



# MUJERES IT

PROYECTO FINAL  
SQL



**Estudiante:** Lilian Virginia Lombardi López

**Profesor:** Miguel Rodas

# Nombre del Proyecto: Mujeres IT, Mujeres en IT

**OBJETIVO:** El objetivo de este Proyecto es crear una base de datos para almacenar la información de las Mujeres que trabajan en IT o tienen conocimientos pero aún no tienen Empleo en dicha área. Se realiza para saber información como el Cargo al que pertenecen, la Empresa en la que trabajan y las Fechas de Ingreso a la misma entre otros datos, es hasta el año 2021, con registros de Uruguay. Con la idea de hacer uso de los datos para su posterior análisis.

# Situación Problemática:

Debido a que pertenezco al **Movimiento Mujeres IT en Uruguay**, y soy parte de su **Directorio**. Es que surge esta Idea como Proyecto Final.(Es un Movimiento sin fines de Lucro) Cada vez son más y más Mujeres que se suman al Mundo del IT, en Uruguay. Y es debido a eso que quise proponerme realizar una BBDD que pueda ayudar a tratar los datos de cada una de ellas, de forma correcta y simple.

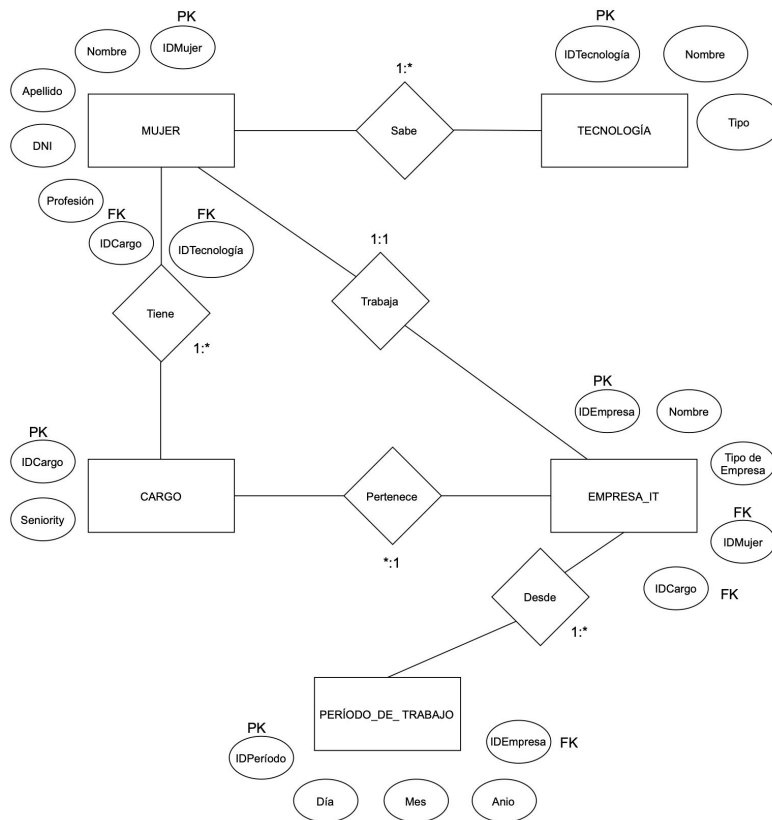
Y también para saber cuántas Mujeres están trabajando actualmente en IT. Al trabajar en una Empresa de Tecnología, tengo el conocimiento de que en dicha Empresa somos el 27% de Mujeres contra un 72% de Hombres, que trabajamos en la misma. Por ende estoy casi segura que no todas las Mujeres que pertenecen al Directorio tienen un trabajo en IT. Ya que al entrar al Directorio, yo aún no contaba con uno, pero si con conocimientos.

# Modelo De Negocio:

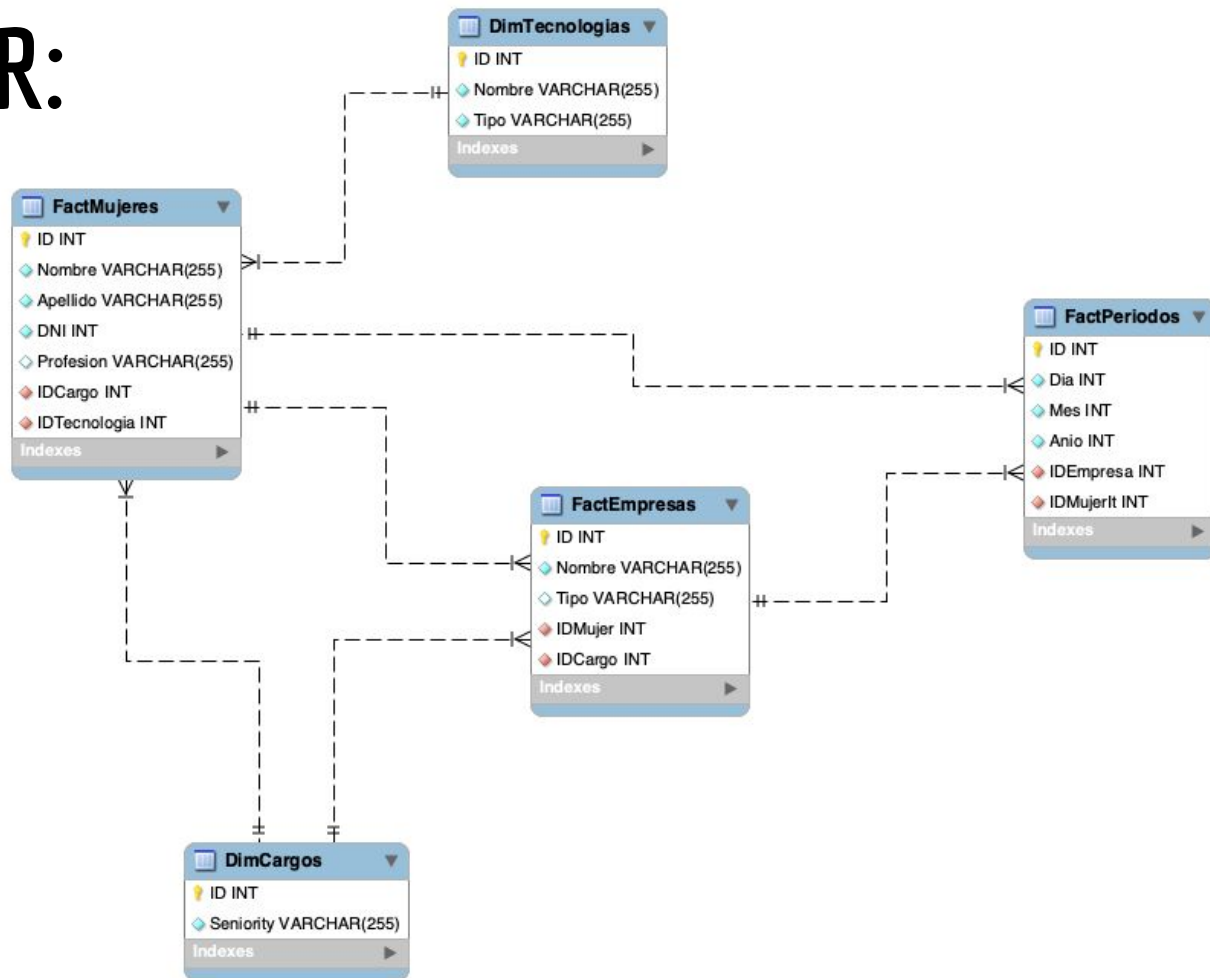
El Modelo de Negocio hace referencia a qué Tipo de Industria o Rubro aplicaría el Script de SQL que cree.

El Cliente cómo dije anteriormente sería un Movimiento de Mujeres en IT, y por ende sería gratis (es decir no cobraría por el mismo), debido a que se trata de una Empresa conformada por Mujeres en IT, y sin fines de Lucro. Sería para aportar un granito de arena a esas 4 Mujeres que iniciaron el Movimiento, que ahora se convirtió en una gran Familia y Apoyo para todas las Mujeres en IT en Uruguay.

# Primer Modelo Entidad - Relación: ("Pienso")



# Modelo ER:



# Tablas del Proyecto:

**DimCargos:** Esta es una tabla dimensional de mi modelo dimensional. Contendrá el ID que la identificará, así como el Seniority al que pertenecerá la MujerIT. (El Seniority es la experiencia que la persona tiene en esa tecnología o cargo en el que se encuentra. Se desarrolla, según los conocimientos que se van adquiriendo con el tiempo.)

Columna	Tipo de Dato	Clave	Características
ID	INT	PK	UNIQUE NOT NULL AUTO_INCREMENT
SENIORITY	VARCHAR(255)		NOT NULL

**DimTecnologias:** Esta es una tabla dimensional de mi modelo dimensional. Contendrá un ID que la identificará. Así como el Nombre de dicha Tecnología y el Tipo al que pertenece. Dos Tecnologías diferentes pueden pertenecer a un mismo Tipo de Tecnología.

Columna	Tipo de Dato	Clave	Características
ID	INT	PK	NOT NULL AUTO_INCREMENT
NOMBRE	VARCHAR(255)		NOT NULL
TIPO	VARCHAR(255)		NOT NULL



**FactMujeres:** Esta es una tabla de hechos de mi modelo dimensional. La cual contendrá su ID que la identificará. También contendrá el Nombre, Apellido y DNI de la MujerIT. Así como su Profesión. Y tendrá Claves Foráneas de las tablas DimCargos y DimTecnologías.

Columna	Tipo de Dato	Clave	Características
ID	INT	PK	NOT NULL AUTO_INCREMENT
NOMBRE	VARCHAR(255)		NOT NULL
APELLIDO	VARCHAR(255)		NOT NULL
DNI	INT		NOT NULL
PROFESION	VARCHAR(255)		NOT NULL
IDCARGO	INT	FK	NOT NULL
IDTECNOLOGIA	INT	FK	NOT NULL

**FactEmpresas:** Esta es una tabla de hechos de mi modelo dimensional. Contendrá un ID que la identificará. Poseerá Nombre de la Empresa, así como el Tipo de Empresa a la que se refiere. Varias Empresas pueden tener el mismo Tipo de Empresa. Contendrá también Claves Foráneas de las tablas FactMujeres y de DimCargos.

Columna	Tipo de Dato	Clave	Características
ID	INT	PK	UNIQUE NOT NULL
NOMBRE	VARCHAR(255)		NOT NULL
TIPO	VARCHAR(255)		
IDMUJER	INT	FK	NOT NULL
IDCARGO	INT	FK	NOT NULL

**FactPeriodos:** Esta es una tabla de hechos de mi modelo dimensional. Contendrá un ID que la identificará. Así como el Día, Mes y Año en el que la MujerIT ingresó a la Empresa. Es por ello que también tendrá las Claves Foráneas de las tablas FactEmpresas y de FactMujeres.

Columna	Tipo de Dato	Clave	Características
ID	INT	PK	NOT NULL AUTO_INCREMENT
DIA	INT		NOT NULL
MES	INT		NOT NULL
ANIO	INT		NOT NULL
IDEMPRESA	INT	FK	NOT NULL
IDMUJERIT	INT	FK	NOT NULL

# Creación del Script SQL - Schema y Uso del Mismo:

```
1      #CREATION OF SCHEMA mujeres_it and Use of it.  
2  *   CREATE SCHEMA IF NOT EXISTS mujeres_it;  
3  
4  *   USE mujeres_it;  
5
```

# Creación del Script SQL - Tablas:

```
10  # CREATION OF TABLES OF 5 TABLES FOR THIS BBDD -> mujeres_it.
11
12  /* Creation of Table DimCargos. */
13  ● ⊖ CREATE TABLE IF NOT EXISTS DimCargos(
14      |                                     ID INT UNIQUE NOT NULL AUTO_INCREMENT PRIMARY KEY,
15      |                                     Seniority VARCHAR(255) NOT NULL);
16
17  /* Creation of Table DimTecnologias. */
18  ● ⊖ CREATE TABLE IF NOT EXISTS DimTecnologias(
19      |                                     ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
20      |                                     Nombre VARCHAR(255) NOT NULL,
21      |                                     Tipo VARCHAR(255) NOT NULL);
22
23  /* Creation of Table FactMujeres. */
24  ● ⊖ CREATE TABLE IF NOT EXISTS FactMujeres(
25      |                                     ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
26      |                                     Nombre VARCHAR(255) NOT NULL,
27      |                                     Apellido VARCHAR(255) NOT NULL,
28      |                                     DNI INT NOT NULL,
29      |                                     Profesion VARCHAR(255),
30      |                                     IDCargo INT NOT NULL,
31      |                                     IDTecnologia INT NOT NULL,
32      |                                     FOREIGN KEY (IDCargo) REFERENCES DimCargos(ID),
33      |                                     FOREIGN KEY (IDTecnologia) REFERENCES DimTecnologias(ID));
34
35  /* Creation of Table FactEmpresas. */
36  ● ⊖ CREATE TABLE IF NOT EXISTS FactEmpresas(
37      |                                     ID INT UNIQUE NOT NULL PRIMARY KEY,
38      |                                     Nombre VARCHAR(255) NOT NULL,
39      |                                     Tipo VARCHAR(255),
40      |                                     IDMujer INT NOT NULL,
41      |                                     IDCargo INT NOT NULL,
42      |                                     FOREIGN KEY (IDMujer) REFERENCES FactMujeres(ID),
43      |                                     FOREIGN KEY (IDCargo) REFERENCES DimCargos(ID));
44
```

```
44
45      /* Creation of Table FactPeriodos. */
46  * ⊖ CREATE TABLE IF NOT EXISTS FactPeriodos(
47      ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
48      Dia INT NOT NULL,
49      Mes INT NOT NULL,
50      Anio INT NOT NULL,
51      IDEmpresa INT NOT NULL,
52      IDMujerIt INT NOT NULL,
53      FOREIGN KEY (IDEmpresa) REFERENCES FactEmpresas(ID),
54      FOREIGN KEY (IDMujerIt) REFERENCES FactMujeres(ID));
55
```

En este caso se crean **5 Tablas** para la **BBDD mujeres\_it**. Cada una con sus respectivos datos que luego serán llenados por la información que le corresponda a cada una.  
Son las Tablas que inicialmente se pensaron en el Primer Modelo Entidad Relación.

# Script SQL - Inserción de Datos en cada Tabla:

```
59
60 #POPULATION OF THE TABLES, WITH 10 RECORDS PER EACH TABLE
61
62 /* Population of the Table mujeres_it.DimCargos. */
63 • INSERT INTO mujeres_it.DimCargos (ID, Seniority)
64 VALUES (NULL, 'Trainee'), (NULL, 'Junior'), (NULL, 'Junior Advanced'), (NULL, 'Semi Senior'), (NULL, 'Senior'),
65 (NULL, 'Senior Advanced'), (NULL, 'Senior Level One'), (NULL, 'Senior Level Two'), (NULL, 'Senior Level Three'), (NULL, 'Senior Level Four');
66
67 /* Population of the mujeres_it.DimTecnologias. */
68 • INSERT INTO mujeres_it.DimTecnologias (ID, Nombre, Tipo)
69 VALUES (NULL, 'iOS', 'Mobile'), (NULL, 'Android', 'Mobile'), (NULL, 'React', 'Web UI'), (NULL, 'MongoDB', 'Web UI'), (NULL, 'VUE JS', 'Web UI'),
70 (NULL, 'JavaScript', 'Web UI'), (NULL, 'Java', 'Programming'), (NULL, 'C++', 'Programming'), (NULL, 'Python', 'Programming'), (NULL, 'C', 'Programming');
71
72 /* Population of the mujeres_it.FactMujeres. */
73 • INSERT INTO mujeres_it.FactMujeres (ID, Nombre, Apellido, DNI, Profesion, IDCargo, IDTecnologia)
74 VALUES (NULL, 'Virginia', 'Lombardi', 46288064, 'iOS Developer', 1, 1), (NULL, 'Lucia', 'Perez', 54568907, 'Systems Engineering', 5, 4),
75 (NULL, 'Mariana', 'Orman', 33458065, 'iOS Developer', 8, 1), (NULL, 'Lorena', 'Alvarez', 56789073, 'Java Developer', 2, 7),
76 (NULL, 'Alicia', 'Gonzalez', 45678753, 'Python Developer', 3, 9), (NULL, 'Martina', 'Bernardi', 56873219, 'Android Developer', 4, 2),
77 (NULL, 'Isabel', 'Vayra', 34623786, 'Web UI Developer', 6, 3), (NULL, 'Micaela', 'Nelson', 56784325, 'Web UI Developer', 7, 5),
78 (NULL, 'Susana', 'Mori', 34868063, 'C Developer', 10, 10), (NULL, 'Natalia', 'Bonjour', 65433456, 'C++ Developer', 9, 8);
79
80 /* Population of the mujeres_it.FactEmpresas. */
81 • INSERT INTO mujeres_it.FactEmpresas (ID, Nombre, Tipo, IDMujer, IDCargo)
82 VALUES (123, 'Globant', 'Global Company', 2, 5), (124, 'TCS', 'Global Company', 2, 5), (125, 'Sabre', 'Global Company', 3, 5),
83 (126, 'd-Local', 'Global Company', 4, 2), (127, 'Overactive', 'Global Company', 5, 3), (128, 'Wabbi', 'Start Up', 6, 4),
84 (129, 'Tienda Inglesa', 'Global Company', 7, 6), (200, 'Nimacloud', 'Start Up', 8, 7), (201, 'Mercado Libre', 'Start Up', 9, 10),
85 (202, 'SparkDigital', 'Start Up', 10, 9);
86
87 /* Population of the mujeres_it.FactPeriodos. */
88 • INSERT INTO mujeres_it.FactPeriodos (ID, Dia, Mes, Anio, IDEmpresa, IDMujerIt)
89 VALUES (NULL, 18, 1, 2021, 123, 10), (NULL, 20, 2, 2019, 125, 2), (NULL, 23, 12, 2020, 127, 4), (NULL, 31, 10, 2010, 129, 6), (NULL, 12, 8, 2015, 201, 8),
90 (NULL, 11, 7, 2021, 202, 3), (NULL, 21, 6, 2021, 124, 5), (NULL, 29, 3, 2013, 126, 7), (NULL, 1, 4, 2014, 128, 9), (NULL, 10, 5, 2016, 200, 1);
91
92
```

**EXPLICACIÓN:** En este caso, al principio se pensó en hacerlo mediante archivos .csv, por ello en la parte de **PDF\_EXPLICACION\_IMPORTACION**, se encuentran los archivos .csv de cada una de las tablas que fueron creadas. Más la explicación de cómo se realiza dicha importación de datos. Se dejan en el repositorio, (aunque no hayan sido utilizadas finalmente), debido a que son parte del proceso que se realizó durante el “pienso” de la BBDD mujeres\_it.

Finalmente, para la inserción de datos, se utilizó la **instrucción INSERT INTO:**

**INSERT INTO Nombre Tabla (PrimeraColumna, SegundaColumna, etc)**

**VALUES (“Dato1”, “Dato2”, etc);**

Se insertan al comienzo 10 Registros para cada una de las Tablas y sus correspondientes columnas.

En los casos en que el ID o algún otro dato es AUTO\_INCREMENT, se coloca como VALUE (en esa posición) NULL. Otro dato, es que debía sacar los ID's que eran Clave Foránea. Para ello fui llenando con datos, las Tablas de las que necesitaba dichas Claves, y luego ingresando cada ID en su parte correspondiente, en las otras Tablas.



# Listado de Vistas: (Se crean con la BBDD seleccionada)

- **Primera Vista:** vista\_profesion\_mujeresit.

**OBJETIVO:** Mostrar los Nombres y Apellidos de las Mujeres IT, su Profesión y el Tipo de Seniority que tiene cada una de ellas.

**DESCRIPCIÓN:** La idea principal de esta vista es que la persona que vaya a utilizar la base de datos sepa quien es la Mujer IT, su Profesión y el Tipo de Seniority que tiene la misma. En este caso es muy importante conocer el Seniority de cada una de ellas, para saber qué experiencia tienen en su profesión. Es debido a eso que se crea esta vista.

**TABLAS QUE COMPONEN A LA VISTA:** Las tablas que componen la vista, vista\_profesion\_mujeresit, son: FactMujeres y DimCargos.

## Creación del Script para la Vista - vista\_profesion\_mujeresit:

```
3  #FIRST VIEW
4  /* Creation of the view vista_profesion_mujeresit. This view shows information of the Name, the Last Name,
5  the Profession and the Seniority of each MujereIt in the data base.
6  This view was created with the intention of showing the relevant information of the MujerIt on her work. */
7  CREATE VIEW
8  vista_profesion_mujeresit
9  AS SELECT
10 FactMujeres.nombre,
11 FactMujeres.apellido,
12 FactMujeres.profesion,
13 DimCargos.seniority
14 FROM
15 mujeres_it.DimCargos,
16 mujeres_it.FactMujeres
17 WHERE
18 FactMujeres.IDCargo = DimCargos.ID;
19
```

## EJEMPLO VISTA, vista\_profesion\_mujeresit:

Result Grid					
Filter Rows: <input type="text" value="Search"/>					
	nombre	apellido	profesion	seniority	
▶	Virginia	Lombardi	iOS Developer	Trainee	
▶	Lorena	Alvarez	Java Developer	Junior	
▶	Siri	Altez	iOS Developer	Junior	
▶	Alicia	Gonzalez	Python Developer	Junior Advanced	
▶	Martina	Bernardi	Android Developer	Semi Senior	
▶	Lucia	Perez	Systems Engineering	Senior	
▶	Isabel	Vayra	Web UI Developer	Senior Advanced	
▶	Micaela	Nelson	Web UI Developer	Senior Level One	
▶	Mariana	Orman	iOS Developer	Senior Level Two	
▶	Natalia	Bonjour	C++ Developer	Senior Level Three	
▶	Susana	Mori	C Developer	Senior Level Four	
▶					

- **Segunda Vista:** vista\_ingreso\_mujeresit\_en\_empresait.

**OBJETIVO:** Mostrar el Nombre, el Apellido de la Mujer IT. Y también incluir información relevante como el Día, el Mes y el Año de ingreso de la Mujer IT a la Empresa que integra. Ordenado de forma DESC por el Año.

**DESCRIPCIÓN:** Debido a que es importante tener una idea de cuándo es que ingresó la Mujer IT a la Empresa correspondiente es que se crea esta vista. Con el fin de registrar el Nombre y Apellido de la misma, la Fecha de ingreso y por último la Empresa en la que trabaja actualmente.

**TABLAS QUE COMPONEN A LA VISTA:** Las tablas que componen la vista, vista\_ingreso\_mujeresit\_en\_empresait, son: FactMujeres, FactPeriodos y FactEmpresas.

## Creación del Script para la Vista - vista\_ingreso\_mujeresit\_en\_empresait:

```
20
21  #SECOND VIEW
22  /* Creation of the view vista_ingreso_mujeresit_en_empresait. This view shows information of the Name, the Last Name of the MujerIt.
23     Also shows the day, the month and the year in which each MujerIt joined the company.
24     And finally the company where she is working at.
25     This view was created with the intention of showing the MujerIt's entry into the company where she currently works at. */
26  • CREATE VIEW
27    vista_ingreso_mujeresit_en_empresait
28  AS SELECT
29    FactMujeres.nombre AS Nombre_Mujer_IT,
30    FactMujeres.apellido AS Apellido,
31    FactPeriodos.dia AS Dia,
32    FactPeriodos.mes AS Mes,
33    FactPeriodos.anio AS Anio,
34    FactEmpresas.nombre AS Empresa
35  FROM
36    mujeres_it.FactMujeres,
37    mujeres_it.FactPeriodos,
38    mujeres_it.FactEmpresas
39  WHERE
40    FactPeriodos.IDMujerIT = FactMujeres.ID
41    AND FactEmpresas.IDMujer = FactMujeres.ID
42    ORDER BY anio DESC;
43
```

## EJEMPLO VISTA, vista\_ingreso\_mujeresit\_en\_empresait:

Result Grid   Filter Rows: <input type="text" value="Search"/> Export:							
	Nombre_Mujer_IT	Apellido	Dia	Mes	Anio	Empresa	
▶	Natalia	Bonjour	18	1	2021	SparkDigital	
▶	Mariana	Orman	11	7	2021	Sabre	
▶	Alicia	Gonzalez	21	6	2021	Overactive	
▶	Lorena	Alvarez	23	12	2020	d-Local	
▶	Lucia	Perez	20	2	2019	TCS	
▶	Virginia	Lombardi	10	5	2016	Globant	
▶	Micaela	Nelson	12	8	2015	Nimacloud	
▶	Susana	Mori	1	4	2014	Mercado Libre	
▶	Isabel	Vayra	29	3	2013	Tienda Inglesa	
▶	Martina	Bernardi	31	10	2010	Wabbi	

- **Tercera Vista:** vista\_\_mujeresit\_\_trabajan\_\_empresa.

**OBJETIVO:** Mostrar el Nombre y el Apellido de la Mujer IT. Y también incluir información relevante como su Profesión, Empresa en la cual trabaja y el Tipo de Empresa.

**DESCRIPCIÓN:** Considerando que hoy en día hay muchos Tipos de Empresas en las que una Mujer IT puede trabajar, es que se crea esta vista. Para poder visualizar en qué Tipo de Empresa y en cual Empresa trabaja cada una.

**TABLAS QUE COMPONEN A LA VISTA:** Las tablas que componen la vista, vista\_\_mujeresit\_\_trabajan\_\_empresa, son: FactMujeres y FactEmpresas.

## Creación del Script para la Vista - vista\_mujeresit\_trabajan\_empresa:

```
44
45  #THIRD VIEW
46  /* Creation of the view vista_mujeresit_trabajan_empresa. This view shows information of the Name, the Last Name of the MujerIt.
47  Also shows the Profession, the Company and the type of Company where she is working at.
48  This view was created with the intention of showing not only what they do within the company,
49  but also in which company they work and the type of company in which they work.*/
50  CREATE VIEW
51  vista_mujeresit_trabajan_empresa
52  AS SELECT
53  FactMujeres.nombre AS Nombre,
54  FactMujeres.apellido AS Apellido,
55  FactMujeres.profesion AS Profesion,
56  FactEmpresas.nombre AS Empresa,
57  FactEmpresas.tipo AS Tipo_Empresa
58  FROM
59  mujeres_it.FactMujeres,
60  mujeres_it.FactEmpresas
61  WHERE
62  FactEmpresas.IDMujer = FactMujeres.ID;
63
```



## EJEMPLO VISTA, vista\_mujeresit\_trabajan\_empresa:

Result Grid   Filter Rows: <input type="text" value="Search"/> Export: 						
	Nombre	Apellido	Profesion	Empresa	Tipo_Empresa	
▶	Virginia	Lombardi	iOS Developer	Globant	Global Company	
●	Lucia	Perez	Systems Engineering	TCS	Global Company	
	Mariana	Orman	iOS Developer	Sabre	Global Company	
●	Lorena	Alvarez	Java Developer	d-Local	Global Company	
	Alicia	Gonzalez	Python Developer	Overactive	Global Company	
●	Martina	Bernardi	Android Developer	Wabbi	Start Up	
	Isabel	Vayra	Web UI Developer	Tienda Inglesa	Global Company	
●	Micaela	Nelson	Web UI Developer	Nimacloud	Start Up	
	Susana	Mori	C Developer	Mercado Libre	Start Up	
●	Natalia	Bonjour	C++ Developer	SparkDigital	Start Up	

- **Cuarta Vista:** vista\_\_mujeresit\_\_saben\_\_tecnologias.

**OBJETIVO:** Mostrar el Nombre, el Apellido de la Mujer IT. Y también incluir información relevante como la Tecnología que manejan y el Tipo de dicha Tecnología.


**DESCRIPCIÓN:** Se crea a partir de la necesidad de saber cuántas Mujeres IT saben determinada Tecnología y el Tipo de la misma. Esto sirve para saber cuántas Mujeres IT conocen la misma Tecnología o el mismo Tipo de Tecnología también.

**TABLAS QUE COMPONEN A LA VISTA:** Las tablas que componen la vista, vista\_\_mujeresit\_\_saben\_\_tecnologias, son: FactMujeres y DimTecnologias.

## Creación del Script para la Vista - vista\_mujeresit\_saben\_tecnologias:

```
64
65 #FOURTH VIEW
66 /* Creation of the view vista_mujeresit_saben_tecnologias. This view shows information of the Name, the Last Name of the MujerIt.
67 Also shows the Technology and the type of Technology she is capable in and know.
68 This view was created with the intention of showing which technologies are handled by the MujerIt and what type they are. */
69 • CREATE VIEW
70 vista_mujeresit_saben_tecnologias
71 AS SELECT
72 FactMujeres.nombre AS Nombre,
73 FactMujeres.apellido AS Apellido,
74 DimTecnologias.nombre AS Tencologia,
75 DimTecnologias.tipo AS Tipo_Tecnologia
76 FROM
77 mujeres_it.DimTecnologias,
78 mujeres_it.FactMujeres
79 WHERE
80 FactMujeres.IDTecnologia = DimTecnologias.ID;
81
```

## EJEMPLO VISTA, vista\_mujeresit\_saben\_tecnologias:

Result Grid					
  Filter Rows: <input type="text" value="Search"/>					
	Nombre	Apellido	Tencologia	Tipo_Tecnologia	
▶	Virginia	Lombardi	iOS	Mobile	
▶	Mariana	Orman	iOS	Mobile	
▶	Siri	Altez	iOS	Mobile	
▶	Isabel	Vayra	React	Web UI	
▶	Lucia	Perez	MongoDB	Web UI	
▶	Micaela	Nelson	VUE JS	Web UI	
▶	Lorena	Alvarez	Java	Programming	
▶	Natalia	Bonjour	C++	Programming	
▶	Alicia	Gonzalez	Python	Programming	
▶	Susana	Mori	C	Programming	

- **Quinta Vista:** vista\_profesion\_tecnologia\_iOS\_de\_mujeresit.

**OBJETIVO:** Mostrar el Nombre, el Apellido de la Mujer IT, filtrando por las Mujeres IT que saben sobre la Tecnología iOS.

**DESCRIPCIÓN:** Esta vista se crea para tener conocimiento de cuantas Mujeres IT hay en la Base de Datos que sepan la tecnología iOS. Incluyendo Nombre y Apellido de las mismas, así como la Profesión y la Tecnología que manejan que en este caso va a ser iOS.

**TABLAS QUE COMPONEN A LA VISTA:** Las tablas que componen la vista, vista\_profesion\_tecnologia\_iOS\_de\_mujeresit, son: FactMujeres y DimTecnologias.

## Creación del Script para la Vista - vista\_profesion\_tecnologia\_iOS\_de\_mujeresit:

```
82
83  #FIFTH VIEW
84  /* Creation of the view vista_profesion_tecnologia_iOS_de_mujeresit. This view shows information of the Name, the Last Name of the MujerIt.
85  Also shows the Profession of the MujerIt and the Technology she works at. With the exception that only is going to show
86  the MujerIt that works on the technology iOS.
87  This view was created with the intention of filtering MujerIt working on the technology named 'iOS' and siplay that information. */
88  CREATE VIEW
89  vista_profesion_tecnologia_iOS_de_mujeresit
90  AS SELECT
91  FactMujeres.nombre AS Nombre,
92  FactMujeres.apellido AS Apellido,
93  FactMujeres.profesion AS Profesion,
94  DimTecnologias.nombre AS Tecnologia
95  FROM
96  mujeres_it.DimTecnologias,
97  mujeres_it.FactMujeres
98  WHERE
99  FactMujeres.IDTecnologia = DimTecnologias.ID
100  AND DimTecnologias.nombre = 'iOS';
101
```

EJEMPLO VISTA, vista\_profesion\_tecnologia\_iOS\_de\_mujeresit:

Result Grid					
Filter Rows: <input type="text" value="Search"/>					
	Nombre	Apellido	Profesion	Tecnologia	
▶	Virginia	Lombardi	iOS Developer	iOS	
▶	Mariana	Orman	iOS Developer	iOS	
▶	Siri	Altez	iOS Developer	iOS	
▶					

**ACLARACIÓN SOBRE LAS VISTAS:** Las Vistas se hacen sin JOIN, ya que es opcional su uso.

# Script SQL - Creación de Funciones:

Con el objetivo de procesar y manipular datos de forma procedural y eficiente, se presentan las siguientes funciones para la Base de Datos MujeresIT.

- **Función Conteo\_Mujeres\_en\_IT\_entre\_periodos:**

**OBJETIVO:** Contar la cantidad de Mujeres IT que ingresaron en determinado período elegido por el usuario.

**DESCRIPCIÓN:** Primero se crea una vista llamada: **vista\_traer\_fecha**. Esta vista lo que hace es traer la fecha en formato por ejemplo: 2012-01-23 (es decir año-mes-día). Para realizar una consulta en la función Conteo\_Mujeres\_en\_IT\_entre\_periodos, se debe ingresar los parámetros FechaIni (del tipo DATE) y FechaFin (del tipo DATE). El resultado que arroja la función es del tipo INT y corresponde a la cantidad de Mujeres IT que entraron en ese período de fechas.



**TABLAS QUE INTERVIENEN EN LA FUNCIÓN:** La tabla que interviene en la función es: FactPeriodos.

**Creación del Script para la Vista - vista\_traer\_fecha:**

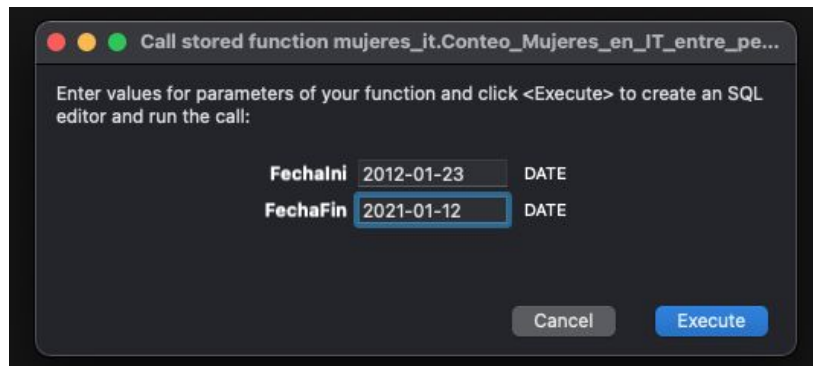
```
1  #CREATION OF THE VIEW TO BRING THE COMPLETE DATE TO THE FUNCTION Conteo_Mujeres_en_IT_entre_periodos:
2  * CREATE VIEW
3  vista_traer_fecha
4  AS SELECT CAST(CONCAT(Anio, '-', Mes, '-', Dia) AS DATE) AS Fecha,
5  IDMujerIt
6  FROM
7  mujeres_it.FactPeriodos;
8
```

## Creación del Script para la Función Conteo\_Mujeres\_en\_IT\_entre\_periodos:

```
1 DELIMITER $$
2 CREATE DEFINER='root'@'localhost' FUNCTION `Conteo_Mujeres_en_IT_entre_periodos`(FechaIni DATE, FechaFin DATE) RETURNS int
3 READS SQL DATA
4 BEGIN
5     DECLARE Total INT;
6     SET Total = (SELECT COUNT(IDMujerIt)
7     FROM vista_traer_fecha
8     WHERE Fecha BETWEEN FechaIni AND FechaFin);
9     RETURN Total;
10 END$$
11 DELIMITER ;
12
```

**PASOS FINALES:** Una vez creado el Script de la Función, se le da a ejecutar (al rayo). Y es ahí cuando se crea la Función, la cual aparecerá en la parte de Funciones de la BBDD. Una vez que tenemos esto ya podemos ejecutarla.

**EJEMPLO DE Conteo\_Mujeres\_en\_IT\_entre\_periodos:** En este caso nos aparecerá un recuadro, dónde **DEBEMOS** recordar que la fecha se escribe de la siguiente manera: 2012-01-23 (año-mes-día). Teniendo esto presente, colocamos las fechas que queremos verificar cuántos ingresos de MujeresIT se han realizado en ese lapso de tiempo, como se muestra en la figura:



Call stored function mujeres\_it.Conteo\_Mujeres\_en\_IT\_entre\_pe...

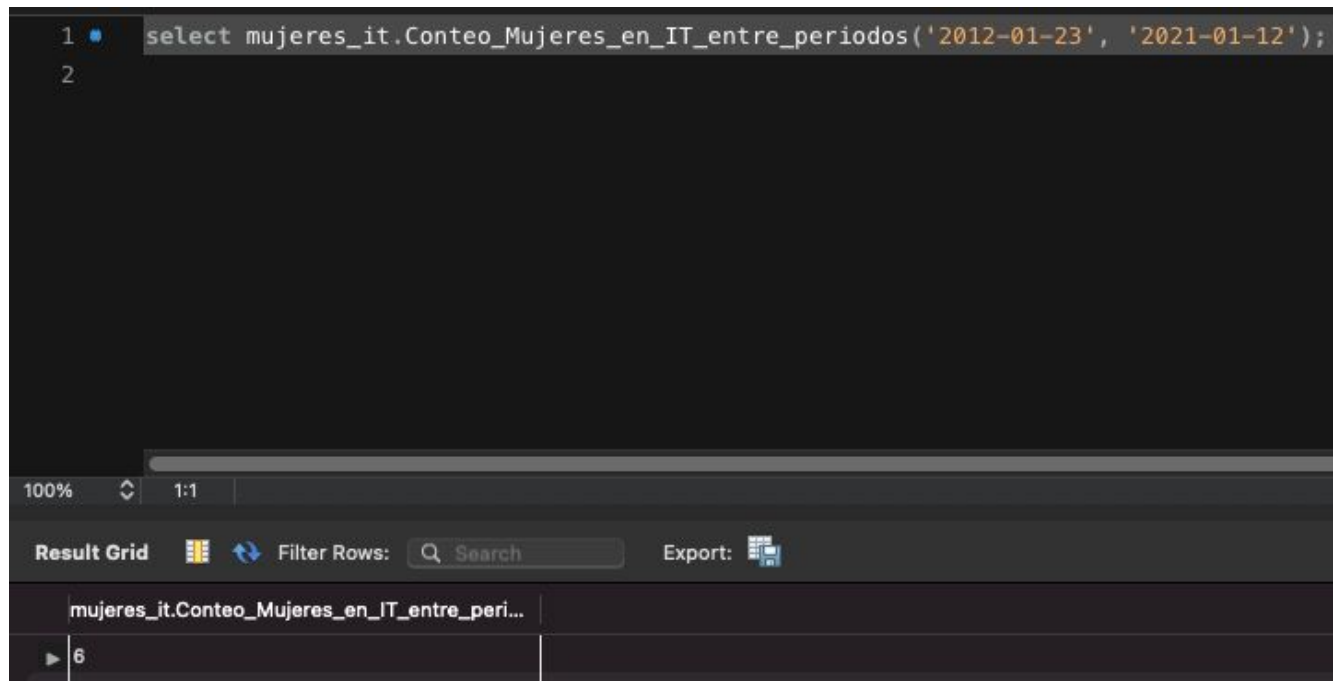
Enter values for parameters of your function and click <Execute> to create an SQL editor and run the call:

FechaIni	<input type="text" value="2012-01-23"/>	DATE
FechaFin	<input type="text" value="2021-01-12"/>	DATE

Cancel Execute

## Resultado de Ejecutar la Función Conteo\_Mujeres\_en\_IT\_entre\_periodos:

```
1 select mujeres_it.Cuento_Mujeres_en_IT_entre_periodos('2012-01-23', '2021-01-12');  
2
```



The screenshot shows a SQL query execution interface. The query is: `select mujeres_it.Cuento_Mujeres_en_IT_entre_periodos('2012-01-23', '2021-01-12');`. The interface includes a "Result Grid" section with a search bar and an "Export" button. The result grid shows a single row with the value 6.

mujeres_it.Cuento_Mujeres_en_IT_entre_peri...
6

Este resultado nos muestra que hubieron 6 Ingresos de MujeresIT, en ese lapso de tiempo.

- **Función Conteo\_Mujeres\_en\_IT\_en\_tecnologia\_tipo:**

**OBJETIVO:** Contar la cantidad de Mujeres IT que manejan X Tipo de Tecnología.

**DESCRIPCIÓN:** Para realizar una consulta en la función Conteo\_Mujeres\_en\_IT\_en\_tecnologia\_tipo, se debe ingresar el parámetro Tipo (de tipo VARCHAR(255)). El resultado que arroja la función es del tipo INT y corresponde a la cantidad de Mujeres IT que manejan ese Tipo de Tecnología dentro de la Base de Datos.

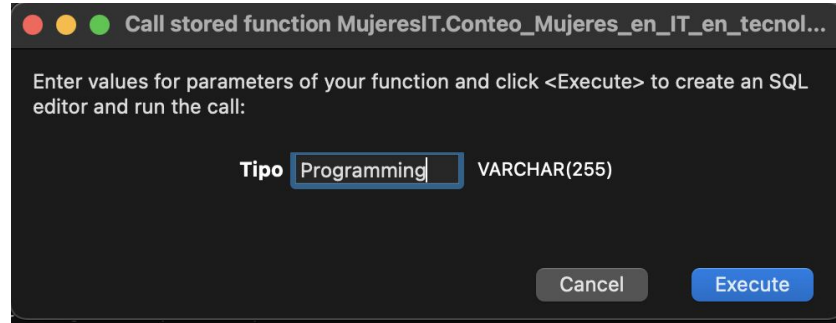
**TABLAS QUE INTERVIENEN EN LA FUNCIÓN:** Las tablas que intervienen en dicha función son DimTecnologias y FactMujeres.

## Creación del Script para la Función Conteo\_Mujeres\_en\_IT\_en\_tecnologia\_tipo:

```
1 DELIMITER $$
2 CREATE DEFINER='root'@'localhost' FUNCTION `Conteo_Mujeres_en_IT_en_tecnologia_tipo`(Tipo VARCHAR(255)) RETURNS int
3 READS SQL DATA
4 BEGIN
5     DECLARE Conteo INT;
6     SET Conteo = (SELECT COUNT(FactMujeres.ID)
7                   FROM FactMujeres
8                   INNER JOIN DimTecnologias
9                   ON DimTecnologias.ID = FactMujeres.IDTecnologia
10                  WHERE DimTecnologias.tipo = Tipo);
11     RETURN Conteo;
12 END$$
13 DELIMITER ;
```

**PASOS FINALES:** Una vez creado el Script de la Función, se le da a ejecutar (al rayo). Y es ahí cuando se crea la Función, la cual aparecerá en la parte de Funciones de la BBDD. Una vez que tenemos esto ya podemos ejecutarla.

**EJEMPLO DE Conteo\_Mujeres\_en\_IT\_en\_tecnologia\_tipo:** Nos saldrá un recuadro como el siguiente donde deberemos colocar el Tipo de Tecnología que queremos. En este caso será uno que tenemos en la BBDD, "Programming".



Call stored function MujeresIT.Conteo\_Mujeres\_en\_IT\_en\_tecnol...

Enter values for parameters of your function and click <Execute> to create an SQL editor and run the call:

Tipo  VARCHAR(255)

Cancel Execute

Resultado de Ejecutar la Función `Conteo_Mujeres_en_IT_en_tecnologia_tipo`:

```
MujeresIT.Conteo_Mujeres_en_IT_en_tecnologia_tipo('Programming')  
▶ 4
```

Este resultado nos muestra que hay 4 MujeresIT que saben el Tipo de Tecnología: “Programming”.



- **Función Mujer\_IT\_Conoce\_Tecnologia:**

**OBJETIVO:** Esta Función surge de pensar por ejemplo si esta BBDD se convierte en un posible Directorio, y sabemos el Nombre de la MujerIT, pero no qué Tecnología maneja, nos puede servir esta Función para resolver ese problema. También podría hacerse con el Nombre y Apellido. O solamente Apellido. Esta es solamente una idea de Función que puede servir y ser mejorada a futuro.

**DESCRIPCIÓN:** Para realizar una consulta en la función Mujer\_IT\_Conoce\_Tecnologia, se debe ingresar el parámetro Nombre (de tipo VARCHAR(255)). El resultado que arroja la función es del tipo VARCHAR(255), el cual es el Nombre de la Tecnología que la MujerIT maneja. Debido a que el parámetro de entrada y el que devuelve es del mismo Tipo, colocamos **DETERMINISTIC** en la Función.

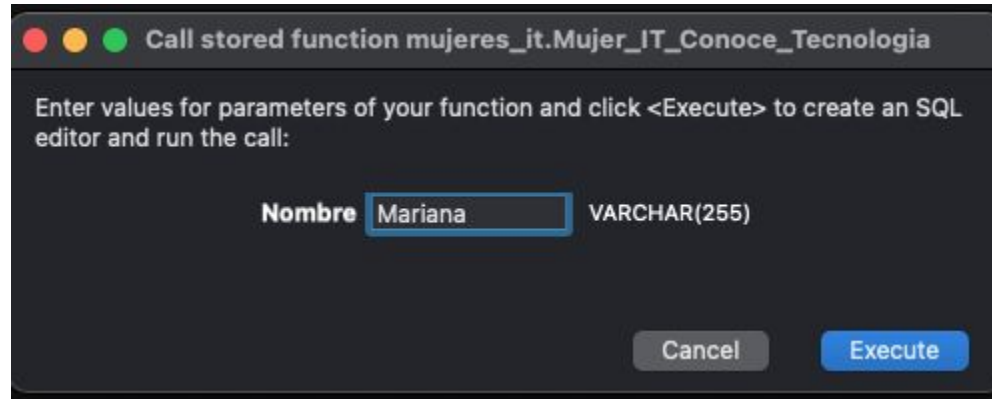
**TABLAS QUE INTERVIENEN EN LA FUNCIÓN:** Las tablas que intervienen en dicha función son DimTecnologias y FactMujeres.

## Creación del Script para la Función Mujer\_IT\_Conoce\_Tecnologia:

```
1  DELIMITER $$
2  CREATE DEFINER=`root`@`localhost` FUNCTION `Mujer_IT_Conoce_Tecnologia`(Nombre VARCHAR(255)) RETURNS varchar(255) CHARSET utf8mb4
3  DETERMINISTIC
4  BEGIN
5      DECLARE Resultado VARCHAR(255);
6      SET Resultado = (SELECT T.Nombre
7                      FROM DimTecnologias AS T
8                      INNER JOIN FactMujeres AS M
9                          ON T.ID = M.IDTecnologia
10                     WHERE M.Nombre = Nombre);
11     RETURN Resultado;
12 END$$
13 DELIMITER ;
```

**PASOS FINALES:** Una vez creado el Script de la Función, se le da a ejecutar (al rayo). Y es ahí cuando se crea la Función, la cual aparecerá en la parte de Funciones de la BBDD. Una vez que tenemos esto ya podemos ejecutarla.

**EJEMPLO DE Mujer\_IT\_Conoce\_Tecnologia:** Nos saldrá un recuadro como el siguiente donde deberemos colocar el Nombre de la MujerIT. En este caso será uno que tenemos en la BBDD, “Mariana”.



Call stored function mujeres\_it.Mujer\_IT\_Conoce\_Tecnologia

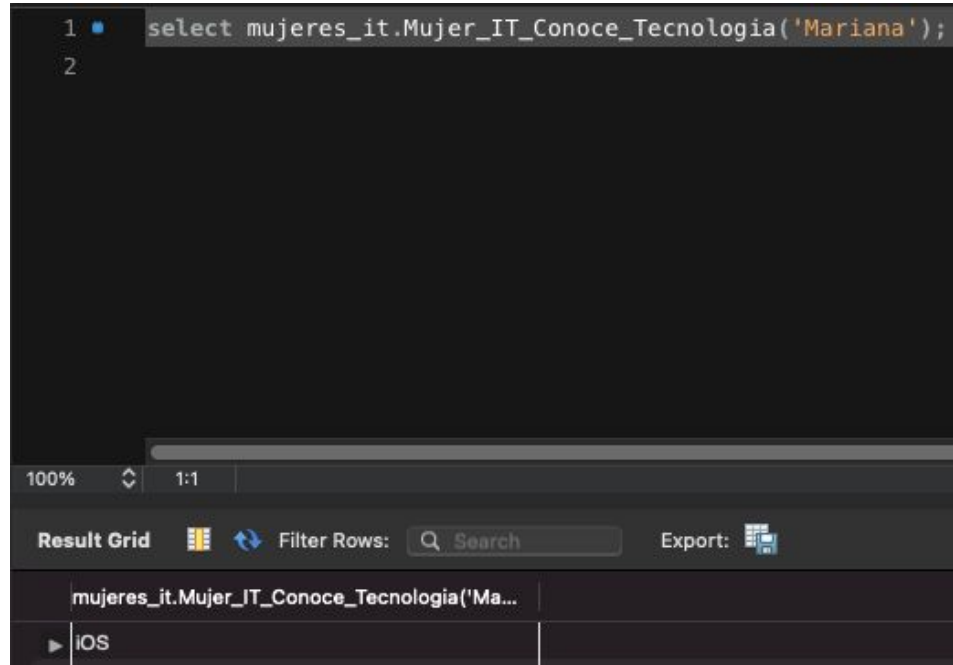
Enter values for parameters of your function and click <Execute> to create an SQL editor and run the call:

Nombre  VARCHAR(255)

Cancel Execute

## Resultado de Ejecutar la Función Mujer\_IT\_Conoce\_Tecnologia:

```
1 select mujeres_it.Mujer_IT_Conoce_Tecnologia('Mariana');  
2
```



The screenshot shows a SQL query execution interface. The query is: `select mujeres_it.Mujer_IT_Conoce_Tecnologia('Mariana');`. The interface includes a "Result Grid" section with a search bar and an "Export" button. The result is displayed in a table with one row and one column.

mujeres_it.Mujer_IT_Conoce_Tecnologia('Ma...
iOS

Este resultado nos muestra que la MujerIT, “Mariana”, tiene conocimientos en la Tecnología iOS.

# Script SQL creación de Procedimientos Almacenados:

- **PROCEDIMIENTO ALMACENADO Eliminar y Agregar Valores a DimCargos:**

**OBJETIVO:** Eliminar y agregar valores a la tabla DimCargos.

**DESCRIPCIÓN:** Para utilizar el procedimiento almacenado, se debe indicar en el primer parámetro el idCargo que se quiere borrar o agregar y en el segundo parámetro el seniority que se desea borrar o agregar, como por ejemplo: Trainee, Software Engineer, etc. El resultado que arrojará el procedimiento será o bien la agregación de un seniority o la eliminación de uno ya existente en la tabla DimCargos.

**TABLAS QUE INTERVIENEN EN EL PROCEDIMIENTO:** La tabla que interviene en este procedimiento es DimCargos.

## Creación del Script para PROCEDIMIENTO ALMACENADO Eliminar y Agregar Valores a DimCargos:

Debido a que estamos tratando con FK's es que debemos incluir en la Función el código:

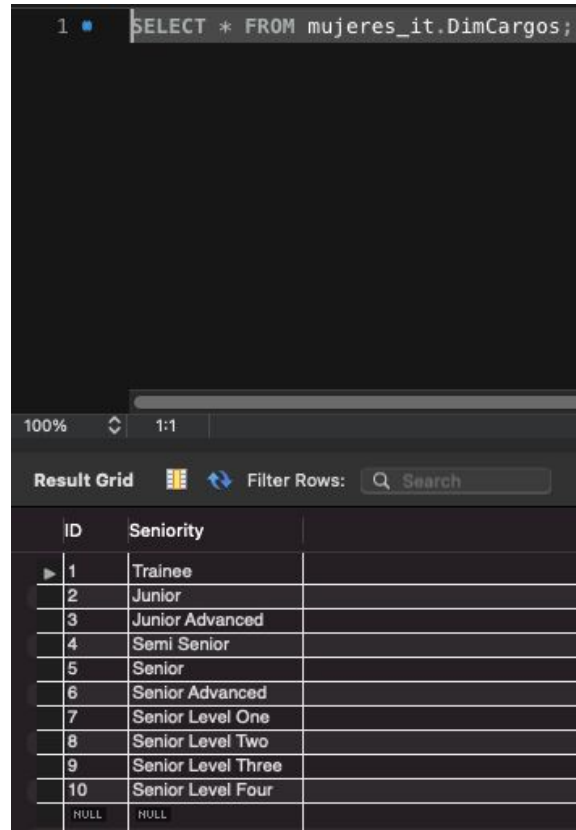
**SET FOREIGN\_KEY\_CHECKS = 0;** Si no, no nos va a dejar eliminar registros.

```
1  DELIMITER $$
2  CREATE DEFINER='root'@'localhost' PROCEDURE `Eliminar y Agregar Valores a DimCargos`(IN idcargo INT, IN seniority VARCHAR(255))
3  BEGIN
4      SET FOREIGN_KEY_CHECKS=0;
5      IF EXISTS (SELECT * FROM mujeres_it.DimCargos WHERE idcargo = ID AND seniority = seniority) THEN
6          DELETE FROM mujeres_it.DimCargos
7              WHERE idcargo = ID AND seniority = seniority;
8      ELSE
9          INSERT INTO mujeres_it.DimCargos (seniority)
10             VALUES (seniority);
11      END IF;
12  END$$
13  DELIMITER ;
```

Veamos primero cómo se ve la Tabla DimCargos (antes de utilizar el SP):

Vemos que se muestran 10

Datos



The screenshot shows a SQL query execution window. At the top, a query is entered: `SELECT * FROM mujeres_it.DimCargos;`. Below the query editor, the results are displayed in a table format. The table has three columns: ID, Seniority, and an empty column. The results show 10 rows of seniority levels, from Trainee to Senior Level Four, followed by a row with NULL values. The interface includes a search bar and a filter row button.

ID	Seniority	
1	Trainee	
2	Junior	
3	Junior Advanced	
4	Semi Senior	
5	Senior	
6	Senior Advanced	
7	Senior Level One	
8	Senior Level Two	
9	Senior Level Three	
10	Senior Level Four	
NULL	NULL	

Ejecutemos el Script para el PROCEDIMIENTO ALMACENADO Eliminar y Agregar Valores a DimCargos - Agregar:

Call stored procedure mujeres\_it.Eliminar y Agregar Valores a Di...

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

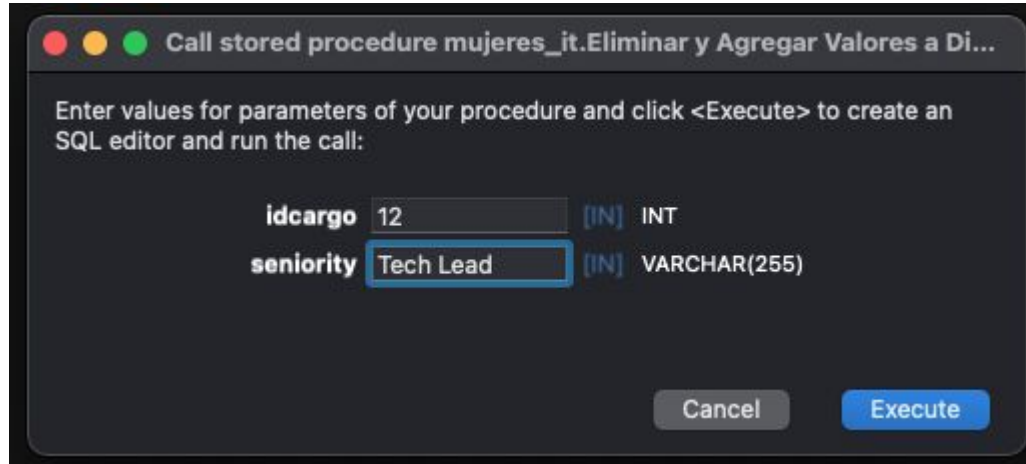
idcargo	11	[IN]	INT
seniority	chitect Designer	[IN]	VARCHAR(255)

Cancel Execute

Luego de dar Execute nos aparece lo siguiente: `call mujeres_it.`Eliminar y Agregar Valores a DimCargos`(11, 'Architect Designer');`



Agregamos otro Dato (mismo procedimiento):



Call stored procedure mujeres\_it.Eliminar y Agregar Valores a Di...

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

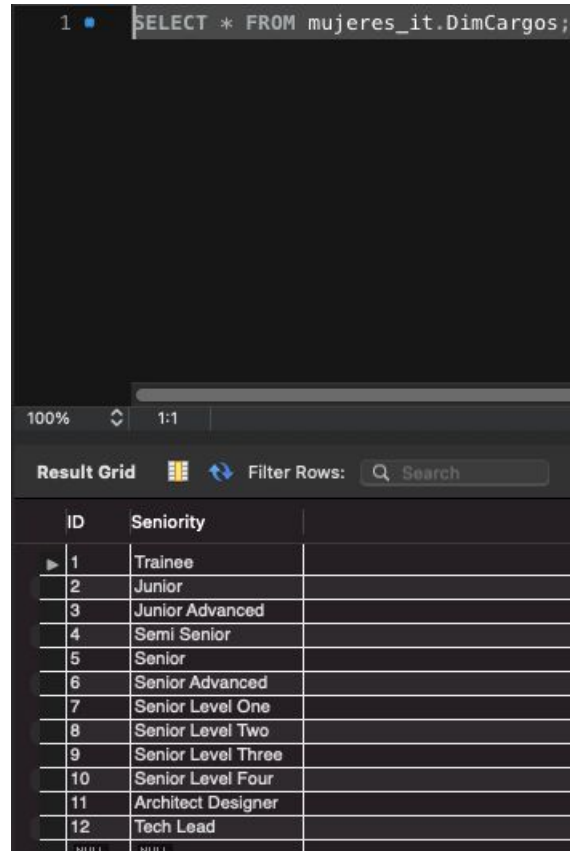
idcargo	<input type="text" value="12"/>	[IN] INT
seniority	<input type="text" value="Tech Lead"/>	[IN] VARCHAR(255)

Cancel Execute

Luego de dar Execute nos aparece lo siguiente: `call mujeres_it.`Eliminar y Agregar Valores a DimCargos`(12, 'Tech Lead');`

Finalmente el Resultado que obtenemos en la Tabla DimCargos es el siguiente:

Agregamos 2 Registros los  
cuales se ven reflejados en la  
imagen.



```
1 SELECT * FROM mujeres_it.DimCargos;
```

100% 1:1

Result Grid Filter Rows: Search

ID	Seniority
1	Trainee
2	Junior
3	Junior Advanced
4	Semi Senior
5	Senior
6	Senior Advanced
7	Senior Level One
8	Senior Level Two
9	Senior Level Three
10	Senior Level Four
11	Architect Designer
12	Tech Lead

Ejecutemos el Script para el PROCEDIMIENTO ALMACENADO Eliminar y Agregar Valores a DimCargos - Eliminar: (Recordemos que se agregaron dos datos anteriormente)

Call stored procedure mujeres\_it.Eliminar y Agregar Valores a Di...

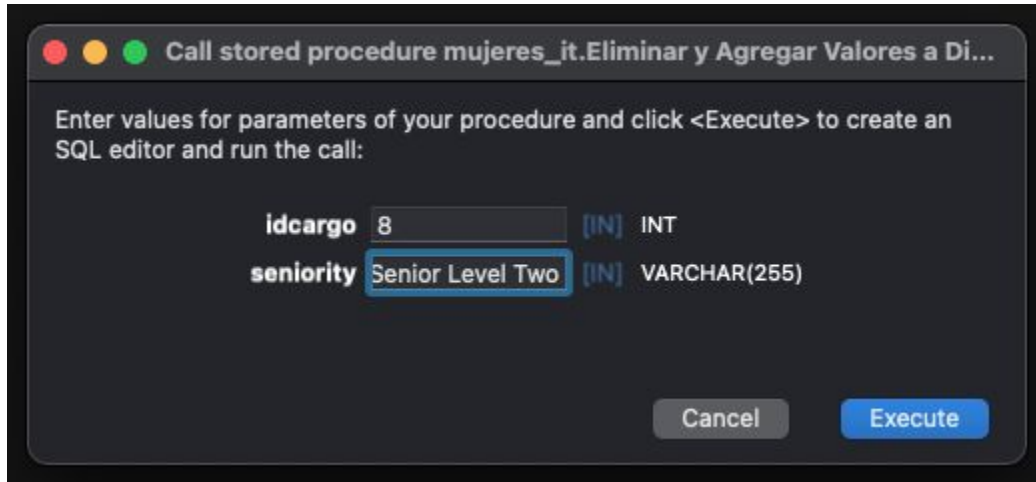
Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

idcargo	1	[IN]	INT
seniority	Trainee	[IN]	VARCHAR(255)

Cancel Execute

Luego de dar Execute nos aparece lo siguiente: `call mujeres_it.`Eliminar y Agregar Valores a DimCargos`(1, 'Trainee');`

Eliminamos otro Dato (mismo procedimiento):



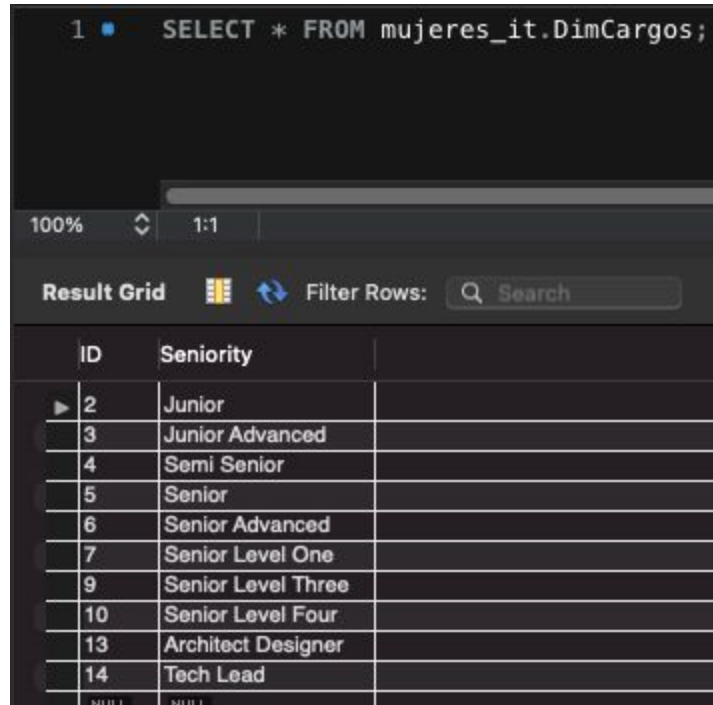
A screenshot of a database application window titled "Call stored procedure mujeres\_it.Eliminar y Agregar Valores a Di...". The window has a dark background and contains the following text: "Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:". Below this, there are two input fields. The first is labeled "idcargo" and contains the value "8", with "[IN] INT" to its right. The second is labeled "seniority" and contains the value "Senior Level Two", with "[IN] VARCHAR(255)" to its right. At the bottom right of the window, there are two buttons: "Cancel" and "Execute".

Luego de dar Execute nos aparece lo siguiente: `call mujeres_it.`Eliminar y Agregar Valores a DimCargos`(8, 'Senior Level Two');`

Finalmente el Resultado que obtenemos en la Tabla DimCargos es el siguiente:

Eliminamos el Registro 1 y 8.

Los últimos Registros se ven así, debido a que me equivoqué y tuve que eliminar los datos agregados. Y como el ID es incremental, se agregan con el valor siguiente.



ID	Seniority
2	Junior
3	Junior Advanced
4	Semi Senior
5	Senior
6	Senior Advanced
7	Senior Level One
9	Senior Level Three
10	Senior Level Four
13	Architect Designer
14	Tech Lead

- **PROCEDIMIENTO ALMACENADO** Ordenamiento sobre Tabla FactMujeres:

**OBJETIVO:** Ordenar la tabla FactMujeres, según el Campo que la persona que esta utilizando la BBDD desee.

**DESCRIPCIÓN:** Para utilizar el procedimiento almacenado, se debe indicar en el primer parámetro que es el campo por el cual se quiere ordenar la tabla FactMujeres. En el segundo parámetro se debe ingresar el tipo de ordenamiento que se quiere hacer, cómo esta realizado con un **ENUM** eso le da la pauta al usuario que esté utilizando la Base de Datos si lo quiere hacer **ASC** o **DESC**. Este procedimiento además tiene una particularidad, que es que tiene dos validaciones por si el usuario no ingresa nada en alguno de los campos descritos anteriormente.

**TABLAS QUE INTERVIENEN EN EL PROCEDIMIENTO:** La tabla que interviene en este procedimiento es FactMujeres.

## EJEMPLO PROCEDIMIENTO ALMACENADO Ordenamiento sobre Tabla FactMujeres:

Call stored procedure MujeresIT.Ordenamiento sobre Tabla Fact...

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

<b>campo</b>	<input type="text" value="Nombre"/>	[IN] VARCHAR(255)
<b>tipo_ordenamiento</b>	<input type="text" value="DESC"/>	[IN] ENUM('ASC', 'DESC')

Finalmente el Resultado que obtenemos en la Tabla FactMujeres es el siguiente:

```
1 • call mujeres_it.`Ordenamiento sobre Tabla FactMujeres`('Nombre', 'DESC');
2
```

100% 1:1

Result Grid Filter Rows: Search Export:

	ID	Nombre	Apellido	DNI	Profesion	IDCargo	IDTecnologia	
▶	1	Virginia	Lombardi	46288064	iOS Developer	1	1	
	9	Susana	Mori	34868063	C Developer	10	10	
	10	Natalia	Bonjour	65433456	C++ Developer	9	8	
	8	Micaela	Nelson	56784325	Web UI Developer	7	5	
	6	Martina	Bernardi	56873219	Android Developer	4	2	
	3	Mariana	Orman	33458065	iOS Developer	8	1	
	2	Lucia	Perez	54568907	Systems Engineering	5	4	
	4	Lorena	Alvarez	56789073	Java Developer	2	7	
	7	Isabel	Vayra	34623786	Web UI Developer	6	3	
	5	Alicia	Gonzalez	45678753	Python Developer	3	9	



# Listado de Triggers:

- PRIMEROS PASOS:

`SELECT DATABASE();` -> Esto lo que hace es mostrarnos la BBDD actualmente seleccionada o en uso.

DATABASE()	
▶	mujeres_it

`SELECT VERSION();` -> Nos muestra la Versión que se está corriendo actualmente.

VERSION()	
▶	8.0.26

- **TRIGGER Aft\_Ins\_MujerIt:**

**OBJETIVO:** Crear un trigger después de la inserción de datos.

**DESCRIPCIÓN:** Se crea primero que nada una tabla llamada **adiciones\_mujeres\_general** la cual va a guardar la información necesaria que se genere al disparar el trigger, como se puede observar en la imagen.

```
8  /* Se crea Tabla mujeres_it.adiciones_mujeres_general.
9  Se selecciona la Tabla `FactMujeres`. */
10 CREATE TABLE mujeres_it.adiciones_mujeres_general(
11     ID_adicion INT AUTO_INCREMENT PRIMARY KEY,
12     Fecha DATE NOT NULL,
13     Hora TIME NOT NULL,
14     Usuario_ID VARCHAR(255) NOT NULL,
15     ID INT,
16     Nombre VARCHAR(255),
17     Apellido VARCHAR(255),
18     Profesion VARCHAR(255),
19     Tipo_operacion VARCHAR(255));
```

Una vez realizada la tabla se procede a escribir el código del trigger. El trigger denominado Aft\_Ins\_MujerIt, lo que hace es que después de la inserción en la tabla FactMujeres se va a aplicar por cada fila insertar en la tabla adiciones\_mujeres\_general (creada en Tables\_For\_Triggers) los datos pasados: Fecha, Hora Usuario\_ID, ID, Nombre, Apellido, Profesion y Tipo\_operacion. Y en VALUES le pasamos los valores que vamos a insertar en dicha tabla. Se realiza una inserción para probar el funcionamiento de este trigger.

**TABLAS QUE INTERVIENEN:** La tabla que interviene es FactMujeres.

**BENEFICIOS:** Esto es beneficioso para por ejemplo cuando un usuario X agrega datos a la tabla FactMujeres, para saber qué datos relevantes se agregaron y la información del usuario así como la fecha y la hora en que se realizaron estos cambios.

## Creación del Script para el TRIGGER Aft\_Ins\_MujerIt:

```
12  CREATE TRIGGER `Aft_Ins_MujerIt`  
13  AFTER INSERT ON `FactMujeres` FOR EACH ROW  
14  INSERT INTO `adiciones_mujeres_general` (Fecha, Hora, Usuario_ID, ID, Nombre, Apellido, Profesion, Tipo_operacion)  
15  VALUES(CURRENT_DATE(), CURRENT_TIME(), USER(), NEW.ID, NEW.Nombre, NEW.Apellido, NEW.Profesion, 'INSERT');  
16  
17  #INSERTAMOS VALORES EN LA TABLA FactMujeres  
18  INSERT INTO mujeres_it.FactMujeres (Nombre, Apellido, DNI, Profesion, IDCargo, IDTecnologia)  
19  VALUES('Siri', 'Altez', 43211285, 'iOS Developer', 2, 1);  
20
```

## EJEMPLO DEL TRIGGER Aft\_Ins\_MujerIt:

```
16
17 #INSERTAMOS VALORES EN LA TABLA FactMujeres
18 • INSERT INTO mujeres_it.FactMujeres (Nombre, Apellido, DNI, Profesion, IDCargo, IDTecnologia)
19 VALUES('Siri', 'Altez', 43211285, 'iOS Developer', 2, 1);
20
```

Como Resultado Obtenemos en la Tabla mujeres\_it.adiciones\_mujeres\_general:

[illegible]

- **TRIGGER Bef\_Del\_MujerIt:**

**OBJETIVO:** Crear un trigger antes de la eliminación de datos.

**DESCRIPCIÓN:** Se crea primero que nada una tabla llamada **eliminaciones\_mujeres\_general** la cual va a guardar la información necesaria que se genere al disparar el trigger, como se puede observar en la imagen.

```
23 ● ⊖ CREATE TABLE mujeres_it.eliminaciones_mujeres_general(  
24     ID_eliminacion INT AUTO_INCREMENT PRIMARY KEY,  
25     Fecha DATE NOT NULL,  
26     Hora TIME NOT NULL,  
27     Usuario_ID VARCHAR(255) NOT NULL,  
28     ID INT,  
29     Nombre VARCHAR(255),  
30     Apellido VARCHAR(255),  
31     Profesion VARCHAR(255),  
32     Tipo_operacion VARCHAR(255));  
33
```

Una vez realizada la tabla se procede a escribir el código del trigger. El trigger denominado Bef\_Del\_MujerIt, lo que hace es que antes de la eliminación en la tabla FactMujeres se va a aplicar por cada fila insertar en la tabla eliminaciones\_\_mujeres\_\_general (creada en Tables\_\_For\_\_Triggers) los datos pasados: Fecha, Hora Usuario\_ID, ID, Nombre, Apellido, Profesion y Tipo\_operacion. Y en VALUES le pasamos los valores que vamos a insertar en dicha tabla. Se realiza una eliminación para probar el funcionamiento de este trigger.

**TABLAS QUE INTERVIENEN:** La tabla que interviene es FactMujeres.

**BENEFICIOS:** Esto es beneficioso para por ejemplo cuando un usuario X elimina datos de la tabla FactMujeres, para saber qué datos relevantes se eliminaron y la información del usuario así como la fecha y la hora en que se realizaron estos cambios.

## Creación del Script para el TRIGGER Bef\_Del\_MujerIt:

```
27
28 • CREATE TRIGGER `Bef_Del_MujerIt`
29   BEFORE DELETE ON `FactMujeres` FOR EACH ROW
30   INSERT INTO `eliminaciones_mujeres_general` (Fecha, Hora, Usuario_ID, ID, Nombre, Apellido, Profesion, Tipo_operacion)
31   VALUES(CURRENT_DATE(), CURRENT_TIME(), USER(), OLD.ID, OLD.Nombre, OLD.Apellido, OLD.Profesion, 'DELETE');
32
```



## EJEMPLO DEL TRIGGER Bef\_Del\_MujerIt:

```
33
34     #USAMOS ESTO PARA NO OBTENER PROBLEMAS CON LA ELIMINACIÓN DEL ID
35     SET FOREIGN_KEY_CHECKS=0;
36     #BORRAMOS UNA FILA DE LA TABLA DimTecnologias DONDE EL ID ES DIEZ
37     DELETE FROM mujeres_it.FactMujeres
38     WHERE ID = 10;
39
```

Como Resultado Obtenemos en la Tabla mujeres\_it.eliminaciones\_mujeres\_general:

Result Grid									
Filter Rows:		Search			Edit:		Export/Import:		
ID_eliminacion	Fecha	Hora	Usuario_ID	ID	Nombre	Apellido	Profesion	Tipo_operacion	
1	2021-12-21	15:35:45	root@localhost	10	Natalia	Bonjour	C++ Developer	DELETE	
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	

- **TRIGGER Bef\_Del\_Tecnologia:**

**OBJETIVO:** Crear un trigger antes de la eliminación de datos de la tabla.

**DESCRIPCIÓN:** Se crea primero que nada una tabla llamada **eliminaciones\_tecnologias\_general** la cual va a guardar la información necesaria que se genere al disparar el trigger, como se puede observar en la imagen.

```
• ⊖ CREATE TABLE MujeresIT.eliminaciones_tecnologias_general(  
    ID_eliminacion INT AUTO_INCREMENT PRIMARY KEY,  
    Fecha DATE NOT NULL,  
    Hora TIME NOT NULL,  
    Usuario_ID VARCHAR(255) NOT NULL,  
    ID INT,  
    Nombre VARCHAR(255),  
    Tipo VARCHAR(255),  
    Tipo_operacion VARCHAR(255));
```

Una vez realizada la tabla se procede a escribir el código del trigger. El trigger denominado `Bef_Del_Tecnologia`, lo que hace es que antes de la eliminación en la tabla `DimTecnologias` se va a aplicar por cada fila insertar en la tabla `eliminaciones_tecnologias_general` (creada en `Tables_For_Triggers`) los datos pasados: `Fecha Hora Usuario_ID`, `ID`, `Nombre`, `Tipo` y `Tipo_operacion`. Y en `VALUES` le pasamos los valores que vamos a insertar en dicha tabla. Se realiza una eliminación para probar el funcionamiento de este trigger. La particularidad que tiene este trigger es que para que no tire un error se debe utilizar el siguiente trozo de código: `SET FOREIGN_KEY_CHECKS=0;`. Es más que nada para que no haya problemas con las FK, y se pueda realizar la eliminación sin problemas.

**TABLAS QUE INTERVIENEN:** La tabla que interviene es `DimTecnologias`.

**BENEFICIOS:** Encuentro que este trigger es muy beneficioso para cuando un usuario X elimina datos de la tabla `DimTecnologias`, porque esto deja registrado quien, cuándo y qué datos fueron eliminados con respecto a la misma.

## Creación del Script para el TRIGGER Bef\_Del\_Tecnologia:

```
48 CREATE TRIGGER `Bef_Del_Tecnologia`  
49 BEFORE DELETE ON `DimTecnologias` FOR EACH ROW  
50 INSERT INTO `eliminaciones_tecnologias_general` (Fecha, Hora, Usuario_ID, ID, Nombre, Tipo, Tipo_operacion)  
51 VALUES(CURRENT_DATE(), CURRENT_TIME(), USER(), OLD.ID, OLD.Nombre, OLD.Tipo, 'DELETE');  
52
```



- **TRIGGER Aft\_Ins\_Tecnologia:**

**OBJETIVO:** Crear un trigger después de la inserción de datos de la tabla.

**DESCRIPCIÓN:** Se crea primero que nada una tabla llamada **agregaciones\_tecnologias\_general** la cual va a guardar la información necesaria que se genere al disparar el trigger, como se puede observar en la imagen.

```
52 • ⊖ CREATE TABLE mujeres_it.agregaciones_tecnologias_general(  
53     ID Eliminacion INT AUTO_INCREMENT PRIMARY KEY,  
54     Fecha DATE NOT NULL,  
55     Hora TIME NOT NULL,  
56     Usuario_ID VARCHAR(255) NOT NULL,  
57     ID INT,  
58     Nombre VARCHAR(255),  
59     Tipo VARCHAR(255),  
60     Tipo_operacion VARCHAR(255));
```

Una vez realizada la tabla se procede a escribir el código del trigger. El trigger denominado Aft\_Ins\_Tecnologia, lo que hace es que antes de la inserción de datos en la tabla FactMujeres se va a aplicar por cada fila insertar en la tabla agregaciones\_tecnologias\_general (creada en Tables\_For\_Triggers) los datos pasados: Fecha, Hora Usuario\_ID, ID, Nombre, Tipo y Tipo\_operacion. Y en VALUES le pasamos los valores que vamos a insertar en dicha tabla. Se realiza una inserción para probar el funcionamiento de este trigger.

**TABLAS QUE INTERVIENEN:** La tabla que interviene es DimTecnologias.

**BENEFICIOS:** Esto es beneficioso para por ejemplo cuando un usuario X agrega datos a la tabla DimTecnologias, para saber qué datos relevantes se agregaron y la información del usuario así como la fecha y la hora en que se realizaron estos cambios.

## Creación del Script para el TRIGGER Aft\_Ins\_Tecnologia:

```
65 • CREATE TRIGGER `Aft_Ins_Tecnologia`  
66 AFTER INSERT ON `DimTecnologias` FOR EACH ROW  
67 INSERT INTO `agregaciones_tecnologias_general` (Fecha, Hora, Usuario_ID, ID, Nombre, Tipo, Tipo_operacion)  
68 VALUES(CURRENT_DATE(), CURRENT_TIME(), USER(), NEW.ID, NEW.Nombre, NEW.Tipo, 'INSERT');  
69
```



## EJEMPLO DEL TRIGGER Aft\_Ins\_Tecnologia:

```
69
70 #INSERTAMOS VALORES EN LA TABLA DimTecnologias
71 • INSERT INTO mujeres_it.DimTecnologias (Nombre, Tipo)
72 VALUES('Angular', 'Web UI');
73
```

Como Resultado Obtenemos en la Tabla mujeres\_it.agregaciones\_tecnologias\_general:

Result Grid

Filter Rows:

Search

Edit:

Export/Import

	ID_eliminacion	Fecha	Hora	Usuario_ID	ID	Nombre	Tipo	Tipo_operacion
<div><div></div><div></div></div>	1	2021-12-21	15:35:45	root@localhost	11	Angular	Web UI	INSERT
<div><div></div><div></div></div>	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

# Creación de Users:

- **PRIMER PASO:** Creación de Users `virginia@localhost` y `mariana@localhost`, como bien se muestra se decide hacerlo con dominio de `localhost` (es decir de forma local). Como por esta vez no se pedía en la entrega, se crearon los usuarios sin ninguna contraseña.
- **SEGUNDO PASO:** Verificar que realmente se crearon los Users detallados anteriormente.
- Las sentencias utilizadas son las siguientes:

```
1  #CREACIÓN DE USERS
2
3  /* Creamos los users que se piden en la entrega esta vez como no es necesario se hacen sin contraseña. */
4  ■ CREATE USER virginia@localhost;
5  ■ CREATE USER mariana@localhost;
6
7  #COMPROBAMOS QUE LOS USER FUERON CREADOS CON ÉXITO
8  /* Para corroborar que los users fueron creados implementamos la siguiente sentencia. */
9  ■ SELECT * FROM mysql.user;
```

El Resultado obtenido es el siguiente:

	Host	User
▶	localhost	mariana
	localhost	mysql.infoschema
	localhost	mysql.session
	localhost	mysql.sys
	localhost	root
	localhost	virginia

- **DAR PERMISOS A LOS USERS CREADOS:**

**OBJETIVO:** Como bien dice el título se trata de crear permisos para cada uno de los usuarios creados anteriormente.

**PRIMER PERMISO OTORGADO A virginia@localhost:** Se otorgan permisos de Lectura para este User de la siguiente manera: **GRANT SELECT ON mujeres\_it.\* TO virginia@localhost;**. Lo que estamos realizando con esta sentencia de GRANT SELECT, es justamente eso, darle permisos de Lectura al usuario virginia@localhost. En la sentencia indicamos el nombre de la BBDD sobre la cual queremos que este User tenga este permiso, en este caso es nuestra BBDD mujeres\_it.

**VERIFICACIÓN:** Con la siguiente sentencia **SHOW GRANTS FOR virginia@localhost;** verificamos si efectivamente se creó el permiso de Lectura.

En las siguientes imágenes vemos las sentencias utilizadas y el resultado obtenido:

```
13 #DAR PERMISOS DE LECTURA AL USER virginia@localhost
14 /* Se nos pide que el primer user o sea virginia@localhost tenga sólo permisos de lectura, escribimos la siguiente sentencia de GRANT para darle
15 permisos de lectura e indicamos el nombre de la base de datos que queremos que tenga los permisos. En nuestro caso será la BBDD mujeres_it. */
16 GRANT SELECT ON mujeres_it.* TO virginia@localhost;
17
18 #MOSTRAR LOS PRIVILEGIOS PARA EL USER virginia@localhost
19 /* Con esta sentencia verificamos que efectivamente los permisos que se le dan a virginia@localhost son de lectura. */
20 SHOW GRANTS FOR virginia@localhost;
```

Grants for virginia@localhost	
▶	GRANT USAGE ON *.* TO `virginia`@`localhost`
▶	GRANT SELECT ON `mujeres_it`.* TO `virginia`@`localhost`

**PRIMEROS PERMISOS OTORGADOS A mariana@localhost:** Se otorgan permisos de Lectura, Inserción y Modificación para este User de la siguiente manera: **GRANT SELECT, INSERT, UPDATE ON mujeres\_it.\* TO mariana@localhost;**. Con esta sentencia estamos otorgando al User mariana@localhost esos permisos anteriormente mencionados al principio. Al igual que en el anterior caso también se indica el nombre de la BBDD sobre la cual queremos que este User tenga dichos permisos (en nuestro caso será mujeres\_it).

**VERIFICACIÓN:** Con la siguiente sentencia **SHOW GRANTS FOR mariana@localhost;** verificamos si efectivamente se crearon los permisos de Lectura, Inserción y Modificación.

La siguiente imagen muestra las sentencias mencionadas:

```
24 #DAR PERMISOS DE LECTURA, INSERCIÓN Y MODIFICACIÓN AL USER mariana@localhost
25 /* Se nos pide dar al segundo user mariana@localhost, permisos de lectura, inserción y modificación, por lo que escribimos la siguiente sentencia
26 de GRANT para darle dichos permisos, indicando siempre el nombre de la base de datos que queremos que tenga los permisos.
27 En nuestro caso será la BBDD mujeres_it. */
28 ■ GRANT SELECT, INSERT, UPDATE ON mujeres_it.* TO mariana@localhost;
29
30 #MOSTRAR LOS PRIVILEGIOS PARA EL SEGUNDO USER mariana@localhost;
31 /* Con este script verificamos que efectivamente los permisos que se le dan a mariana@localhost; son de lectura, inserción y modificación. */
32 ■ SHOW GRANTS FOR mariana@localhost;
```

En la siguiente imagen vemos el resultado obtenido:

Grants for mariana@localhost	
▶	GRANT USAGE ON *.* TO `mariana`@`localhost`
▶	GRANT SELECT, INSERT, UPDATE ON `mujeres_it`.* TO `mariana`@`localhost`
▶	
▶	

**SENTENCIAS DE REVOKE PARA AMBOS USERS:** Sólo a modo de prueba se crean dos sentencias de REVOKE, para ambos Users. Debido a que no tienen derechos de Eliminación no se van a ver reflejados en ningún lado. Pero como método de prueba parecía interesante hacerlo.

La siguiente imagen muestra las sentencias utilizadas para este caso de REVOKE:

```
36  # Ninguno de ellos podrá eliminar registros de ninguna tabla.
37  /* Creamos una sentencia de REVOKE para ambos users así no podrán eliminar en ningún caso ningún dato de la BBDD mujeres_it.
38  Esto es simplemente para asegurarnos de que no puedan eliminar nada. (A modo de prueba) */
39
40  #PARA EL PRIMER USER
41  REVOKE DELETE ON mujeres_it.* FROM virginia@localhost;
42
43  #PARA EL SEGUNDO USER
44  REVOKE DELETE ON mujeres_it.* FROM mariana@localhost;
```



# Sentencias Del Sublenguaje TCL:

- **OBJETIVO:** Elegir dos tablas de mi BBDD mujeres\_it. Realizar una serie de modificaciones en los registros, controladas por transacciones.
- **PRIMERA PARTE:** En la **Primera Tabla**, la cual es **DimCargos**, como la misma tiene registros, se deben eliminar algunos de ellos iniciando previamente una transacción. Se debe dejar en una línea siguiente, comentado la sentencia Rollback, y en una línea posterior, la sentencia Commit.
  - **PRIMER PASO:** Para comenzar cualquier transacción debemos escribir la siguiente sentencia: **START TRANSACTION;**
  - **SEGUNDO PASO:** Escribimos la sentencia **SET FOREIGN\_KEY\_CHECKS=0;**, para no tener problemas a la hora de realizar la eliminación con las FK.

- **TERCER PASO:** Realizar la eliminación de algunos registros que ya tenemos en nuestra BBDD mujeres\_it, en nuestra tabla DimCargos. Para ello usamos la siguiente sentencia:

**DELETE FROM mujeres\_it.DimCargos**

**WHERE ID = nº del ID;**

De esta forma vamos variando los ID's y así borrando registros de los diferentes Cargos. Se terminan **borrando 5 registros** de la tabla **DimCargos**.

- **CUARTO PASO:** Verificar que esos registros recién eliminados, efectivamente se hayan borrado (ordenando los registros por el ID) con la siguiente sentencia:

**SELECT \***

**FROM mujeres\_it.DimCargos**

**ORDER BY ID;**

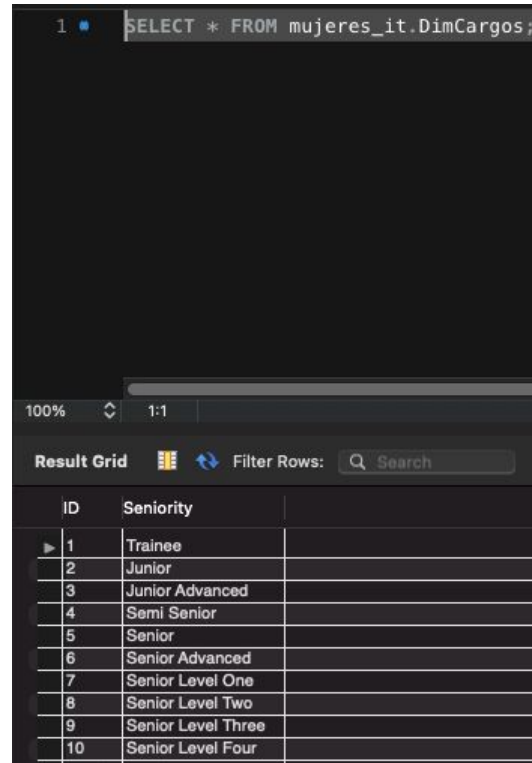
- **ÚLTIMOS PASOS:** Para terminar debemos dejar comentadas las sentencias **ROLLBACK;** y **COMMIT;.**

En la siguiente imagen vemos lo realizado y explicado en los pasos anteriores:

```
9
10  #Comenzamos con un START TRANSACTION
11  ■ START TRANSACTION;
12
13  #PARA NO TENER PROBLEMAS CON LAS FK Y SE PUEDA REALIZAR LA ELIMINACIÓN SIN PROBLEMAS UTILIZAMOS:
14  ■ SET FOREIGN_KEY_CHECKS=0;
15
16  #TABLA A ELEGIR DimCargos, REALIZAMOS LA ELIMINACIÓN DE ALGUNOS DATOS DE LA MISMA
17  ■ DELETE FROM mujeres_it.DimCargos
18      WHERE ID = 2;
19
20  ■ DELETE FROM mujeres_it.DimCargos
21      WHERE ID = 4;
22
23  ■ DELETE FROM mujeres_it.DimCargos
24      WHERE ID = 6;
25
26  ■ DELETE FROM mujeres_it.DimCargos
27      WHERE ID = 8;
28
29  ■ DELETE FROM mujeres_it.DimCargos
30      WHERE ID = 10;
31
32  #NOS FIJAMOS QUE ESTÉN EFECTIVAMENTE ELIMINADOS LOS DATOS CON LA SIGUIENTE SENTENCIA
33  ■ SELECT *
34      FROM mujeres_it.DimCargos
35      ORDER BY ID;
36
37  # HACEMOS UN ROLLBACK QUE DEJAMOS COMENTADO COMO LO PIDE LA CONSIGNA Y AL IGUAL QUE EL COMMIT
38  #ROLLBACK;
39  #COMMIT;
```

La siguiente imagen muestra el antes de la tabla DimCargos:

Podemos observar como existen  
10 registros en la misma.



```
1 SELECT * FROM mujeres_it.DimCargos;
```

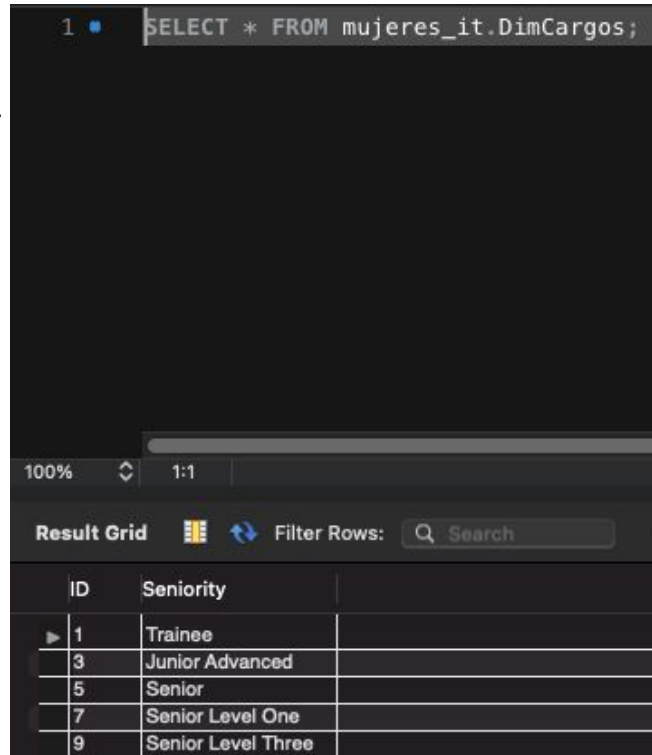
100% 1:1

Result Grid Filter Rows: Search

ID	Seniority
1	Trainee
2	Junior
3	Junior Advanced
4	Semi Senior
5	Senior
6	Senior Advanced
7	Senior Level One
8	Senior Level Two
9	Senior Level Three
10	Senior Level Four

La siguiente imagen muestra el después de la tabla DimCargos al realizar los pasos anteriores:

Y ahora observamos que  
solamente quedan 5 registros.



The screenshot shows a SQL query execution interface. At the top, a query editor displays the command: `SELECT * FROM mujeres_it.DimCargos;`. Below the editor, a toolbar shows a zoom level of 100% and a 1:1 aspect ratio. The main area displays the results in a 'Result Grid' format. The grid has two columns: 'ID' and 'Seniority'. There are five rows of data, each with a play button icon in the first column. The data rows are: (1, Trainee), (3, Junior Advanced), (5, Senior), (7, Senior Level One), and (9, Senior Level Three).

ID	Seniority
1	Trainee
3	Junior Advanced
5	Senior
7	Senior Level One
9	Senior Level Three

- **SEGUNDA PARTE:** En la **segunda tabla**, la cual es **DimTecnologias**, insertamos 8 nuevos registros iniciando también una transacción. Agregaremos un savepoint a posteriori de la inserción del registro **#4** y otro savepoint a posteriori del registro **#8**. También debemos agregar en una línea comentada la sentencia de eliminación del savepoint de los primeros 4 registros insertados
  - **PRIMER PASO:** Para comenzar cualquier transacción debemos escribir la siguiente sentencia: **START TRANSACTION;**
  - **SEGUNDO PASO:** Realizamos la inserción de los **primeros 4 registros**, debido a que hay que colocar un **SAVEPOINT** a posteriori de la inserción del registro **#4**.
  - **TERCER PASO:** Colocar la sentencia de **SAVEPOINT + nombre;**, en nuestro caso es **SAVEPOINT numero\_cuatro;**. El nombre se debe a lo explicado en el **SEGUNDO PASO**.
  - **CUARTO PASO:** Realizamos la inserción de los **últimos 4 registros**.
  - **QUINTO PASO:** Colocamos la sentencia del **SAVEPOINT**: **SAVEPOINT numero\_ocho;**, el nombre se debe a que hay que agregar un **SAVEPOINT** a posteriori del registro **#8**.

- **SEXTO PASO:** Para verificar que los 8 registros efectivamente se agregaron realizamos:

**SELECT \* FROM mujeres\_it.DimTecnologias;**

- **ÚLTIMOS PASOS:** Escribimos como comentario la sentencia para eliminar el SAVEPOINT de los primeros 4 registros insertados: **RELEASE SAVEPOINT numero\_cuatro;**

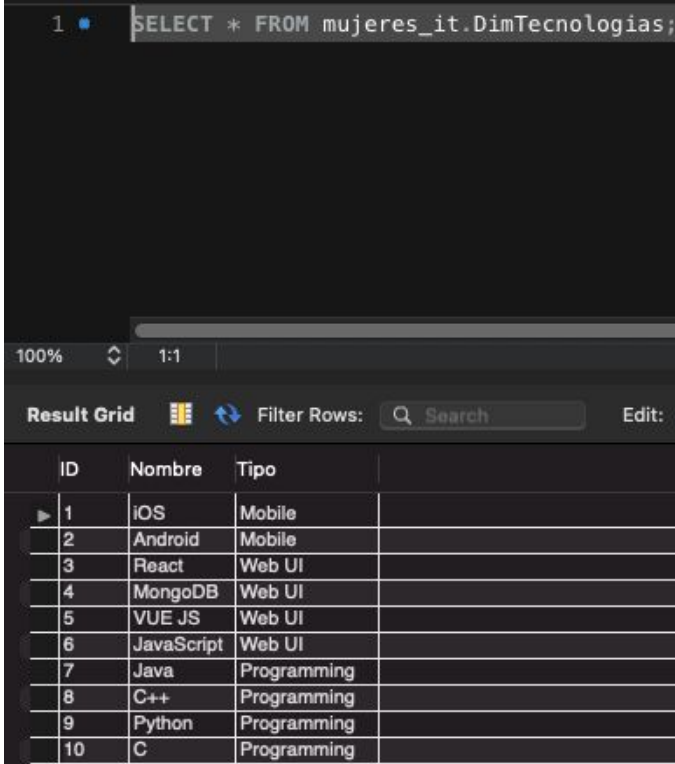


En las siguiente imagen vemos lo realizado y explicado en los pasos anteriores:

```
42
43 #SEGUNDA PARTE
44 /*En la segunda tabla, inserta ocho nuevos registros iniciando también una transacción.
45   Agrega un savepoint a posteriori de la inserción del registro #4 y otro savepoint a posteriori del registro #8
46   Agrega en una línea comentada la sentencia de eliminación del savepoint de los primeros 4 registros insertados */
47
48 #Comenzamos con un START TRANSACTION
49 ■ START TRANSACTION;
50
51 #TABLA A ELEGIR DimTecnologias, REALIZAMOS LA INSERCIÓN DE 8 NUEVOS REGISTROS
52 ■ INSERT INTO mujeres_it.DimTecnologias (ID, Nombre, Tipo)
53   VALUES (NULL, 'Angular', 'Web UI'), (NULL, 'NodeJS', 'Web UI'), (NULL, 'Laravel', 'Web UI'), (NULL, 'CSS', 'Web UI');
54 ■ SAVEPOINT numero_cuatro;
55 ■ INSERT INTO mujeres_it.DimTecnologias (ID, Nombre, Tipo)
56   VALUES (NULL, 'HTML', 'Web UI'), (NULL, 'MySQL', 'Programming'), (NULL, 'Salesforce', 'Programming'), (NULL, 'C#', 'Programming');
57 ■ SAVEPOINT numero_ocho;
58
59 #HACEMOS UNA CONSULTA A LA TABLA PARA VER LOS REGISTROS
60 ■ SELECT * FROM mujeres_it.DimTecnologias;
61
62 #SENTENCIA DE ELIMINACIÓN DEL SAVEPOINT DE LOS PRIMEROS 4 REGISTROS INSERTADOS
63 #RELEASE SAVEPOINT numero_cuatro;
```

La siguiente imagen muestra el antes de la tabla DimTecnologias:

Podemos observar como existen  
10 registros en la misma.



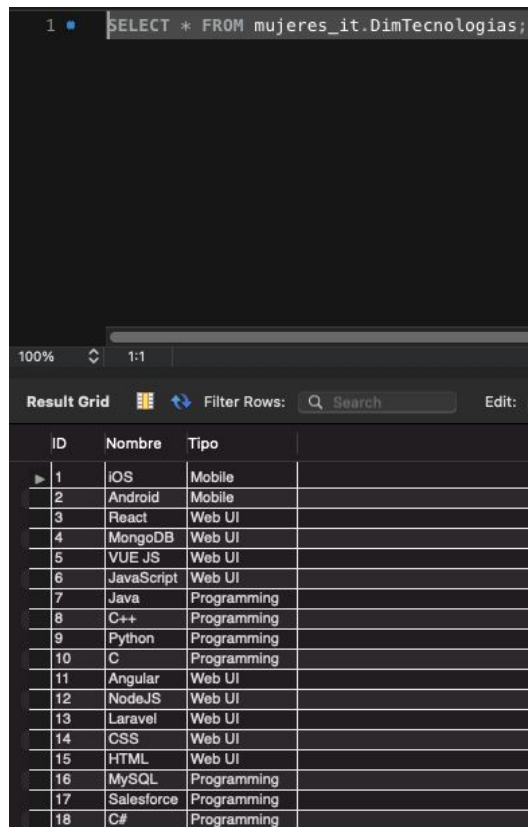
```
1 SELECT * FROM mujeres_it.DimTecnologias;
```

ID	Nombre	Tipo
1	iOS	Mobile
2	Android	Mobile
3	React	Web UI
4	MongoDB	Web UI
5	VUE JS	Web UI
6	JavaScript	Web UI
7	Java	Programming
8	C++	Programming
9	Python	Programming
10	C	Programming

La siguiente imagen muestra el después de la tabla DimTecnologias al realizar los pasos anteriores:

Y ahora observamos que

tenemos 18 registros.



```
1 SELECT * FROM mujeres_it.DimTecnologias;
```

ID	Nombre	Tipo
1	iOS	Mobile
2	Android	Mobile
3	React	Web UI
4	MongoDB	Web UI
5	VUE.JS	Web UI
6	JavaScript	Web UI
7	Java	Programming
8	C++	Programming
9	Python	Programming
10	C	Programming
11	Angular	Web UI
12	NodeJS	Web UI
13	Laravel	Web UI
14	CSS	Web UI
15	HTML	Web UI
16	MySQL	Programming
17	Salesforce	Programming
18	C#	Programming

# Backup y Restauración:

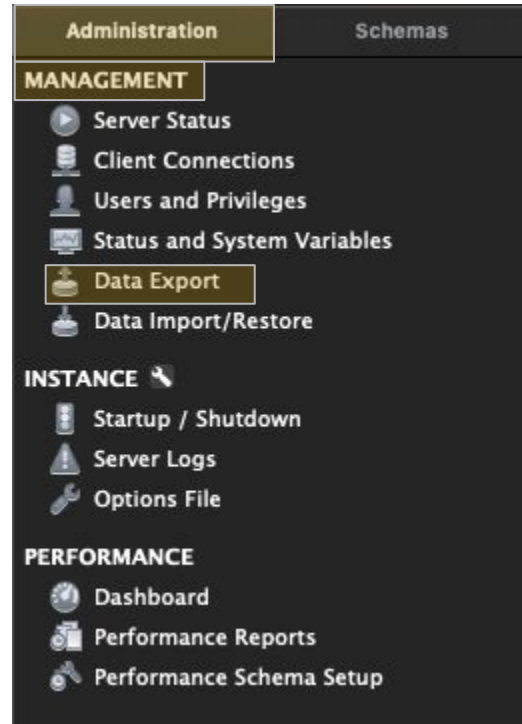
- **OBJETIVO:** Generar un Backup de la BBDD mujeres\_it, incluyendo en éste solamente las tablas. El backup debe incluir sólo los datos, dejando de lado su estructura. No se incluye debido a que ya tenemos la BBDD creada cuando se hace el Backup. También realizar la Importación o Restauración de esos Datos salvados.

## Backup:

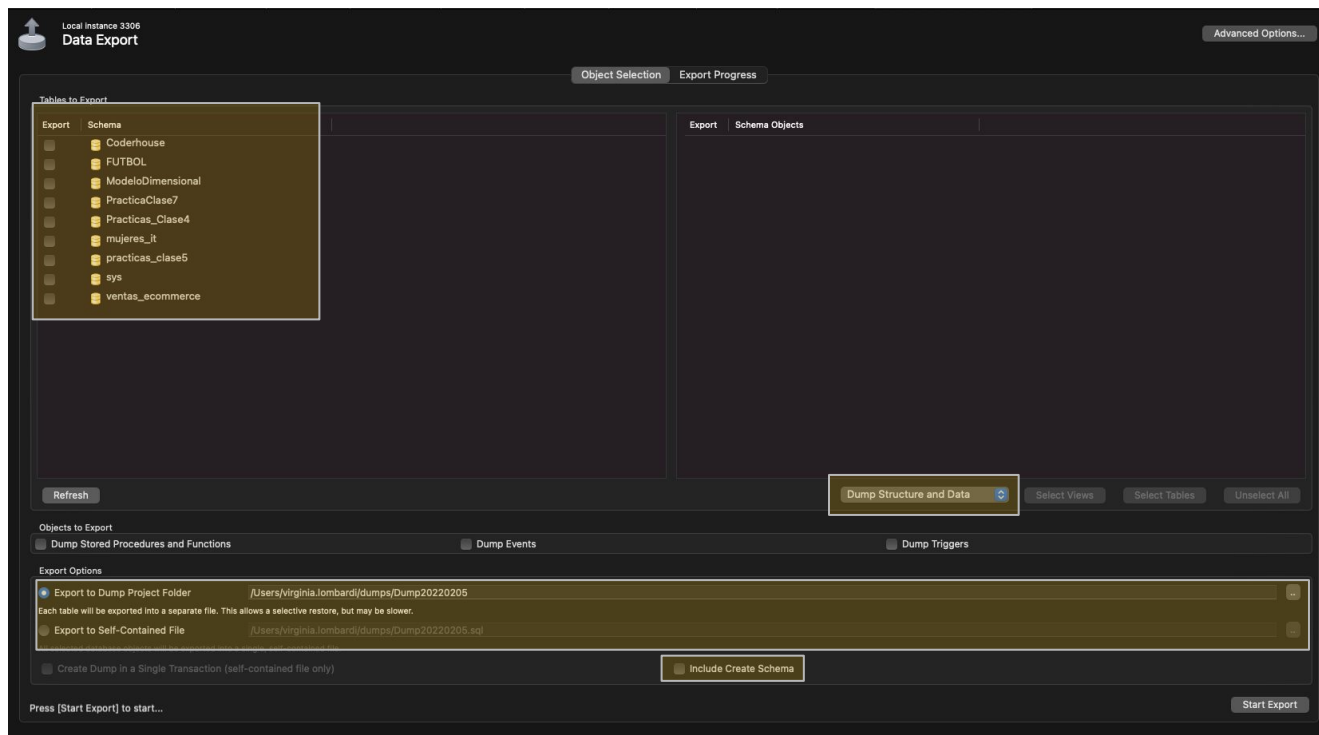
- **PRIMER PASO:** Para realizar el Backup de nuestra BBDD, lo que hacemos primero que nada es ir a la pestaña de Administración (en la parte izquierda al lado de Schemas).
- **SEGUNDO PASO:** Nos dirigimos a la parte en la que dice MANAGEMENT, ahí le damos click en donde dice "Data Export".

Como se muestra en la siguiente imagen.

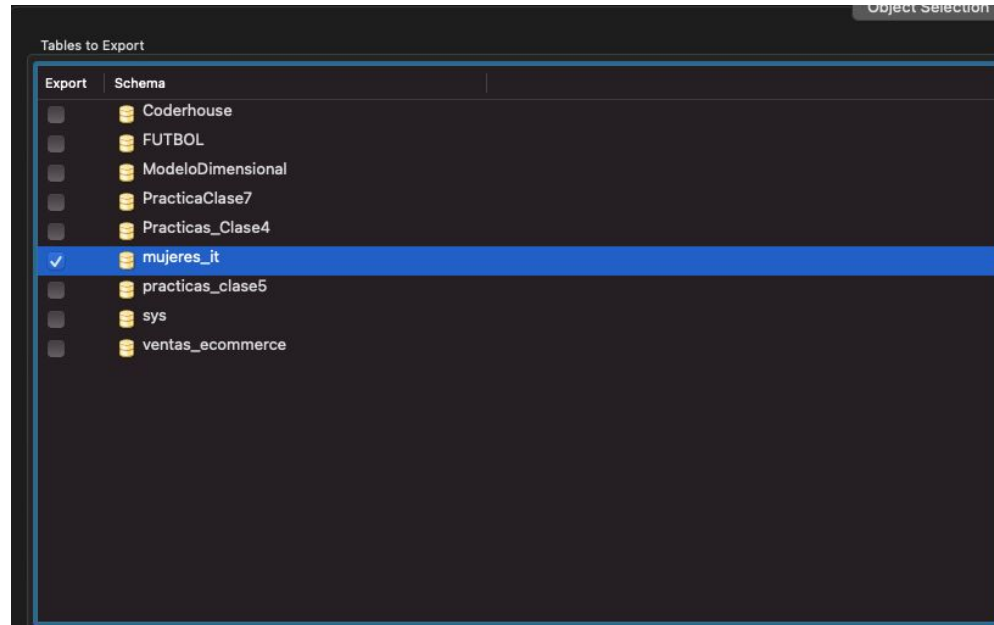
Aquí podemos ver la pantalla y los pasos anteriores.



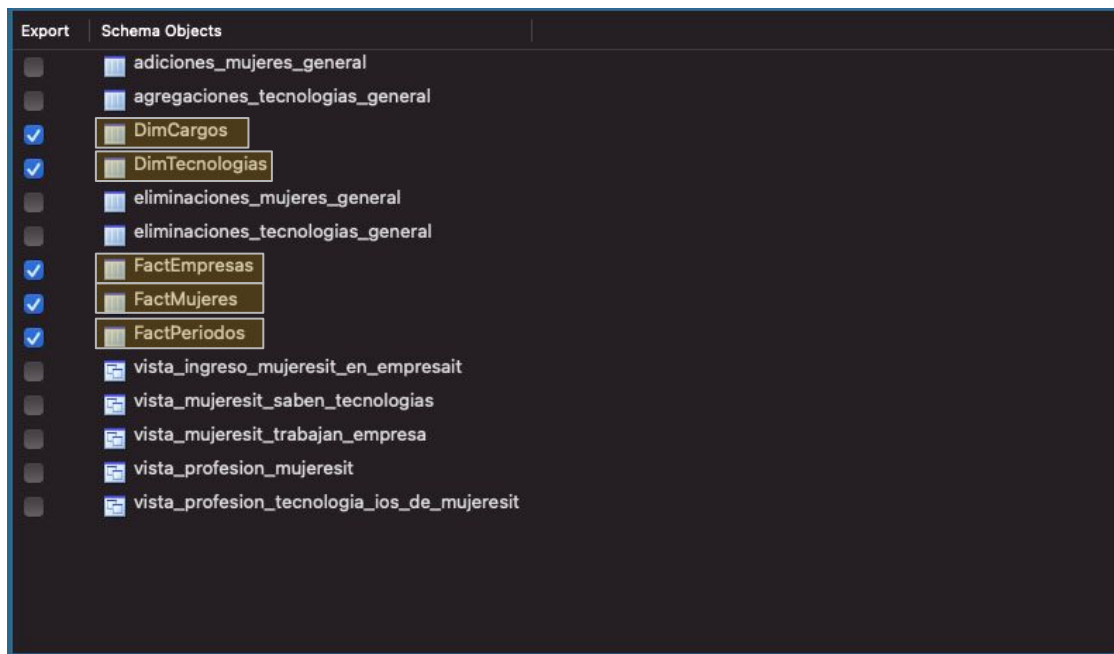
- **TERCER PASO:** Se nos abrirá la siguiente ventana, conteniendo todos los Schemas que tenemos en ese momento, y una serie de opciones para Exportar los Datos. Están **destacados** los más **importantes**.



- **CUARTO PASO:** En esta parte elegimos el Schema que queremos Exportar o al cual queremos hacerle el Backup. En nuestro caso es la BBDD de mujeres\_it. Así que simplemente la seleccionamos.

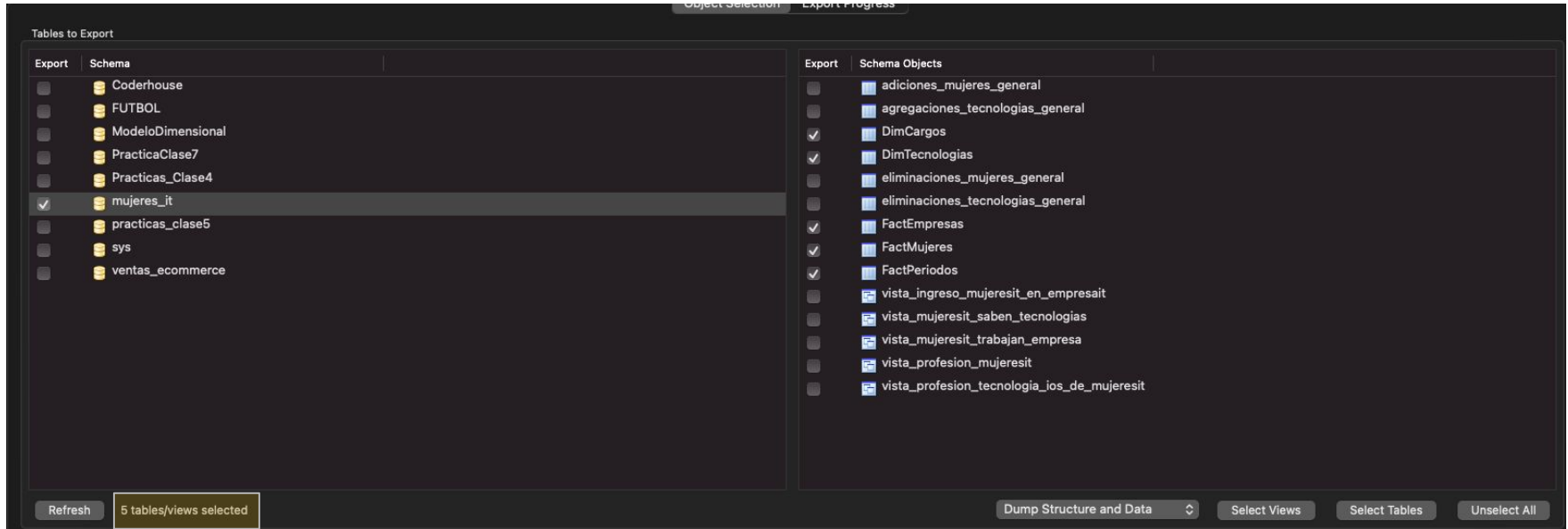


- **QUINTO PASO:** Una vez que seleccionamos el Schema, cómo lo hicimos en el paso anterior, en el lado derecho de nuestra pantalla nos aparecen las Tablas que queremos Exportar. En este caso, vamos a elegir las siguientes Tablas que se muestran en la imagen (debido a que son las principales Tablas del Proyecto):

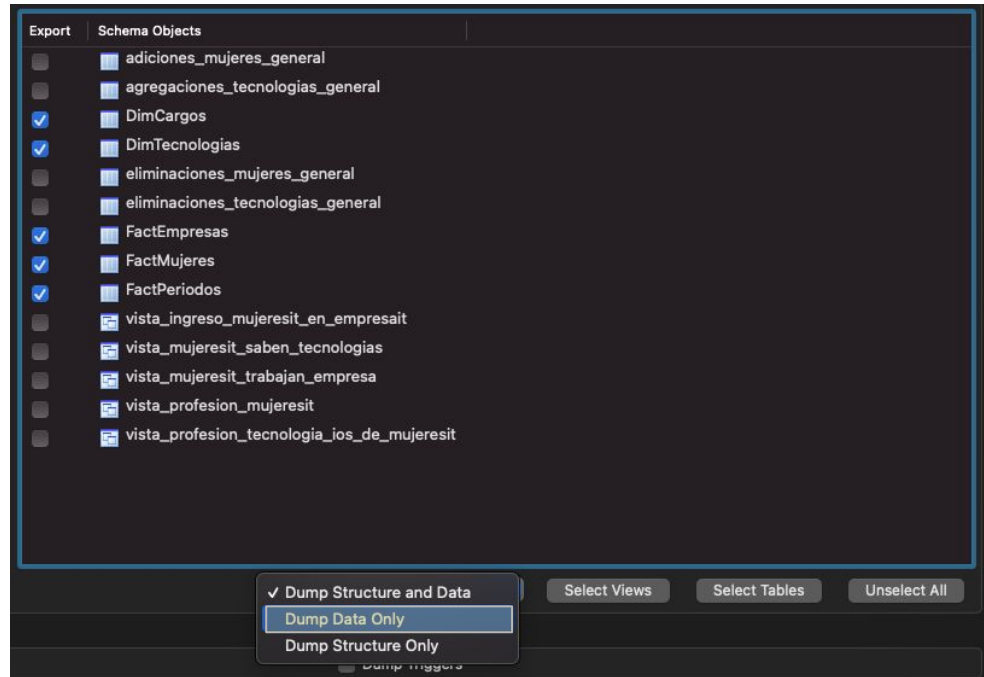




Podemos observar en la pantalla que nos dice cuántas Tablas/Views fueron seleccionadas:



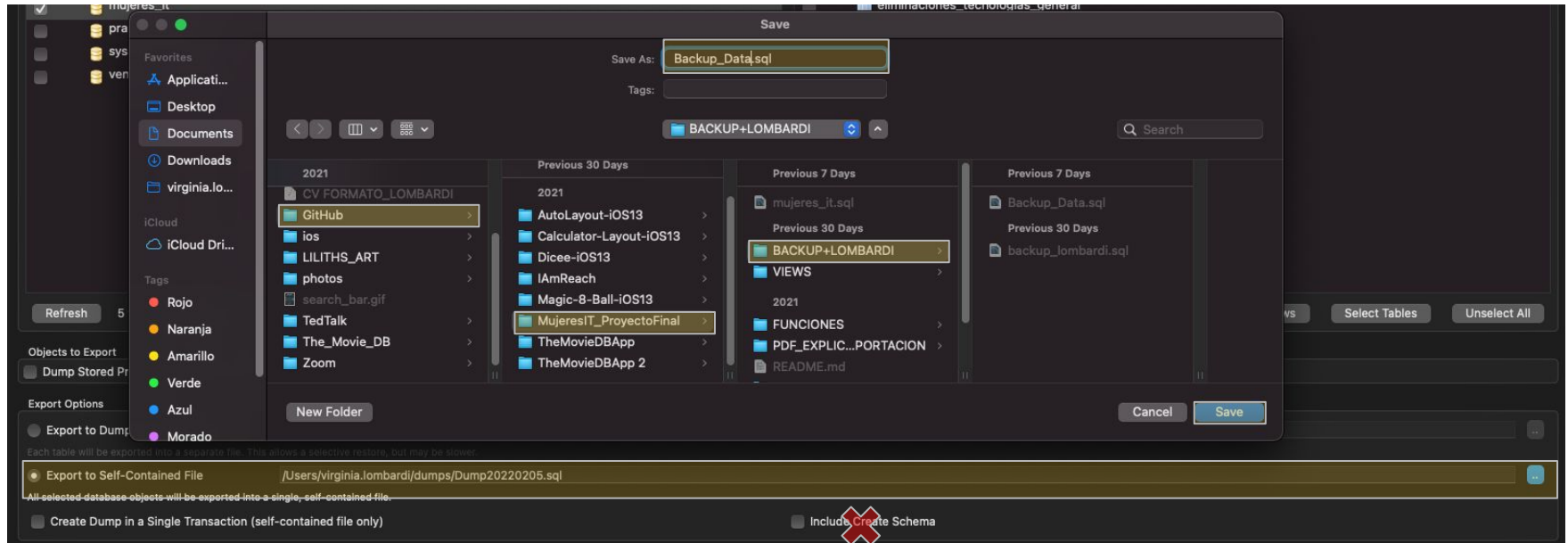
- **SEXTO PASO:** Ahora es momento de elegir si queremos Exportar los Datos con la Estructura, la Estructura sola o los Datos solos. En nuestro caso se nos había pedido SOLAMENTE los Datos de las Tablas. Por lo que Elegimos la Opción: “Dump Data Only” como se muestra en la imagen:



- **SEPTIMO PASO:** En este paso vamos a elegir cómo queremos guardar esos Datos exportados. Tenemos dos opciones. La primera opción es la que guarda las Tablas en diferentes archivos, esto puede ser útil en caso que queramos elegir una en especial para Restaurar, etc. Pero en nuestro caso vamos a utilizar la segunda opción que las guarda en un sólo archivo .sq, lo que nos permite Restaurarlas a todas de una vez. Por defecto se guardarán en la Carpeta Dump con la Fecha en la que realizamos la Exportación. En nuestro caso ya que tenemos un Repositorio y una Carpeta para cada Script, la guardamos en la dirección que se indica en la imagen y le damos a Save.

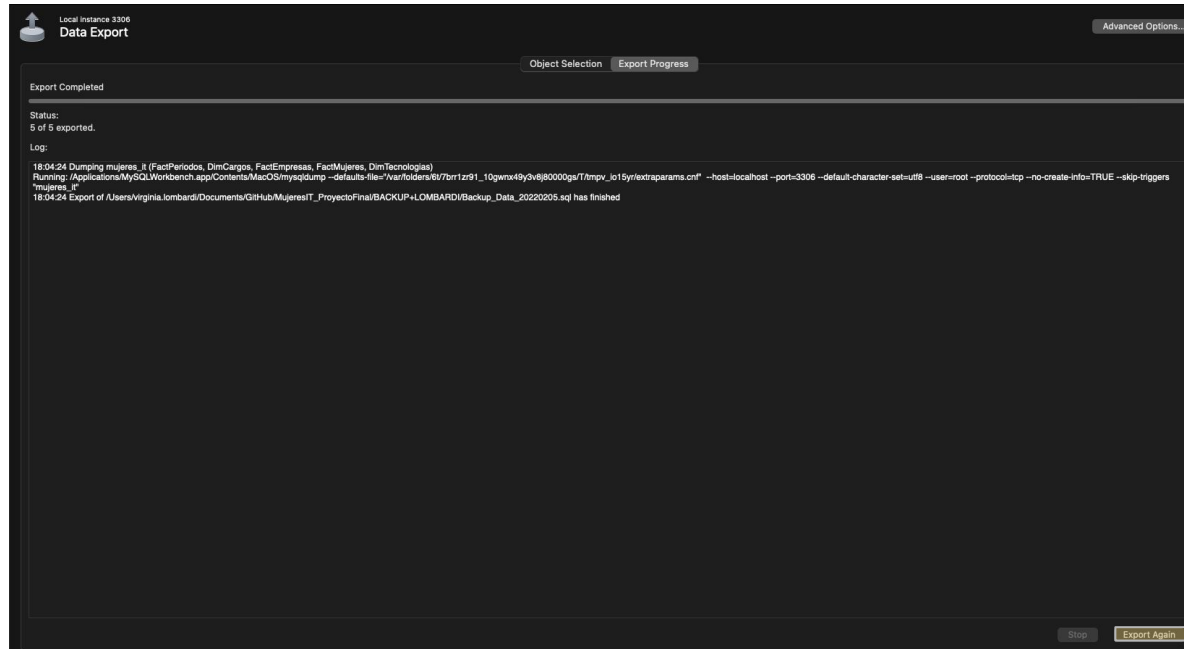
Como dijimos al principio no elegimos el checkbox que dice “Include Create Schema”.

Aquí vemos lo que dijimos en este paso:



En la Carpeta le agregamos también la fecha para que quedara más prolijo.

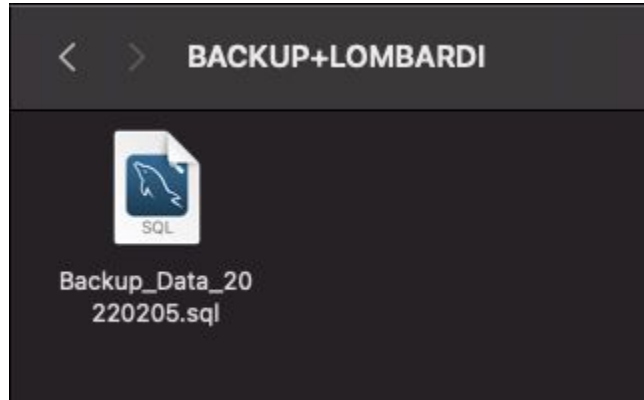
- **ÚLTIMOS PASOS:** A esta altura ya estamos listos para darle al Botón Start Export. Una vez que hagamos esto. En la pestaña “Export Progress” vamos a ver el Progreso de la Exportación, así como el Status que muestra la cantidad de archivos exportados. Así como la hora de realización de la Exportación, las Tablas elegidas, entre otra información, como se muestra en la imagen:



También tenemos un Botón que dice “Export Again”, por si queremos realizar la Exportación nuevamente. Esto sirve por si por ejemplo nos olvidamos de agregar algo que queremos salvar, lo hacemos en la Pestaña de “Object Selection” y lo volvemos a Exportar.

## Restauración:

- **PRIMER PASO:** Una vez realizado el Backup tenemos un archivo .sql con las Tablas que deseábamos Recuperar. Como se ve en la imagen, tenemos un solo archivo .sql con las Tablas a Recuperar:



# Lo que podemos hacer en un principio es abrirlo en un Editor de Texto para ver lo que contiene, que en nuestro caso es lo siguiente:

```
-- MySQL dump 10.13 Distrib 8.0.26, for macos11 (x86_64)

--

-- Host: localhost  Database: mujeres_it

-- -----

-- Server version          8.0.26

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;

/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;

/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;

/*!50503 SET NAMES utf8 */;

/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;

/*!40103 SET TIME_ZONE='+00:00' */;

/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;

/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;

/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;

/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
```

```
-  
  
Dumping data for table `DimCargos`
```

```
--
```

```
LOCK TABLES `DimCargos` WRITE;
```

```
/*!40000 ALTER TABLE `DimCargos` DISABLE KEYS */;
```

```
INSERT INTO `DimCargos` VALUES (2,'Junior'),(3,'Junior Advanced'),(4,'Semi Senior'),(5,'Senior'),(6,'Senior Advanced'),(7,'Senior Level One'),(9,'Senior Level Three'),(10,'Senior Level Four'),(13,'Architect Designer'),(14,'Tech Lead');
```

```
/*!40000 ALTER TABLE `DimCargos` ENABLE KEYS */;
```

```
UNLOCK TABLES;
```

```
--
```

```
-- Dumping data for table `DimTecnologias`
```

```
--
```

```
LOCK TABLES `DimTecnologias` WRITE;
```

```
/*!40000 ALTER TABLE `DimTecnologias` DISABLE KEYS */;
```

```
INSERT INTO `DimTecnologias` VALUES (1,'iOS','Mobile'),(3,'React','Web UI'),(4,'MongoDB','Web UI'),(5,'VUE JS','Web UI'),(6,'JavaScript','Web UI'),(7,'Java','Programming'),(8,'C++','Programming'),(9,'Python','Programming'),(10,'C','Programming'),(11,'Angular','Web UI');
```

```
/*!40000 ALTER TABLE `DimTecnologias` ENABLE KEYS */;
```

```
UNLOCK TABLES;
```



--

-- Dumping data for table `FactEmpresas`

--

LOCK TABLES `FactEmpresas` WRITE;

/\*!40000 ALTER TABLE `FactEmpresas` DISABLE KEYS \*/;

INSERT INTO `FactEmpresas` VALUES (123,'Globant','Global Company',1,1),(124,'TCS','Global Company',2,5),(125,'Sabre','Global Company',3,5),(126,'d-Local','Global Company',4,2),(127,'Overactive','Global Company',5,3),(128,'Wabbi','Start Up',6,4),(129,'Tienda Inglesa','Global Company',7,6),(200,'Nimacloud','Start Up',8,7),(201,'Mercado Libre','Start Up',9,10),(202,'SparkDigital','Start Up',10,9);

/\*!40000 ALTER TABLE `FactEmpresas` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Dumping data for table `FactMujeres`

--

LOCK TABLES `FactMujeres` WRITE;

/\*!40000 ALTER TABLE `FactMujeres` DISABLE KEYS \*/;

INSERT INTO `FactMujeres` VALUES (1,'Virginia','Lombardi',46288064,'iOS Developer',1,1),(2,'Lucia','Perez',54568907,'Systems Engineering',5,4),(3,'Mariana','Orman',33458065,'iOS Developer',8,1),(4,'Lorena','Alvarez',56789073,'Java Developer',2,7),(5,'Alicia','Gonzalez',45678753,'Python Developer',3,9),(6,'Martina','Bernardi',56873219,'Android Developer',4,2),(7,'Isabel','Vayra',34623786,'Web UI Developer',6,3),(8,'Micaela','Nelson',56784325,'Web UI Developer',7,5),(9,'Susana','Mori',34868063,'C Developer',10,10),(12,'Siri','Altez',43211285,'iOS Developer',2,1);

/\*!40000 ALTER TABLE `FactMujeres` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Dumping data for table `FactPeriodos`

--

LOCK TABLES `FactPeriodos` WRITE;

/\*!40000 ALTER TABLE `FactPeriodos` DISABLE KEYS \*/;

INSERT INTO `FactPeriodos` VALUES

(1,18,1,2021,123,10),(2,20,2,2019,125,2),(3,23,12,2020,127,4),(4,31,10,2010,129,6),(5,12,8,2015,201,8),(6,11,7,2021,202,3),(7,21,6,2021,124,5),(8,29,3,2013,126,7),(9,1,4,2014,128,9),(10,10,5,2016,200,1);

/\*!40000 ALTER TABLE `FactPeriodos` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!40103 SET TIME\_ZONE=@OLD\_TIME\_ZONE \*/;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;

/\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

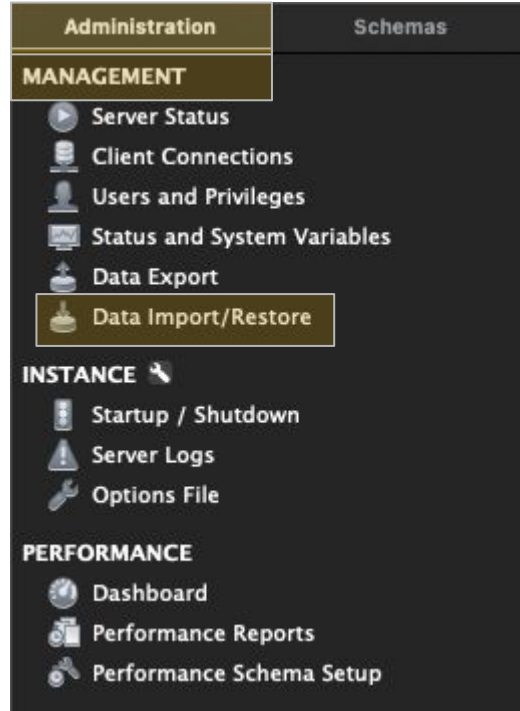
/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

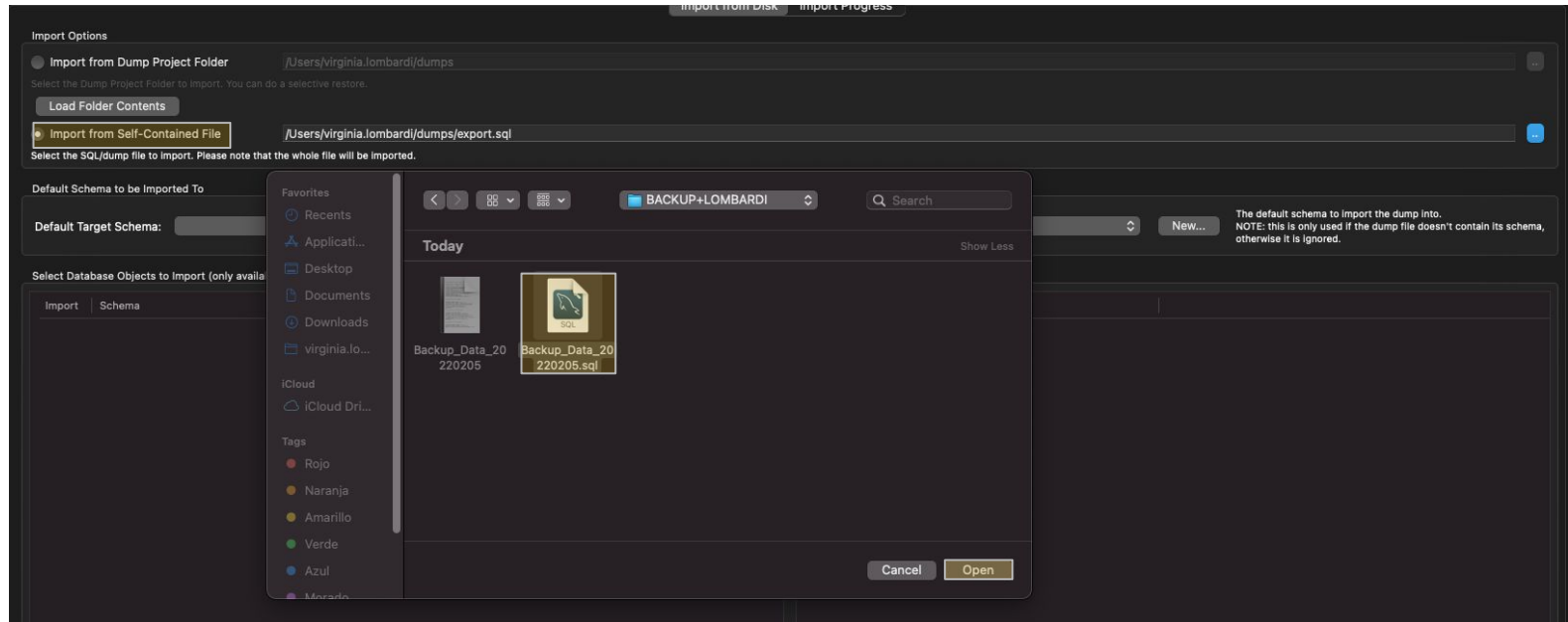
/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;

-- Dump completed on 2022-02-05 18:04:24

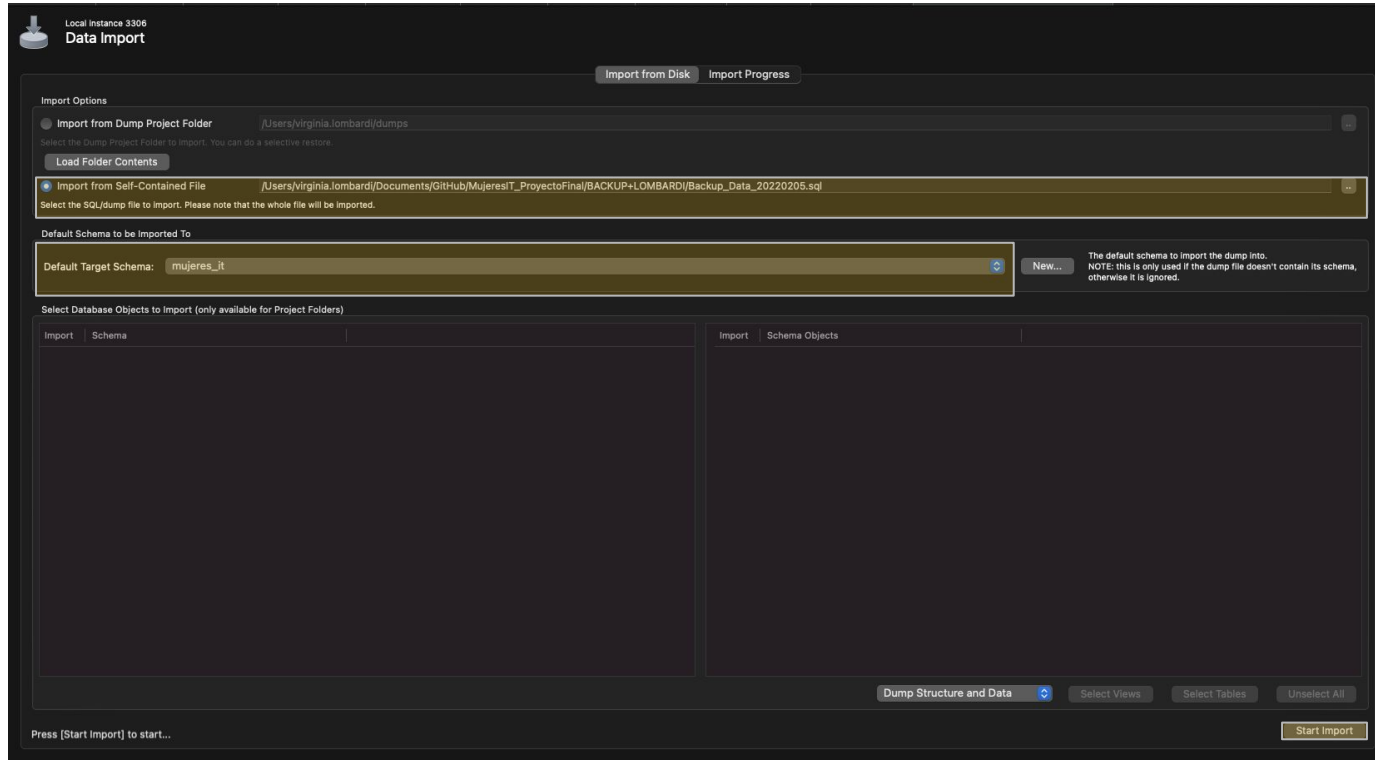
- **SEGUNDO PASO:** Realizaremos la Importación de los Datos. Para ello nos dirigimos a la misma Pestaña que para Exportar, sólo que esta vez elegimos “Data Import/Restore” como se muestra en la imagen:



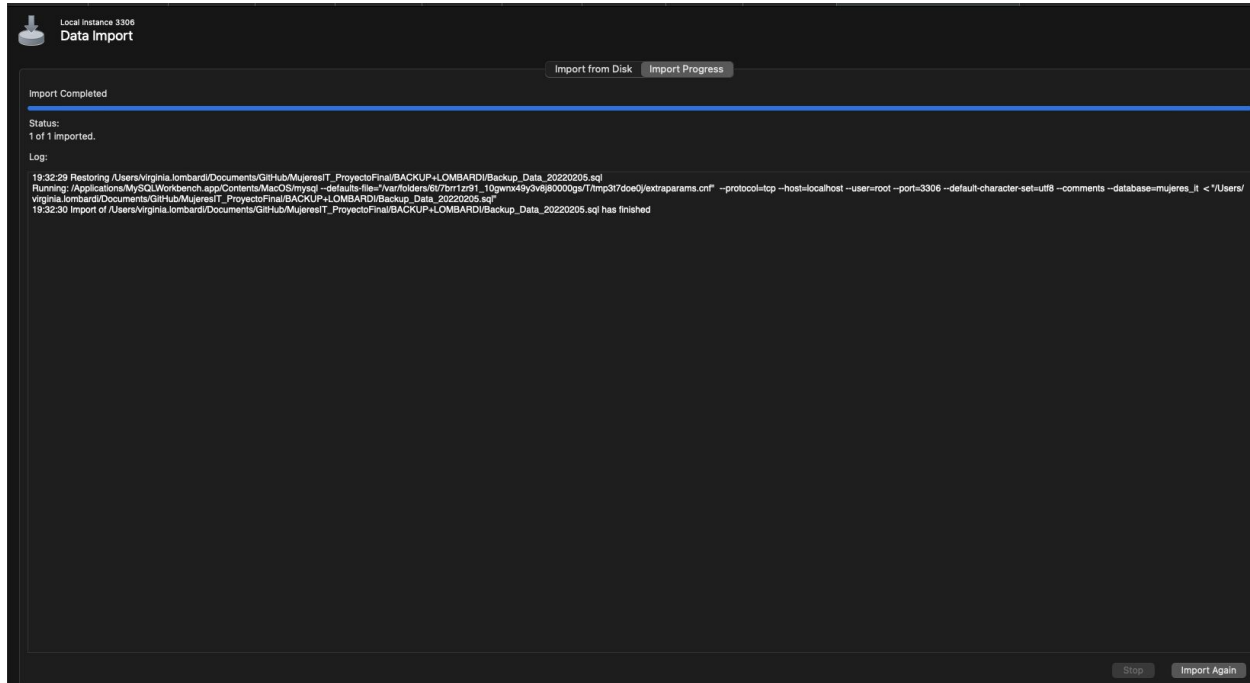
- **TERCER PASO:** Se nos abrirá la siguiente ventana, conteniendo la forma de Importación. Hay dos opciones, la primera es para si tenemos archivos por separado de las Tablas importarlas de esa forma. La segunda sería la que vamos a utilizar, ya que es la que permite seleccionar el archivo .sql que tenemos con todas las Tablas. Una vez realizado esto elegimos el archivo, en nuestro caso va a ser el que se muestra en la imagen:



Aquí lo que hacemos es elegir la BBDD (mujeres\_it) y antes de realizar nada yo borré las Tablas que voy a Importar, y las cree nuevamente pero sin los datos, de esa forma no recibiré ningún error de Datos Duplicados, etc. Y le damos a “Start Import”:



- **ÚLTIMOS PASOS:** En la pestaña “Import Progress” vamos a ver el Progreso de la Importación, así como el Status que muestra la cantidad de archivos exportados. Así como la hora de realización de la Importación, entre otra información, como se muestra en la imagen:



# Informes generados en base a la información almacenada en las tablas:

- Según la Vista `vista_ingreso_mujeresit_en_empresait`, podemos concluir que desde 2010 hasta 2021 en Uruguay los ingresos fueron de tan sólo 10 Mujeres a diferentes Empresas de IT. Por lo que podemos sacar la conclusión de que no son muchas las Mujeres que ingresan a Empresas IT. Por supuesto pueden faltar Datos/Registros, pero es algo que está comprobado el hecho de que no todas las Mujeres que tienen conocimiento en IT obtienen trabajo en esa área.



- De la Vista, **vista\_mujeresit\_saben\_tecnologias**, podemos concluir que por ejemplo en el Tipo de Tecnología Mobile, la Tecnología destacada es iOS, lo cual tiene mucho sentido, ya que en Uruguay hay una altísima demanda de personas que tengan conocimientos en dicha Tecnología. Debido a que son muy difíciles de obtener, las Empresas tales como Globant (por ejemplo), tienen una iniciativa de contratar personas con conocimientos en IT (no necesariamente en iOS), y capacitarlas en dicha Tecnología.
- Mediante la Vista **vista\_mujeresit\_trabajan\_empresa**, concluimos que la mayoría de las Empresas en la que trabajan las Mujeres IT, son del Tipo “Global Company”. Lo cual es muy llamativo dado que no son muchas las Empresas de este Tipo, digamos que se pueden contar “con los dedos de la mano”. Claramente eso ha venido evolucionando con el tiempo. Más y más Empresas del Tipo Global eligen Uruguay como destino para hacer negocios.
- Si observamos las Vistas **vista\_profesion\_mujeresit** y **vista\_ingreso\_mujeresit\_en\_empresait**, teniendo en cuenta el Tiempo de Ingreso vs Seniority no se observa que ambos evolucionan en el mismo sentido. Está comprobado que a las Mujeres en el Mundo IT les cuesta mucho más obtener ascensos o cambios en su Seniority. Aún cuando la propia Empresa forma a Mujeres y Hombres por igual.

- Según la Vista, **vista\_profesion\_mujeresit**, si tomamos en cuenta la Profesión al Ingreso y la evolución en el Seniority. La Profesión tampoco se es respetada al momento de obtener los cambios de Seniority.

También en este sentido la Profesión de la Mujer al Ingreso, en Uruguay, tampoco es tenida en cuenta para obtener mejores Senioritis a diferencia de lo que ocurre con los Hombres.

# Herramientas y Tecnologías:

- Para el Model ER, el inicial se utilizó la Herramienta **app.diagrams.net**.
- Para realizar todo el Script SQL y el segundo Modelo ER, se trabajó con el **Gestor MySQL**.
- Para los Datos, que al final no se utilizaron, para la Importación de Datos, se utilizó **Excel Online**.
- Para los PDF's, se utilizó **Google Presentation**.



# ARCHIVOS .SQL Y PDF'S:

LINK REPO:

[https://github.com/LVLVirginiaLombardi/MujeresIT\\_ProyectoFinal](https://github.com/LVLVirginiaLombardi/MujeresIT_ProyectoFinal)

Se envió invite al repo como colaborador.



# MUJERES IT

FIN DEL PROYECTO  
SQL



**Estudiante:** Lilian Virginia Lombardi López

**Profesor:** Miguel Rodas