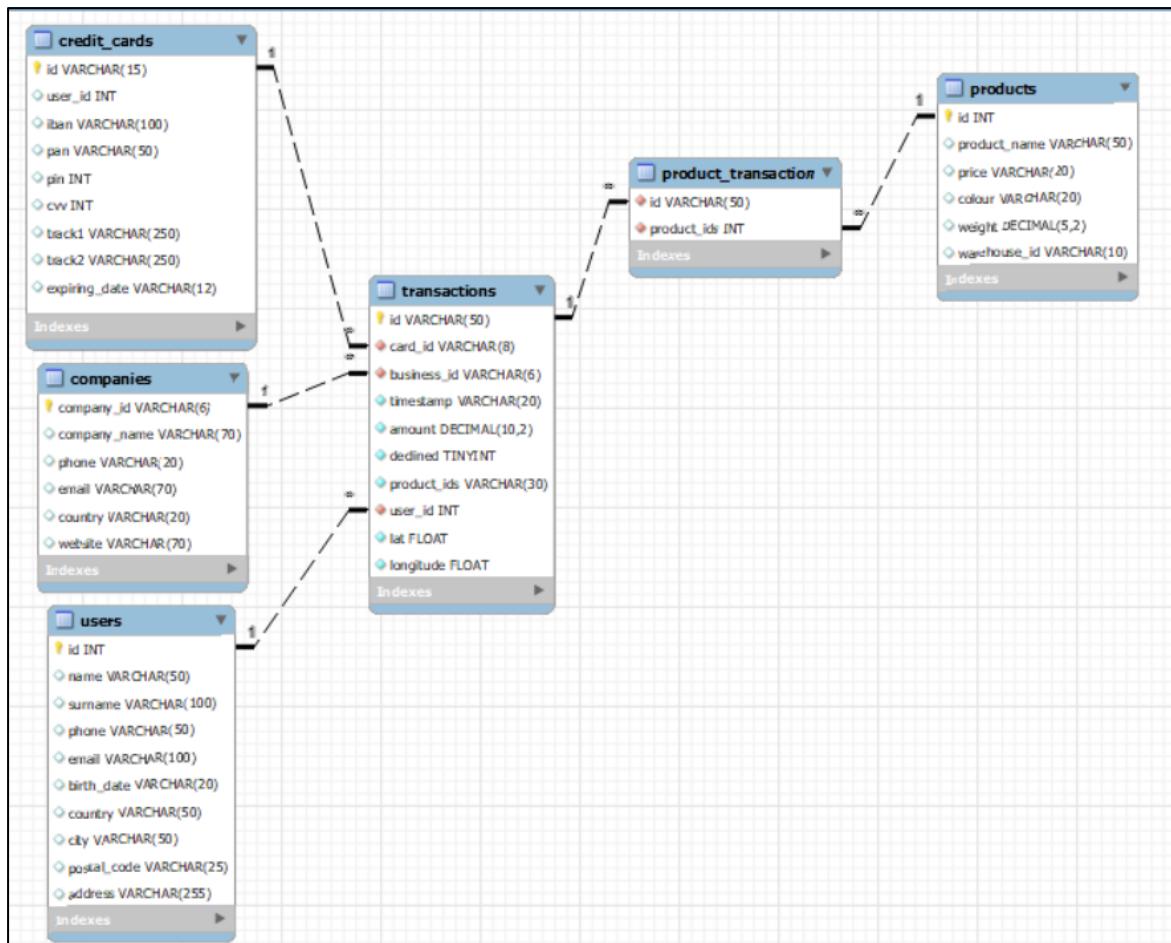


NIVEL 1



- Ejercicio 1

Realiza una subconsulta que muestre a todos los usuarios con más de 30 transacciones utilizando al menos 2 tablas.

Para este ejercicio he utilizado las tablas users y transactions, el resultado me da 4 líneas:

```

6 •  SELECT u.*, total_transactions
7   FROM users u
8   JOIN (
9     SELECT user_id, COUNT(id) AS total_transactions
10    FROM transactions
11   GROUP BY user_id
12  ) AS conteo ON u.id = conteo.user_id
13 WHERE total_transactions > 30;
14
15
  
```

result Grid | Filter Rows: Export: Wrap Cell Content:

ID	Name	Surname	Phone	Email	Birth Date	Country	City	Postal Code	Address	Total Transactions
92	Lynn	Riddle	1-387-885-4057	vitae.aliquet@outlook.edu	Sep 21, 1984	United States	Bozeman	61871	P.O. Box 712, 7907 Est St.	39
267	Ocean	Nelson	079-481-2745	aenean@yahoo.com	Dec 26, 1991	Canada	Charlottetown	85X 3P4	Ap #732-8357 Pede, Rd.	52
272	Hedwig	Gilbert	064-204-8788	sem.eget@icloud.edu	Apr 16, 1991	Canada	Tuktoyaktuk	Q4C 3G7	P.O. Box 496, 5145 Sapien Road	76
275	Kenyon	Hartman	082-871-7248	convallis.ante.lectus@yahoo.com	Aug 3, 1982	Canada	Richmond	R8H 2K2	8564 Facilisi, St.	48

- Ejercicio 2

Muestra el promedio de la suma de transacciones por IBAN de las tarjetas de crédito en la compañía Donec Ltd. utilizando al menos 2 tablas.

Lo primero que hago es encontrar el iban que corresponda con el nombre de la compañía que nos proporciona el ejercicio.

```
22 •   SELECT cc.iban                                #Por medio de esta consulta llego a encontrar el iban correspondiente a la empresa
23     FROM credit_cards cc
24   JOIN transactions t ON cc.id = t.card_id
25   JOIN companies c ON t.business_id = c.company_id
26 WHERE c.company_name = "Donec Ltd";
```

He tenido en cuenta 2 criterios:

1º Considerando únicamente las transacciones que no han sido rechazadas.

```
28   #Una vez encontrado el iban que es: 'PT87806228135092429456346' hago la busquede de la media utilizando 3 tablas.
29 •   SELECT ROUND(AVG(t.amount),2)
30     FROM transactions t
31   JOIN credit_cards cc ON cc.id = t.card_id
32   JOIN companies c ON t.business_id = c.company_id
33 WHERE cc.iban = 'PT87806228135092429456346' AND t.declined = 0 ;    #Aquí solo tengo en cuenta las transacciones aprobadas.
34
```

Result Grid | Filter Rows: _____ | Export: _____ | Wrap Cell Content:
ROUND(AVG(t.amount),2)
42.82

2º Considerando todas las transacciones indistintamente si fueron rechazadas o no.

```
35   #Una vez encontrado el iban que es: 'PT87806228135092429456346' hago la busquede de la media utilizando 3 tablas.
36 •   SELECT ROUND(AVG(t.amount),2)
37     FROM transactions t
38   JOIN credit_cards cc ON cc.id = t.card_id
39   JOIN companies c ON t.business_id = c.company_id
40 WHERE cc.iban = 'PT87806228135092429456346' ;                      #Aquí incluyo las aprobadas y rechazadas.
41
```

Result Grid | Filter Rows: _____ | Export: _____ | Wrap Cell Content:
ROUND(AVG(t.amount),2)
203.72

NIVEL 2

Crea una nueva tabla que refleje el estado de las tarjetas de crédito basado en si las últimas tres transacciones fueron declinadas y genera la siguiente consulta:

La explicación está puesta paso a paso al lado del código, considerando el criterio indicado de tomar en cuenta solo aquellas tarjetas con un mínimo de 3 transacciones hechas en total. Para luego revisar las 3 últimas transacciones si se rechazaron o no y con base en esto: Si 1 de las 3 últimas transacciones se ha aprobado, se considera la tarjeta operativa de lo contrario no.

⚠ SI QUISIERA CONSIDERAR TODAS LAS TARJETAS (INCLUYENDO AQUELLAS QUE TIENEN MENOS DEL MÍNIMO DE 3 TRANSACCIONES HECHAS) SOLO TENDRÍA QUE QUITAR LA PARTE FINAL DEL CÓDIGO (HAVING COUNT(*) = 3)

```
43 • CREATE TABLE last_card_movements AS          #1º Creo la tabla last_card_movements
44   WITH card_status AS (                         #2º Defino una CTE llamada card_status
45     SELECT card_id, timestamp, declined,        #selecciono las columnas card_id, timestamp, y declined de la tabla transactions.
46       ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) #con la función ROW NUMBER asigno un número de fila a cada transacción dentro de cada card_id,
47         AS row_num                           #ordenando las transacciones por timestamp en orden descendente.
48   FROM transactions
49 )
50   SELECT card_id,                                #2º Selecciono card_id de la CTE card_status
51   CASE                                           #utilizo CASE para calcular el estado de la tarjeta
52     WHEN SUM(declined) <= 2 THEN 'tarjeta operativa'  #basándome en la suma total de (declined) donde indico que la suma del campo declined sea <= 2
53     ELSE 'tarjeta no operativa'                  #de ser así 'tarjeta operativa' de lo contrario 'tarjeta no operativa'
54   END AS status
55   FROM card_status
56   WHERE row_num <= 3
57   GROUP BY card_id
58   HAVING COUNT(*) = 3                          #desde la CTE card_status
59   ;                                         #todo esto considerando que la row_num <=3 para que tome las 3 últimas transacciones
60                                         #agrupando por card_id
61                                         #y con HAVING hago que cuente todas las filas de cada card_id y que esta sea =3 para asegurar
62                                         #que la tarjeta tiene 3 transacciones
```

Ejercicio 1

¿Cuántas tarjetas están activas?

Esta query me da como resultado 9 tarjetas

Teniendo la tabla anterior solo tenemos que hacer un conteo de los card_id filtrando por aquellos que hayan sido considerados como 'tarjeta operativa'

```
64 •   SELECT COUNT(card_id) as 'total tarjetas operativas'      #1º Hago un COUNT de los card_id de la tabla que he creado (last_card_movements)
65     FROM last_card_movements
66     WHERE status = 'tarjeta operativa';                         # para ver cuantos encuentra.....
67                                         # filtrando solo las que el campo status sea = 'tarjeta operativa'
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
total tarjetas operativas			
9			

NIVEL 3

Crea una tabla con la que podamos unir los datos del nuevo archivo products.csv con la base de datos creada, teniendo en cuenta que desde transaction tienes product_ids. Genera la siguiente consulta:

El proceso que he seguido ha sido:

1º He creado una tabla derivada de la tabla products que haga de nexo entre transactions y products a la cual he llamado product_transaction .

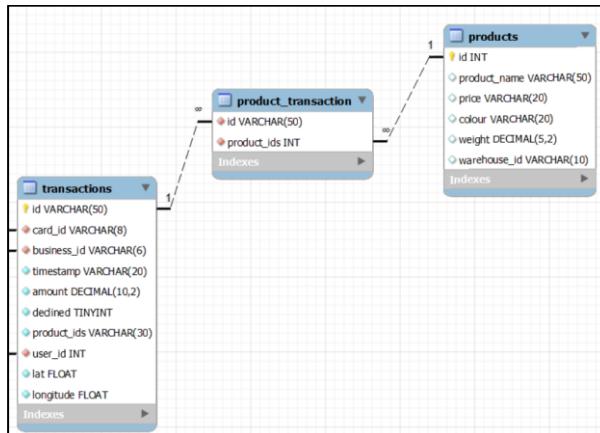
En la que he dejado únicamente dos campos que considero son los que necesito:

(Ídem, product_ids).

2º He indexado las dos columnas de esta nueva tabla para poder hacer la relación

3º He declarado como foreign key los dos campos de esta nueva tabla y hago referencia hacia las tablas con las que quiero crear la relación.

```
10 • CREATE TABLE product_transaction (
11     id varchar(50) NOT NULL,
12     product_ids int NOT NULL
13 );
14
15     #CREACION DE INDICES
16 • ALTER TABLE product_transaction      #agrego indice a product_transaction.id
17     ADD INDEX idx_id (id);
18
19 • ALTER TABLE product_transaction      #agrego indice a product_transaction.product_ids
20     ADD INDEX idx_product_ids (product_ids);
21
22     #CREACION DE FOREIGN KEY'S
23 • ALTER TABLE product_transaction
24     ADD FOREIGN KEY (id) REFERENCES transactions(id);           #creo foreign key product_transaction---->transactions
25
26 • ALTER TABLE product_transaction #ACHTUN...!!!!
27     ADD FOREIGN KEY (product_ids) REFERENCES products(id);       #creo foreign key product_transaction---->products
```



Ejercicio 1

Necesitamos conocer el número de veces que se ha vendido cada producto.

Hago una Subquery donde utilizo la función ROW NUMBER , el proceso está detallado al lado del código. Resultado en 26 líneas:

```
82 • SELECT product_ids producto, COUNT(row_num) total_vendido FROM          #2º La query que está por encima muestra el product_ids 'producto' y hace un conteo de la row_num
83   (SELECT product_ids ,
84    ROW_NUMBER() OVER (PARTITION BY product_ids ) AS row_num           #1º Hago una subquery seleccionando product_ids don una función
85    FROM product_transaction) dd                                     # ROW NUMBER para que vaya generando un conteo de las filas por cada product_ids
86   GROUP BY product_ids                                         #3º Agrupo el resultado por el campo product_ids
87   ORDER BY producto ASC                                         #4º Y lo ordeno para que sea mas visual
88 ;
89
90
```

result Grid | Filter Rows: Export: Wrap Cell Content:

producto	total_vendido
1	61
2	65
3	51
5	49
7	54

Result Grid | Filter Rows: Export: Wrap Cell Content: