<u>Challenge Engineer - Primera Parte – SQL</u>

Para este challenge diseñé un DER (Diagrama de Entidad – Relación) que responde al modelo del negocio ecommerce solicitado.

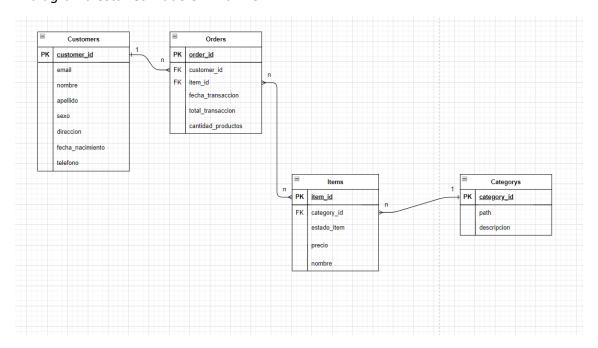
Customers: Representa a los usuarios del sitio de ecommerce. Contiene información personal como nombre, apellido, email, dirección, etc. Cada Customer puede realizar múltiples Orders.

Orders: Refleja las transacciones generadas dentro del sitio. Cada compra realizada por un Customer se registra como una Order. Cada Order contiene información sobre la fecha de la transacción, el Customer que realizó la compra y los Items comprados.

Items: Contiene información sobre los productos publicados en el marketplace. Cada Item está asociado a una Category y puede tener múltiples Orders relacionadas si ha sido comprado por los Customers.

Categorys: Describe las categorías de los Items. Cada Item está asociado a una Category.

El diagrama está realizado en Draw.io



Customers cuenta con una PK que se relaciona con la FK customer_id de Orders.

Items posee una PK que se relaciona con la FK item_id de Orders.

Orders cuenta con su propia PK.

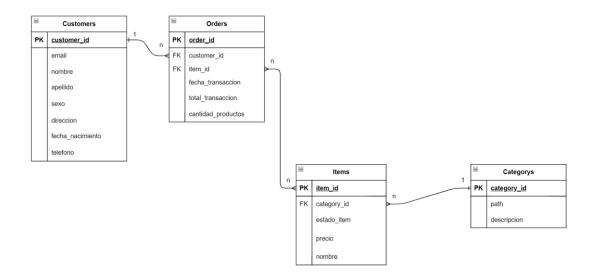
Categorys cuenta con una PK que se relaciona con la FK category id de Items.

En cuanto a la relación y la cardinalidad de estas 4 entidades destaco lo siguiente.

Customers tiene una relación de 1 a muchos con Orders, ya que un comprador puede hacer múltiples ordenes, pero una orden pertenece a un solo comprador.

Orders e Items tienen una relación de mucho a muchos ya que una orden puede tener varios ítems, y un item puede estar en múltiples órdenes.

Item y Category tienen una relación de muchos a 1, ya que un item puede tener una sola categoría, pero una categoría puede tener diferentes ítems.



Enlace del drwai.io

https://viewer.diagrams.net/?tags=%7B%7D&highlight=0000ff&edit=blank&layers=1& nav=1&title=dise%C3%B1o DER.drawio#R7Z1db6M4FIZ%2FTS6niiGQ9HKStjurabWjdg WZvao84CTWAI6MO03669cOJh8YEpMJkNaWKjU4xgGf9%2FjxxzH03Em8%2FlvCxfyBhC jqOf1w2XNveo4DHNfj%2F0TKKkvxfZkwoziUmbYJT%2FgNycS%2BTH3BIUr3MjJCloYX%2 B4kBSRIUsL00SCl53c82JdH%2Bry7gDCkJTwGM1NTvOGTzLHXk9bfpXxCezfNfBn35TQzzz DIhncOQvO4loSW7lwmTl%2FgN0RgmKGH8mwdlfvHa827njlk7%2Fdxz7vjfVOS%2BmhE yixBc4PQqIDFPDlKe5W4KYxyJat4paCwL4j%2Fn3vbcCSWEZZ%2Fi5QRFwla5GbJruqv4dl MPVJSrccKPp4gN07dPX2%2BHD2T8No2%2Bv3365Eij%2F4bRi6zgyUvKSIxoKmuJrfKq5x W2EB8Z%2FCmSximDlEmFuH2ewG3OIE747bk3YH0cRXCR4nX2LGWOo%2FAersgLywvKj 8ZTvEThYyYQkZdr5Z4XJg5F4aKqn%2BTFiK9hhGcJ%2Fxzw%2Bxe%2FOKYo5ddyD1Mmc8 xZHMmP8i4RZWhZWX9gYxXuPYjXAaMrnkWe4EuNSL8BA3n8ulUhGMm0%2BY4CneFAql %2FKarYpemst%2FkEarI7xfMV4h232KOQ%2BnhOK34SlIlmzu3ZcH7%2FiOOJ6%2FYJgW Egak3W7sbYHjqIJiYgwdkISpNhbZAopWfwL6QwxmbAgWPqXN%2BZ%2FvL4m%2FSuv5 %2FFrnfBjsD3mfyI7ZROSpIxyXYkyEDfvKxImHjOykIVGaJqXT2XFi88%2FCeNKrqOAAz6i6kI KwdXUQZ7v%2FDIYKjL49rVSCPx2GYbRI2%2BbYTKLMrOtm2q4NVuJbUtre1PDxaovOivhF T%2BN1q3tHIch4o47fp1jhp4WMBCZXjmtTnDZA36hGmzHQm5NA8nCtrVWuzQY8VYqg Yy70EsSporVN9f5B0IYKUIIZGP%2BzE%2FccK8rUeQtdpZ3nHLT42R2n53pF1TjXYpqltVu7p 1VRVrFtSGja4sVLaz0m8SK3zVW8oJ1ZNBO%2B4FCnJfXPlOuzWSKCxQVoBji6NJRchHiMA 4drmPR0T06ritHh2vRUXQKA9ExUFSQkPgnRZYdGuowjx3qVKRIR%2BvsAE7n8Kgxq2kAP CrM9eHhoU5qcpNHEQ6JxYeGPozDx3WNPqfFR2P48HTx4TSEj2u112kwPjKvqI%2BPA9Y5 AR%2BF0tpoDdTOZIqWFh062qiLjpO1olVcC2IBwLLjEtgxcjoeegBg4aH6hXmDDwBUfISYoi DAJLEMOZGIceMPAGw01gVAxAHdQ0SdujAaIoZGYwGghmNNUTCHzwlvsmPMg82OR7S UYiBLbAjWJbBkoMuSpiazNqsxliU7fmHedBZw1DAshri5SGIZoqUQ4ya1XLX38Q8N7bYeR RF5E1t7X0%2B%2FqRGEa%2BmvR%2F8%2FXDodHaR%2F5%2Ft6Bir7zd7X4xoagz1Q4U 9EU77e1HPZ8O90R0%2BVXowbT%2BbjGAuURoeTR4DS%2BY6egbrMedc1ULocTmZuYS BP1FXO97VP9CJUYh5FbID2BVBEd3OP0xhF1HVOsylyWoT2IfvUp0ixtDaaA3WZk9dwbA mirZCaBDldMVrFtSEZdU7SEqR1ggDQ%2BUCkxgSnAQCpMNdHH4bkjZISH8F9J0mhDbbT FYpxIxFQ8qQjC5L2QaK7VNZcrF2dZxV9fJJIvzAPJaDkYUWMcD%2B3LKkrFfNgUhlpYWHSP kyGncOkTtyFATCpMtiHh0lJ4EUAE4ZDGD4vKAlfAkaqw6iMpEmVVoyjiafGavzNUGyj7vYF4 RWepj0oe%2BzMsMSk4Lqp5t9zbDeghag7r0IYB6LuynTQWC%2FAU4MkzI66y%2FzCvG6 Ap0ZJvJ%2F1rU7j7qoUc6AzMDhvZ0CnuDYkZCMm2hhZHkFKWdxdu0ixERMlbmEgUdSli YD%2F4IzQ1fugykWoxDyK2PnJC6BIWdxduxSxs5OKU5jHkLwzs%2Fs%2BBu7YIXkWgxML EA2JGAcQX52TtABpHSClT9Yul8igKYL4NWY4Pz5B%2FApzHWnzD1mnPkGKpbXRGqgTnA uKAmwfJKGjjprwOF0tWsW1EmtnJ7EugR6dz2KBOq%2Bb%2Ffj4AFUPj%2FroIxBQ8r5Z% 2B14ffX2YN%2FxQF9ImctrThkXsi2KUzzIdCosoC4oDjW339C3%2B2wiLyJykVlhEq8GRvgp %2Fs8MifEPx77%2F3RaxOQyOqVGNcnKRvF7XaGFUewUrnjyTy7aKW4hTmMSWfx9idko T8Gi%2BcJBehDePIMbSrWRdAjs7fMT60i1mKUxhIDnUxK0RpQPHC7vjVIIhxABlpvOQNhT OUDxJ5tWK2ekQRZFxTt9tvxigJP1O6tt%2FtYwyTVY6VbWqmG57xDkd75Nk5RkvMfkj7is %2F%2FifQrTx7dLHey3eQc4hdFVz92D3bOEofb09ZH%2BXnZnYrbq1SPTErJCw3QgXrM1 4NYTrqqjBrP2QT9Em1sEum67n%2FvX%2FIBAX4TuN1K28kfeJSPevyC6LJblWdtBacU5Hq FgoaFgrKqUAricoCrnWyyO1B5wW7hgh35PrqtI2QIntctNCZ6z%2BYWMvWIX5yo8VP86Y x%2BkT8n86hfaOysatQv3GFBZsXwAl2%2F8Iu7Od1BI37hFS9Y9jirrssr%2BKvreLXuwxkeK

X9QKN8%2Fc%2F7CuykG%2Fb38zbQDIPf7MzcEGQlPagfeJx893XZAYymh0XbAr5BZ%2FX agUJBfXJWsaAfOJ111OKAOAnjPme13qpVlXtG%2FxgGMPssvYt47Xyt6vf67M0bYH9iLsTt 8YSTNZhXAWsuU%2FEKFYUTJyKKn068%2Fsqe%2BX%2Bw1qF36uoPAP4wYUnva5phj6B a8Kl9J78wc%2BVuojDSHUwit2MxWdWcOtbECxphjpHQZO%2FcOdfxjjjmcQs%2B0LNy0Z XOURBkaYw6vyJ7mvIMfUkLYbkeMwsX8gYRI5Pgf

Generación del script DDL

Generación del script DDL para la creación de cada una de las tablas representadas en el DER, previamente cree una base de datos ecommerse y la utilicé para realizar ahí la creación de las tablas.

- -- SCRIPT DDL para la creación de mi base datos ecommerce en donde voy a trabajar CREATE DATABASE ecommerce;
- -- SCRIPT DDL para usar la base de datos USE ecommerce;
- -- Script DDL para la creación de la tabla Customers

```
customer_id INT PRIMARY KEY,
email VARCHAR(100),
nombre VARCHAR(50),
apellido VARCHAR(50),
sexo CHAR(1),
direccion VARCHAR(100),
fecha_nacimiento DATE,
telefono VARCHAR(20)
);
```

```
-- Script DDL para la creación de la tabla Categorys
CREATE TABLE Categorys (
  category_id INT PRIMARY KEY,
  nombre VARCHAR(100),
  descripcion VARCHAR(255)
);
-- Script DDL para la creación de la tabla Items
CREATE TABLE Items (
  item_id INT PRIMARY KEY,
  category_id INT,
  nombre VARCHAR(100),
  precio DECIMAL(10, 2),
  estado VARCHAR(20),
  FOREIGN KEY (category_id) REFERENCES Categorys(category_id)
);
-- Script DDL para la creación de la tabla Orders
CREATE TABLE Orders (
  order_id INT PRIMARY KEY,
  customer_id INT,
  item_id INT,
  fecha_transaccion DATE,
  total_transaccion DECIMAL(10, 2),
  cantidad_productos INT,
  FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),
  FOREIGN KEY (item_id) REFERENCES Items(item_id)
);
```

Dato importante: Es importante el orden en que creo las entidades y tablas, ya que si ejecutara primero la creación de la tabla Orders por ejemplo, antes de crear la de Customer o Items, marcaria un error, dado a que necesito las otras tablas como referencias para las Foreign Key.

Datos complementarios:

El primer script de create, crea la tabla Customers con las siguientes columnas:

- customer id: Identificador único entero para cada cliente (clave primaria).
- email: Dirección de correo electrónico del cliente.
- nombre: Nombre del cliente.
- apellido: Apellido del cliente.
- sexo: Sexo del cliente ('M' para masculino, 'F' para femenino).
- dirección: Dirección postal del cliente.
- fecha_nacimiento: Fecha de nacimiento del cliente.
- telefono: Número de teléfono del cliente.

El segundo script de create, crea la tabla Categorys con las siguientes columnas:

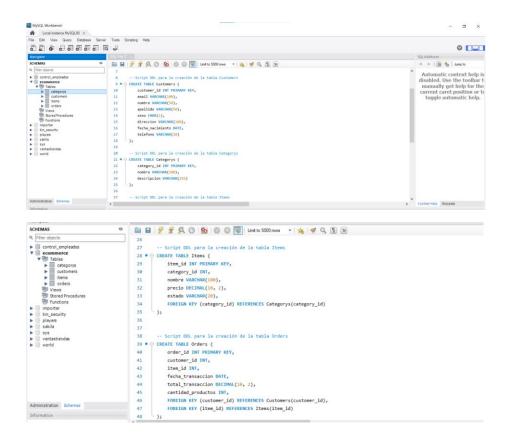
- category id: Identificador único entero para cada categoría (clave primaria).
- nombre: Nombre de la categoría.
- descripcion: Descripción breve de la categoría.

El tercer script de create, crea la tabla Items con las siguientes columnas:

- item id: Identificador único entero para cada item (clave primaria).
- category_id: Identificador de la categoría a la que pertenece el item (clave foránea que referencia a la tabla Categorys).
- nombre: Nombre del item.
- precio: Precio del item (en formato decimal con dos decimales).
- estado: Estado del item (por ejemplo, "activo", "inactivo", "descontinuado").
- La restricción de clave foránea FOREIGN KEY (category_id) REFERENCES
 Categorys(category_id) asegura que la categoría a la que se asocia un item debe existir en la tabla Categorys.

El cuarto script de créate, crea la tabla Orders con las siguientes columnas:

- order_id: Identificador único entero para cada pedido (clave primaria).
- customer_id: Identificador del cliente que realizó el pedido (clave foránea que referencia a la tabla Customer).
- item_id : Identificador del item que contiene la orden (clave foránea que referencia a la tabla Items)
- fecha transaccion: Fecha de transacción de la orden.
- total_transaccion: Total de la transacción de la orden.
- cantidad_productos: Numero entero que muestra la cantidad de productos que contiene la orden, este atributo lo terminé agregando mas tarde luego de hacer la query para el caso2 ya que era mas eficiente para la consulta que la tabla orders presente la cantidad de productos que contiene una orden.
- La restricción de clave foránea FOREIGN KEY (customer_id) REFERENCES
 Customers(customer_id) asegura que el comprador al que se asocia una orden debe existir en la tabla Customers.
- La restricción de clave foránea FOREIGN KEY (item_id) REFERENCES
 Items(item_id) asegura que el item al que se asocia una orden debe existir en la tabla Items.



Casos que resolver

- 1. Listar los usuarios que cumplan años el día de hoy cuya cantidad de ventas realizadas en enero 2020 sea superior a 1500.
- 2. Por cada mes del 2020, se solicita el top 5 de usuarios que más vendieron (\$) en la categoría Celulares. Se requiere el mes y año de análisis, nombre y apellido del vendedor, cantidad de ventas realizadas, cantidad de productos vendidos y el monto total transaccionado.
- 3. Se solicita poblar una nueva tabla con el precio y estado de los Ítems a fin del día. Tener en cuenta que debe ser reprocesable. Vale resaltar que en la tabla Item, vamos a tener únicamente el último estado informado por la PK definida. (Se puede resolver a través de StoredProcedure)

Caso 1.

Selecciono las columnas que se mostrarán en los resultados. En este caso, selecciono el nombre, apellido y fecha de nacimiento del cliente con tres agregaciones, que son la cantidad de ventas realizadas por cada cliente (COUNT(o.order_id)), la cantidad total de productos vendidos (SUM(o.cantidad_productos) y el monto total transaccionado por cada cliente de los productos vendidos en cada orden (SUM(o.total_transaccion)).

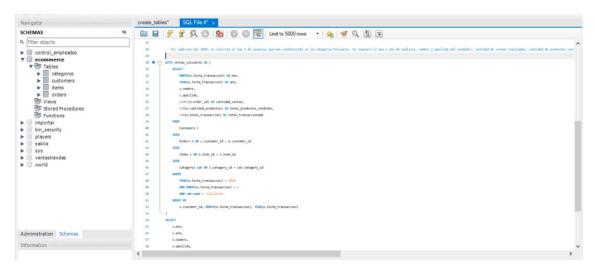
Especifico las tablas que se utilizarán en la consulta y pongo las condiciones de unión (joins). En este caso, utilizo las tablas Customers y Orders. Realizo la unión de las tablas según las claves primarias y foráneas.

Filtro los registros usando la condición WHERE para encontrar clientes cuya fecha de nacimiento coincida con el día y el mes actuales, también el año coincida con el 2020, por eso uso YEAR, DAY y MONTH para que coincidan con los datos que quiero comparar. También uso la función CURDATE para representar la fecha actual del sistema.

Utilizo GROUP BY para agrupar los resultados por el identificador único del cliente.

Utilizo HAVING COUNT(o.order_id) > 1500 para aplicar un filtro adicional y seleccionar solo a los clientes que hicieron más de 1500 órdenes.

Caso 2.



Utilizo la cláusula WITH para definir una subconsulta llamada ventas_celulares.

Dentro de esta subconsulta, selecciono el mes de la fecha de transacción (utilizando la función MONTH), el año de la fecha de transacción (utilizando la función YEAR), el nombre y apellido del cliente, la cantidad de órdenes realizadas por el cliente (utilizando la función COUNT), la suma de la cantidad de productos vendidos en todas las órdenes (utilizando la función SUM), la suma del monto total de todas las órdenes (utilizando la función SUM).

Hago un JOIN entre las tablas Customers, Orders, Items y Categorys para obtener información sobre las órdenes relacionadas con la categoría "Celulares".

Coloco el WHERE para filtrar las órdenes realizadas con las condiciones requeridas como el año 2020 y en el mes 1 (enero).

Agrupo los resultados por customer_id, mes y año.

Después realizo la consulta principal, en donde selecciono todas las columnas de la subconsulta ventas_celulares.

Ordeno los resultados por año, mes y total transaccionado en orden descendente.

Finalmente limito los resultados a los primeros 5 registros.

Caso 3.

```
SCHEMAS
                                                   🛅 🖬 | 🗲 📝 👰 🔘 | 🗞 | 🥥 🔞 📳 | Limit to 5000 rows 🕝 🏂 | 🥩 🝳 🐧 🖫
Control_empleados

Control_empleados
                                                               -- 3.Se solicita poblar una nueva tabla con el precio y estado de los Ítems a fin del día. Tener en cuenta que debe ser reproce
                                                    72
                                                             item_id INT PRIMARY KEY,
                                                    74
                                                               precio DECIMAL(10, 2),
                                                                estado VARCHAR(50),
                                                    77 fe
                                                                fecha_actualizacion DATE
       Views
Stored Procedures
Functions
    Functions
importar
kin_security
players
sakila
sys
ventastiendas
world
                                                             DELIMITER //
                                                    82 • CREATE PROCEDURE ActualizarPrecioEstadoItems()
                                                     83 ⊝ BEGIN
                                                     84
                                                                     -- Eliminar los registros anteriores de la tabla
                                                     85
                                                                  TRUNCATE TABLE PrecioEstadoItems;
                                                     87
                                                     88
                                                                   INSERT INTO PrecioEstadoItems (item_id, precio, estado, fecha_actualizacion)
                                                     89
                                                                  SELECT
                                                                        precio,
                                                                         estado,
                                                                         CURDATE() AS fecha actualizacion
```

Con esta query creo una tabla llamada PrecioEstadoItems para almacenar los precios y estados más recientes de los ítems, y define un procedimiento almacenado llamado ActualizarPrecioEstadoItems que actualiza esta tabla con los datos más recientes de la tabla Items.

Utilizo el comando CREATE TABLE para crear una nueva tabla llamada PrecioEstadoltems.

La tabla tiene cuatro columnas: item id, precio, estado y fecha actualizacion.

La columna item_id se define como la clave primaria de la tabla utilizando PRIMARY KFY.

Las columnas precio y estado son del tipo DECIMAL (10, 2) y VARCHAR(50) respectivamente.

La columna fecha actualizacion es del tipo DATE.

Previo a crear el procedure ActualizarPrecioEstadoltems, utilizo DELIMITER para cambiar el delimitador de comandos de "";" a "//". Esto lo vi necesario colocar porque MySQL interpreta cada; como el final del procedimiento almacenado o la función, en lugar de como el final de una sentencia SQL individual dentro del procedimiento, entonces generaba algunos errores si no cambiaba el delimitador.

Defino el procedimiento almacenado ActualizarPrecioEstadoItems utilizando CREATE PROCEDURE.

Dentro del procedimiento almacenado, ejecuto las siguientes acciones:

Utilizo TRUNCATE TABLE para eliminar todos los registros existentes de la tabla PrecioEstadoltems.

Luego, realizo una inserción en la tabla PrecioEstadoItems utilizando INSERT INTO.

Los valores por insertar se obtienen de la tabla Items mediante una consulta SELECT.

La subconsulta SELECT busca los precios y estados más recientes de cada ítem.

Utilizo CURDATE () para obtener la fecha actual como fecha_actualizacion.

La condición (item_id, fecha_actualizacion) IN la utilizo para seleccionar solo los registros más recientes de cada ítem.

Finalmente restauro el delimitador original utilizando DELIMITER;.