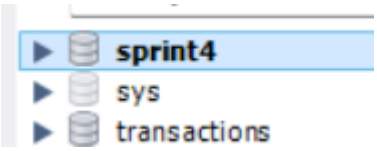
Tasca S4.01. Creació de Base de Dades

Nivel 1

Preliminares

Primero cree un esquema nuevo para este sprint:



Luego, las tablas necesarias para importar los datos CSV:

```
CREATE TABLE `transaction` (
  `id` varchar(40) NOT NULL,
  `card_id` varchar(20) NOT NULL,
  `bussiness_id` varchar(10) NOT NULL,
  `timestamp` timestamp NULL DEFAULT NULL,
  `amount` decimal(10,2) DEFAULT NULL,
  `declined` int DEFAULT NULL,
  `product_ids` varchar(100) NOT NULL,
  `user_id` int NOT NULL,
  `lat` float DEFAULT NULL,
  `longitude` float DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id` (`id`),
  KEY `card_id` (`card_id`),
  KEY `bussiness_id` (`bussiness_id`),
  KEY `product_ids` (`product_ids`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `fk_transaction_american_users_id` FOREIGN KEY (`user_id`) REFERENCES `user` (`id`),
  CONSTRAINT `fk_transaction_bussiness_id` FOREIGN KEY (`bussiness_id`) REFERENCES `company` (`company_id`),
  CONSTRAINT `fk_transaction_card_id` FOREIGN KEY (`card_id`) REFERENCES `credit_card` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

```
CREATE TABLE `credit_card` (
  `id` varchar(20) NOT NULL,
  `user_id` int NOT NULL,
  `iban` varchar(50) DEFAULT NULL,
  `pan` varchar(50) DEFAULT NULL,
  `pin` int NOT NULL,
  `cvv` int NOT NULL,
  `track1` varchar(255) DEFAULT NULL,
  `track2` varchar(255) DEFAULT NULL,
  `expiring_date` varchar(25) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id` (`id`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `fk_credit_card_user_id` FOREIGN KEY (`user_id`) REFERENCES `user` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

```sql
CREATE TABLE `company` (
  `company_id` varchar(20) NOT NULL,
  `company_name` varchar(25) NOT NULL,
  `phone` varchar(100) DEFAULT NULL,
  `email` varchar(50) DEFAULT NULL,
  `country` varchar(20) NOT NULL,
  `website` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`company_id`),
  UNIQUE KEY `company_id` (`company_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

```sql
CREATE TABLE `user` (
  `id` int NOT NULL,
  `name` varchar(10) NOT NULL,
  `surname` varchar(10) NOT NULL,
  `phone` varchar(14) DEFAULT NULL,
  `email` varchar(25) DEFAULT NULL,
  `birth_date` varchar(30) DEFAULT NULL,
  `country` varchar(25) DEFAULT NULL,
  `city` varchar(20) DEFAULT NULL,
  `postal_code` int DEFAULT NULL,
  `adress` varchar(25) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

```sql
CREATE TABLE `product` (
  `id` varchar(100) NOT NULL,
  `product_name` varchar(30) NOT NULL,
  `price` varchar(20) NOT NULL,
  `colour` varchar(25) DEFAULT NULL,
  `weight` float DEFAULT NULL,
  `warehouse_id` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Ya que en la tabla transactions puede haber varios product_ids por cada transacción, por ejemplo:

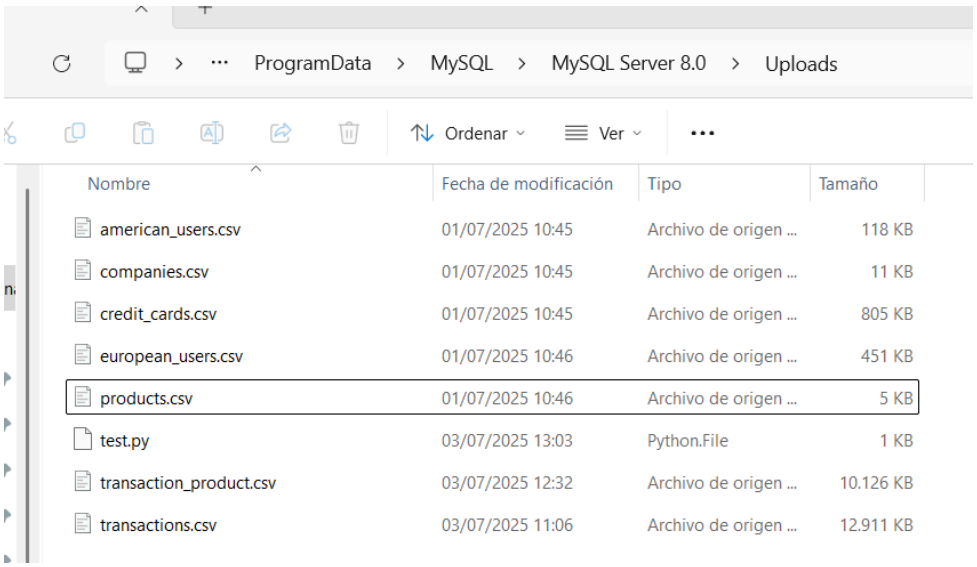
```
;75, 73, 98;
```

Cree una tabla intermedia, de muchos a muchos, entre transaction y product:

```sql
CREATE TABLE `transaction_product` (
  `id` int NOT NULL AUTO_INCREMENT,
  `transaction_id` varchar(40) NOT NULL,
  `product_id` varchar(100) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `uq_transaction_product` (`transaction_id`,`product_id`),
  KEY `fk_product_id` (`product_id`),
  CONSTRAINT `fk_product_id` FOREIGN KEY (`product_id`) REFERENCES `product` (`id`),
  CONSTRAINT `fk_transaction_id` FOREIGN KEY (`transaction_id`) REFERENCES `transaction` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=262141 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Para importar los archivos CSV, evitando usar el wizard, seguí estos pasos:

- Tuve restricciones de seguridad, que no supe suprimir, a la hora de importar los CSV directamente desde la interfaz gráfica de Workbench. Así que añadí los CSV a la carpeta Uploads (carpeta propia de MySQL):



- Ejecuté la siguiente línea, para trabajar con MySQL en CMD:
  "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysql.exe" -u root -p --local-infile=1

- Y ejecuté los INSERT desde la propia carpeta de Uploads, por ejemplo:

LOAD DATA LOCAL INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv'

INTO TABLE product

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

IGNORE 1 LINES;

```
mysql> LOAD DATA LOCAL INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv'
,' LINES TERMINATED BY '\n' IGNORE 1 LINES;
Query OK, 100 rows affected (0.05 sec)
Records: 100  Deleted: 0  Skipped: 0  Warnings: 0

mysql>
```

- Usé el siguiente script para transformar los registros de **transactions** que tengan varios products [xex: {transaction 1: products: (30, 20)}] en registros únicos con ids repetidas [xex: {transaction 1: products: (30)}, {transaction 1: products: (20)}]:

```python
import pandas as pd

transaction = pd.read_csv("transaction.csv")
company = pd.read_csv("company.csv")
credit_card = pd.read_csv("credit_card.csv")
transaction_product = pd.read_csv("transaction_product.csv")
product = pd.read_csv("product.csv")

# --- Limpieza de espacios en claves ---
transaction['business_id'] = transaction['business_id'].astype(str).str.strip()
company['company_id'] = company['company_id'].astype(str).str.strip()

transaction['card_id'] = transaction['card_id'].astype(str).str.strip()
credit_card['id'] = credit_card['id'].astype(str).str.strip()

transaction_product['transaction_id'] = transaction_product['transaction_id'].astype(str).str.strip()
transaction['id'] = transaction['id'].astype(str).str.strip()

transaction_product['product_id'] = transaction_product['product_id'].astype(str).str.strip()
product['id'] = product['id'].astype(str).str.strip()

# --- Uniones ---
# 1. transaction + company
df = transaction.merge(company, left_on='business_id', right_on='company_id', how='left')

# 2. + credit_card
df = df.merge(credit_card, left_on='card_id', right_on='id', how='left', suffixes=('', '_cc'))

# 3. + transaction_product
df = df.merge(transaction_product, left_on='id', right_on='transaction_id', how='left')

# 4. + product
df = df.merge(product, left_on='product_id', right_on='id', how='left', suffixes=('', '_product'))
```

```python
# 4. + product
df = df.merge(product, left_on='product_id', right_on='id', how='left', suffixes=('', '_product'))

# --- Seleccionar y renombrar ---
transaction_flattened = df[[
    'id', 'date', 'amount', 'business_id',
    'company_name', 'country',
    'card_id', 'card_number', 'expiration_date', 'card_type',
    'product_id', 'name', 'price', 'color'
]].rename(columns={
    'id': 'transaction_id',
    'date': 'transaction_date',
    'name': 'product_name',
    'price': 'product_price',
    'color': 'product_color'
})

# Mostrar primeras filas
print(transaction_flattened.head())

# Opcional: guardar a CSV
# transaction_flattened.to_csv("transaction_flattened.csv", index=False)
```

- Usé el siguiente script en python para montar un CSV, con los datos de la tabla muchos a muchos, transaction_product:

```python
import pandas as pd

input_csv = "transaction_flattened.csv"
output_csv = "transaction_product.csv"
df = pd.read_csv(input_csv)

rows = []

for _, row in df.iterrows():
    transaction_id = row['transaction_id']
    product_ids_str = str(row['product_ids']).strip()

    if ',' in product_ids_str:
        product_ids = product_ids_str.split(',')
    else:
        product_ids = product_ids_str.split()

    for pid in product_ids:
        pid = pid.strip()
            rows.append({'transaction_id': transaction_id, 'product_id': pid})

flat_df = pd.DataFrame(rows)

flat_df.to_csv(output_csv, index=False)

print(f"Archivo {output_csv} creado con {len(flat_df)} filas.")
```

Ejercicio 1

```
SELECT id, name FROM user
WHERE id IN (
        SELECT user_id FROM transaction
  GROUP BY user_id
  HAVING count(id) > 80
  )
```

```sql
SELECT id, name FROM user
WHERE id IN (
    SELECT user_id FROM transaction
    GROUP BY user_id
    HAVING count(id) > 80
    )
```

Result Grid | Filter Rows: | Edit: | Exp

id	name
185	couk,"Dec
289	hwcru@exam
318	astuw@exam
454	xgvfridxs@
NULL	NULL

Ejercicio 2

```sql
SELECT
  cc.iban,
  AVG(t.amount) AS avg_amount
FROM
  credit_card cc
  JOIN transaction t ON t.card_id = cc.id
  JOIN company c ON t.bussiness_id = c.company_id
WHERE
  c.company_name = 'Donec Ltd'
GROUP BY
  cc.iban;
```

SQL File 9* | SQL File 7*

Don't Limit

```sql
SELECT
  cc.iban,
  AVG(t.amount) AS avg_amount
FROM
  credit_card cc
  JOIN transaction t ON t.card_id = cc.id
  JOIN company c ON t.bussiness_id = c.company_id
WHERE
  c.company_name = 'Donec Ltd'
GROUP BY
  cc.iban;
```

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

iban	avg_amount
XX9114064011255863075868 05	356.246667
SK9446370242474562577506	142.960000
XX7767529178459529755556 40	257.370000
XX4138273622897193049089 90	139.590000
XX3477872460707696107803 08	240.410000
XX6887684365430908948546 02	188.580000
MK28368851538688349	439.390000

Result 3

Nivel 2

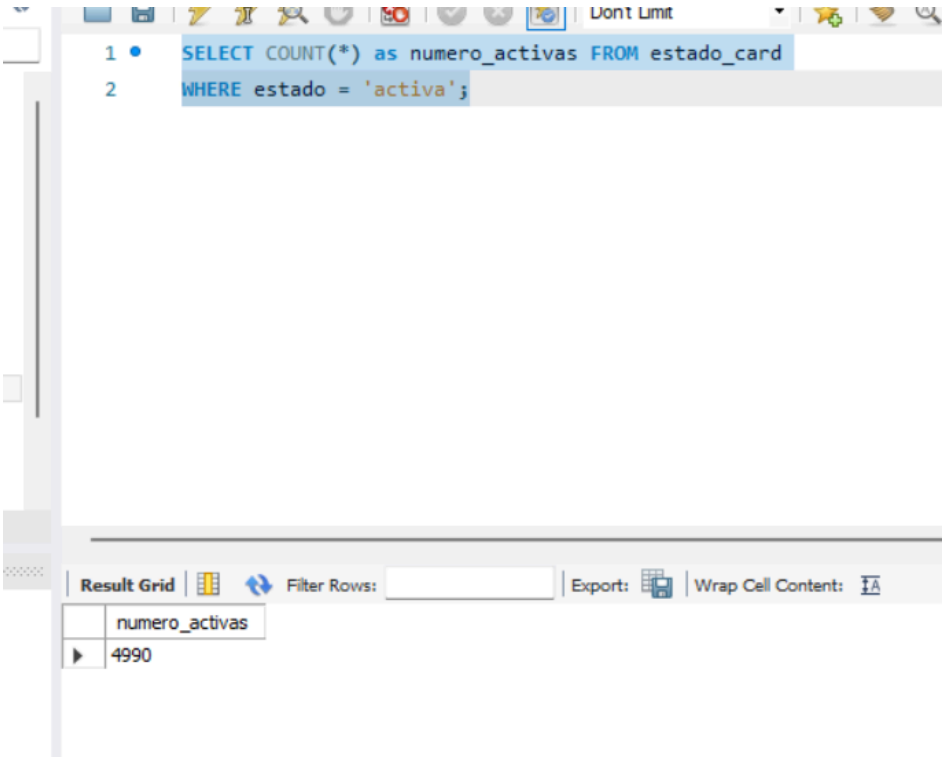
tabla estado_card:

```sql
CREATE TABLE IF NOT EXISTS estado_card (
 id VARCHAR(20) NOT NULL,
 estado ENUM('activa', 'inactiva'),
 iban VARCHAR(50) NOT NULL,
 KEY(id),
 PRIMARY KEY (id)
);
```

INSERT en la tabla estado_card, validando la condición de activa o inactiva (no se si había una solución más simple; pero me ha costado bastante):

```
INSERT INTO estado_card (
        id,
    estado,
    iban
    )
SELECT
        cc.id,
    CASE
                WHEN
                (SELECT COUNT(*)
        FROM transaction t
        WHERE t.card_id = cc.id
        AND t.declined = 1
        ORDER BY t.timestamp DESC
        LIMIT 3) = 3
                THEN 'inactiva'
                ELSE 'activa'
        END AS estado,
        cc.iban
FROM credit_card cc;
```

Ejercicio 1



Nivel 3

Entiendo que la tabla pedida corresponde a la tabla transaction_product, muchos a muchos, hecha en el Nivel 1:

DDL for sprint4.transaction_product

```
CREATE TABLE `transaction_product` (
  `id` int NOT NULL AUTO_INCREMENT,
  `transaction_id` varchar(40) NOT NULL,
  `product_id` varchar(100) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `uq_transaction_product` (`transaction_id`,`product_id`),
  KEY `fk_product_id` (`product_id`),
  CONSTRAINT `fk_product_id` FOREIGN KEY (`product_id`) REFERENCES `product` (`id`),
  CONSTRAINT `fk_transaction_id` FOREIGN KEY (`transaction_id`) REFERENCES `transaction` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=262141 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Ejercicio 1

SELECT COUNT(*) as numero_transacciones, tp.product_id FROM transaction_product tp
JOIN transaction t
ON t.id = tp.transaction_id
WHERE t.declined = 0
GROUP BY product_id
ORDER BY product_id DESC;

```
SELECT COUNT(*) as numero_transacciones, tp.product_id FROM transaction_product tp
JOIN transaction t
ON t.id = tp.transaction_id
WHERE t.declined = 0
GROUP BY product_id
ORDER BY product_id DESC;
```

Result Grid

numero_transacciones	product_id
2503	99
2557	98
2520	97
2500	96
2516	95