehouse.

# Tablas de Dimensiones y de Hecho en Arquitecturas Estrella y Snowflake

Al diseñar un Data Warehouse, es importante entender la diferencia entre las tablas de dimensiones y las tablas de hecho. Las tablas de dimensiones contienen información descriptiva, mientras que las tablas de hecho contienen medidas cuantitativas. En este artículo, hablaremos sobre cómo estas tablas se utilizan en las arquitecturas de Data Warehousing Estrella y Snowflake.



# Datos almacenados del COVID-19 en todo el mundo

La pandemia del COVID-19 ha afectado al mundo entero de diversas maneras, y almacenar la información relacionada es crucial para entender el impacto del virus. En el Data Warehouse se almacenan datos como los casos confirmados, muertes, recuperados y activos, así como la cantidad de vacunas aplicadas. También se registra la variedad de sintomatologías presentadas alrededor del mundo, información valiosa para prevenir futuros brotes.

## Casos confirmados

La cantidad de casos confirmados es un indicador importante para la toma de decisiones y la implementación de medidas preventivas.

## Muertes

El número de fallecimientos es un reflejo directo de la gravedad de la pandemia y su impacto en la sociedad.

## Sintomatologías

La variedad de síntomas reportados en distintas partes del mundo ayuda a los médicos a identificar patrones en la presentación del COVID-19.

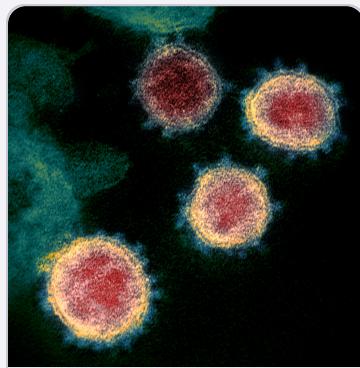
# Datos específicos de México

El Data Warehouse también almacena información relevante al COVID-19 específica de México. Esto incluye datos sobre el tipo de paciente, los resultados de pruebas y enfermedades relacionadas.

Tipo de paciente	Resultados de prueba	Enfermedades relacionadas
Paciente ambulatorio	Positivo	Diabetes
Paciente hospitalizado	Negativo	Hipertensión
Paciente con COVID-19	Positivo	Cáncer

# Dataframes

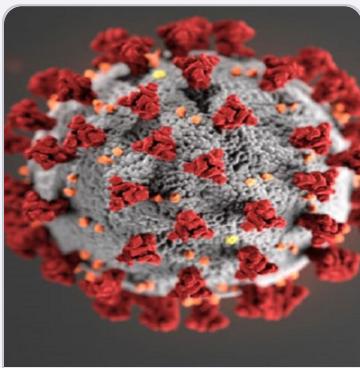
Para apoyar a mi Data Warehouse de COVID-19, he recopilado una lista de URLs de fuentes confiables de información. Algunas de estas fuentes incluyen la Organización Mundial de la Salud (OMS) y los Centros para el Control y la Prevención de Enfermedades (CDC) de los Estados Unidos. Con esta información, podemos asegurarnos de que estamos utilizando los datos más precisos y actualizados para el Data Warehouse tenga la suficiente información de análisis y toma de decisiones.



 [www.kaggle...](https://www.kaggle.com/datasets/imdevskp/corona-virus-report) ↗

**COVID-19...**

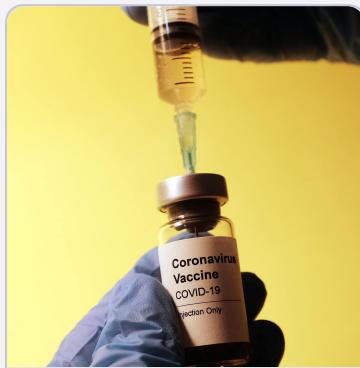
Number of  
Confirmed, Deat...



 [www.kaggle...](https://www.kaggle.com/datasets/iamhungundji/covid19-symptoms-checker) ↗

**COVID-19...**

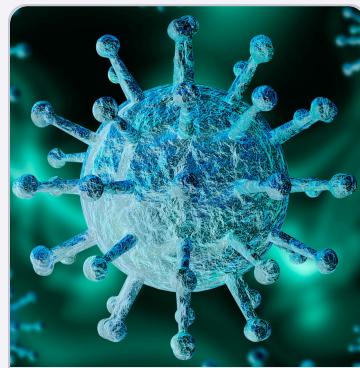
Predict whether  
someone has...



 [www.kaggle...](https://www.kaggle.com/datasets/gprena/covid-world-vaccination-progress) ↗

**COVID-19...**

Daily and Total  
Vaccination for...



 [www.kaggle...](https://www.kaggle.com/datasets/lalish99/covid19-mx) ↗

**COVID-19 MX**

Official information  
on COVID-19...

"  
[https://www.kaggle...](https://www.kaggle.com/datasets/imdevskp/corona-virus-report)  
.com/datasets/imdevskp/corona-virus-report"

"  
[https://www.kaggle...](https://www.kaggle.com/datasets/iamhungundji/covid19-symptoms-checker)  
.com/datasets/iamhungundji/covid19-symptoms-checker"

"[https://www.kaggle...](https://www.kaggle.com/datasets/gprena/covid-world-vaccination-progress)  
.com/datasets/gprena/covid-world-vaccination-progress"

"[https://www.kaggle...](https://www.kaggle.com/datasets/lalish99/covid19-mx)  
.com/datasets/lalish99/covid19-mx"

# Arquitectura del Data Warehouse

La arquitectura del Data Warehouse es fundamental para la eficiencia y efectividad del almacenamiento y análisis de los datos. En el caso del COVID-19, la arquitectura del Data Warehouse está diseñada para procesar grandes cantidades de datos en tiempo real, lo que permite la monitorización constante del virus en distintas partes del mundo.

"Para tener una arquitectura de Data Warehouse efectiva es necesario pensar en el diseño, la infraestructura, la seguridad, el modelado de datos y la implementación de ETL. En el caso del COVID-19, la arquitectura debe ser flexible y capaz de adaptarse a los cambios en la pandemia".

- John Doe, Experto en Data Warehouse

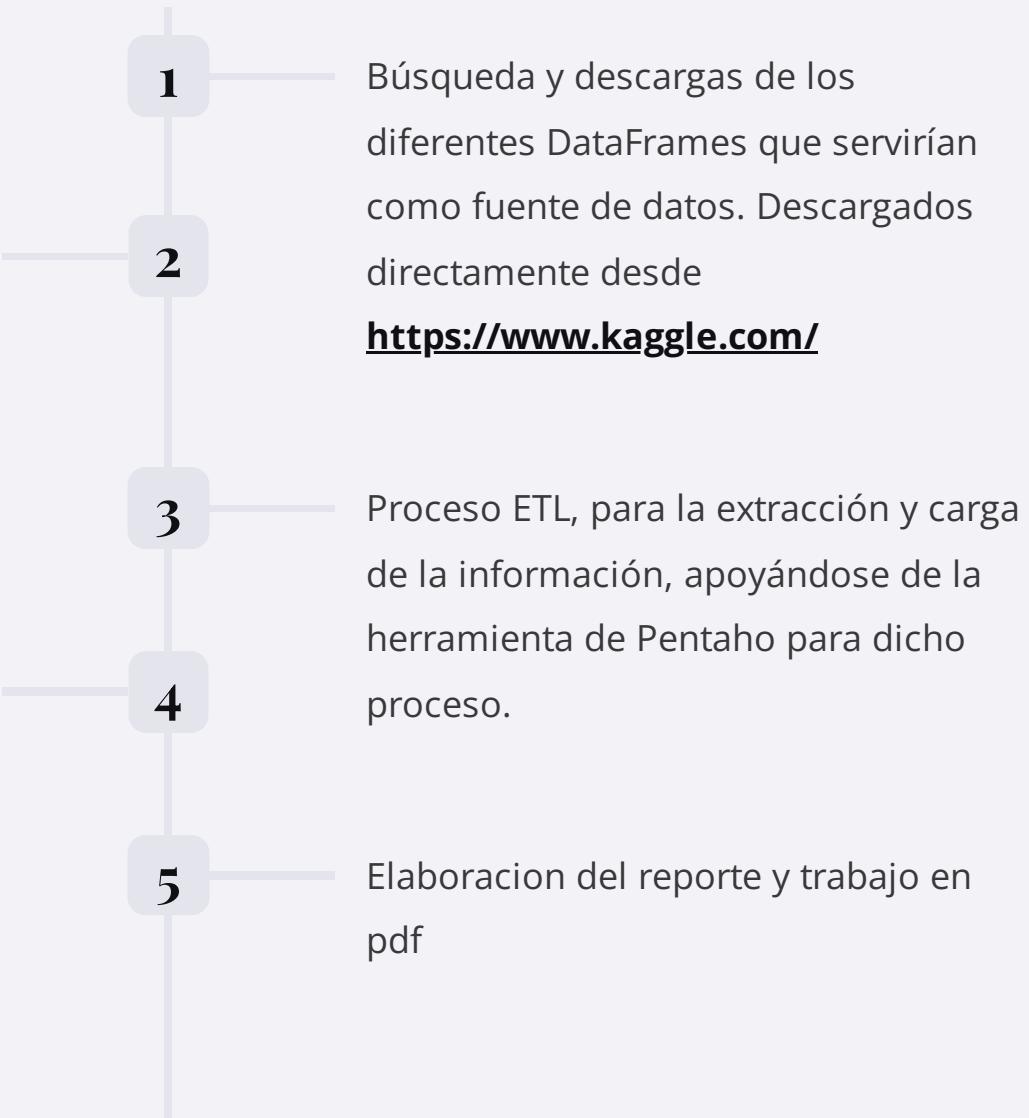
# Desarrollo del proyecto

El proyecto tuvo varias etapas hasta llegar a la construcción del Data Warehouse

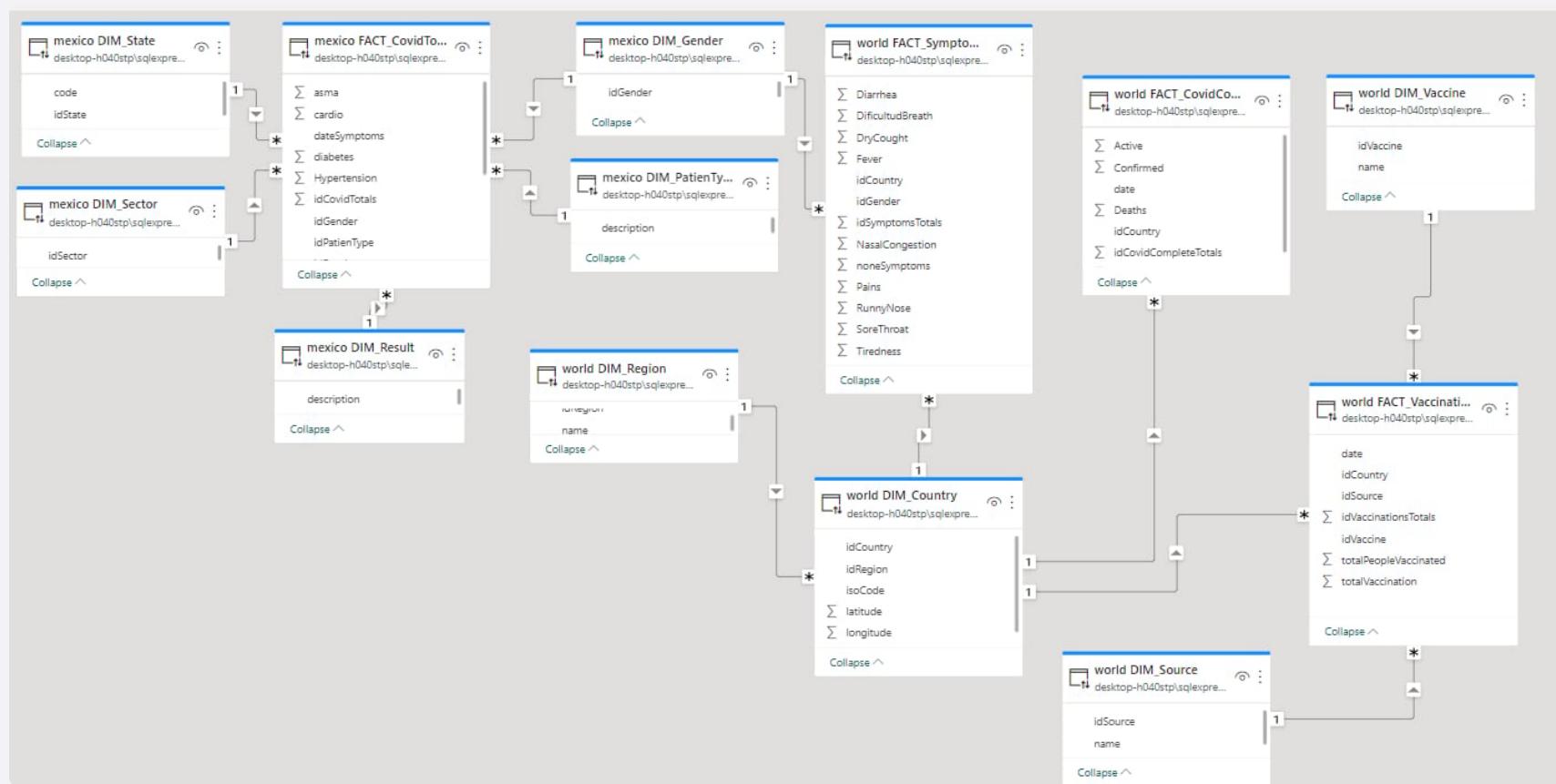
Interpretación, análisis y construcción de toda la arquitectura del Data Warehouse, la cual fue hecha en SQL

SERVER

Costrucción de los distintos querys que servirían para la creación de las diferentes vistas, funciones y stored procedures



# Diagrama del Data Warehouse



## Explicación

La arquitectura del Data Warehouse consta de un arquitectura de tipo SnowFlake esto para representar en el proyecto la jerarquización de alguna dimensión y no solo dejar joins directos entre las tablas. Aunque cabe aclarar que prefiero la arquitectura de tipo estrella cuando se trata de la creación de este tipo de proyectos.

La arquitectura consta de las siguientes tablas:

### 1 Fact Tables

- mexico.FACT\_CovidTotals
- world.FACT\_CovidCompleteTotals
- world.FACT\_SymptomsTotals
- world.FACT\_VaccinationsTotals

### 2 Dimension Tables

- mexico.DIM\_Gender
- mexico.DIM\_PatientType
- mexico.DIM\_Result
- mexico.DIM\_Sector
- mexico.DIM\_State
- world.DIM\_Country
- world.DIM\_Region
- world.DIM\_Source
- world.DIM\_Vaccine

Se observa que las tablas cuentan con un esquema, para este proyecto decide crear dos esquemas bases para identificar la relación o bien identificar la procedencia de la información en las tablas por lo que se creo los siguientes esquemas

1. **world** (tendrá las tablas, stored, vistas y funciones relacionadas con información del covid a nivel mundial)
2. **mexico** (tendrá las tablas, stored, vistas y funciones relacionadas con información del covid en México)

# Dimension Tables

## **mexico.DIM\_Gender**

Tabla que almacena el genero, se usa como características en **mexico.FACT\_CovidTotals** y **world.FACT\_SymptomsTotals**

- idGender (int)
- name (varchar(20))

```
CREATE TABLE [mexico].[DIM_Gender](
    [idGender] [int] IDENTITY(1,1) NOT NULL,
    [name] [varchar](20) NOT NULL,
    CONSTRAINT [PK_DIM_Gender] PRIMARY KEY CLUSTERED
    (
        [idGender] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
    ON [PRIMARY]
) ON [PRIMARY]
```

# Dimension Tables

## **mexico.DIM\_PatientType**

Tabla que almacena los tipos de paciente, se usa como características en **mexico.FACT\_CovidTotals**

- idPatientType (int)
- description (varchar(100))

```
CREATE TABLE [mexico].[DIM_PatientType](
    [idPatientType] [int] IDENTITY(1,1) NOT NULL,
    [description] [varchar](100) NOT NULL,
    CONSTRAINT [PK_DIM_PatientType] PRIMARY KEY CLUSTERED
    (
        [idPatientType] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
    ON [PRIMARY]
) ON [PRIMARY]
```

# Dimension Tables

## **mexico.DIM\_Result**

Tabla que almacena los resultados de las pruebas covid, se usa como características en **mexico.FACT\_CovidTotals**

- idResult (int)
- description (varchar(200))

```
CREATE TABLE [mexico].[DIM_Result](
    [idResult] [int] IDENTITY(1,1) NOT NULL,
    [description] [varchar](200) NOT NULL,
    CONSTRAINT [PK_DIM_Result] PRIMARY KEY CLUSTERED
    (
        [idResult] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
    ON [PRIMARY]
) ON [PRIMARY]
```

# Dimension Tables

## **mexico.DIM\_Sector**

Tabla que almacena los sectores donde se ha registrado casos de las pruebas covid, se usa como características en **mexico.FACT\_CovidTotals**

- idResult (int)
- description (varchar(200))

```
CREATE TABLE [mexico].[DIM_Result](
    [idResult] [int] IDENTITY(1,1) NOT NULL,
    [description] [varchar](200) NOT NULL,
    CONSTRAINT [PK_DIM_Result] PRIMARY KEY CLUSTERED
    (
        [idResult] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
    ON [PRIMARY]
) ON [PRIMARY]
```

# Dimension Tables

## **mexico.DIM\_State**

Tabla que almacena las entidades federativas de México, se usa como características en

### **mexico.FACT\_CovidTotals**

- idState (int)
- code (varchar(20))
- name (VARCHAR(150))

```
CREATE TABLE [mexico].[DIM_State](
    [idState] [int] IDENTITY(1,1) NOT NULL,
    [code] [varchar](20) NOT NULL,
    [name] [varchar](150) NOT NULL,
    CONSTRAINT [PK_DIM_State] PRIMARY KEY CLUSTERED
    (
        [idState] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
    ON [PRIMARY]
) ON [PRIMARY]
```

# Dimension Tables

## world.DIM\_Country

- ⓘ Esta tabla es la única dimensión que cuenta con otro índice a parte de su índice por default de su llave foránea, esto porque siempre es bueno poner índices en los campos que tiene como foreing key.

Tabla que almacena los países de todo el mundo los cuales tiene relación con los casos de COVID-19, se usa como características en **world.FACT\_CovidCompleteTotals**, **world.FACT\_SymptomsTotals**, **world.FACT\_VaccinationsTotals**

- idCountry (int)
- idRegion (int)
- isoCode (varchar(20))
- name (VARCHAR(100))

```
CREATE TABLE [world].[DIM_Country](
    [idCountry] [int] IDENTITY(1,1) NOT NULL,
    [idRegion] [int] NOT NULL,
    [isoCode] [varchar](20) NULL,
    [name] [varchar](100) NOT NULL,
    [latitude] [decimal](10, 8) NULL,
    [longitude] [decimal](10, 8) NULL,
    CONSTRAINT [PK_DIM_Country] PRIMARY KEY CLUSTERED
    (
        [idCountry] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
    ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
ALTER TABLE [world].[DIM_Country] WITH CHECK ADD CONSTRAINT
[FK_DIM_Country_DIM_Region] FOREIGN KEY([idRegion])
REFERENCES [world].[DIM_Region] ([idRegion])
GO
```

```
ALTER TABLE [world].[DIM_Country] CHECK CONSTRAINT [FK_DIM_Country_DIM_Region]
```

# Dimension Tables

## world.DIM\_Region

Tabla que almacena que almacena las regiones que tienen relación con los países del mundo los cuales tiene relación con los casos de COVID-19, se usa como llave foránea en **world.DIM\_Country**.

- idRegion (int)
- name (VARCHAR(100))

```
CREATE TABLE [world].[DIM_Region](
    [idRegion] [int] IDENTITY(1,1) NOT NULL,
    [name] [varchar](100) NOT NULL,
    CONSTRAINT [PK_DIM_Region] PRIMARY KEY CLUSTERED
    (
        [idRegion] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
    ON [PRIMARY]
) ON [PRIMARY]
```

# Dimension Tables

## **world.DIM\_Source**

Tabla que almacena que almacena las fuentes de donde proviene información de la vacuna que , se usa como caracteristica en **world.FACT\_VaccinationsTotals**.

- idSource (int)
- name (VARCHAR(150))
- webSite (VARCHAR(150))

```
CREATE TABLE [world].[DIM_Source](
    [idSource] [int] IDENTITY(1,1) NOT NULL,
    [name] [varchar](150) NOT NULL,
    [webSite] [varchar](150) NULL,
    CONSTRAINT [PK_DIM_Source] PRIMARY KEY CLUSTERED
    (
        [idSource] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
    ON [PRIMARY]
) ON [PRIMARY]
```

# Dimension Tables

## **world.DIM\_Vaccine**

Tabla que almacena que almacena la información referente a las vacunas que se ha puesto en algún país, se usa como característica en **world.FACT\_VaccinationsTotals**.

- idVaccine (int)
- name (VARCHAR(150))

```
CREATE TABLE [world].[DIM_Vaccine](
    [idVaccine] [int] IDENTITY(1,1) NOT NULL,
    [name] [varchar](150) NOT NULL,
    CONSTRAINT [PK_DIM_Vaccine] PRIMARY KEY CLUSTERED
    (
        [idVaccine] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
    ON [PRIMARY]
) ON [PRIMARY]
```

# Fact Tables

- 💡 Todas las tablas de hechos presentadas a continuación, se les genero índices por cada llave foránea o relación que maneje con sus tablas dimensión, esto es buena practica para el performance de las consultas.

## mexico.FACT\_CovidTotals

Tabla que almacena que almacena los hechos de los totales relacionados con el covid en México.

### 1 Campos

- idCovidTotals
- idState
- dSector
- idGender
- idPatientType
- idResult
- dateSymptoms
- intubated
- pneumonia

### 2 Campos

- diabetes
- asma
- inmusupr
- Hypertension
- obecity
- cardio
- tabaquismo

```
CREATE TABLE [mexico].[FACT_CovidTotals](
    [idCovidTotals] [int] IDENTITY(1,1) NOT NULL,
    [idState] [int] NOT NULL,
    [dSector] [int] NOT NULL,
    [idGender] [int] NOT NULL,
    [idPatientType] [int] NOT NULL,
    [idResult] [int] NOT NULL,
    [dateSymptoms] [date] NOT NULL,
    [intubated] [int] NOT NULL,
    [pneumonia] [int] NOT NULL,
    [diabetes] [int] NOT NULL,
    [asma] [int] NOT NULL,
    [inmusupr] [int] NOT NULL,
    [Hypertension] [int] NOT NULL,
    [obecity] [int] NOT NULL,
    [cardio] [int] NOT NULL,
    [tabaquismo] [int] NOT NULL,
    CONSTRAINT [PK_FACT_CovidTotals] PRIMARY KEY CLUSTERED
    (
        [idCovidTotals] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
    ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
ALTER TABLE [mexico].[FACT_CovidTotals] WITH CHECK ADD CONSTRAINT
[FK_FACT_CovidTotals_DIM_Gender] FOREIGN KEY([idGender])
REFERENCES [mexico].[DIM_Gender] ([idGender])
GO
```

```
ALTER TABLE [mexico].[FACT_CovidTotals] CHECK CONSTRAINT
[FK_FACT_CovidTotals_DIM_Gender]
GO
```

```
ALTER TABLE [mexico].[FACT_CovidTotals] WITH CHECK ADD CONSTRAINT
[FK_FACT_CovidTotals_DIM_PatientType] FOREIGN KEY([idPatientType])
REFERENCES [mexico].[DIM_PatientType] ([idPatientType])
GO
```

```
ALTER TABLE [mexico].[FACT_CovidTotals] CHECK CONSTRAINT
[FK_FACT_CovidTotals_DIM_PatientType]
GO
```

```
ALTER TABLE [mexico].[FACT_CovidTotals] WITH CHECK ADD CONSTRAINT
[FK_FACT_CovidTotals_DIM_Result] FOREIGN KEY([idResult])
REFERENCES [mexico].[DIM_Result] ([idResult])
GO
```

```
ALTER TABLE [mexico].[FACT_CovidTotals] CHECK CONSTRAINT
[FK_FACT_CovidTotals_DIM_Result]
GO
```

```
ALTER TABLE [mexico].[FACT_CovidTotals] WITH CHECK ADD CONSTRAINT
[FK_FACT_CovidTotals_DIM_Sector] FOREIGN KEY([idSector])
REFERENCES [mexico].[DIM_Sector] ([idSector])
GO
```

# Fact Tables

## world.FACT\_CovidCompleteTotals

Tabla que almacena los hechos de los totales de indicadores a nivel mundial referente al COVID-19.

### 1 Campos

- idCovidCompleteTotals (int)
- idCountry (int)
- date (date)
- Confirmed (int)
- Deaths (int)
- Recovered (int)
- Active (int)

```
CREATE TABLE [world].[FACT_CovidCompleteTotals](
    [idCovidCompleteTotals] [int] IDENTITY(1,1) NOT NULL,
    [idCountry] [int] NOT NULL,
    [date] [date] NOT NULL,
    [Confirmed] [int] NOT NULL,
    [Deaths] [int] NOT NULL,
    [Recovered] [int] NOT NULL,
    [Active] [int] NOT NULL,
    CONSTRAINT [PK_FACT_CovidCompleteTotals] PRIMARY KEY CLUSTERED
    (
        [idCovidCompleteTotals] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
    ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
ALTER TABLE [world].[FACT_CovidCompleteTotals] WITH CHECK ADD CONSTRAINT
[FK_FACT_CovidCompleteTotals_DIM_Country] FOREIGN KEY([idCountry])
REFERENCES [world].[DIM_Country] ([idCountry])
GO
```

```
ALTER TABLE [world].[FACT_CovidCompleteTotals] CHECK CONSTRAINT
[FK_FACT_CovidCompleteTotals_DIM_Country]
```

# Fact Tables

## world.FACT\_VaccinationsTotals

Tabla que almacena que almacena los hechos de los totales de las vacunas recibidas y aplicadas a nivel mundial referente al COVID-19.

### 1 Campos

- idVaccinationsTotals (int)
- idCountry (int)
- idVaccine (int)
- idSource (int)
- date (date)
- totalVaccination (bigint)
- totalPeopleVaccinated (bigint)

ⓘ Aquí decidí que los campos de los totales fueran de tipo bigint debido a la información de los datasets estaban dando problemas al momento de hacer el pase.

```
CREATE TABLE [world].[FACT_VaccinationsTotals] (
    [idVaccinationsTotals] [int] IDENTITY(1,1) NOT NULL,
    [idCountry] [int] NOT NULL,
    [idVaccine] [int] NOT NULL,
    [idSource] [int] NOT NULL,
    [date] [date] NOT NULL,
    [totalVaccination] [bigint] NOT NULL,
    [totalPeopleVaccinated] [bigint] NOT NULL,
    CONSTRAINT [PK_FACT_VaccinationsTotals] PRIMARY KEY CLUSTERED
    (
        [idVaccinationsTotals] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
    ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
ALTER TABLE [world].[FACT_VaccinationsTotals] WITH CHECK ADD CONSTRAINT
[FK_FACT_VaccinationsTotals_DIM_Country] FOREIGN KEY([idCountry])
REFERENCES [world].[DIM_Country] ([idCountry])
GO
```

```
ALTER TABLE [world].[FACT_VaccinationsTotals] CHECK CONSTRAINT
[FK_FACT_VaccinationsTotals_DIM_Country]
GO
```

```
ALTER TABLE [world].[FACT_VaccinationsTotals] WITH CHECK ADD CONSTRAINT
[FK_FACT_VaccinationsTotals_DIM_Source] FOREIGN KEY([idSource])
REFERENCES [world].[DIM_Source] ([idSource])
GO
```

```
ALTER TABLE [world].[FACT_VaccinationsTotals] CHECK CONSTRAINT
[FK_FACT_VaccinationsTotals_DIM_Source]
GO
```

```
ALTER TABLE [world].[FACT_VaccinationsTotals] WITH CHECK ADD CONSTRAINT
[FK_FACT_VaccinationsTotals_DIM_Vaccine] FOREIGN KEY([idVaccine])
REFERENCES [world].[DIM_Vaccine] ([idVaccine])
GO
```

```
ALTER TABLE [world].[FACT_VaccinationsTotals] CHECK CONSTRAINT
[FK_FACT_VaccinationsTotals_DIM_Vaccine]
```

# Fact Tables

## world.FACT\_SymptomsTotals

Tabla que almacena que almacena los hechos de los totales de indicadores a nivel mundial referente al COVID-19.

### 1 Campos

- idSymptomsTotals
- idCountry
- idGender
- Fever
- Tiredness
- DryCough
- DificultudBreath

### 2 Campos

- SoreThroat
- Pains
- NasalCongestion
- RunnyNose
- Diarrhea
- noneSymptoms

```
CREATE TABLE [world].[FACT_SymptomsTotals](
[idSymptomsTotals] [int] IDENTITY(1,1) NOT NULL,
[idCountry] [int] NOT NULL,
[idGender] [int] NOT NULL,
[Fever] [int] NOT NULL,
[Tiredness] [int] NOT NULL,
[DryCough] [int] NOT NULL,
[DificultudBreath] [int] NOT NULL,
[SoreThroat] [int] NOT NULL,
[Pains] [int] NOT NULL,
[NasalCongestion] [int] NOT NULL,
[RunnyNose] [int] NOT NULL,
[Diarrhea] [int] NOT NULL,
[noneSymptoms] [int] NOT NULL,
CONSTRAINT [PK_FACT_SymptomsTotals] PRIMARY KEY CLUSTERED
(
    [idSymptomsTotals] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
ALTER TABLE [world].[FACT_SymptomsTotals] WITH CHECK ADD CONSTRAINT
[FK_FACT_SymptomsTotals_DIM_Country] FOREIGN KEY([idCountry])
REFERENCES [world].[DIM_Country] ([idCountry])
GO
```

```
ALTER TABLE [world].[FACT_SymptomsTotals] CHECK CONSTRAINT
[FK_FACT_SymptomsTotals_DIM_Country]
GO
```

```
ALTER TABLE [world].[FACT_SymptomsTotals] WITH CHECK ADD CONSTRAINT
[FK_FACT_SymptomsTotals_DIM_Gender] FOREIGN KEY([idGender])
REFERENCES [mexico].[DIM_Gender] ([idGender])
GO
```

```
ALTER TABLE [world].[FACT_SymptomsTotals] CHECK CONSTRAINT
[FK_FACT_SymptomsTotals_DIM_Gender]
```

# Vistas

Las vistas son importantes elementos dentro de un Data Warehouse, esto facilita a los programas de análisis de datos y generador de reportes, así la información se presenta de una manera mas eficiente y directa.

- ⓘ Se crearon 4 vistas para el proyecto, una para cada Fact Table, las cuales comienzan con **VW** en su nomenclatura, esto lo hago para poder tener una identificación de la vista rápida y es parte del buen diseño de la base de datos.

## mexico.VW\_Covid\_Totals\_Mexico\_Data

La query de esta vista se hace a partir de la fact table **mexico.FACT\_CovidTotals**

```
SELECT
    CONVERT(VARCHAR, ctm.dateSymptoms, 6) AS 'Fecha'
    ,st.[name] AS 'EntidadFederativa'
    ,se.[name] AS 'Sector'
    ,ge.[name] AS 'Genero'
    ,pt.[description] AS 'TipoPaciente'
    ,ctm.intubated AS 'Entubados'
    ,ctm.pneumonia AS 'Neumonia'
    ,ctm.diabetes AS 'Diabetes'
    ,ctm.asma AS 'Asma'
    ,ctm.inmusupr 'Inmuno'
    ,ctm.Hypertension 'Impertencion'
    ,ctm.obecity 'Obesidad'
    ,ctm.cardio 'Cardio'
    ,ctm.tabaquismo 'Tabaquismo'
    ,r.[description] AS 'Resultado'
FROM
    mexico.FACT_CovidTotals ctm
    INNER JOIN
        mexico.DIM_State st ON st.idState = ctm.idState
    INNER JOIN
        mexico.DIM_Sector se ON se.idSector = ctm.idSector
    INNER JOIN
        mexico.DIM_Gender ge ON ge.idGender = ctm.idGender
    INNER JOIN
        mexico.DIM_PatientType pt ON pt.idPatientType = ctm.idPatientType
    INNER JOIN
        mexico.DIM_Result r ON r.idResult = ctm.idResult
```

- ⓘ Se observa que se ponen alias en las tablas, esto para mi me resulta mas fácil poner un alias corto para poder realizar las consultas de una manera mas eficiente.

Y los nombres de las consultas me gusta ponerlo en comillas simples para una mejor visualización e interpretación del código.

- ⓘ `SELECT * FROM mexico.VW_Covid_Totals_Mexico_Data`

	Results	Messages													
	Fecha	EntidadFederativa	Sector	Genero	TipoPaciente	Entubados	Neumonia	Diabetes	Asma	Inmuno	Impertencion	Obesidad	Cardio	Tabaquismo	Resultado
1	10 Apr 20	CHIHUAHUA	IMSS	MUJER	HOSPITALIZADO	2	1	2	2	2	1	1	2	2	Positivo SARS-CoV-2
2	17 Apr 20	VERACRUZ DE IGNACIO DE LA LLAVE	IMSS	MUJER	AMBULATORIO	97	2	2	2	2	2	2	2	2	Positivo SARS-CoV-2
3	01 Jun 20	MEXICO	ESTATAL	HOMBRE	HOSPITALIZADO	2	1	2	2	2	2	2	2	2	Positivo SARS-CoV-2
4	05 Jun 20	MEXICO	ESTATAL	MUJER	AMBULATORIO	97	2	2	2	2	2	2	2	2	Positivo SARS-CoV-2
5	08 Jun 20	MEXICO	ESTATAL	HOMBRE	AMBULATORIO	97	1	2	2	2	2	2	2	2	Positivo SARS-CoV-2
6	25 Mar 20	OAXACA	IMSS	MUJER	AMBULATORIO	97	2	2	2	2	2	2	2	2	Positivo SARS-CoV-2
7	20 Mar 20	JALISCO	IMSS	HOMBRE	HOSPITALIZADO	2	1	2	2	1	1	2	2	2	Positivo SARS-CoV-2
8	13 Apr 20	YUCATAN	IMSS	HOMBRE	AMBULATORIO	97	2	2	2	2	2	2	2	2	Positivo SARS-CoV-2

# Vistas

**world.VW\_Covid\_Symptoms\_Totals\_Mundo\_Data**

La query de esta vista se hace a partir de la fact table **world.FACT\_SymptomsTotals** con relación de sus dimensiones .

```
SELECT
cpr.Region
,cpr.Pais
,(
    SELECT
        gd.[name]
    FROM
        mexico.DIM_Gender AS gd
    WHERE
        gd.idGender = st.idGender
) AS 'Genero'
,st.Fever AS 'Fiebre'
,st.Tiredness AS 'Fatiga'
,st.DryCough AS 'TosSeca'
,st.DifficultyBreath AS 'DificultadRespiratoria'
,st.SoreThroat AS 'DolorGarganta'
,st.Pains AS 'Dolores'
,st.NasalCongestion AS 'CongestionNasal'
,st.RunnyNose AS 'GoteoNasal'
,st.Diarrhea AS 'Diarrea'
,st.noneSymptoms AS 'SinSintomas'
FROM
world.FACT_SymptomsTotals st
CROSS APPLY
(
    SELECT
        c.[name] AS 'Pais'
        ,re.[name] AS 'Region'
    FROM
        world.DIM_Country AS c
    INNER JOIN
        world.DIM_Region AS re ON re.idRegion = c.idCountry
    WHERE
        c.idCountry = st.idCountry
) cpr
```

-  Aquí se observa que se esta usando una subquery en la consulta para obtener el genero, esto es mas practico y ayuda al perfomance de la consulta.

- SELECT \* FROM world.VW\_Covid\_Symptoms\_Totals\_Mundo\_Data

# Vistas

## world.VW\_Covid\_Totals\_Mundo\_Data

La query de esta vista se hace a partir de la fact table **world.FACT\_CovidCompleteTotals** con relación de sus dimensiones .

```
SELECT
    CONVERT(VARCHAR,cct.date, 11) AS 'Fecha'
    ,cpr.Region
    ,cpr.Pais
    ,cct.Confirmed AS 'Confirmados'
    ,cct.Deaths AS 'Muertes'
    ,cct.Recovered AS 'Recuperados'
    ,cct.Active AS 'Activos'
FROM
    world.FACT_CovidCompleteTotals AS cct
CROSS APPLY
(
    SELECT
        c.[name] AS 'Pais'
        ,re.[name] AS 'Region'
    FROM
        world.DIM_Country AS c
    INNER JOIN
        world.DIM_Region AS re ON re.idRegion = c.idRegion
    WHERE
        c.idCountry=cct.idCountry
) cpr
```

- ⓘ Aquí se observa que se esta usando un CROSS APPLY, en lo particular me gusta usar este tipo de funciones para cuando tengo que extraer mas de un dato que esta en tablas distintas y están relacionados con la tabla principal.

- ✓ `SELECT * FROM world.VW_Covid_Totals_Mundo_Data`

	Fecha	Region	Pais	Confirmados	Muertes	Recuperados	Activos
1	20/01/22	Eastern Mediterranean	Afghanistan	0	0	0	0
2	20/01/22	Europe	Albania	0	0	0	0
3	20/01/22	Africa	Algeria	0	0	0	0
4	20/01/22	Europe	Andorra	0	0	0	0
5	20/01/22	Africa	Angola	0	0	0	0
6	20/01/22	Americas	Antigua and Barbuda	0	0	0	0
7	20/01/22	Americas	Argentina	0	0	0	0
8	20/01/22	Europe	Armenia	0	0	0	0
9	20/01/22	Western Pacific	Australia	0	0	0	0
10	20/01/22	Western Pacific	Australia	0	0	0	0

# Vistas

## world.VW\_Covid\_Vaccines\_Totals\_Mundo\_Data

La query de esta vista se hace a partir de la fact table **world.FACT\_VaccinationsTotals** con relación de sus dimensiones .

```
SELECT
    CONVERT(VARCHAR, vt.[date], 6) AS 'Fecha'
    ,cpr.Region
    ,cpr.Pais
    ,va.[name] AS 'Vacunas'
    ,so.[name] AS 'Fuente'
    ,vt.totalVaccination AS 'TotalVacunas'
    ,vt.totalPeopleVaccinated AS 'TotalVacunados'
FROM
    world.FACT_VaccinationsTotals vt WITH(NOLOCK)
    INNER JOIN
        world.DIM_Vaccine va WITH(NOLOCK) ON va.idVaccine = vt.idVaccine
    INNER JOIN
        world.DIM_Source so WITH(NOLOCK) ON so.idSource = vt.idSource
    CROSS APPLY
    (
        SELECT
            c.[name] AS 'Pais'
            ,re.[name] AS 'Region'
        FROM
            world.DIM_Country AS c WITH(NOLOCK)
        INNER JOIN
            world.DIM_Region AS re WITH(NOLOCK) ON re.idRegion = c.idRegion
        WHERE
            c.idCountry = vt.idCountry
    ) cpr
```

- ⓘ Se observa que aquí se uso el WITH(NOLOCK) el cual es lo recomendable para cuando se realizan consultas y esas involucran joins, pues así se asegura no bloquear las tablas cuando se esta realizando una consulta.

- ⓘ SELECT \* FROM world.VW\_Covid\_Vaccines\_Totals\_Mundo\_Data

	Fecha	Region	Pais	Vacunas	Fuente	TotalVacunas	TotalVacunados
1	22 Feb 21	Eastern Mediterranean	Afghanistan	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioN...	World Health Organization	0	0
2	23 Feb 21	Eastern Mediterranean	Afghanistan	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioN...	World Health Organization	0	0
3	24 Feb 21	Eastern Mediterranean	Afghanistan	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioN...	World Health Organization	0	0
4	25 Feb 21	Eastern Mediterranean	Afghanistan	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioN...	World Health Organization	0	0
5	26 Feb 21	Eastern Mediterranean	Afghanistan	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioN...	World Health Organization	0	0
6	27 Feb 21	Eastern Mediterranean	Afghanistan	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioN...	World Health Organization	0	0
7	28 Feb 21	Eastern Mediterranean	Afghanistan	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioN...	World Health Organization	8200	8200
8	01 Mar 21	Eastern Mediterranean	Afghanistan	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioN...	World Health Organization	0	0
9	02 Mar 21	Eastern Mediterranean	Afghanistan	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioN...	World Health Organization	0	0
10	03 Mar 21	Eastern Mediterranean	Afghanistan	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioN...	World Health Organization	0	0

# Funciones

Las funciones son de gran apoyo sobre todo para cuando se esta manejando stored procedure, esto ayuda a seccionar parte del código y de las tareas y las vuelve reutilizables.

- ⓘ Se creo una única función solo para representar esta parte dentro del Data Warehouse

## ***world.fn\_CaculaPorcentajeEfectividadUsoVacunas***

Esta función regresa un calculo del porcentaje de efectividad en el uso adecuado de vacunas.

- ⓘ Porcentaje = totalPeopleVaccinated / totalVaccination

```
c FUNCTION [world].[fn_CaculaPorcentajeEfectividadUsoVacunas]
(
    @TotalVacunas AS BIGINT,
    @TotalVacunados AS BIGINT
)
RETURNS DECIMAL(20,2)

AS

BEGIN

    DECLARE @Porcentaje DECIMAL(20,2) = 0

    SET @Porcentaje = CASE WHEN @TotalVacunas = 0 THEN 0 ELSE (CAST(@TotalVacunados AS
decimal(20,2))/CAST(@TotalVacunas AS decimal(20,2)))*100 END

    RETURN @Porcentaje

END
```

- ⓘ SELECT world.fn\_CaculaPorcentajeEfectividadUsoVacunas(1203087,672765)

Results	
	(No column name)
1	55.92

# Stored Procedure

Los stored procedure son de gran importancia para poder apoyarse cuando se requiere realizar consultas muy complejas o poderosas. Cuando se esta generando un proceso transaccional los stored procedure son indispensables , pues así puedes codificar la lógica de dicho proceso de inserción, actualización y eliminación.

- ⓘ Se crearon tres tipo de stored, uno de consulta, otro de inserción y actualización y el otro de eliminacion.

## ***mexico.stp\_DatosInternosContraMundialesMexicoPorEntidadConsulta***

Este stored procedure realiza una conexión entre la vista principal de los datos del covid en Mexico con la información referente a la vacuna.

```
ALTER PROCEDURE [mexico].[stp_DatosInternosContraMundialesMexicoPorEntidadConsulta]
    @EntidadFederativa VARCHAR(150) = NULL
AS
BEGIN
    SET NOCOUNT ON;

    SELECT
        re.[name] AS 'RegionMundo'
        ,vmt.Fecha AS 'Fecha'
        ,vmt.EntidadFederativa
        ,vmt.Sector
        ,vmt.Entubados
        ,vmt.Neumonia
        ,vmt.Asma
        ,vmt.Inmuno
        ,vmt.Impertencion
        ,vmt.Obesidad
        ,vmt.Cardio
        ,vmt.Tabaquismo
        ,vt.totalVaccination
        ,vt.totalPeopleVaccinated
        ,(
            SELECT
                world.fn_CaculaPorcentajeEfectividadUsoVacunas(totalVaccination,totalPeopleVaccinated)
            ) AS 'PorcentajeEfectividadUsoVacuna'
        FROM mexico.VW_Covid_Totals_Mexico_Data AS vmt
        INNER JOIN
            world.DIM_Country AS ct WITH(NOLOCK) ON ct.[name] = 'MEXICO'
        INNER JOIN
            world.FACT_VaccinationsTotals as vt WITH(NOLOCK) ON vt.idCountry = ct.idCountry
        INNER JOIN
            world.DIM_Region AS re WITH(NOLOCK) ON re.idRegion = ct.idRegion
        WHERE vmt.EntidadFederativa LIKE CASE WHEN @EntidadFederativa IS NULL THEN
            vmt.EntidadFederativa ELSE @EntidadFederativa END
        END
```

### ⓘ Demostración de funcionamiento

```
EXEC mexico.stp_DatosInternosContraMundialesMexicoPorEntidadConsulta 'sonora'
```

	Results	Messages													
1	RegionMundo	Fecha	EntidadFederativa	Sector	Entubados	Neumonia	Asma	Inmuno	Impertencion	Obesidad	Cardio	Tabaquismo	totalVaccination	totalPeopleVaccinated	PorcentajeEfectividadUsoVacuna
2	Americas	26 Mar 20	SONORA	IMSS	97	2	2	2	1	2	2	2	6824	6824	100.00
3	Americas	26 Mar 20	SONORA	IMSS	97	2	2	2	1	2	2	2	9579	9579	100.00
4	Americas	26 Mar 20	SONORA	IMSS	97	2	2	2	1	2	2	2	18529	18529	100.00
5	Americas	26 Mar 20	SONORA	IMSS	97	2	2	2	1	2	2	2	24998	24998	100.00
6	Americas	26 Mar 20	SONORA	IMSS	97	2	2	2	1	2	2	2	0	0	0.00
7	Americas	26 Mar 20	SONORA	IMSS	97	2	2	2	1	2	2	2	0	0	0.00
8	Americas	26 Mar 20	SONORA	IMSS	97	2	2	2	1	2	2	2	190441475	85575818	44.94
													191907868	85580293	44.59

# Stored Procedure

## *mexico.stp\_GeneroInserta*

Este stored procedure realiza el proceso de insertar un registro nuevo, para este ejemplo inserta o bien actualiza un registro referente al genero.

```
ALTER PROCEDURE [mexico].[stp_GeneroInserta]
    @idGender INT = NULL
    , @name VARCHAR(150)
AS
BEGIN
    SET NOCOUNT ON;

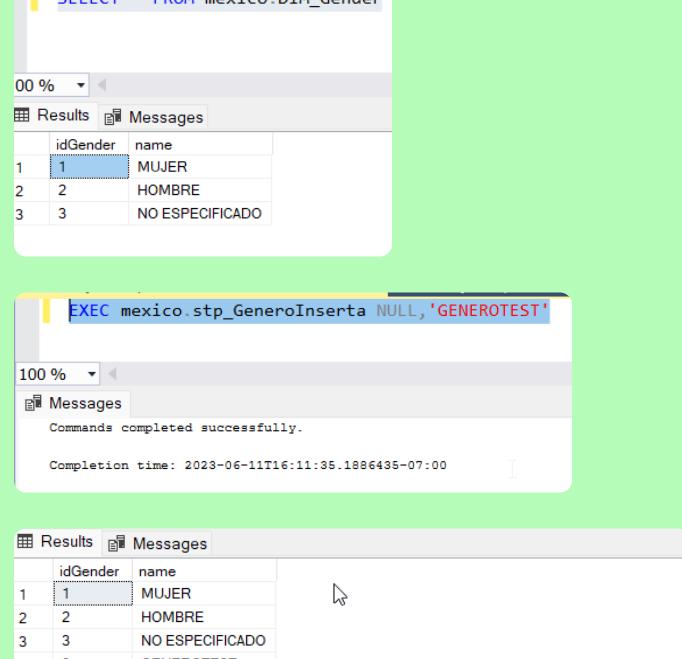
    BEGIN TRY

        IF @idGender IS NOT NULL
        BEGIN
            UPDATE G
                SET [name] = @name
                FROM
                    mexico.DIM_Gender G
                WHERE
                    G.idGender = @idGender
        END
        ELSE
        BEGIN
            IF(SELECT COUNT(*) FROM mexico.DIM_Gender WHERE [name] = @name) != 0
            BEGIN
                INSERT INTO mexico.DIM_Gender ([name]) VALUES (@name)
            END
        END
    END TRY
    BEGIN CATCH

        DECLARE @Err NVARCHAR(1000)
        SET @Err = ' (' + CAST(ERROR_NUMBER() AS varchar(6)) + ')' + CONVERT(varchar(500),
        ERROR_MESSAGE())

        RETURN @Err
    END CATCH
END
```

### ⌚ Demostración de funcionamiento



The screenshot shows three windows from SQL Server Management Studio:

- A top window titled "SELECT \* FROM mexico.DIM\_Gender" displays a table with three rows: idGender (1), name (MUJER); idGender (2), name (HOMBRE); and idGender (3), name (NO ESPECIFICADO).
- A middle window titled "EXEC mexico.stp\_GeneroInserta NULL, 'GENEROTEST'" shows the command being run.
- A bottom window titled "Messages" shows the output: "Commands completed successfully." and "Completion time: 2023-06-11T16:11:35.1886435-07:00".

# Stored Procedure

## ***world.stp\_SymptomsTotalElimina***

Este stored procedure que realiza la tarea de eliminar un registro de la tabla de world.FACT\_SymptomsTotals.

```
ALTER PROCEDURE [world].[stp_SymptomsTotalElimina]
    @idSymptomsTotal INT
AS
BEGIN
    SET NOCOUNT ON;
    BEGIN TRY

        DELETE FROM world.FACT_SymptomsTotals WHERE idSymptomsTotals = @idSymptomsTotal

    END TRY
    BEGIN CATCH

        DECLARE @Err NVARCHAR(1000)
        SET @Err = ' (' + CAST(ERROR_NUMBER() AS varchar(6)) + ')' + CONVERT(varchar(500),
        ERROR_MESSAGE())

        RETURN @Err

    END CATCH
END
```

### ⓘ Demostración de funcionamiento

The screenshot shows three separate queries in a SQL Server Management Studio window:

- Query 1:** A SELECT statement retrieves all columns from the world.FACT\_SymptomsTotals table where idSymptomsTotals equals 13799. The result set contains one row with the following values:

idSymptomsTotals	idCountry	idGender	Fever	Tiredness	DryCough	DificultudBreath	SoreThroat	Pains	NasalCongestion	RunnyNose	Diarrhea	noneSymptoms
13799	249	3	1	0	0	0	0	0	1	0	0	0
- Query 2:** An EXEC statement runs the stored procedure stp\_SymptomsTotalElimina with the parameter value 13799.
- Query 3:** A final SELECT statement retrieves the same data as Query 1, but this time there is no row for idSymptomsTotals 13799, indicating it has been successfully deleted.

# Conclusiones

El Data Warehouse es una herramienta valiosa en la recopilación, almacenamiento y análisis de grandes cantidades de datos relacionados al COVID-19. La implementación de una arquitectura efectiva y el uso de herramientas como SQL Server permiten un manejo eficiente y efectivo del Data Warehouse. Esto, a su vez, ayuda a tomar decisiones informadas y a prevenir futuros brotes del virus.

## 1    **Implementación de arquitectura efectiva**

La arquitectura del Data Warehouse es fundamental para un almacenamiento y análisis efectivo de datos relacionados al COVID-19.

## 2    **Uso de herramientas como SQL Server**

SQL Server es una herramienta esencial en el manejo de datos almacenados en el Data Warehouse, permitiendo un análisis y utilización más eficiente.

## 3    **Toma de decisiones informadas**

El almacenamiento y análisis de datos en el Data Warehouse permite la toma de decisiones informadas en la lucha contra el COVID-19.