

Tecnológico Nacional de México
Instituto Tecnológico de Mexicali

Ingeniería en Sistema Computacionales



Taller de Base de Datos

Reporte de práctica. Consultas SQL relacionadas al proyecto.

Barba Navarro Luis Rodrigo, 20490687
Gurrola Bernal Johan Antonio, 20490703
Lira López Alejandro, 16491160
Pérez García Sofía, 20491083

No. Equipo: 4
Grupo: CZA 5
Profesor: Carlos Alberto López Castellanos

Mexicali, Baja California a sábado, 22 de octubre de 2022.

Índice.

Introducción.	2
Desarrollo.	3
Adaptaciones realizadas a la base de datos.	3
Inserciones de datos de la base de datos mediante archivos CSV.	6
Desarrollo de las consultas planteadas.	11
Conclusiones personales.	22
Conclusión general.	22

I. Introducción.

Básicamente, SQL es un lenguaje de programación que puede permitir acceder a una base de datos y adquirir información. Esta información puede ser sobre productos, existencias, pedidos, entre otros datos relacionados a la empresa dada. Es ampliamente utilizado por las empresas no sólo para almacenar información, sino también para recuperar y manipular datos; sobre todo, mejorar la gestión de un gran cúmulo de datos.

Hoy en día, las pequeñas y medianas empresas pueden beneficiarse de SQL porque está creado específicamente para sus necesidades de administración de bases de datos. Por otro lado, en un mundo donde la gran cantidad de datos abundan, las empresas grandes deben ser capaces de procesar grandes volúmenes de datos todos los días. Tener herramientas como SQL y gente capacitada en programación de ese lenguaje, pueden garantizar que lidiar con consultas de datos complicadas sea lo más fácil posible.

Durante el transcurso de la primera unidad, se estableció una problemática relacionada a la gestión de un sistema de máquinas expendedoras, por lo que se diseñó una base de datos que cumplieran las necesidades planteadas por el cliente. Sin embargo, una parte fundamental para la realización de consultas es que nuestra base de datos tenga un cúmulo de datos decente, por lo que se integró datos de una manera en específico que se hablará con más detalle en el transcurso del reporte.

En el presente reporte de práctica, se pretende atender las necesidades de las consultas planteadas por el cliente, y hacer las respectivas adaptaciones a nuestra base de datos, que nos permitirá realizar nuestras consultas de manera

satisfactoria; explicando el por qué de esas modificaciones. Por otro lado, se plantea explicar cómo fue el proceso de inserciones a nuestra base de datos mediante archivos de tipo CSV, y por último, llevar a cabo la ejecución de nuestras consultas, donde se explicará brevemente el funcionamiento de cada una de ellas.

II. Desarrollo.

A. Adaptaciones realizadas a la base de datos.

Al momento de analizar las consultas planteadas en la práctica dirigida al proyecto, nos percatamos que tenían algunas irregularidades que con la estructura previa definida de nuestra base de datos, hecha a partir de la primera unidad, no podría suplir con la obtención de cierta información, por que se planteó hacer ciertas adaptaciones al diseño de nuestra base de datos para que realización de las consultas se haga de manera correcta.

Primeramente, surgió la necesidad de agrupar las tarjetas de los clientes por universidad; nuestro diseño no nos permitía realizar eso ya que no se encontraba relacionado los clientes con una respectiva universidad, ya que en la primera unidad, la institución donde estaban registrados los clientes era despreciable, por lo que no se tomó en cuenta. Ahora, se requirió establecer una llave foránea en la tabla de los clientes, para que cada cliente estuviera asociado a una universidad o no (acepta valores nulos en caso de que sea una persona no ligada directamente a una institución de educación).

```
/*Tabla: Cliente*/  
DROP TABLE IF EXISTS maquina_expendedora.cliente;  
CREATE TABLE IF NOT EXISTS maquina_expendedora.cliente (  
    id_cliente INT NOT NULL,  
    id_institucion INT NULL,  
    nombre_cliente VARCHAR(64) NOT NULL,  
    apellido_paterno VARCHAR(64) NOT NULL,  
    apellido_materno VARCHAR(64) NOT NULL,  
    PRIMARY KEY (id_cliente),
```

Imagen 1.1. Estructura de creación de la tabla de los clientes.

Asimismo, a la tabla de los clientes se le añadió un índice visible apuntando directamente al identificador de la institución. Además, de que a la llave foránea del identificador de la institución se le agregaron como restricciones, no realizar acción alguna al momento de borrar o actualizar un dato de la tabla padre, en este caso, de la tabla de las instituciones.

```

INDEX institucion_idx (id_institucion ASC) VISIBLE,
CONSTRAINT institucion_cliente
FOREIGN KEY (id_institucion)
REFERENCES maquina_expendedora.institucion (id_institucion)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

Imagen 1.2. Estructura de la creación de índice y restricciones de la llave foránea en la tabla de los clientes.

Por otro lado, otra de las problemáticas era que las promociones deberían estar ligadas a una serie de productos de la máquina expendedora que conforman el paquete, y no a una categoría como se acordó en la primera unidad; creemos que una de las razones de este cambio tan repentino fue debido a que, sería imposible establecer las promociones cuando existen diferentes variedades de categorías, asociadas a múltiples productos con diferentes precios dependiendo de la máquina expendedora.

Para atender esta problemática, lo que se optó realizar fue crear una tabla de unión entre la promoción, la máquina y el producto, que se encargaría del detalle de la promoción, evidentemente esta tabla contiene las llaves foráneas de las tablas anteriormente mencionadas (estas a su vez, forman parte de la llave compuesta) y un campo donde se establecería el precio reducido de cada uno de los productos por separado, con eso podríamos obtener el precio del paquete de la promoción. Se hizo de esta manera ya que, como los precios de los productos varían dependiendo de la máquina, sería despreciable darle un precio a un paquete que en otras máquinas adquirirlo podría ser desventajoso, en cuestión de costos de adquisición.

```

/*Tabla de Unión: Detalle-Promocion*/
DROP TABLE IF EXISTS maquina_expendedora.detalle_promocion;
CREATE TABLE IF NOT EXISTS maquina_expendedora.detalle_promocion (
id_promocion INT NOT NULL,
id_maquina VARCHAR(8) NOT NULL,
id_producto INT NOT NULL,
precio_reducido INT NULL,
PRIMARY KEY (id_promocion, id_maquina, id_producto),

```

Imagen 1.3. Estructura de creación de la tabla de los detalles de la promoción.

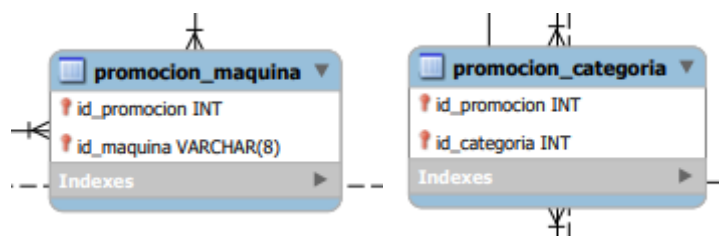
Consiguientemente, en la tabla de los detalles de las promociones se le añadió tres índices visibles; cada uno apuntando a su respectiva llave foránea de la tabla. A la llave foránea del identificador la promoción se le agregaron como restricciones, no realizar acción alguna al momento de borrar o actualizar un dato de la tabla padre, en este caso, de la tabla de las promociones. Este mismo proceso,

se realizó con su respectiva tabla y llave foránea del identificador de la máquina y el producto.

```
INDEX promocion_idx (id_promocion ASC) VISIBLE,  
INDEX producto_idx (id_producto ASC) VISIBLE,  
INDEX maquina_idx (id_maquina ASC) VISIBLE,  
CONSTRAINT promocion_pp  
FOREIGN KEY (id_promocion)  
REFERENCES maquina_expendedora.promocion (id_promocion)  
ON DELETE NO ACTION  
ON UPDATE NO ACTION,  
  
CONSTRAINT producto_pp  
FOREIGN KEY (id_producto)  
REFERENCES maquina_expendedora.producto (id_producto)  
ON DELETE NO ACTION  
ON UPDATE NO ACTION,  
  
CONSTRAINT maquina_pp  
FOREIGN KEY (id_maquina)  
REFERENCES maquina_expendedora.maquina (id_maquina)  
ON DELETE NO ACTION  
ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

Imágenes 1.4, 1.5, 1.6 respectivas. Estructura de la creación de los índices y restricciones de las llaves foráneas en la tabla de los detalles de la promoción .

Ya que la tabla de los detalles de la promoción ya contenía en qué máquina expendedora se encontraría determinada promoción, la tabla de unión de promoción y máquina fue descartada de nuestro diseño. Así como la tabla de unión de promoción y categoría, ya que no se iban a agrupar las promociones por categorías, sino por productos.



Imágenes 1.7. Tablas previamente mencionadas que fueron descartadas de nuestro diseño de base de datos, ya que no atendieron a la problemática.

En las tablas posteriores, se muestra la estructura de la tabla de los detalles de la promoción y la tabla de los clientes respectivamente, donde podemos observar

el tipo de datos, si acepta valores nulos o no, y si es perteneciente a una llave primaria o foránea.

	Field	Type	Null	Key	Default	Extra
►	id_promocion	int	NO	PRI	NULL	
	id_maquina	varchar(8)	NO	PRI	NULL	
	id_producto	int	NO	PRI	NULL	
	precio_reducido	int	YES		NULL	

	Field	Type	Null	Key	Default	Extra
►	id_cliente	int	NO	PRI	NULL	
	id_institucion	int	YES	MUL	NULL	
	nombre_cliente	varchar(64)	NO		NULL	
	apellido_paterno	varchar(64)	NO		NULL	
	apellido_materno	varchar(64)	NO		NULL	

Imágenes 1.8. Estructura de las tablas de detalles de la promoción y tabla de los clientes.

B. Inserciones de datos de la base de datos mediante archivos CSV.

Cuando nosotros construimos una base de datos, es con la necesidad de almacenar información relacionada a la empresa, por lo que se requiere hacer inserciones de datos. La manera tradicional de hacer inserciones es con una sentencia con la palabra clave INSERT, donde especificamos la tabla donde haremos nuestras inserciones y qué valores dados serán correspondientes a sus columnas de la tabla. Por ejemplo, para hacer una inserción a la tabla de las tarjetas, es necesario primero conocer cuáles serán los campos necesarios para hacer la inserción; se observa que son el identificador de la tarjeta y del cliente, la fecha de expedición de la tarjeta, los puntos de la tarjeta y su estatus.

	Field	Type	Null	Key	Default
►	id_tarjeta	int	NO	PRI	NULL
	id_cliente	int	NO	MUL	NULL
	fecha_expedicion	date	NO		NULL
	puntos	int	NO		NULL
	estatus	varchar(32)	NO		NULL

```
DESCRIBE tarjeta;
INSERT INTO tarjeta VALUES
(
1,
20490687,
"2022-10-31",
2500,
"ACTIVA"
);
```

Imágenes 2.1. Estructura de la tabla de las tarjetas y sentencia de la inserción de los datos a dicha tabla.

Posteriormente, verificamos en la tabla de las tarjetas si la inserción se realizó de manera correcta; mediante una instrucción de tipo SELECT que nos muestre todos los registros de la tabla de las tarjetas, se pudo observar que el identificador de tarjeta con el número 1 existe, como se aprecia en la siguiente imagen.

	id_tarjeta	id_cliente	fecha_expedicion	puntos	estatus
►	1	20490687	2022-10-31	2500	ACTIVA

Imagen 2.2. Verificación de la inserción en la tabla de las tarjetas.

No obstante, esto es sólo un ejemplo en una tabla en específico; como necesitamos llenar nuestra base de datos con un tamaño de información decente para realizar las consultas planteadas, sería casi imposible llevar a cabo el método tradicional y hacer inserción por inserción, nos tomaría demasiado tiempo y sería poco eficiente (por el momento, no desarrollamos un sistema de captura de información)

Es por ello que, se decidió llevar a cabo un método más conveniente. En primera instancia, se llenó un documento de excel con información relevante de cada una de las tablas de nuestra base de datos, como se muestra en el siguiente ejemplo. Se deberá colocar el campo perteneciente a la tabla hasta arriba de cada columna del documento de excel, haciendo esto con tantos campos existan en la tabla. Posterior a eso, se ingresan datos debajo de cada campo con su información correspondiente.

	A	B	C
1	id_categoria	nombre_categoria	descripcion_categoria
2	2010	Botellas de agua	Agua potable envasada en botellas individuales
3	2011	Bebidas energeticas	Bebida envasada estimulante en botellas individuales
4	2012	Bebidas vitaminadas	Bebidas envasada vitaminada en botellas individuales
5	2013	Refrescos dieteticos	Refresco envasado con poco contenido de azucars
6	2014	Refrescos mineral	Refresco envasado con características minerales
7	2015	Refrescos normal	Refresco envasado de consumo popular
8	2016	Cafe caliente	Bebida envasada de variantes de cafe caliente
9	2017	Cafe frio	Bebida envasada de variantes de cafe frio
10	2018	Meriendas	Alimento generalmente ligero que se toma a media tarde
11	2019	Pastelillos	Pastelillo de diversos sabores dulces o salados
12	2020	Galletas	Producto alimenticio horneado y moldeado a base de harina
13	2021	Gelatinas	Mezcla coloide gel de diferentes sabores
14	2022	Chocolates	Alimento que se elabora con una pasta de cacao y azucar
15	2023	Gominolas	Alimento coloide gel que poseen diferentes sabores
16	2024	Papas fritas	Alimento procesado a base de papa con diversos sabores
17	2025	Nueces	Semilla comestible con diferentes formas y sabores

Imagen 2.3. Información almacenada en documento de excel.

Una vez concluido con el llenado de los documentos de excel, se tiene que exportar el documento como archivo de tipo CSV. Por lo que en el apartado de guardar, se debe seleccionar esta extensión para cada tabla, como se muestra en el siguiente ejemplo.

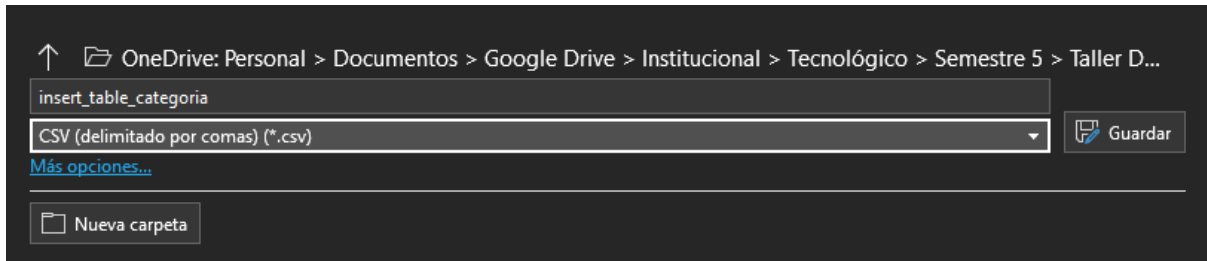


Imagen 2.4. Exportación de documento de excel como archivo de tipo CSV.

Ahora desde la aplicación de workbench, seleccionamos nuestra base de datos desde el apartado de esquemas, en nuestro caso, maquina_expendedora, y desplegamos la información del árbol de las tablas, y seleccionamos la tabla donde queremos importar nuestros datos. En este ejemplo, seleccionamos la tabla de categorías y le damos click derecho; se desplegará un menú y se debe seleccionar la opción con el nombre de “Table Data Import Wizard” para hacer la respectiva importación del archivo de tipo CSV.

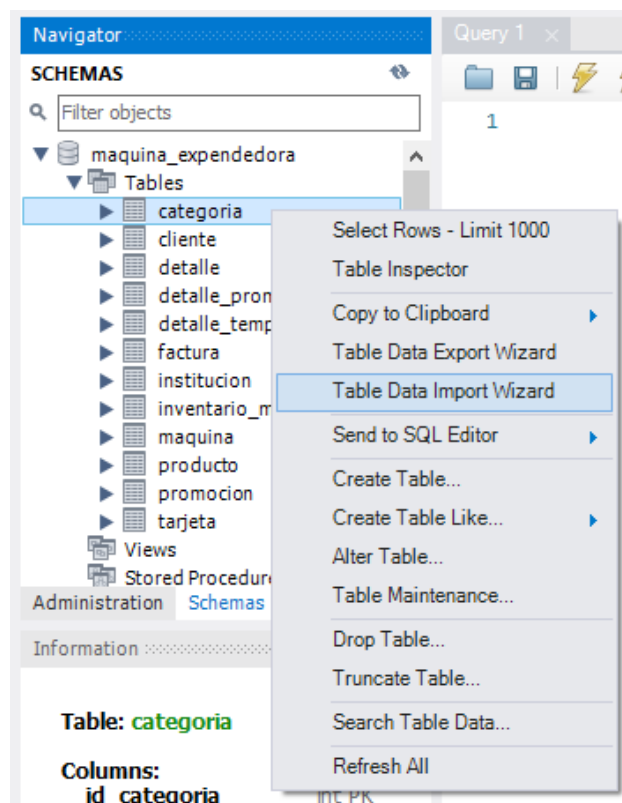


Imagen 2.5. Selección de la opción para importar el archivo de tipo CSV en la tabla.

Por consiguiente, se nos despliega una ventana donde se nos pide que seleccionemos la ubicación del archivo de tipo CSV, que en este caso tiene como nombre “insert_table_categoria”.

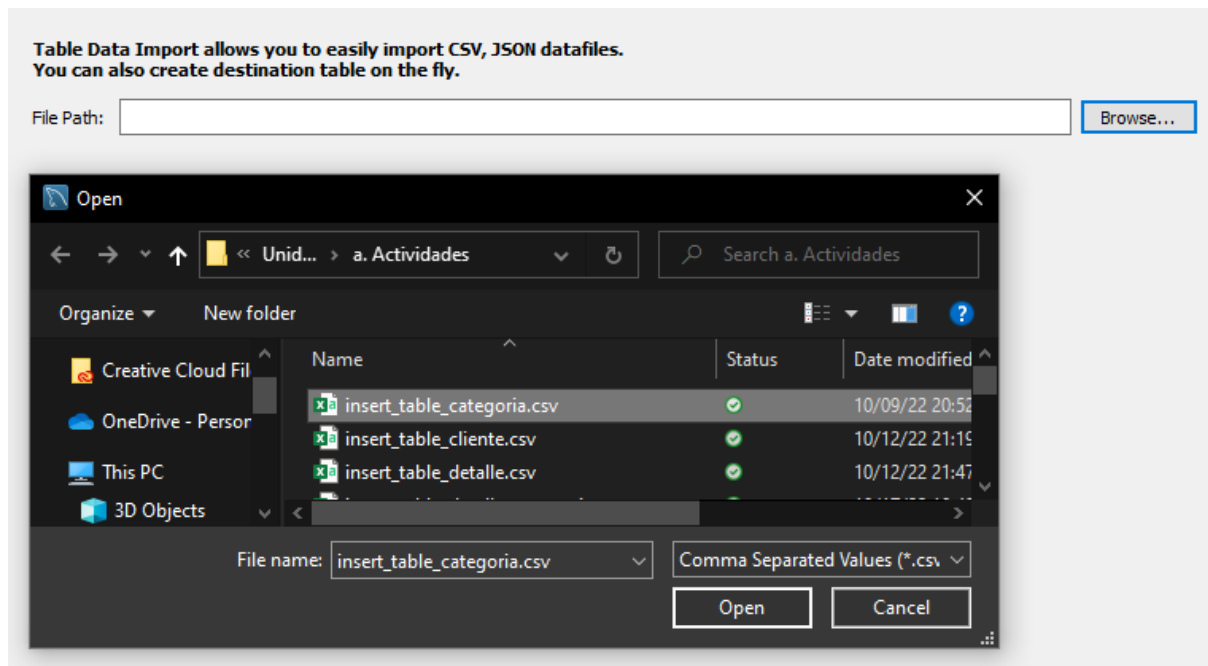


Imagen 2.6. Selección del archivo de tipo CSV.

Posterior a eso, se selecciona el destino de los datos; hay dos opciones, usar una tabla existente o crear una nueva tabla para la importación, nosotros como ya tenemos las tablas creadas en nuestra base de datos, sólo se seleccionará las respectivas tablas existentes, como se muestra en el siguiente ejemplo.

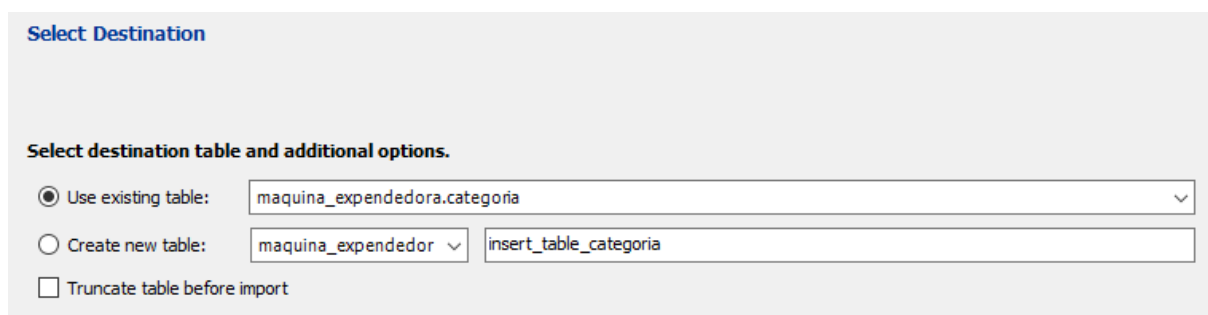
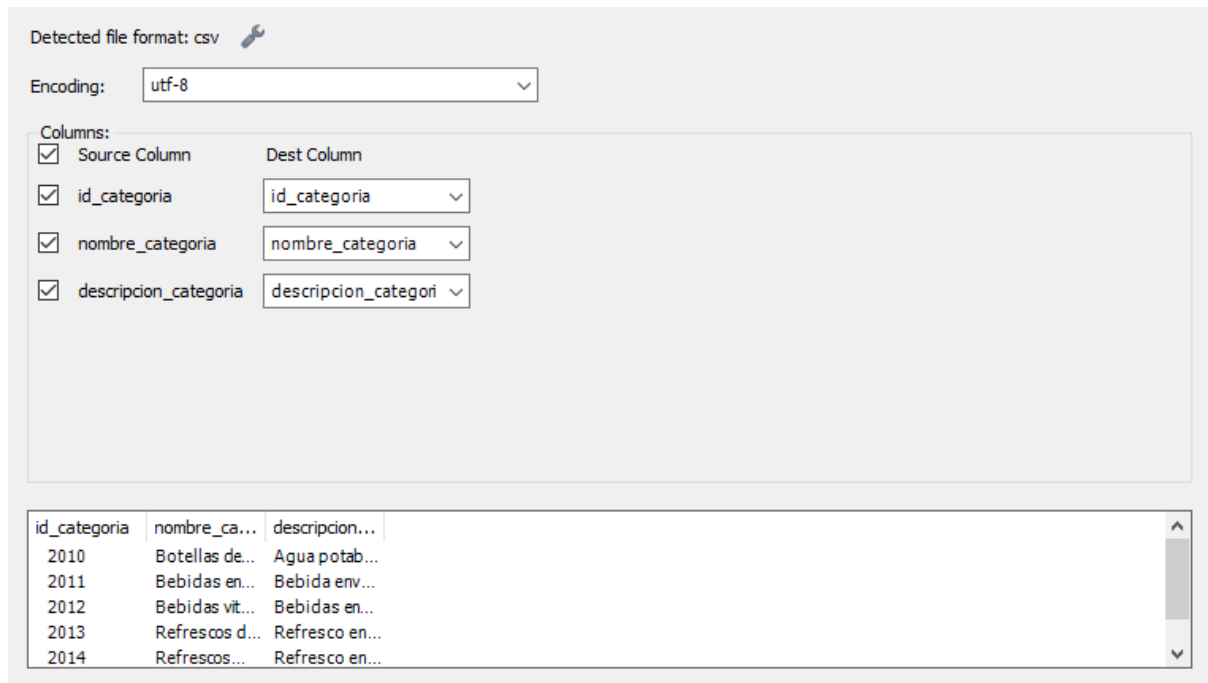


Imagen 2.7. Selección de la tabla destino para la inserción de datos.

Después, si establecimos bien las columnas en nuestra base de datos, la aplicación de workbench detectará de manera automática las columnas destino, solo es cuestión de verificar si las columnas fuente corresponden a las columnas destino, y si la información de ejemplo plasmada en el cuadro posterior es correcta.



Detected file format: csv

Encoding: utf-8

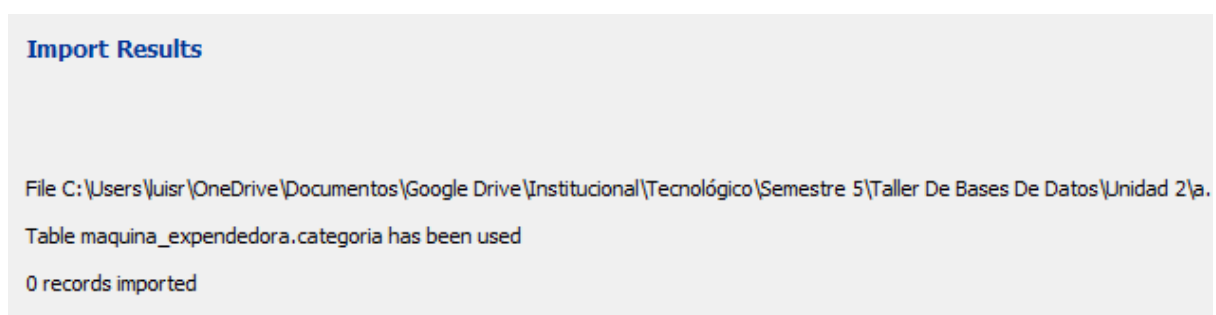
Columns:

Source Column	Dest Column
<input checked="" type="checkbox"/> id_categoria	id_categoria
<input checked="" type="checkbox"/> nombre_categoria	nombre_categoria
<input checked="" type="checkbox"/> descripcion_categoria	descripcion_categoria

id_categoria	nombre_categoria	descripcion_categoria
2010	Botellas de...	Agua potab...
2011	Bebidas en...	Bebida env...
2012	Bebidas vit...	Bebidas en...
2013	Refrescos d...	Refresco en...
2014	Refrescos...	Refresco en...

Imagen 2.8. Verificación de las columnas fuente con las columnas destino.

Finalmente, cuando le damos click izquierdo al botón siguiente, se empezará el proceso de importación, y si todo sale de manera correcta, se desplegará la siguiente información en la ventana de importación.



Import Results

File C:\Users\juisr\OneDrive\Documentos\Google Drive\Institucional\Tecnológico\Semestre 5\Taller De Bases De Datos\Unidad 2\...

Table maquina_expendedora.categoria has been used

0 records imported

Imagen 2.9. Resultado final de la importación de los datos.

C. Desarrollo de las consultas planteadas.

1. A partir de un producto, lista todas las máquinas expendedoras que lo venden indicando para cada una, la ubicación, la cantidad de producto y el precio unitario del mismo.

Sentencia SQL

```
SELECT id_maquina, id_institucion, nombre_inst, id_producto,
nombre_producto, cantidad, precio_unitario
FROM institucion
INNER JOIN maquina using(id_institucion)
INNER JOIN inventario_maquina using(id_maquina)
INNER JOIN producto using(id_producto)
WHERE nombre_producto = "Coca-Cola";
```

Resultado

	id_maquina	id_institucion	nombre_inst	id_producto	nombre_producto	cantidad	precio_unitario
▶	4042	1040	Universidad Xochicalco Mexicali	159010	Coca-Cola	10	27
	4041	1040	Universidad Xochicalco Mexicali	159010	Coca-Cola	10	27
	4040	1040	Universidad Xochicalco Mexicali	159010	Coca-Cola	7	27
	4032	1030	Tecnologico Nacional de Mexico Campus Mexicali	159010	Coca-Cola	4	17



3 13:52:14 SELECT id_maquina, id_institucion, nombre_inst, id_producto,

Explicación

Primero buscamos un producto deseado para realizar esta consulta, el cual resultó siendo "Coca-Cola" y usando su nombre podemos buscar qué máquinas lo contienen, en que cantidad y a qué precio, por medio de un intersección con INNER JOIN de las tablas máquina, inventario_maquina y producto.

2. Liste todas las tarjetas vigentes, indicando el id de la tarjeta, los datos del cliente y el saldo, la lista debe estar ordenada por universidad y los clientes de forma alfabética.

Sentencia SQL

```
select id_tarjeta, id_institucion, id_cliente, nombre_cliente,  
       apellido_paterno, apellido_materno, puntos  
from cliente  
       inner join tarjeta using(id_cliente)  
where estatus = "Habilitada"  
order by id_institucion, nombre_cliente;
```

Resultado

id_tarjeta	id_institucion	id_cliente	nombre_cliente	apellido_paterno	apellido_materno	puntos
10020523	1010	20490698	Alejandro Walton	Daugherty	Banks	674
10020513	1010	20490688	Ashlee Durham	Nash	Patton	50
10020518	1010	20490693	Aston Bull	Alexander	Serrano	121
10020516	1010	20490691	Celia Cottrell	Perkins	Zimmerman	542
10020517	1010	20490692	Habibah Guerra	Brooks	White	999



1 21:30:58 select id_tarjeta, id_institucion, id_cliente, nombre_cliente, apellido_paterno, apellido_materno, puntos

Explicación

Otra simple, usando inner joins de cliente - institución - tarjeta podemos averiguar quienes son los clientes con una tarjeta habilitada utilizando el where. Por último tan solo lo ordenamos acorde a como indica el problema, por universidad y luego por los clientes de forma alfabética (en el caso de los clientes con que escribamos el campo "nombre_clientes" en el orden by lo va a hacer por orden alfabético).

3. Indique el total de saldo de las tarjetas por universidad

Sentencia SQL

```
select id_institucion, sum(puntos) as Total_SaldoXInstitucion
from tarjeta
    inner join cliente using(id_cliente)
group by id_institucion;
```

Resultado

	id_institucion	Total_SaldoXInstitucion
▶	1010	20714
	1020	19795
	1030	18243
	1040	16973



1 21:40:30 select id_institucion, sum(puntos) as Total_SaldoXInstitucion from tarjeta

Explicación

Igualmente como la anterior consulta, utilizando los ids de las instituciones y agrupando por ellas, se calcula la suma de todos los puntos de clientes pertenecientes a sus respectivas instituciones.

4. Indique cuál máquina expendedora ha vendido más en un periodo específico.

Sentencia SQL

```
SELECT id_institucion, nombre_inst, id_maquina, ubicacion, sum(cantidad) AS total_ventas
FROM factura
INNER JOIN detalle using(id_factura)
    INNER JOIN maquina using(id_maquina)
    INNER JOIN institucion using(id_institucion)
WHERE fecha_expedicion BETWEEN '20210212' AND '20211229'
GROUP BY id_maquina
HAVING total_ventas IN
(
    SELECT max(table_1.total_ventas)
    FROM (
        SELECT sum(cantidad) AS total_ventas
        FROM factura
        INNER JOIN detalle using(id_factura)
        WHERE fecha_expedicion BETWEEN '20210212' AND '20211229'
        GROUP BY id_maquina
    ) AS table_1
);
```

Resultado

	id_institucion	nombre_inst	id_maquina	ubicacion	total_ventas
▶	1030	Tecnologico Nacional de Mexico Campus Mexicali	4031	Edificio U	41

Explicación

Para conseguir la maquina (o maquinas) que han vendido más dentro de un periodo es algo diferente.

En una subconsulta necesitamos conseguir TODAS las ventas de productos de máquinas y sumarlas, agrupando por su respectiva máquina MIENTRAS esté dentro de la fecha dada que en este caso fue “2021-02-12” y “2021-12-29”, así que todas las ventas hechas dentro de estas fechas fueron contadas.

Después le sacamos el máximo para finalmente poner toda esa consulta dentro de un having el cual está diseñado para encontrar en la suma de las ventas que se hizo anteriormente el valor máximo que se lista en la subconsulta y así listando el registro en el resultado.

5. Indique el total de ventas por categoría de producto de cada máquina expendedora

Sentencia SQL

```
select nombre_categoria, id_maquina, sum(cantidad) as TotalVentasXCategoria
from factura
    inner join detalle using(id_factura)
        inner join producto using(id_producto)
            inner join categoria using(id_categoria)
group by id_categoria, id_maquina
order by id_categoria, id_maquina;
```

Resultado

nombre_categoria	id_maquina	TotalVentasXCategoria
Botellas de agua	4010	5
Botellas de agua	4011	5
Botellas de agua	4012	5
Botellas de agua	4030	5
Botellas de agua	4031	9

✓ 1 21:45:51 select nombre_categoria, id_maquina, sum(cantidad) as TotalVentasXCategoria from factu...

Explicación

En esta, para que salga bien, tiene que haber una agrupación por la categoría y otra por la id de máquina ya que se pretende encontrar la cantidad de ventas por categoría de cada máquina, no en general. El resto es sencillo, se les hace join a factura - detalle - producto - categoría para tener los datos necesarios para hacer las sumatorias correctas y mostrar el resultado.

6. Indique el total de ventas por categoría de producto de cada universidad

Sentencia SQL

```
SELECT nombre_categoria, id_institucion, nombre_inst,
       sum(cantidad) AS total_ventas_universidad
FROM factura
INNER JOIN detalle using(id_factura)
     INNER JOIN producto using(id_producto)
          INNER JOIN categoria using(id_categoria)
               INNER JOIN maquina using(id_maquina)
                    INNER JOIN institucion using(id_institucion)
GROUP BY id_categoria, id_institucion
ORDER BY id_categoria, nombre_inst;
```

Resultado

	nombre_categoria	id_institucion	nombre_inst	total_ventas_universidad
►	Botellas de agua	1010	CETYS Universidad	15
	Botellas de agua	1030	Tecnologico Nacional de Mexico Campus Mexicali	27
	Botellas de agua	1040	Universidad Xochicalco Mexicali	4
	Bebidas energeticas	1010	CETYS Universidad	18
	Bebidas energeticas	1030	Tecnologico Nacional de Mexico Campus Mexicali	3
	Bebidas energeticas	1020	Universidad Autonoma de Baja California	12
	Bebidas energeticas	1040	Universidad Xochicalco Mexicali	20

Explicación

Similar a la anterior pero esta vez la agrupación se hace con categoría y institución en vez de categoría y máquina, lo cual hace que se requieran otras tablas para hacerse correctamente pero la lógica es la misma.

Se realiza la suma de las ventas de cada categoría perteneciente a sus instituciones.

7. Indique cual cliente es quien más gastado dinero en un periodo determinado

Sentencia SQL

```
SELECT id_cliente,
concat(nombre_cliente, " ", apellido_paterno, " ", apellido_materno) AS nombre,
sum(precio_unitario * detalle.cantidad) AS total_gastado
FROM cliente
INNER JOIN factura using(id_cliente)
    INNER JOIN detalle using(id_factura)
    INNER JOIN inventario_maquina using(id_maquina, id_producto)
WHERE fecha_expedicion BETWEEN '20210212' AND '20211229'
GROUP BY id_cliente
HAVING total_gastado IN
(
    SELECT max(table_1.total_gastado)
    FROM (
        SELECT sum(precio_unitario * detalle.cantidad) AS total_gastado
        FROM cliente
        INNER JOIN factura using(id_cliente)
            INNER JOIN detalle using(id_factura)
            INNER JOIN inventario_maquina using(id_maquina, id_producto)
        WHERE fecha_expedicion BETWEEN '20210212' AND '20211229'
        GROUP BY id_cliente
    ) AS table_1
);
```

Resultado

	id_cliente	nombre	total_gastado
▶	20490687	Sarah Collins Romero Spears	724

Explicación

La lógica en este es exactamente la misma que la de la resolución de la consulta 4: Hacemos una subconsulta para encontrar el máximo del dinero gastado por cada uno de los clientes dentro de un periodo de tiempo específico, recordando que se debe agrupar por cliente para que la suma y el máximo funcionen correctamente, luego comparamos ese valor máximo utilizando un having a la misma suma de todo el dinero gastado por cada cliente para encontrar el que gasto (o los que gastaron) más.

8. Liste a todos los clientes de una universidad en particular, indique los datos completos de cada cliente (incluya el total de tarjetas vigentes y no vigentes) y la relación de compras por categoría señalando cuanto ha gastado en cada una.

Sentencia SQL

```
SELECT
    T1.id_cliente,
    T1.id_institucion,
    T1.nombre_cliente,
    T1.apellido_paterno,
    T1.apellido_materno,
    COUNT(CASE
        WHEN estatus = 'habilitada' THEN 1
    END) AS TarjetasHabilitadas,
    COUNT(CASE
        WHEN estatus = 'deshabilitada' THEN 1
    END) AS TarjetasDeshabilitadas,
    nombre_categoria,
    PuntosUsadosXCategoria
FROM
    (SELECT
        id_cliente,
        id_institucion,
        nombre_cliente,
        apellido_paterno,
        apellido_materno,
        estatus
    FROM cliente
    INNER JOIN tarjeta USING (id_cliente)
    WHERE id_institucion = 1030
    GROUP BY id_cliente, id_tarjeta) AS T1
INNER JOIN
```

```

(SELECT
id_cliente,
id_categoria,
nombre_categoria,
SUM(d.cantidad * precio_unitario) AS PuntosUsadosXCategoria
FROM factura
    INNER JOIN
detalle AS d USING (id_factura)
    INNER JOIN inventario_maquina USING (id_maquina , id_producto)
    INNER JOIN producto USING (id_producto)
    INNER JOIN categoria USING (id_categoria)
GROUP BY id_cliente , id_categoria
ORDER BY id_cliente , id_categoria) AS T2 USING (id_cliente)
GROUP BY id_categoria, id_cliente;

```

Resultado

id_cliente	id_institucion	nombre_cliente	apellido_paterno	apellido_materno	TarjetasHabilitadas	TarjetasDeshabilitadas	nombre_categoria	PuntosUsadosXCategoria
20490713	1030	Efe Southern	Bright	Pierce	1	2	Bebidas energeticas	56
20490713	1030	Efe Southern	Bright	Pierce	1	2	Refrescos mineral	159
20490713	1030	Efe Southern	Bright	Pierce	1	2	Pastelillos	52
20490714	1030	Shelby Mccullough	Oneal	Huynh	1	2	Refrescos mineral	128
20490714	1030	Shelby Mccullough	Oneal	Huynh	1	2	Galletas	55

Explicación

En esta tenemos 2 problemáticas que resolver:

La primera se trata de obtener todos los clientes y sus diferentes tarjetas dado una institución en particular para poder resolver la primera parte de esta consulta la cual trata sobre el conteo de tarjetas habilitadas y deshabilitadas.

La segunda es por los puntos usados por categoría, se deben de obtener todos los puntos usados organizados por sus diferentes categorías acorde al cliente.

Es muy importante implementar bien las agrupaciones en este ya que para resolverse eficientemente se tuvo que crear 2 diferentes consultas, una para el conteo de las tarjetas y otra para el cálculo de los puntos usados. Hubo muchos errores en las agrupaciones al principio, pero ya que se implementaron bien se pudo realizar un inner join entre las 2 consultas para correlacionar los datos de los puntos usados con los clientes de la universidad en particular y sus tarjetas.

Para terminar, se implementó un case dentro de los count para contar solamente las tarjetas habilitadas.

9. Liste todas las promociones existentes, indicando el precio del paquete, el total de los productos si se compraran de forma individual y el porcentaje de ahorro. Ordénese por Universidad y máquina expendedora.

Sentencia SQL

```
SELECT
id_promocion,
id_institucion,
nombre_inst,
id_maquina,
sum(precio_reducido) AS precio_paquete,
sum(precio_unitario) AS total_individual,
concat((((sum(precio_unitario) - sum(precio_reducido))
/ sum(precio_unitario)) * 100), "%") AS ganancia
FROM inventario_maquina
INNER JOIN detalle_promocion using(id_producto, id_maquina)
    INNER JOIN maquina using(id_maquina)
    INNER JOIN institucion using(id_institucion)
GROUP BY id_promocion, id_maquina
ORDER BY id_institucion, id_maquina;
```

Resultado

	id_promocion	id_institucion	nombre_inst	id_maquina	precio_paquete	total_individual	ganancia
▶	5010	1010	CETYS Universidad	4010	45	56	19.6429%
	5013	1010	CETYS Universidad	4010	66	83	20.4819%
	5017	1010	CETYS Universidad	4010	67	84	20.2381%
	5021	1010	CETYS Universidad	4010	69	86	19.7674%

✓ 20 20:08:54 SELECT id_promocion, id_institucion, nombre_inst, id_maquina, sum(precio_reducido),

Explicación

En relación a esta consulta, se muestra el precio del paquete junto al total si los productos se compran individualmente sin el descuento aplicado, y el respectivo porcentaje de ahorro, debido a que la mayoría de los paquetes poseen productos con una disminución de 20% con respecto al precio original, podemos ver que la ganancia se encuentra dentro del rango del 19% y 20%. Asimismo, para obtener la ganancia se realizó mediante una fórmula de uso aritmético común. Para obtener los datos se realizó intersecciones con inventario_maquina, detalle_promocion, maquina e institución, ya que en detalle_promocion se encontraría el precio unitario con descuento de las promociones y en inventario_maquina se obtiene el precio original. Por otra parte, además se agrupó por id_promocion y id_maquina debido a que necesitamos listar todas las promociones existentes con sus respectivas columnas de contraste. Finalmente, se ordena por institucion y después por máquina, en ese orden.

10. Incremente el saldo de los clientes de acuerdo a los siguientes criterios:
1. Un 10% de las compras realizadas de un producto en particular durante un periodo determinado

Sentencia SQL

```
UPDATE tarjeta t,
  (SELECT
    id_cliente,
    id_tarjeta,
    (puntos + (total_gastado * .1)) AS puntos_aumentados
  FROM
    (SELECT
      id_cliente,
      SUM(precio_unitario * detalle.cantidad) AS total_gastado
    FROM
      cliente
    INNER JOIN factura USING (id_cliente)
    INNER JOIN detalle USING (id_factura)
    INNER JOIN inventario_maquina USING (id_maquina , id_producto)
    WHERE
      fecha_expedicion BETWEEN '20210212' AND '20211229'
      AND id_producto = 159012
    GROUP BY id_cliente) AS table_1
    INNER JOIN tarjeta USING (id_cliente)
    WHERE
      estatus = 'Habilitada') AS temp
SET
  puntos = puntos_aumentados
WHERE
  t.id_tarjeta = temp.id_tarjeta;
```

Resultado

	id_tarjeta	id_cliente	fecha_expedicion	puntos	estatus
►	10020520	20490695	2022-06-11	568	Habilitada
	10020523	20490698	2022-06-23	708	Habilitada
	10020544	20490719	2022-09-15	173	Habilitada
	10020550	20490725	2022-10-09	268	Habilitada
	10020512	20490687	2022-05-10	179	Habilitada

	id_tarjeta	id_cliente	fecha_expedicion	puntos	estatus
▶	10020520	20490695	2022-06-11	573	Habilitada
	10020523	20490698	2022-06-23	726	Habilitada
	10020544	20490719	2022-09-15	175	Habilitada
	10020550	20490725	2022-10-09	283	Habilitada
	10020512	20490687	2022-05-10	197	Habilitada

Explicación

En una subconsulta, sacamos el total gastado de una parte de la factura de los clientes, la parte de un artículo específico que indicamos. Después lo unimos con tarjetas para tener un registro de la id de la tarjeta, del cliente y el aumento de sus puntos mediante un cálculo en el cual obtenemos el 10% del total gastado en los artículos y lo sumamos a los puntos existentes de sus tarjetas, para finalmente reemplazarlo en el set, dado que el id de la tarjeta en la tabla tarjeta sea igual al de la subconsulta.

10. Incremente el saldo de los clientes de acuerdo a los siguientes criterios:

2. Un 5% del saldo actual de los clientes de la universidad que más ha consumido en dinero en un periodo en particular

Sentencia SQL

```
UPDATE tarjeta t,
  (SELECT
    id_cliente
  FROM
    cliente
  WHERE
    id_institucion = (SELECT
      id_institucion
    FROM
      (SELECT
        id_institucion, MAX(table_1.total_gastado)
      FROM
        (SELECT
          SUM(precio_unitario * detalle.cantidad) AS total_gastado
        FROM
          cliente
        INNER JOIN factura USING (id_cliente)
        INNER JOIN detalle USING (id_factura)
        INNER JOIN inventario_maquina USING (id_maquina , id_producto)
        WHERE
          fecha_expedicion BETWEEN '20210212' AND '20211229'
        GROUP BY id_institucion) AS table_1) AS table_2)) AS temp
  SET
    puntos = (puntos * 1.05)
  WHERE
    t.id_cliente = temp.id_cliente
    AND estatus = 'habilitada';
```

Resultado

	id_tarjeta	id_cliente	fecha_expedicion	puntos	estatus
▶	10020512	20490687	2022-05-10	170	Habilitada
	10020513	20490688	2022-05-14	50	Habilitada
	10020514	20490689	2022-05-18	325	Habilitada
	10020515	20490690	2022-05-22	707	Habilitada
	10020516	20490691	2022-05-26	542	Habilitada
	10020517	20490692	2022-05-30	999	Habilitada

	id_tarjeta	id_cliente	fecha_expedicion	puntos	estatus
▶	10020512	20490687	2022-05-10	179	Habilitada
	10020513	20490688	2022-05-14	53	Habilitada
	10020514	20490689	2022-05-18	341	Habilitada
	10020515	20490690	2022-05-22	742	Habilitada
	10020516	20490691	2022-05-26	569	Habilitada
	10020517	20490692	2022-05-30	1049	Habilitada

Explicación

Esta consulta requiere 4 diferentes subconsultas acorde a nuestro juicio:

- Una para calcular el total gastado entre todas las universidades dentro de un periodo de tiempo.
- La segunda para obtener el máximo junto con el id de la institución.
- Otra para obtener solo la columna de la institución (la cual nada mas tenia un registro).
- Y por último una cuarta para sacar todas las ids de los clientes pertenecientes a esa institución obtenida.

Finalmente tan solo le decimos a los puntos que se multipliquen por 1.05 de acuerdo a la condición en la cual el id cliente de la tarjeta sea igual al de la subconsulta y que su tarjeta esté habilitada.

10. Incremente el saldo de los clientes de acuerdo a los siguientes criterios:
3. Un 2% del saldo actual de los clientes a los clientes que han comprado diariamente en un periodo en particular.

Sentencia SQL

```
UPDATE tarjeta t,  
(  
    SELECT id_cliente  
    FROM factura  
    WHERE fecha_expedicion IN (20210212, 20210213)  
) table_1  
SET puntos = (puntos * 1.02)  
WHERE t.id_cliente = table_1.id_cliente  
AND t.estatus = "Habilitada";
```

Resultado

	id_tarjeta	id_cliente	fecha_expedicion	puntos	estatus
▶	10020512	20490687	2022-05-10	170	Habilitada
	10020562	20490687	2018-05-08	323	Deshabili...
	10020612	20490687	2019-09-28	816	Deshabili...
	10020513	20490688	2022-05-14	50	Habilitada
	10020563	20490688	2018-05-18	853	Deshabili...
	10020613	20490688	2019-10-08	644	Deshabili...

	id_tarjeta	id_cliente	fecha_expedicion	puntos	estatus
▶	10020512	20490687	2022-05-10	173	Habilitada
	10020562	20490687	2018-05-08	323	Deshabili...
	10020612	20490687	2019-09-28	816	Deshabili...
	10020513	20490688	2022-05-14	51	Habilitada
	10020563	20490688	2018-05-18	853	Deshabili...
	10020613	20490688	2019-10-08	644	Deshabili...

✔ 191 21:22:22 UPDATE tarjeta t, (SELECT id_cliente FROM factura WHERE:

Explicación

Con respecto a la sentencia, se actualiza los puntos de las tarjetas de la tabla tarjeta donde los clientes hayan comprado los días 12 y 13 de febrero de 2022, para eso se utilizó una subconsulta dentro del UPDATE para obtener el id_cliente que hayan comprado diariamente en esas fechas, después se explica en el SET que los puntos de los clientes obtenidos en la subconsulta será igual al mismo saldo pero con 2% de

aumento, por lo que se decidió multiplicarse por 1.02 para obtener el precio con el incremento agregado. Finalmente, en la sección WHERE se estableció que se hagan las actualizaciones de los clientes de la tabla tarjeta, que coincidan con los clientes obtenidos en la subconsulta, así como que el estatus de la tarjeta debe estar habilitada. En los resultados obtenidos se puede ver un contraste de los puntos de las tarjetas pertenecientes a los clientes con identificación 20490687 y 20490688.

11. Traslade el saldo de las tarjetas inactivas a las activas de cada cliente. (las tarjetas inactivas deben quedar en cero y las activas concentrar todo el saldo que hubiera entre activas e inactivas), (considere que cada cliente solo tiene una tarjeta activa)

Sentencia SQL

```
UPDATE tarjeta t,
(
    SELECT id_tarjeta, (puntos + saldo_inactivo) AS puntos
    FROM tarjeta
    INNER JOIN
    (
        SELECT id_cliente, sum(puntos) AS saldo_inactivo
        FROM tarjeta
        WHERE NOT estatus="Habilitada"
        GROUP BY id_cliente
    ) AS table_1 USING (id_cliente)
    WHERE estatus="Habilitada"

    UNION

    SELECT id_tarjeta, (puntos * 0) AS puntos
    FROM tarjeta
    WHERE NOT estatus="Habilitada"
) temp
SET t.puntos = temp.puntos
WHERE t.id_tarjeta = temp.id_tarjeta;
```

Resultado

	id_tarjeta	id_cliente	fecha_expedicion	puntos	estatus
▶	10020512	20490687	2022-05-10	1309	Habilitada
	10020513	20490688	2022-05-14	1547	Habilitada
	10020514	20490689	2022-05-18	1858	Habilitada
	10020515	20490690	2022-05-22	2545	Habilitada
	10020516	20490691	2022-05-26	2085	Habilitada
	10020517	20490692	2022-05-30	2164	Habilitada

	id_tarjeta	id_cliente	fecha_expedicion	puntos	estatus
	10020560	20490735	2022-11-18	1447	Habilitada
	10020561	20490736	2022-11-22	1619	Habilitada
	10020562	20490687	2018-05-08	0	Deshabili...
	10020563	20490688	2018-05-18	0	Deshabili...
	10020564	20490689	2018-05-28	0	Deshabili...

✓ 16 19:50:05 UPDATE tarjeta t, (SELECT id_tarjeta, (puntos + saldo_inactivo) AS puntos FROM tarjeta INNER JOIN (SELEC...

Explicación

Con relación a esta sentencia, se hace una actualización del saldo de las tarjetas de manera general; el saldo de las tarjeta inactivas de un cliente pasa directamente a la tarjeta activa, y por ende, las tarjetas desactivadas de dicho cliente, quedan sin saldo. Para eso se menciona que se actualice los registros de la tabla tarjeta, con base a la subconsulta dada.

Básicamente, la subconsulta hace la unión de dos consultas, la primera consulta se encarga de brindar las tarjetas habilitadas de los clientes con el saldo de las inactivas agregado. Esto se hace mediante la suma del saldo de las tarjetas inactivas del cliente, esto se agrupa por id_cliente para que me muestre el total de saldo de las tarjetas inactivas por cliente, y por ende, ese saldo se le sumará a la única tarjeta que tiene habilitada el cliente; eso en sí nos genera las tarjetas con el saldo agregado de las inactivas.

En la misma subconsulta, la segunda consulta nos genera todas las tarjetas de los clientes que estén inhabilitadas, pero el saldo de estas se dejará en 0, por eso el saldo es multiplicado por 0.

Finalmente, en la sección de SET, se muestra que los puntos de las tarjetas en la tabla serán ahora los puntos de la "tabla temporal" generada por la subconsulta, donde el identificador de ambas tablas anteriormente mencionadas sean iguales.

III. Conclusiones personales.

A. Sofia Perez Garcia

Mediante el análisis de cada consulta a completar sigo notando el hecho de una problemática con las subconsultas pero con la ayuda de mis compañeros he logrado entender las cuales se resolvieron sin problema alguno, espero seguir mejorando.

B. Johan Antonio Gurrola Bernal

Realizando consultas logre finalmente entender unos conceptos que me faltaban como el group by además de implementar un nuevo tipo de comando para mi: Update, el cual tiene una sintaxis muy diferente a los select lo cual me confundió bastante cuando se tenían que hacer las consultas de la problemática 10 y 11.

C. Alejandro Lira

Aprendí a cambiar el tipo de dato que puede aceptar una llave foránea como el debido uso y diferenciación de como realizar un 'on update cascade' y cómo utilizar un 'on delete set null' , como realizar una importación csv como realizar un insert a través de una consulta, cuento con un mejor dominio de los comandos en sql, la utilización de el inner join para poder tomar datos de otras tablas teniendo en consideración lo que se necesita para su debido uso.

D. Luis Rodrigo Barba Navarro

Mediante la realización de estas consultas, me pude percatar de la complejidad que se puede llegar a presentar en la creación de las consultas con una simple definición de problema, esto ya que a veces era necesario acceder desde un extremo de relación de una tabla a otra, y en ocasiones las consultas llegaron a ser algo extensas. Asimismo, durante la realización de la práctica, aprendí cómo utilizar la importación de los archivos de tipo CSV a las tablas especificadas en la base de datos; la verdad es que resultó demasiado eficiente hacer eso en contraste de hacer inserción por inserción manualmente. Otro aspecto a destacar es que, pude darme cuenta que en un UPDATE se puede colocar subconsultas, era algo nuevo para mi, pero tenía sentido de que algo así se podría hacer. En definitiva, esta práctica me ayudó bastante al momento de practicar mis conocimientos sobre lenguaje SQL.

IV. Conclusión general.

- A. Si bien desde un inicio se nos hizo fácil, conforme se fue avanzando tuvimos que buscar ayuda entre nosotros y en diversas fuentes para poder lograr el objetivo principal, pasando desde hacer todo por partes hasta incluso pasarnosla preguntando acerca de las problemáticas que surgían durante el desarrollo de la actividad; con el surgimiento de algunos problemas de incompatibilidad en las descripción de las consultas en cuestión de nuestra estructura de la base de datos, se tuvo que hacer adaptaciones para que se completarán de manera satisfactoria, por lo que nos redujo el tiempo, y nos percatamos que el cliente puede cambiar de parecer con respecto a su problemática (algo muy común si no preguntamos debidamente sobre qué es lo que quiere realmente). Sin más que hablar al respecto esperamos que nuestro trabajo sea de buen agrado, haya servido para aumentar nuestro

conocimiento relacionado a consultas de SQL y nos guíe por el buen camino del conocimiento.