SPRINT 3

Descripció

En aquest sprint, es simula una situació empresarial en la qual has de realitzar diverses manipulacions en les taules de la base de dades. Al seu torn, hauràs de treballar amb índexs i vistes. En aquesta activitat, continuaràs treballant amb la base de dades que conté informació d'una empresa dedicada a la venda de productes en línia. En aquesta tasca, començaràs a treballar amb informació relacionada amb targetes de crèdit.

Nivell 1

- Exercici 1

La teva tasca és dissenyar i crear una taula anomenada "credit_card" que emmagatzemi detalls crucials sobre les targetes de crèdit. La nova taula ha de ser capaç d'identificar de manera única cada targeta i establir una relació adequada amb les altres dues taules ("transaction" i "company"). Després de crear la taula serà necessari que ingressis la informació del document denominat "dades_introduir_credit". Recorda mostrar el diagrama i realitzar una breu descripció d'aquest.

Inicialment fem una ullada a un dels registres de la taula datos_introducir_credit per a veure com son les dades i d'aquest manera assignar adequadament el tipus de camp per a cadascuna:

credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2938', 'TR301950312213576817638661', '5424465566813633', '3257', '984', '10/30/22')

Creem la taula credit_card amb els següents criteris:

credit_card					
Camp	**Tipus**	**Null**	**Clau**	**Valor per defecte**	**Extra**
id	varchar(15)	NO	PRIMARY KEY		identificador únic de la targeta de crèdit. Alfanumèric
iban	varchar(15)	YES			IBAN pot tenir fins a 34 caràcters segons la norma ISO 13616
pan	varchar(30)	YES			PAN (Primary Account Number), normalment de 16 dígits. Incrementem a 30 per precaució
pin	char(4)	YES			PIN de 4 dígits
cvv	char(3)	YES			CVV normalment de 3 dígits
expiring_date	varchar(10)	YES			Conté la data de caducitat de la targeta de crèdit

Utilitzarem la comanda CREATE TABLE IF NOT EXISTS credit_card

A destacar:

Assignarem a id com a VARCHAR(15) fent-lo coincidir amb el tipus de dada que hi ha al camp credit_card_id de la taula transaction i la establirem com a Primary KEY

El camp expiring_date l'establirem com a VARCHAR(10) ja que observem que a la taula datos_introducir_credit el format es MM/DD/YY i el format de DATE es YYYY/MM/DD

El camp **pan** el crearem com a VARCHAR(30) per a tenir un marge més llarg ja que en proves anteriors ens ha donat error al crear-lo estrictament com a CHAR(16) ja que els Primary Account Number (pan) solen tenir 16 dígits.

```sql
CREATE TABLE IF NOT EXISTS credit_card (
id VARCHAR(15) PRIMARY KEY,            -- Identificador únic de la targeta, alfanumèric
    iban VARCHAR(34) ,          -- IBAN pot tenir fins a 34 caràcters segons la norma ISO 13616
    pan VARCHAR(30) ,           -- PAN (Primary Account Number), normalment de 16 dígits. Incrementat a 30 Ha donat problemes, per tant hem i
    pin CHAR(4) ,               -- PIN de 4 dígits
    cvv CHAR(3) ,               -- CVV normalment de 3 dígits
    expiring_date VARCHAR(10)   -- Data d'expiració en format VARCHAR(10) perque les dades a introduir MM/DD/YY i donaria problemes ja que es
    );
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	20:34:29	CREATE TABLE IF NOT EXISTS credit_card (id VARCHAR(15) PRIMARY KEY, -- Id...	0 row(s) affected	0.937 sec

Fem una ullada a la taula credit_card amb DESCRIBE

```sql
DESCRIBE credit_card;
```

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

Field	Type	Null	Key	Default	Extra
id	varchar(15)	NO	PRI	NULL	
iban	varchar(34)	YES		NULL	
pan	varchar(30)	YES		NULL	
pin	char(4)	YES		NULL	
cvv	char(3)	YES		NULL	
expiring_date	varchar(10)	YES		NULL	

Result 22

Output

Action Output

#	Time	Action	Message	Duration / F
1	20:43:12	DESCRIBE credit_card	6 row(s) returned	0.000 sec /

Seguidament insertem les dades de **datos_introducir_credit** executant l'arxiu **datos_introducir_credit.sql**

```sql
-- Insertamos datos de credit_card
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (  'CcU-2938', 'TR301950312213576817
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (  'CcU-2945', 'DO26854763748537475
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (  'CcU-2952', 'BG45IVQL527105256082
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (  'CcU-2959', 'CR724247724433584153
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (  'CcU-2966', 'BG72LKTQ156276283773
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (  'CcU-2973', 'PT878062281350924294
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (  'CcU-2980', 'DE392418818830862771
```

disabled. Use the toolba
manually get help for t
current caret position o
toggle automatic help

Context Help | Snippets

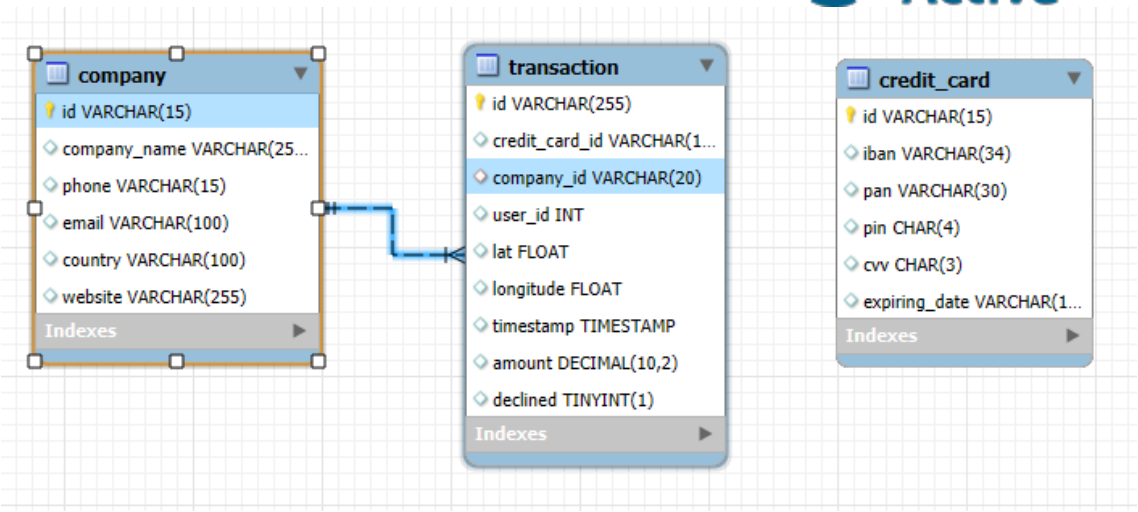
Output

Action Output

#	Time	Action	Message	Duration / Fetch
264	11:38:53	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-4779', 'FI9109...	1 row(s) affected	0.094 sec
265	11:38:53	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-4786', 'SI5170...	1 row(s) affected	0.109 sec
266	11:38:53	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-4793', 'HU952...	1 row(s) affected	0.031 sec
267	11:38:53	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-4800', 'SI9782...	1 row(s) affected	0.047 sec
268	11:38:53	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-4807', 'LB192...	1 row(s) affected	0.141 sec
269	11:38:53	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-4814', 'MR484...	1 row(s) affected	0.031 sec
270	11:38:53	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-4821', 'LT253...	1 row(s) affected	0.094 sec
271	11:38:53	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-4828', 'BG11IL...	1 row(s) affected	0.078 sec
272	11:38:53	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-4835', 'PT345...	1 row(s) affected	0.109 sec
273	11:38:53	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-4842', 'SA215...	1 row(s) affected	0.063 sec
274	11:38:53	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-4849', 'SE281...	1 row(s) affected	0.109 sec
275	11:38:53	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-4856', 'TR373...	1 row(s) affected	0.062 sec

Podem veure a pantalla que s'han incorporat 275 registres

Ara podem fer un diagrama EER per a veure la relació entre taules



Podem observar que no hi ha restricció entre la taula **transaction** i la taula **credit_card**, per tant haurem de crear la restricció de **transaction(credit_card_id)** a **credit_card(id)**

ALTER TABLE transaction ADD CONSTRAINT credit_card_transaction FOREIGN KEY(credit_card_id) REFERENCES credit_card(id);

(A partir de l'SPRINT 4 adoptaré la convenció de anomenar les Foreign Keys d'una manera més indicativa posant primer la taula de origen seguida per la taula objectiu, separat per un guió Ex. En aquest cas transaction_credit_card)

ALTER TABLE transaction

ALTER TABLE = Modifica l'estructura d'una taula existent.

transaction = És la taula que volem modificar.

Aquesta part indica que volem fer canvis a la taula transaction

ADD CONSTRAINT credit_card_transaction

ADD CONSTRAINT = Afegeix una restricció a la taula.

credit_card_transaction = Nom que donem a la restricció (aquesta és arbitrària i serveix per identificar la regla).

Donem un nom a la restricció perquè sigui més fàcil referenciar en el futur

FOREIGN KEY (credit_card_id)

FOREIGN KEY = Declara que una columna contindrà valors que han d'existir en una altra taula.

(credit_card_id) = És la columna de la taula transaction que serà una clau forana.

***credit_card_id** no pot tenir valors que no existeixin a la taula **credit_card**.*

REFERENCES credit_card(id)

REFERENCES = Indica que la clau forana ha d'existir a una altra taula.

credit_card(id) = La taula **credit_card** i la columna **id** a la qual fa referència.

*assegura que cada valor de **credit_card_id a transaction** existeixi com a **id** a **credit_card**.*

```
57 •    ALTER TABLE transaction ADD  CONSTRAINT credit_card_transaction FOREIGN KEY(credit_card_id) REFERENCES credit_card(id);
58
59
60
61
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	12:56:00	ALTER TABLE transaction ADD CONSTRAINT credit_card_transaction FOREIGN KEY(credit_card_i...	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0	2.796 sec

Ara podem veure mitjançant DESCRIBE transaction; com s'ha creat la restricció

```
62      -- Observem com queda la taula transaction amb la nova Foreign Key afegida.
63 •    DESCRIBE transaction;
64
```

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

Field	Type	Null	Key	Default	Extra
id	varchar(255)	NO	PRI	NULL	
credit_card_id	varchar(15)	YES	MUL	NULL	
company_id	varchar(20)	YES	MUL	NULL	
user_id	int	YES		NULL	
lat	float	YES		NULL	
longitude	float	YES		NULL	
timestamp	timestamp	YES		NULL	
amount	decimal(10,2)	YES		NULL	
declined	tinyint(1)	YES		NULL	

Result Grid | Form Editor

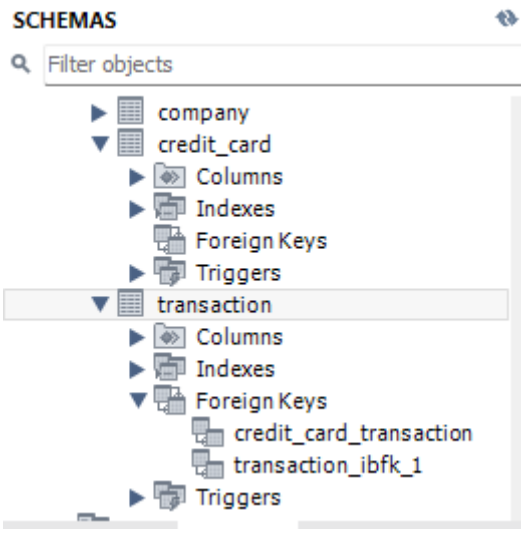
Result 3 | Read Only

Output

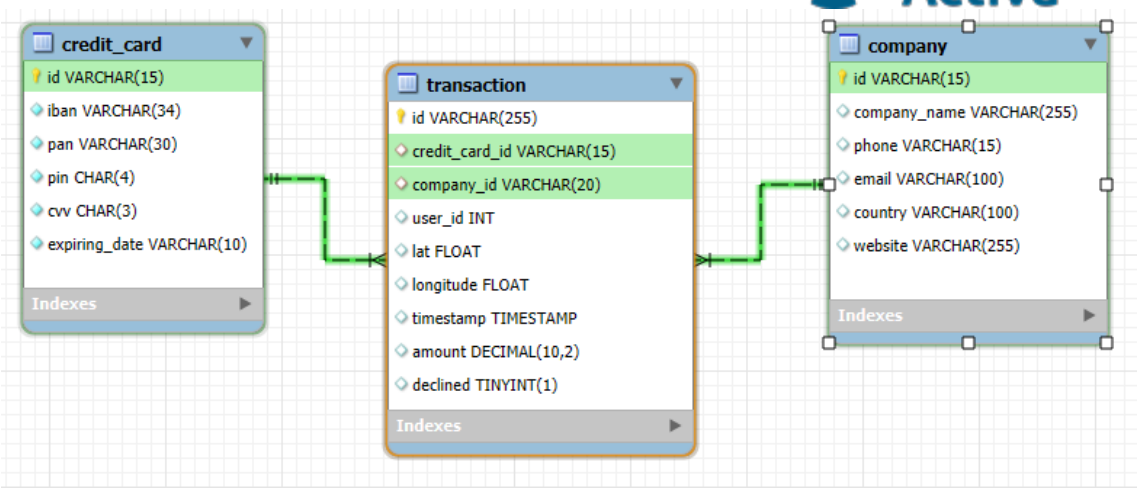
Action Output

#	Time	Action	Message	Duration / Fetch
1	12:59:35	DESCRIBE transaction	9 row(s) returned	0.016 sec / 0.000 sec

També podem observar l'esquema incloent la nova restricció **credit_card_transaction**



Ara tornem a fer el diagrama EER per a visualitzar la base de dades transactions



Relació entre **credit_card** i **transaction**:

Cada transacció (transaction) està vinculada a una targeta de crèdit a través del **camp credit_card_id**, que és una **clau forana (FK)** referenciant **credit_card(id)**.

Relacions a transaction:

credit_card_id connecta amb **credit_card(id)**.

company_id connecta amb **company(id).**

Relació **company** amb **transaction:**
Cada transacció està associada a una empresa a través de la clau forana **company_id**.

Explicació de les relacions

Transaction(credit_card_id) → credit_card(id)

Cada transacció està associada a una única targeta de crèdit.

Una mateixa targeta pot tenir múltiples transaccions. ***(Relació 1:N)***.

Transaction(company_id) → company(id)

Cada transacció està vinculada a una empresa on es realitza el pagament.

Una empresa pot rebre moltes transaccions. ***(Relació 1:N)***

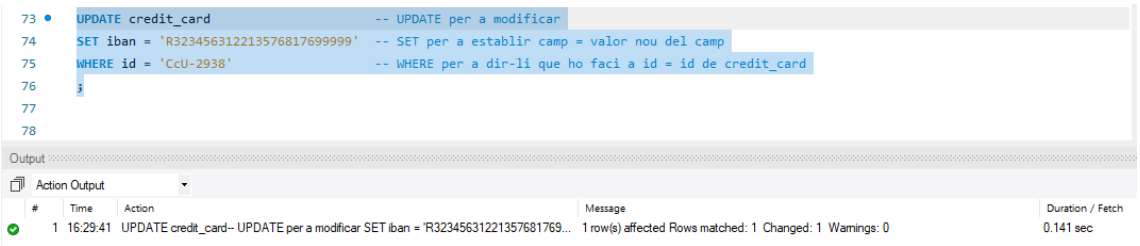
- Exercici 2

El departament de Recursos Humans ha identificat un error en el número de compte de l'usuari amb ID CcU-2938. La informació que ha de mostrar-se per a aquest registre és: R323456312213576817699999. Recorda mostrar que el canvi es va realitzar.

UPDATE credit_card → UPDATE per a modificar

SET iban = 'R323456312213576817699999' → SET per a establir camp = valor nou del camp

WHERE id = 'CcU-2938' → WHERE per a dir-li que ho faci a id = id de credit_card



Fem la comprovació per a verificar que el canvi s'ha produït correctament

```
78    -- Comprovem el resultat per a verificar el canvi
79    SELECT *
80    FROM credit_card AS cc
81    WHERE cc.id = 'CcU-2938'
82    ;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	iban	pan	pin	cvv	expiring_date
CcU-2938	R323456312213576817699999	5424465566813633	3257	984	10/30/22
NULL	NULL	NULL	NULL	NULL	NULL

credit_card 2 | Apply | Revert

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	16:39:38	SELECT * FROM credit_card AS cc WHERE cc.id = 'CcU-2938' LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

- Exercici 3

En la taula "transaction" ingressa un nou usuari amb la següent informació:

Id	**108B1D1D-5B23-A76C-55EF-C568E49A99DD**
credit_card_id	**CcU-9999**
company_id	**b-9999**
user_id	**9999**
lat	**829.999**
longitude	**-117.999**
amount	**111.11**
declined	**0**

Afegim el nou usuari amb la comanda **INSERT INTO transaction** amb els camps proporcionats

```
111   -- Ingressem el nou usuari amb la comanda INSERT INTO transaction
112   INSERT INTO transaction (id, credit_card_id, company_id , user_id, lat, longitude,  amount, declined)
113   VALUES( '108B1D1D-5B23-A76C-55EF-C568E49A99DD' , 'CcU-9999' , 'b-9999' , '9999', '829.999' ,  '-117.999', '111.11','0')
114   ;
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	17:09:24	INSERT INTO transaction (id, credit_card_id, company_id , user_id, lat, longitude, amo...	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`trans...	0.047 sec

Al executar la comanda el sistema ens retorna un error ja que no existeix el **id** = '**b-9999'** a la taula company.

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`transactions`.`transaction`, CONSTRAINT `transaction_ibfk_1` FOREIGN KEY (`company_id`) REFERENCES `company` (`id`))

Per tant primerament crearem id = '**b-9999'** a la taula **company**

INSERT INTO company (id, company_name, phone, email, country, website)

VALUES ('b-9999','company_name', 'phone', 'email', 'country', 'website');

Comprovem que id ='b-9999' ara sí existeix a company.

```
SELECT *
FROM company
WHERE id = 'b-9999'
;
```

id	company_name	phone	email	country	website
b-9999	company_name	phone	email	country	website

#	Time	Action	Message	Duration / Fetch
1	18:13:11	SELECT * FROM company WHERE id = 'b-9999' LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.015 sec

Ara sí tornem a executar la comanda per a ingressar el nou usuari

```
-- Ingressem de nou l'usuari amb la comanda INSERT INTO transaction
INSERT INTO transaction (id, credit_card_id, company_id , user_id, lat, longitude,  amount, declined)
VALUES( '108B1D1D-5B23-A76C-55EF-C568E49A99DD' , 'CcU-9999' , 'b-9999' , '9999', '829.999' ,  '-117.999', '111.11','0')
;
```

#	Time	Action	Message	Duration / Fetch
1	17:38:05	INSERT INTO transaction (id, credit_card_id, company_id , user_id, lat, longitude, amo...	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`trans...	0.047 sec

Aquest error significa que estem intentant inserir o actualitzar un registre a la taula transaction, però el valor que estem proporcionant per a **credit_card_id** no existeix a la taula **credit_card**

Creem **id = 'CcU-9999'** a la taula **credit_card**

```
-- Creem el id 'CcU-9999' a taula credit_card
INSERT INTO credit_card (id , iban , pan, pin, cvv, expiring_date )
VALUES ('CcU-9999', 'iban','pan', 'pin', 'cvv', 'dd/mm/yy' )
;
```

#	Time	Action	Message	Duration / Fetch
1	21:27:21	INSERT INTO credit_card (id , iban , pan, pin, cvv, expiring_date) VALUES ('CcU-9999...	1 row(s) affected	0.140 sec

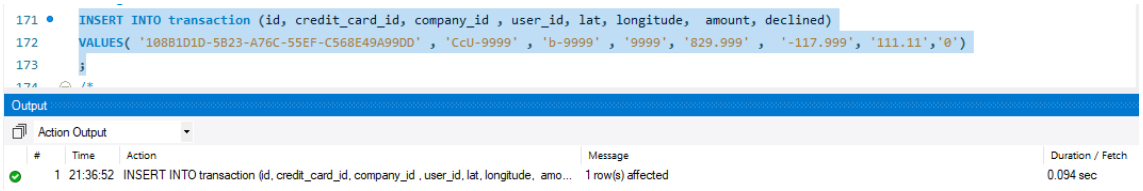
Verifiquem el registre

```
-- Verifiquem id = 'CcU-9999'
select * from credit_card
where id='CcU-9999';
```

id	iban	pan	pin	cvv	expiring_date
CcU-9999	iban	pan	pin	cvv	mm/dd/yy

#	Time	Action	Message	Duration / F
1	21:40:38	select * from credit_card where id='CcU-9999' LIMIT 0, 1000	1 row(s) returned	0.000 sec /

Ara si ingressem el registre de la transacció

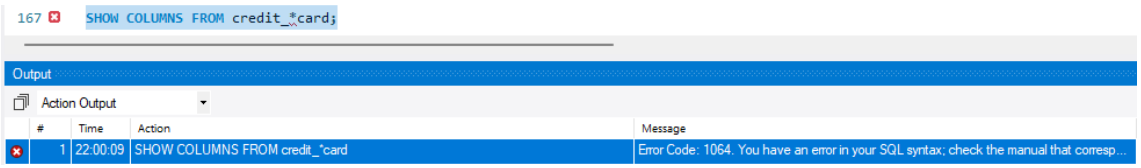


- Exercici 4

Des de recursos humans et sol·liciten eliminar la columna "pan" de la taula credit_*card. Recorda mostrar el canvi realitzat.

Abans de fer el canvi visualitzem les columnes de credit_*card

SHOW COLUMNS FROM credit_*card;



Dona un error degut a que no existeix la taula credit_*card . Hi ha un error a '*card' a l'asterix

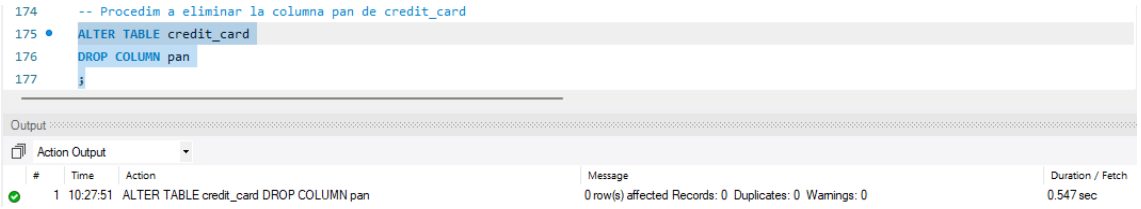
Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '*card' at line 1

No obstant, sí que existeix aquest camp en una taula similar que es **credit_card.** Per tant procedirem a eliminar la columna **pan** de **credit_card** amb la comanda

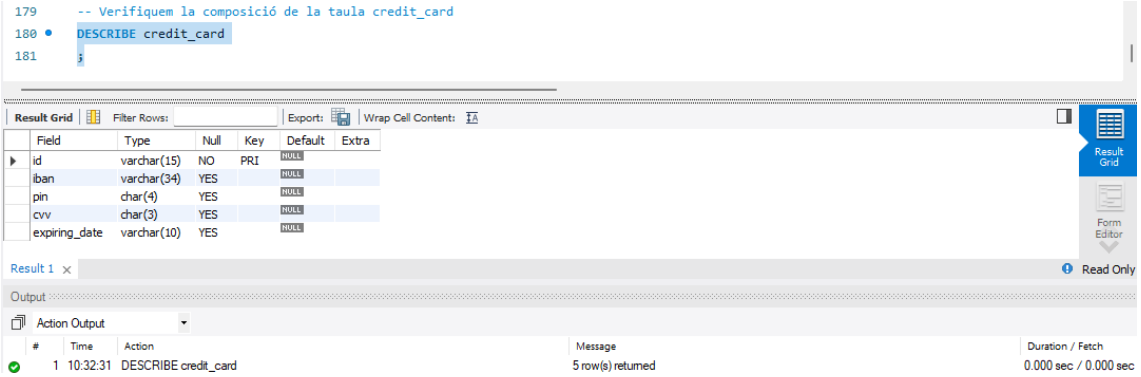
ALTER TABLE credit_card →Indica que volem modificar l'estructura de la taula **credit_card.**

DROP COLUMN pan →Esborra completament la columna.

;



Verifiquem la nova estructura de la taula credit_card amb un **DESCRIBE credit_card**



Nivell 2

Exercici 1

Elimina de la taula transaction el registre amb ID 02C6201E-D90A-1859-B4EE-88D2986D3B02 de la base de dades.

Procedim a eliminar el registre de la taula amb la comanda

DELETE FROM transaction → Indica que volem eliminar dades de la taula transaction.

WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02' Filtra la eliminació perquè només afecti el registre amb aquest id específic.

Si no afegíssim WHERE, **esborraríem tots els registres de la taula!**

Acció	Comanda SQL	Què fa?
Eliminar un registre	DELETE FROM taula WHERE condició;	Elimina una o més files però manté l'estructura de la taula.
Eliminar una columna	ALTER TABLE taula DROP COLUMN columna;	Modifica l'estructura de la taula eliminant una columna i totes les seves dades.

Exercici 2

La secció de màrqueting desitja tenir accés a informació específica per a realitzar anàlisi i estratègies efectives. S'ha sol·licitat crear una vista que proporcioni detalls clau sobre les companyies i les seves transaccions. Serà necessària que creïs una vista anomenada VistaMarketing que contingui la següent informació: Nom de la companyia. Telèfon de contacte. País de residència. Mitjana de compra realitzat per cada companyia. Presenta la vista creada, ordenant les dades de major a menor mitjana de compra.

CREATE VIEW VistaMarketing AS → Crea una vista anomenada **VistaMarketing**.

SELECT → Aquesta part defineix quines dades volem veure dins la vista.

c.company_name → El nom de l'empresa.

c.phone → El telèfon de l'empresa.

c.country → El país de l'empresa.

ROUND(AVG(t.amount),2) AS Mitjana_compra

AVG(t.amount) → Calcula la mitjana dels imports (amount) de les transaccions fetes per cada empresa.

ROUND(...,2) → Arrodoneix el resultat a **dues xifres decimals**.

AS Mitjana_compra → Li dona l'àlies "Mitjana_compra" a la columna resultant.

FROM company AS c →Agafem dades de la taula company. Li donem l'àlies **c** per poder referir-nos-hi més fàcilment.

INNER JOIN transaction AS t → Uneix (JOIN) la taula company amb la taula transaction.

Es fa servir **INNER JOIN**, que només retorna les empreses que tenen almenys una transacció.

t és l'àlies per **transaction**. Si utilitzéssim només **JOIN** el resultat seria el mateix.

ON c.id = t.company_id → Connecta les dues taules basant-se en l'**ID de l'empresa**.

company_id a **transaction** fa referència a la **id** de **company**, establint una relació entre elles.

GROUP BY c.id, c.company_name, c.phone, c.country

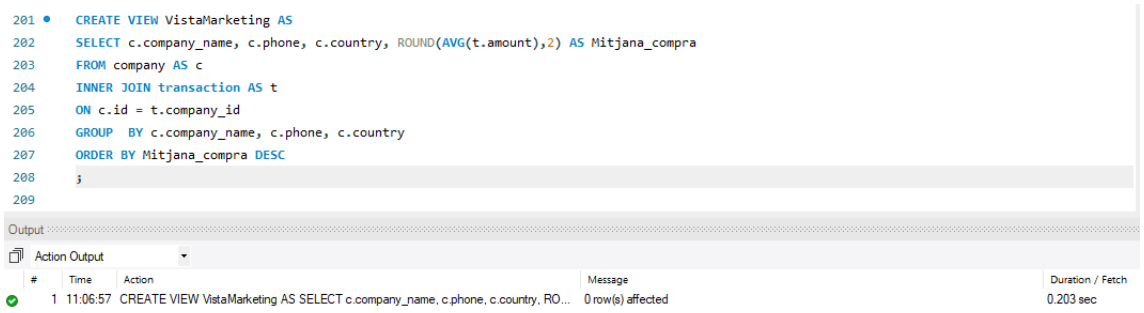
Agrupa els registres per **empresa** (**id**), perquè estem calculant la mitjana de amount.

Totes les columnes que no formen part d'una funció d'agregació **(AVG)** han d'estar en el **GROUP BY**.

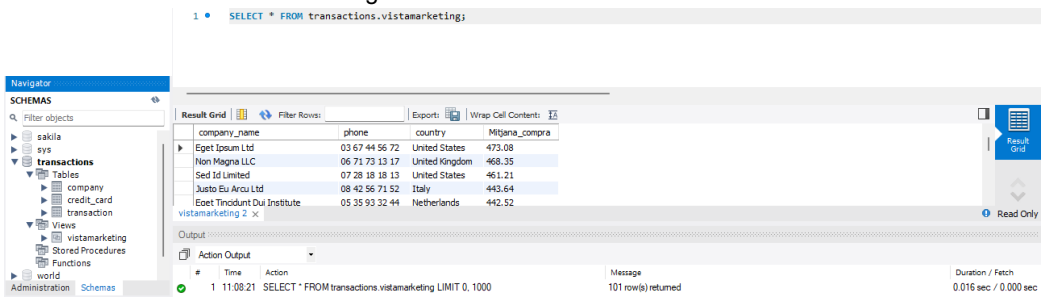
ORDER BY Mitjana_compra DESC

Ordena els resultats **de més a menys (DESC)** segons la mitjana de les transaccions (**Mitjana_compra**).

Així, les empreses que han gastat més apareixen primer.



Visualitzem la Vista VistaMarketing



Exercici 3

Filtra la vista VistaMarketing per a mostrar només les companyies que tenen el seu país de residència en "Germany"

Fem el filtrat amb

SELECT *

FROM VistaMarketing

WHERE country = 'Germany' → Aquí és on fem el filtre limitant els països a 'Germany'

ORDER BY Mitjana_compra DESC

;

```
192     -- Exercici 3
193     -- Filtra la vista VistaMarketing per a mostrar només les companyies que tenen el seu país de residència en "Germany"
194
195 •   SELECT *
196     FROM VistaMarketing
197     WHERE country = 'Germany'
198     ;
```

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

company_name	phone	country	Mitjana_compra
Aliquam PC	01 45 73 52 16	Germany	385.27
Ac Industries	09 34 65 40 60	Germany	289.65
Rutrum Non Inc.	02 66 31 61 09	Germany	266.90
Nunc Interdum Incorporated	05 18 15 48 13	Germany	244.03
Augue Foundation	06 88 43 15 63	Germany	240.80
Ac Fermentum Incorporated	06 85 56 52 33	Germany	206.47
Auctor Mauris Corp	05 62 87 14 41	Germany	184.31

VistaMarketing 2

Output

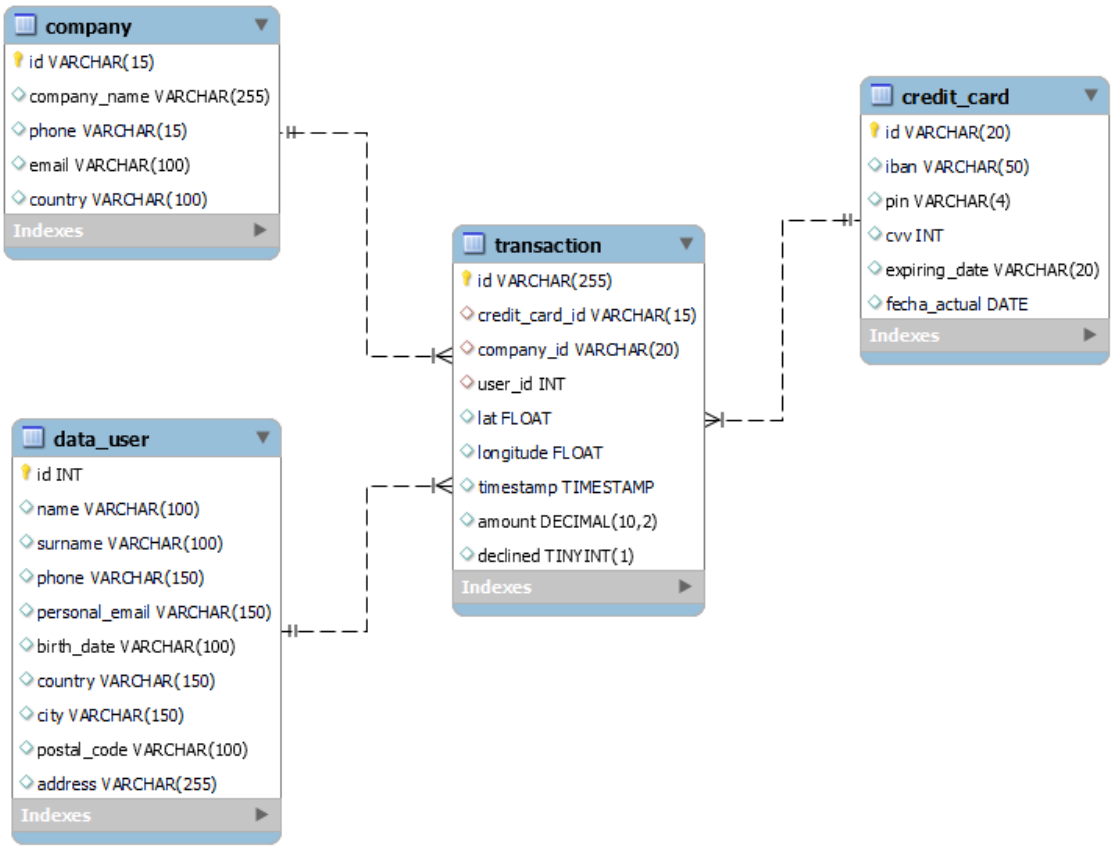
Action Output

#	Time	Action	Message
1	12:16:58	SELECT * FROM VistaMarketing WHERE country = 'Germany' -- ORDER BY Mitjana_...	8 row(s) returned

Nivell 3

Exercici 1

La setmana vinent tindràs una nova reunió amb els gerents de màrqueting. Un company del teu equip va realitzar modificacions en la base de dades, però no recorda com les va realitzar. Et demana que l'ajudis a deixar els comandos executats per a obtenir el següent diagrama:



Recordatori

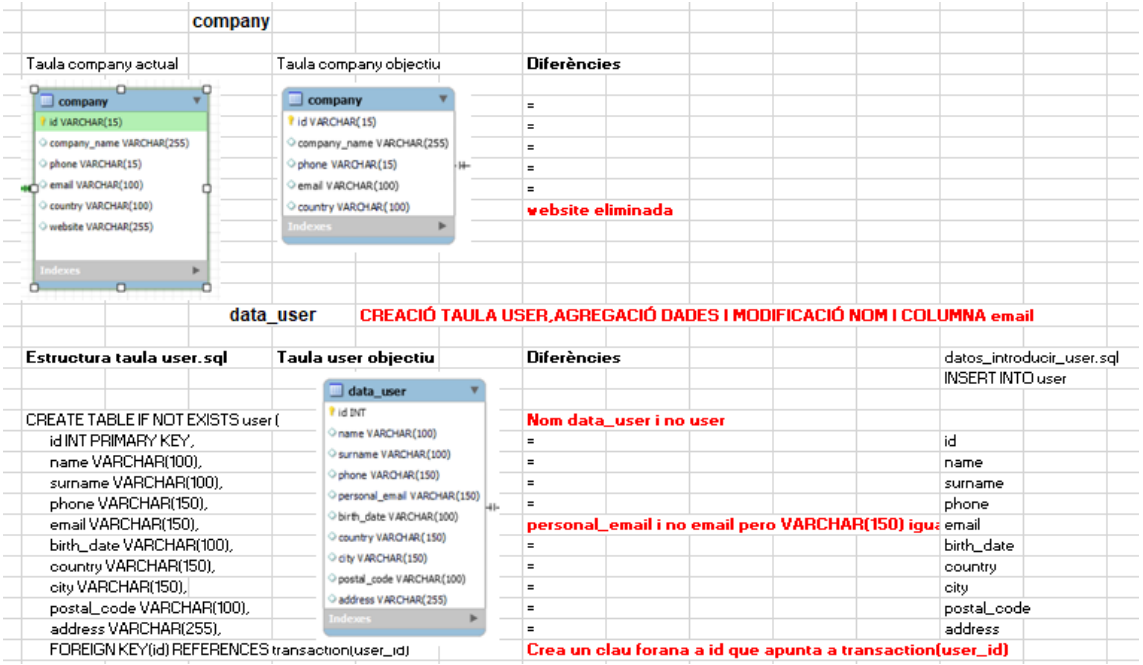
En aquesta activitat, és necessari que descriguis el "pas a pas" de les tasques realitzades. És important realitzar descripcions senzilles, simples i fàcils de comprendre. Per a realitzar aquesta

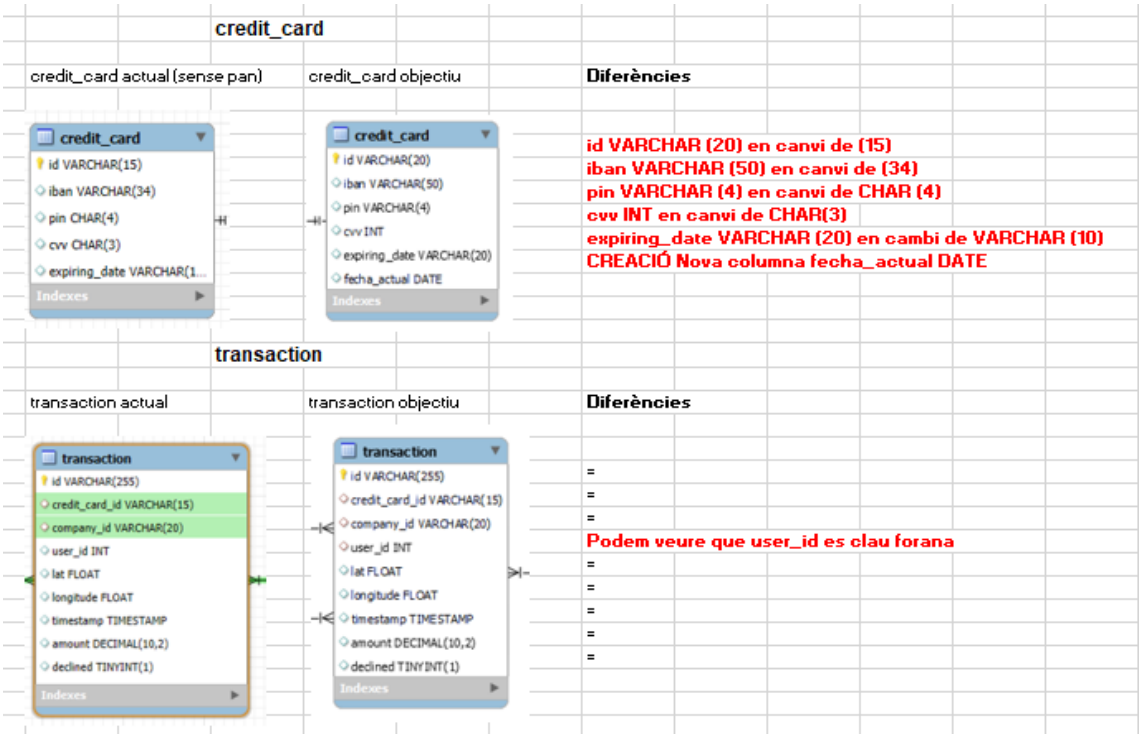
activitat hauràs de treballar amb els arxius denominats "estructura_dades_user" i "dades_introduir_user"

L'objectiu es descriure el procediment i executar-lo per a partir de la situació **actual** arribar a una estructura **objectiu** com la de la imatge.

Passos:

Visualitzem i fem una comparativa de l'estructura i del contingut de cada taula:





1) Crearem la taula **user** executant **estructura_dades_user.sql**

```sql
CREATE INDEX idx_user_id ON transaction(user_id);

CREATE TABLE IF NOT EXISTS user (
        id INT PRIMARY KEY,
        name VARCHAR(100),
        surname VARCHAR(100),
        phone VARCHAR(150),
        email VARCHAR(150),
        birth_date VARCHAR(100),
        country VARCHAR(150),
        city VARCHAR(150),
        postal_code VARCHAR(100),
        address VARCHAR(255),
        FOREIGN KEY(id) REFERENCES transaction(user_id)
    );
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	17:39:08	CREATE INDEX idx_user_id ON transaction(user_id)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	1.125 sec
2	17:39:09	CREATE TABLE IF NOT EXISTS user (id INT PRIMARY KEY, name VARCH...	0 row(s) affected	0.578 sec

Fem un **DESCRIBE user;** per a veure el contingut de la taula

```sql
DESCRIBE user;
```

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

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	
name	varchar(100)	YES		NULL	
surname	varchar(100)	YES		NULL	
phone	varchar(150)	YES		NULL	
email	varchar(150)	YES		NULL	
birth_date	varchar(100)	YES		NULL	
country	varchar(150)	YES		NULL	
city	varchar(150)	YES		NULL	
postal_code	varchar(100)	YES		NULL	
address	varchar(255)	YES		NULL	

Result 1

Output

Action Output

#	Time	Action	Message
1	17:46:20	DESCRIBE user	10 row(s) returned

2) Agregarem les dades executant l'arxiu **dades_introduir_user.sql**

```sql
SET foreign_key_checks = 0;

-- Insertamos datos de user
INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (  "1", "Zeus", "Gamble", "1
INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (  "2", "Garrett", "Mcconnel
INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (  "3", "Ciaran", "Harrison",
INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (  "4", "Howard", "Stafford",
INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (  "5", "Hayfa", "Pierce", "
INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (  "6", "Joel", "Tyson", "(7
INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (  "7", "Rafael", "Jimenez",
INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (  "8", "Nissim", "Franks",
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
269	18:04:12	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal...	1 row(s) affected	0.078 sec
270	18:04:12	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal...	1 row(s) affected	0.141 sec
271	18:04:12	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal...	1 row(s) affected	0.031 sec
272	18:04:12	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal...	1 row(s) affected	0.156 sec
273	18:04:12	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal...	1 row(s) affected	0.188 sec
274	18:04:12	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal...	1 row(s) affected	0.094 sec
275	18:04:13	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal...	1 row(s) affected	0.047 sec
276	18:04:13	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal...	1 row(s) affected	0.125 sec
277	18:04:13	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal...	1 row(s) affected	0.063 sec
278	18:04:13	SET foreign_key_checks = 1	0 row(s) affected	0.000 sec

3)Visualitzem les dades de user amb un SELECT * FROM user;

```
230 •  SELECT * FROM user;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	name	surname	phone	email	birth_date	country	city	postal_code	address
1	Zeus	Gamble	1-282-581-0551	interdum.enim@protonmail.edu	Nov 17, 1985	United States	Lowell	73544	348-7818 Sagittis St.
2	Garrett	Mcconnell	(718) 257-2412	integer.vitae.nibh@protonmail.org	Aug 23, 1992	United States	Des Moines	59464	903 Sit Ave
3	Ciaran	Harrison	(522) 598-1365	interdum.feugiat@aol.org	Apr 29, 1998	United States	Columbus	56518	736-2063 Tellus St.
4	Howard	Stafford	1-411-740-3269	ornare.egestas@icloud.edu	Feb 18, 1989	United States	Kailua	77417	Ap #545-2244 Erat. Rd.
5	Hayfa	Pierce	1-554-541-2077	et.malesuada.fames@hotmail.org	Sep 26, 1998	United States	Sandy	31564	341-2821 Ultrices Av.
6	Joel	Tyson	(718) 288-8020	gravida.nunc.sed@yahoo.ca	Oct 15, 1989	United States	Nashville	96838	888-2799 Amet Street
7	Rafael	Jimenez	(817) 689-0478	eget@outlook.ca	Dec 4, 1981	United States	Hillsboro	29874	8627 Malesuada Rd.
8	Nissim	Franks	(692) 157-3469	egestas.aliquam.fringilla@google.ca	Aug 1, 1993	United States	Jackson	61750	Ap #251-7144 Integer St.

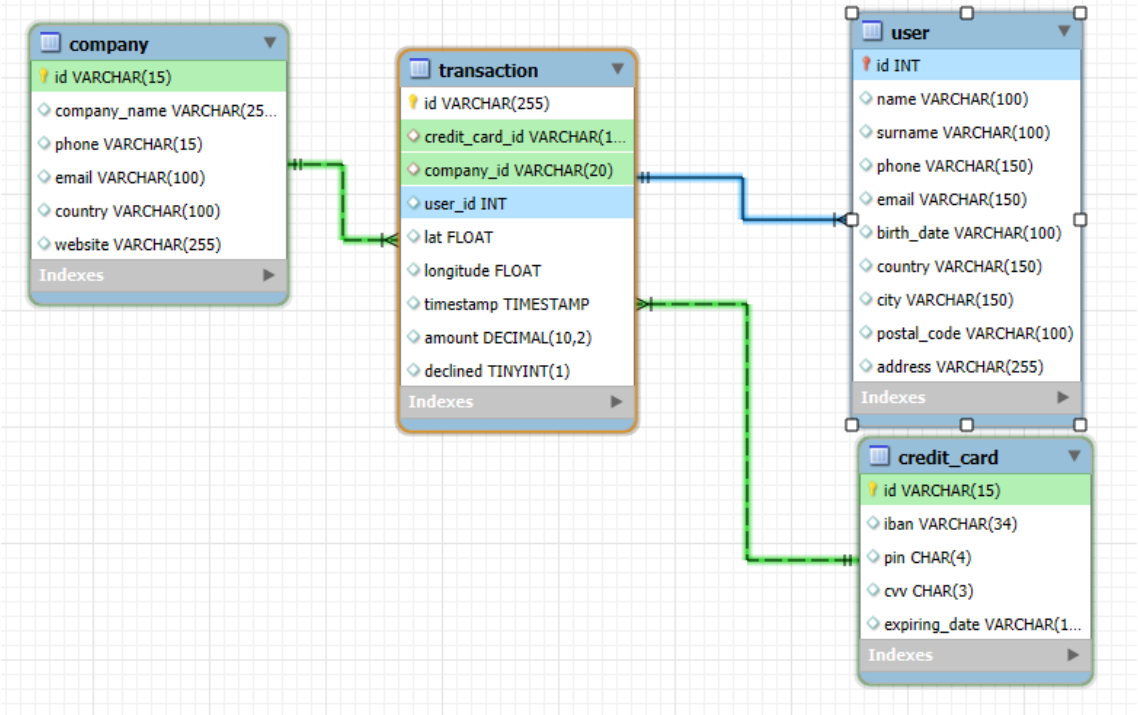
user 2 | Apply | Revert

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	18:11:55	SELECT * FROM user LIMIT 0, 1000	275 row(s) returned	0.015 sec / 0.000 sec

4) Crearem diagrama EER per a veure nova estructura



Diferències estructurals de la base de dades actual:

```
FOREIGN KEY(id) REFERENCES transaction(user_id)
```

La línia contínua entre **user** i transaction es produeix perquè al crear la taula **user** es determina que **id** de **user** és la primary key i al mateix temps es **FOREIGN KEY** de si mateixa que apunta a **transaction.user_id** quan hauria de ser a la inversa, de **transaction.user_id** una **FOREIGN KEY** que apunti a **user.id**

Per tant el primer que farem serà eliminar la Foreign Key de **user** anomenada **user_ibfk_1** ja que es la clau forana que esta mal assignada.

```
237     -- Eliminem la Foreign Key
238 •  ALTER TABLE user DROP FOREIGN KEY user_ibfk_1
239     ;
240
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	10:38:44	ALTER TABLE user DROP FOREIGN KEY user_ibfk_1	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.156 sec

Després mirem de crear la FOREIGN KEY a transaction apuntant a **user.id**

```
245 •  ALTER TABLE transaction ADD  CONSTRAINT user_transaction FOREIGN KEY(user_id) REFERENCES user(id)
246    ;
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	10:51:21	ALTER TABLE transaction ADD CONSTRAINT user_transaction FOREIGN KEY(user_i...	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (trans...	1.343 sec

Ens trobem amb un error degut a que a que no pot crear la Foreign Key a transaction ja que no existeix el user(id) = "9999". Això es produeix perquè anteriorment hem creat un usuari a transaction que contenia un user_id =’9999’

```
-- Ingressem de nou l'usuari amb la comanda INSERT INTO transaction
INSERT INTO transaction (id, credit_card_id, company_id , user_id, lat, longitude,  amount, declined)
VALUES( '108B1D1D-5B23-A76C-55EF-C568E49A99DD' , 'CcU-9999' , 'b-9999' , '9999', '829.999' ,  '-117.999', '111.11','0')
;
```

però aquest id no existeix a la taula user, per tant el crearem

```
241 •  INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address)
242    VALUES ("9999", "name", "surname", "(0034) 614-1654", "user@email.net", "Aug 31, 1981", "Catalunya", "Barcelona", "08036", "Ap #836-9508 Vita
243    ;
244
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	11:42:16	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_c...	1 row(s) affected	0.125 sec

Ara si crearem la Foreign Key a la taula **transaction** anomenada **user_transaction**

```
246    -- Creem la FOREIGN KEY a transaction que apunti a id de user. Li direm transaction_user
247
248 •  ALTER TABLE transaction ADD  CONSTRAINT user_transaction FOREIGN KEY(user_id) REFERENCES user(id)
249    ;
250
251
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	11:45:41	ALTER TABLE transaction ADD CONSTRAINT user_transaction FOREIGN KEY(user_i...	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0	3.329 sec

Eliminarem la columna **website** de la taula **company**

```
243 •  ALTER TABLE company
244    DROP COLUMN website;
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	23:50:51	ALTER TABLE company DROP COLUMN website	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.344 sec

Modificarem el nom de la taula **user** per **data_user**

```
233    -- Cambiem el nom de la taula user
234 •  ALTER TABLE user RENAME TO data_user
235    ;
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	23:38:23	ALTER TABLE user RENAME TO data_user	0 row(s) affected	0.375 sec

Modificarem el nom de la columna **email** per **personal_email** de la taula **data_user**

ALTER TABLE data_user → Modifica la taula **data_user.**

CHANGE email personal_email VARCHAR(150) → Canvia el nom de la columna **email per personal_email**, mantenint el tipus de dades **VARCHAR(150).**

```
237     -- Cambiem el nom de la columna
238 •   ALTER TABLE data_user
239     CHANGE email personal_email VARCHAR(150)
240     ;
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	23:45:04	ALTER TABLE data_user CHANGE email personal_email VARCHAR(150)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.328 sec

Modificarem les columnes de la taula **credit_card** de la següent forma:

ALTER TABLE credit_card

MODIFY COLUMN id VARCHAR(20), →modifica la columna id de VARCHAR(15) a VARCHAR(20).

MODIFY COLUMN iban VARCHAR(50), →modificar la columna iban de VARCHAR(34) a VARCHAR(50)

MODIFY COLUMN pin VARCHAR(4), → modificar la columna pin de CHAR(4) a VARCHAR(4)

MODIFY COLUMN cvv INT, → modifica la columna cvv de CHAR(3) a INT

MODIFY COLUMN expiring_date VARCHAR(20) → modifica la columna cvv de CHAR(3) a CHAR(3)

;

Però abans haurem de modificar el valor de cvv a id 'CcU-9999' que hem creat abans.

```
257     -- Modifiquem el valor de cvv a 000 on id 'CcU-9999'
258 •   UPDATE credit_card
259     SET cvv = 000
260     WHERE id = 'CcU-9999'
261     ;
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	00:28:18	UPDATE credit_card SET cvv = 000 WHERE id = 'CcU-9999'	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.187 sec

Ara si modifiquem totes les columnes que volem modificar de credit_card

```
246     -- Modifiquem les columnes de la taula credit_card de la següent forma
247 •   ALTER TABLE credit_card
248     MODIFY COLUMN id VARCHAR(20),
249     MODIFY COLUMN iban VARCHAR(50),
250     MODIFY COLUMN pin VARCHAR(4),
251     MODIFY COLUMN cvv INT,
252     MODIFY COLUMN expiring_date VARCHAR(20)
253     ;
254
255
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	00:30:04	ALTER TABLE credit_card MODIFY COLUMN id VARCHAR(20), MODIFY COLUMN ib...	276 row(s) affected Records: 276 Duplicates: 0 Warnings: 0	1.719 sec

Creem una nova columna **fecha_actual** com a DATE a **credit card**

```
255 •   ALTER TABLE credit_card
256     ADD COLUMN
257     fecha_actual DATE
258     ;
259
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	00:34:40	ALTER TABLE credit_card ADD COLUMN fecha_actual DATE	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.406 sec

Després de les modificacions tornem a fer el diagrama EER i el visualitzem:

DIAGRAMA EER FINAL

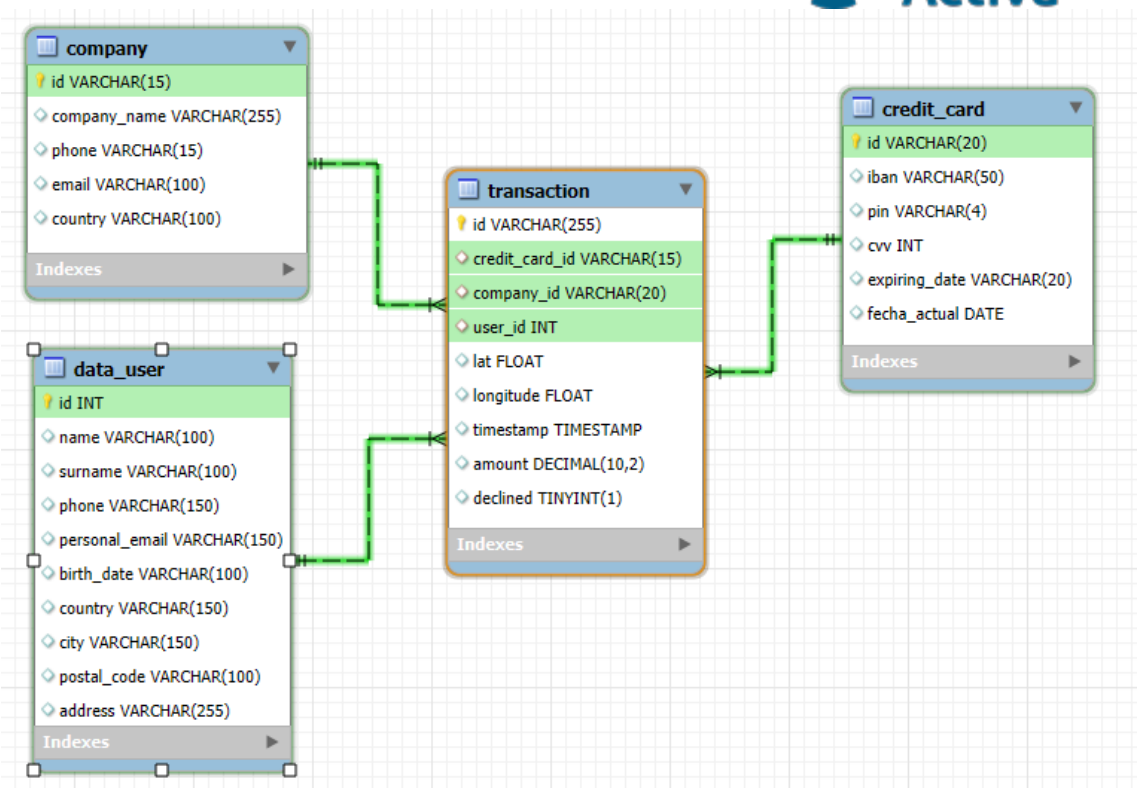
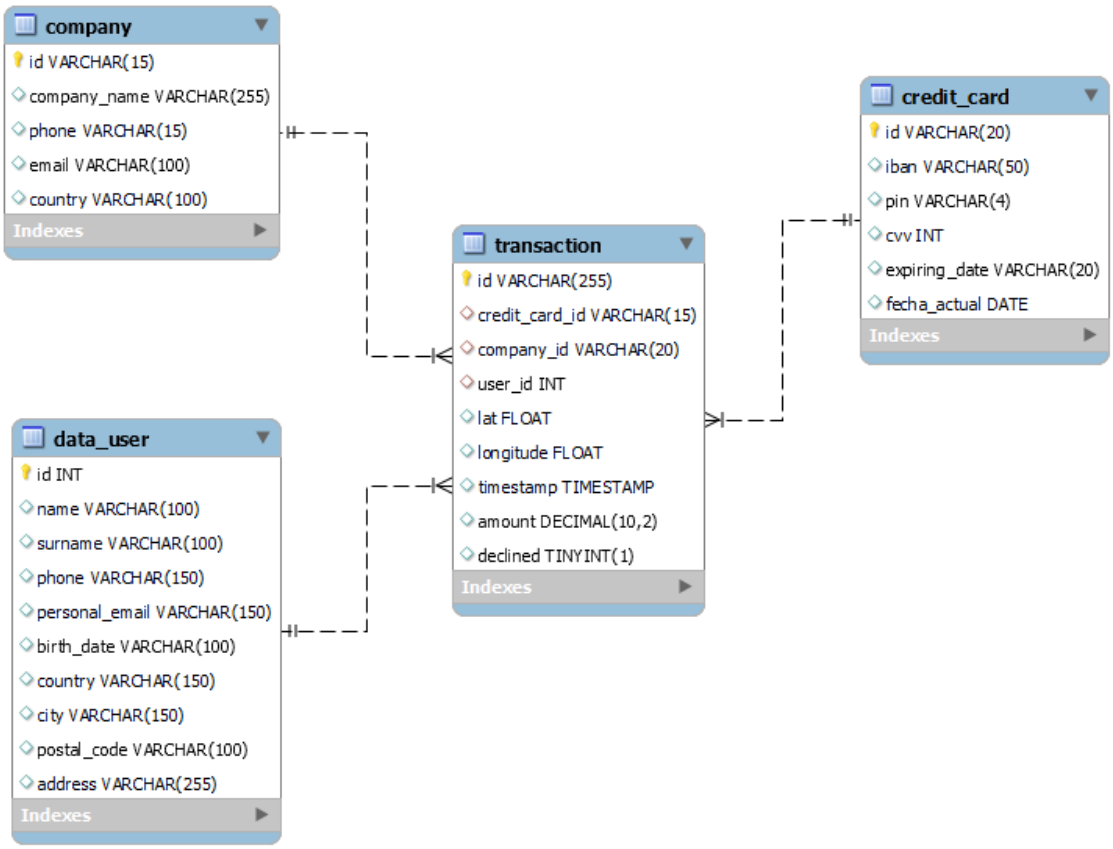


DIAGRAMA EER OBJECTIU



Anàlisi del Diagrama EER

El diagrama EER (Enhanced Entity-Relationship Diagram), mostra un esquema relacional de base de dades amb quatre taules principals: data_user, transaction, company i credit_card. A continuació analitzem cada taula, les relacions entre elles i la integritat de les dades.

1. Taules i Primary Keys

Taula data_user

- **Clau Primària (PK):** id (INT)
- **Atributs:** Conté informació personal dels usuaris, com el nom, cognoms, telèfon, email, data de naixement, ubicació, etc.
- **Funció:** Emmagatzema dades dels usuaris del sistema.

Taula transaction

- **Clau Primària (PK):** id (VARCHAR(255))
- **Atributs:** Conté informació sobre una transacció, incloent la ubicació (lat, longitude), el timestamp, l'import (amount) i si ha estat rebutjada (declined).
- **Foreign Keys (FK):**
    - user_id → referència a data_user(id), per associar una transacció a un usuari.
    - credit_card_id → referència a credit_card(id), per identificar amb quina targeta s'ha fet la transacció.
    - company_id → referència a company(id), per associar la transacció amb una empresa.

Taula company

- **Clau Primària (PK):** id (VARCHAR(15))
- **Atributs:** Conté informació de les empreses, com nom, telèfon, email i país.
- **Funció:** Relaciona transaccions amb empreses.

Taula credit_card

- **Clau Primària (PK):** id (VARCHAR(20))
- **Atributs:** Conté informació sobre les targetes de crèdit, com IBAN, PIN, CVV, data d'expiració i data actual.
- **Funció:** Permet fer transaccions associades a una targeta.

2. Relacions entre les Taules

Les relacions entre les taules estan definides per les claus foranes (FOREIGN KEY):

1. **transaction → data_user**
    - **Relació de molts a un (N:1):** Moltes transaccions poden pertànyer a un únic usuari.
    - **user_id** a **transaction** fa referència a **id** a **data_user.**
2. **transaction → credit_card**
    - **Relació de molts a un (N:1):** Moltes transaccions poden estar associades a una mateixa targeta de crèdit.
    - **credit_card_id** a **transaction** fa referència a **id** a **credit_card.**
3. **transaction → company**

- **Relació de molts a un (N:1**): Moltes transaccions poden estar associades a una empresa.
- **company_id** a **transaction** fa referència a **id** a **company.**

Aquest model permet associar transaccions amb usuaris, targetes de crèdit i empreses, facilitant la traçabilitat de cada pagament.

3.Restriccions i Integritat de Dades

- **Claus primàries:** Cada taula té una PRIMARY KEY única que garanteix que cada registre sigui únic.
- **Claus foranes:**
  - **user_id** a **transaction** assegura que només es poden crear transaccions per usuaris existents.
  - **credit_card_id** a **transaction** evita transaccions amb targetes inexistent.
  - **company_id** a **transaction** assegura que només es registren transaccions d'empreses existents.
- **Tipus de dades:** Són variats **(VARCHAR, INT, DECIMAL, TIMESTAMP, FLOAT)**, la qual cosa permet emmagatzemar diferents tipus d'informació.
- **Integritat referencial:** Les foreign keys (user_id, company_id, credit_card_id) asseguren que només es poden registrar transaccions si existeix un usuari, una empresa i una targeta de crèdit vàlida.

El model de base de dades correspon a un **Star Schema**:

1. Taula de fets (**Fact Table**): on s'emmagatzemen dades mesurables.
2. Taules de dimensions (**Dimension Tables**): que contenen dades descriptives.

1. Identificació de la taula de fets

La taula **transaction** sembla ser la taula de fets en aquest esquema perquè:

- Conté dades quantitatives i mesurables com amount, declined, timestamp, etc.
- Està connectada a diverses taules que emmagatzemen informació descriptiva (dimensions).

2. Identificació de les dimensions

- **data_user** → Representa la dimensió dels clients/usuaris.
- **credit_card** → Representa la dimensió dels mètodes de pagament.
- **company** → Representa la dimensió de les empreses o venedors.

3. Característiques del model que encaixen amb un Star Schema

- Estructura centralitzada: **transaction** està al centre i està envoltada de taules que la descriuen.
- **Relacions 1:N**: Cada transacció està associada a un únic usuari, targeta i empresa, però cada usuari, targeta o empresa pot tenir múltiples transaccions.
- **Dades desnormalitzades en dimensions**: Tot i que sembla bastant normalitzat, les dimensions contenen informació descriptiva que s'usa per fer anàlisi.

Exercici 2

L'empresa també et sol·licita crear una vista anomenada "InformeTecnico" que contingui la següent informació:

- **ID de la transacció**
- **Nom de l'usuari/ària**
- **Cognom de l'usuari/ària**
- **IBAN de la targeta de crèdit usada.**
- **Nom de la companyia de la transacció realitzada.**
- **Assegura't d'incloure informació rellevant de totes dues taules i utilitza àlies per a canviar de nom columnes segons sigui necessari.**

Mostra els resultats de la vista, ordena els resultats de manera descendent en funció de la variable ID de transaction.

Primer crearem una taula per entendre quina informació es sol·licita i l'origen de les dades

Dada demanada	Taula dades	Nom columna
ID de la transacció	transaction	id
Nom de l'usuari/ària	data_user	name
Cognom de l'usuari/ària	data_user	surname
IBAN de la targeta de crèdit usada.	credit_card	iban
Nom de la companyia de la transacció realitzada.	company	company_name

Creem la Vista InformeTecnico amb la següent comanda , com demana incloure informació rellevant inclourem la columna **amount** amb l'àlies **Import_Transferència** de la taula **transaction,** però no l'inclourem al **ORDER BY** perquè està ordenat per id de transferència i per tant cada registre tindrà un import diferent. (segons demana l'enunciat tot i que a opinió meva s'hauria de ordenar per l'àlies **Import_Transfrència** que es més rellavant) .Ho fem seguint les condicions esmentades:

```
CREATE VIEW Informe_Tecnico AS

SELECT t.id AS Identificador_transacció, d_u.name AS Nom_usuari, d_u.surname AS Cognom_usuari, c_c.iban AS Iban,

 c.company_name AS Nom_de_la_empresa, t.amount AS Import_Transferència

FROM transaction AS t

JOIN data_user AS d_u ON d_u.id = t.id

JOIN credit_card AS c_c ON c_c.id = t.credit_card_id

JOIN company as c ON c.id = t.company_id

ORDER BY Identificador_transacció DESC
```

```
;
-- Creem una vista que inclou els amount per transferencia ja que es demana que inclogui informació rellevant

CREATE VIEW Informe_Tecnico AS
SELECT t.id AS Identificador_transacció, d_u.name AS Nom_usuari, d_u.surname AS Cognom_usuari, c_c.iban AS Iban,
 c.company_name AS Nom_de_la_empresa, t.amount AS Import_Transferència
FROM transaction AS t
JOIN data_user AS d_u ON d_u.id = t.id
JOIN credit_card AS c_c ON c_c.id = t.credit_card_id
JOIN company as c ON c.id = t.company_id
ORDER BY Identificador_transacció DESC
;
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	13:30:09	CREATE VIEW Informe_Tecnico AS SELECT t.id AS Identificador_transacció, d_u.nam...	0 row(s) affected	0.203 sec

Visualitzem la VISTA Infrome_Tecnico

```
SELECT * FROM transactions.informe_tecnico;
```

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

Identificador_transacció	Nom_usuari	Cognom_usuari	Iban	Nom_de_la_empresa	Import_Transferència
9FBB3D61-D3C2-E5BB-4BC3-6CC83C718D34	Mannix	Mcclain	SI90187218272592688	Aliquam PC	490.19
9F698741-2C27-2C18-9CD3-E3C7F9FBE1CB	Mannix	Mcclain	CR7242477244335841535	Lorem Eu Incorporated	84.20
9EE639A3-2836-87C2-207F-AF3941834DB2	Mannix	Mcclain	VG1468087984174645729577	Ut Semper Foundation	368.83
9DED76AE-FE36-ACC3-7377-AE98A6D95247	Mannix	Mcclain	FI7847379817377733	Malesuada PC	408.58
9D4AB77C-9F51-FD95-EAB4-A37163E6D3E2	Mannix	Mcclain	SE8424235059254817742452	Nunc Interdum Incorporated	111.51
9D38BED9-558B-EF5A-E4B3-C9668587A15B	Mannix	Mcclain	BH6271442836806676529 4	Enim Condimentum Ltd	178.59
9CEA5069-EEDB-11BB-F8DE-9E8585CA56B8	Mannix	Mcclain	SE2813123487163628531121	Nunc Interdum Incorporated	308.12
9CB62BC2-FFA6-449F-59E4-379EDA5F29C6	Mannix	Mcclain	HR5688747222861164805	Erat LLP	448.44
9C4B45E7-C159-18E5-DB43-9D6F897775D4	Mannix	Mcclain	HU1416500002852287044442312 8	Nunc Interdum Incorporated	183.11

informe_tecnico 2

Output

Action Output

#	Time	Action	Message	Duration /
1	13:34:19	SELECT * FROM transactions.informe_tecnico LIMIT 0, 1000	201 row(s) returned	0.016 sec

Reptes, dificultats afrontades i aspectes a millorar

-Jerarquies de les comandes:

Comanda SQL	Acció
SELECT	Escull les columnes que es mostraran en el resultat.
FROM	Indica la taula d'on es prendran les dades.
JOIN	Uneix dades de dues o més taules segons una relació comuna.
WHERE	Filtra les files segons una condició específica.
GROUP BY	Agrupa les files segons valors comuns d'una o més columnes.
HAVING	Filtra els grups creats per GROUP BY.
ORDER BY	Ordena els resultats segons una o més columnes.
LIMIT	Restringeix el nombre de files retornades.

- Comandes de manipulació de taules:

Comanda SQL	Acció
INSERT	Insereix noves files en una taula.
UPDATE	Modifica valors d'una taula existent.
DELETE	Elimina files d'una taula.
CREATE TABLE	Crea una nova taula en la base de dades.
ALTER TABLE	Modifica l'estructura d'una taula (afegir, eliminar columnes, etc.).
DROP TABLE	Elimina completament una taula de la base de dades.

- Fer una selecció en un a VISTA

- Per a poder crear una restricció és necessari que els camps existeixin prèviament a les taules.

-Comprensió de la informació gràfica dels diagrames EER, relacions entre taules, claus primàries grogues i vermelles, línies continues i discontinues i el seu significat

-Diferents models de Schema de bases de dades: OLTP, Star Schema, Snowflake Schema