SPRINT 3

MANIPULACIÓ DE TAULES

PREPARED BY:

CECIL LUNA

CHECKED AND REVISED BY:

Micaela Giaroli

This sprint simulated a business situation where I had to perform various manipulations on database tables while also working with indexes and views.

In this activity, I continued working on the previous database, which contains information about a company dedicated to online product sales. Additionally, I started working with information related to credit cards.

LEVEL 1

Exercise 1:

GOAL 1: Design and create a table called "credit_card" that stores crucial details about credit cards. The new table must be able to uniquely identify each card…

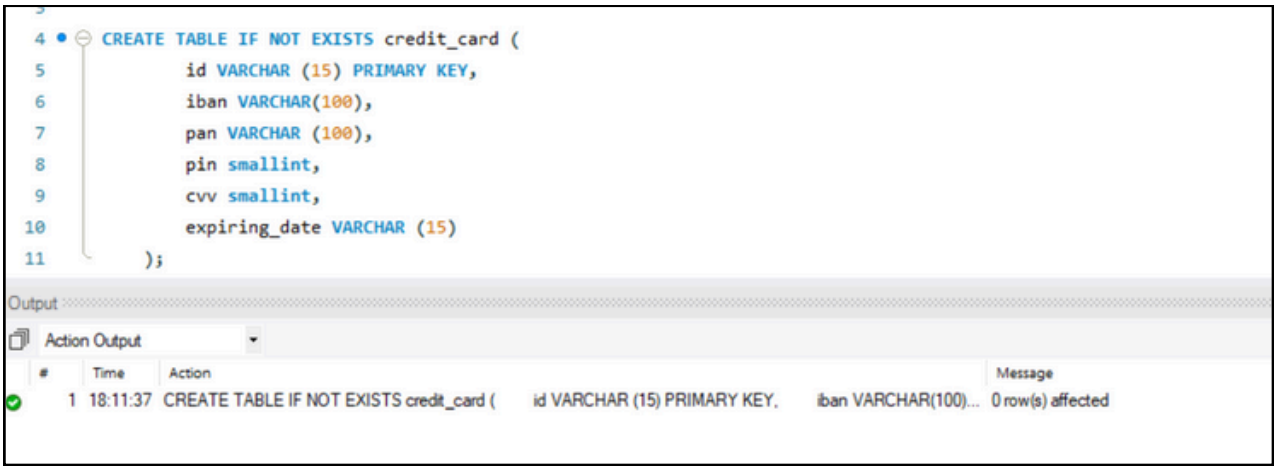


Figure 1.1.1.1 Table Creation

EXPLANATION:

CREATE TABLE: In the data record provided, there are six columns in the credit_card table. Upon checking each data, here are the names and data types of each column:

- **id VARCHAR(15) PRIMARY KEY:** This ID should be unique as it serves as the primary key of this table. It is defined as VARCHAR because it contains letters, special characters, and numbers. The length is limited to 15 to avoid errors in data entry.
- **iban VARCHAR(100):** This column uses VARCHAR because the data contains letters and numbers. The length is limited to 100 to prevent errors in data entry.

- **pan VARCHAR(100):** There was an error in inserting data, so the data type was changed to VARCHAR to allow more flexible data entry, but it was limited to 100 characters to ensure consistency.

- **pin SMALLINT:** After checking and understanding the data, the values in the pin column usually have 10 to 16 bits. Typically, credit card PINs are not longer than 16 bits. This type is used to limit data entry and prevent incorrect entries.

- **cvv SMALLINT:** Generally, CVVs are 3-digit numbers. Using TINYINT would not be effective as it only stores numbers ranging from 0 to 255. If the CVV exceeds 255, the database won't accept the entry, which is why SMALLINT is the best option, allowing for numbers from 0 to 16 bits.

- **expiring_date VARCHAR(15):** The provided data has different date formats, so VARCHAR is used for flexibility, and it is limited to 15 characters.

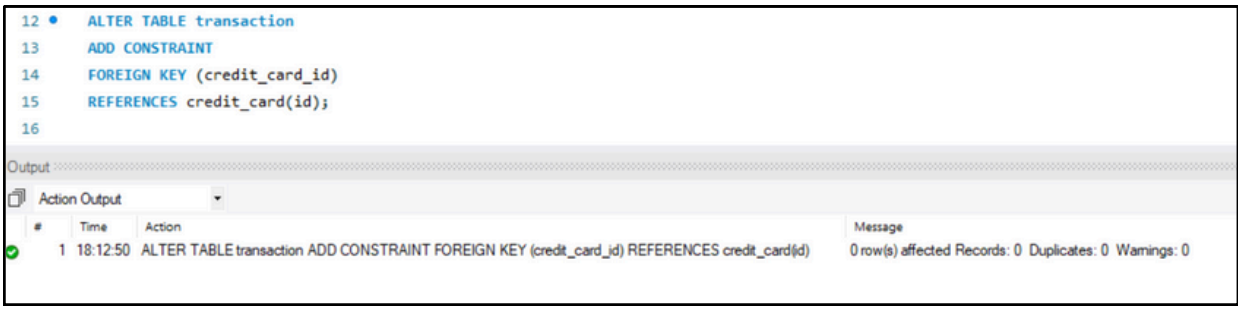


Figure 1.1.1.2 Adding Foreign Key

IMPORTANT NOTE:

This is a revised task. After experiencing problems in relationships, updating and deleting tables, I decided to assign the foreign key after creating the credit_card table.

LEVEL 1

Exercise 1:

GOAL 2: and establish a proper relationship with the other two tables ("transaction" and "company"). After creating the table you will need to enter the information from the document named "data_enter_credit". Remember to show the diagram and make a brief description of it.

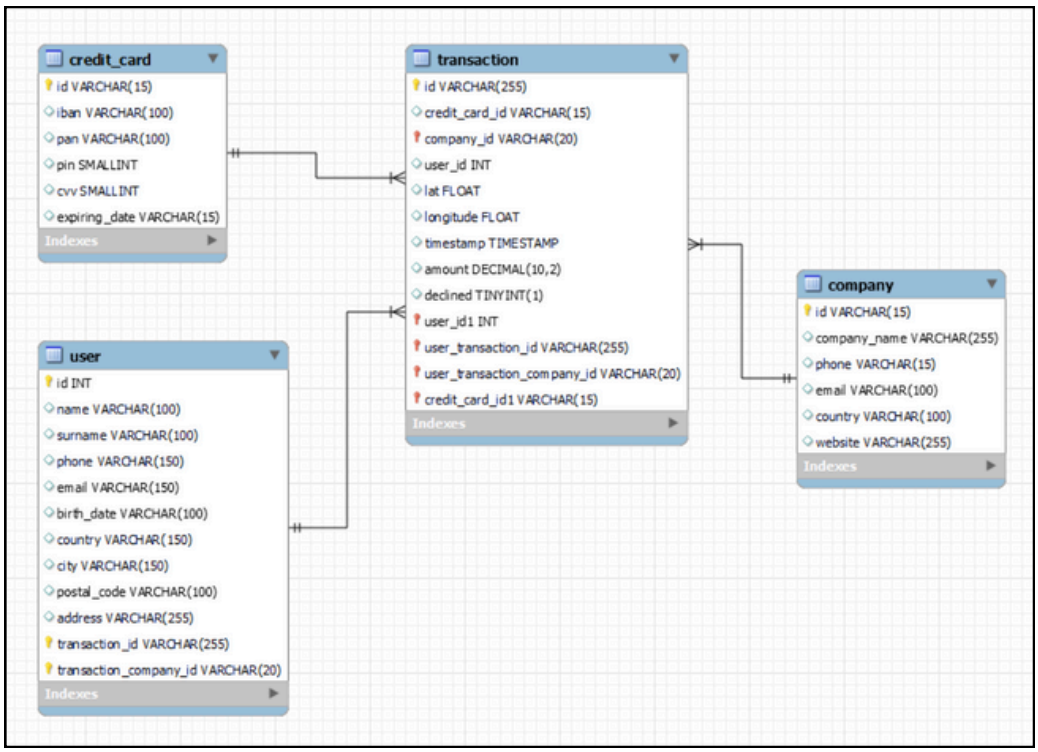


Figure 1.1.2 Relationship

EXPLANATION:

OVERALL RELATIONSHIPS: The database structure resembles a star schema, where the transaction table serves as the fact table because it contains more records and multiple foreign keys. Each of the company, user, and credit_card tables has a natural key.

ONE-TO-MANY RELATIONSHIPS:

- **Company to Transaction:** A company can have one or more transactions.
- **User to Transaction:** A user can have one or more transactions.
- **Credit Card to Transaction:** A credit card can be used in one or more transactions.

LEVEL 1

Exercise 2:

GOAL 1: The Human Resources department has identified an error in the account number for user ID CcU-2938. The information to be displayed for this record is: R323456312213576817699999. Remember to show that the change was made.

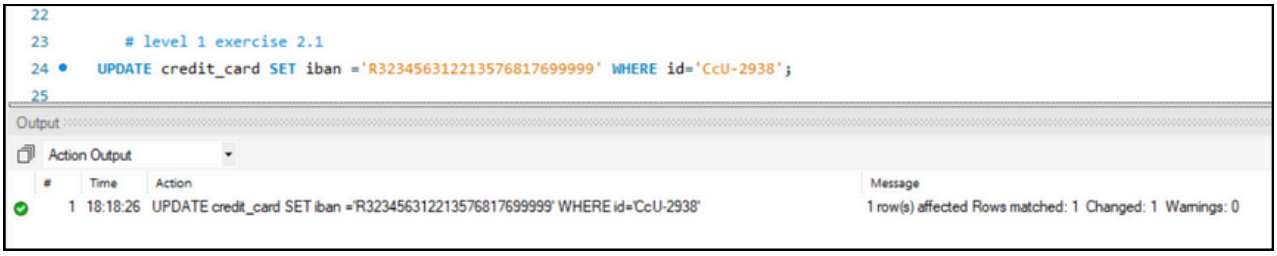


Figure 1.2.1 Query + Result

EXPLANATION:

To change the data entry, updating the table is necessary. First, understand the data and identify the table and the columns where the data should be updated. In this case, the table is credit_card, and the columns are iban and id due to the nature of the data. Then, set the data to be updated and use the WHERE clause to specify the precise location in the column.

LEVEL 1

Exercise 3 :

GOAL 1: In the "transaction" table enter a new user with the following information:

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

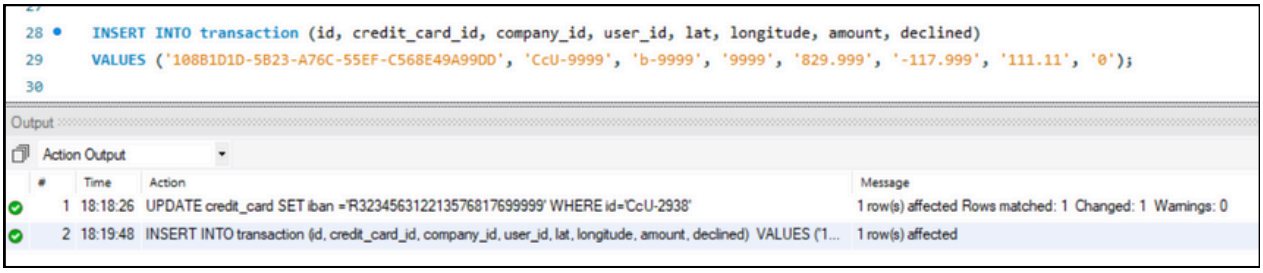


Figure 1.3.1 Query + Result

EXPLANATION:

To do this, use the INSERT INTO command followed by the table where you want to add the data. It is necessary to specify the columns where you want to insert the data. Afterward, add the values, ensuring they follow the order of the columns to correctly allocate the data to its corresponding fields.

LEVEL 1

Exercise 4 :

GOAL 1: From human resources they ask you to delete the column "pan" from the credit_card table. Remember to show the change made.

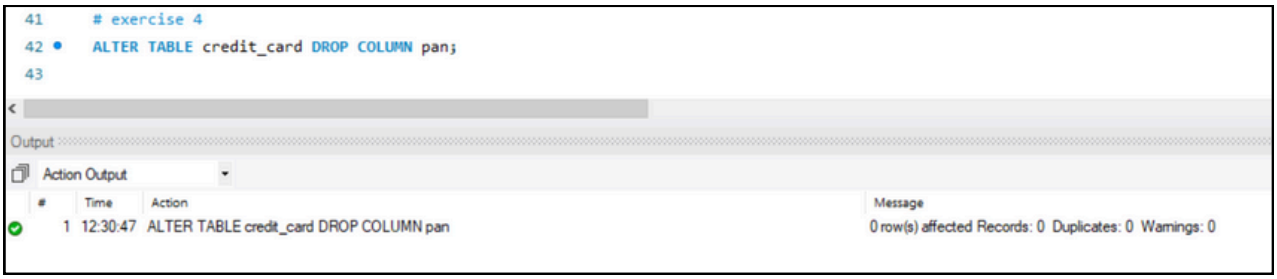


Figure 1.4.1 Relationship

EXPLANATION:

To delete a column, the DROP command is needed. First, choose the table that needs to be altered. Then, use the DROP command to delete the chosen column. In this task, the column that needs to be deleted is pan in the credit_card table.

LEVEL 2

Exercise 1 :

GOAL 1: Delete the record with ID 02C6201E-D90A-1859-B4EE-88D2986D3B02 from the transaction table in the database.

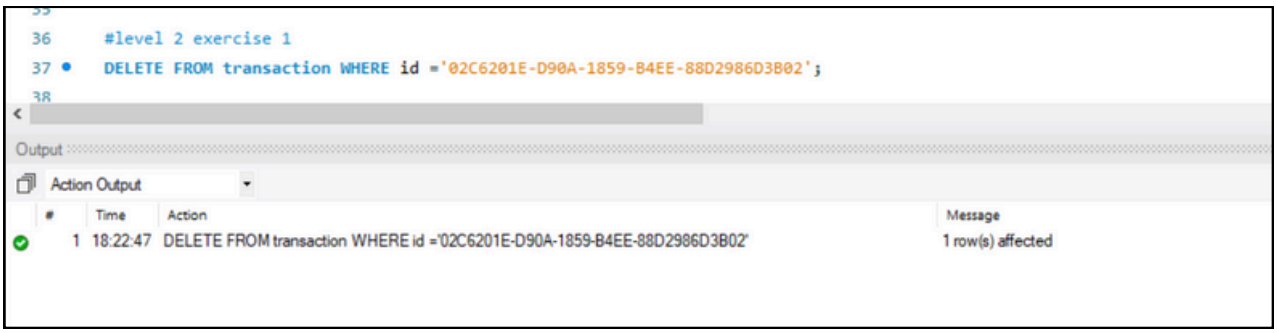


Figure 2.1.1.1 Alter Table

EXPLANATION:
Using the DELETE FROM command requires identifying table where data resides. In this case, it's transaction table. And set the condition in which column would you like to delete this data. In this case, the data to be delete if from id column from transaction table

LEVEL 2

Exercise 2 :

GOAL 1: The marketing section would like to have access to specific information for effective analysis and strategy. You have been asked to create a view that provides key details about the companies and their transactions. You will need to create a view called MarketingView that contains the following information: Company name. Contact phone number. Country of residence. Average purchase made by each company. Present the created view, sorting the data from highest to lowest average purchase.

Figure 2.2.1.1 Relationship

EXPLANATION:

To do this view, first, create a query that fetches the following information:

- Company name
- Contact phone number
- Country of residence
- Average purchase made by each company from highest to lowest

Using the **INNER JOIN**, connect the tables company and transaction where this information is located. These tables are linked by the id column in the company and the company_id column in the transaction. To show the average purchase made from highest to lowest, order the average using a **DESC command.**

Once the results meet the required information, create a view by writing the query above the SELECT query and naming it MarketingView.

LEVEL 2

```
47 •  SELECT *
48    FROM transactions.marketingview;
```

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

company_name	phone	country	AveragePurchase
Eget Ipsum Ltd	03 67 44 56 72	United States	473.075000
Non Magna LLC	06 71 73 13 17	United Kingdom	468.345000
Sed Id Limited	07 28 18 18 13	United States	461.210000
Justo Eu Arcu Ltd	08 42 56 71 52	Italy	443.635000
Eget Tincidunt Dui Institute	05 35 93 32 44	Netherlands	442.520000

marketingview 13 ×

Output

Action Output

#	Time	Action	Message
1	18:23:43	CREATE VIEW MarketingView AS SELECT company.company_name, company.phone, company.country, AVG...	0 row(s) affected
2	18:24:11	SELECT * FROM transactions.marketingview	100 row(s) returned

Figure 2.2.1.2 Relationship

EXPLANATION:

To confirm that the view has been created, you can run a query to retrieve all the rows from the marketingview view table.

LEVEL 2

Exercise 3 :

GOAL 1: Filter the MarketingView to show only companies that have their country of residence in "Germany".

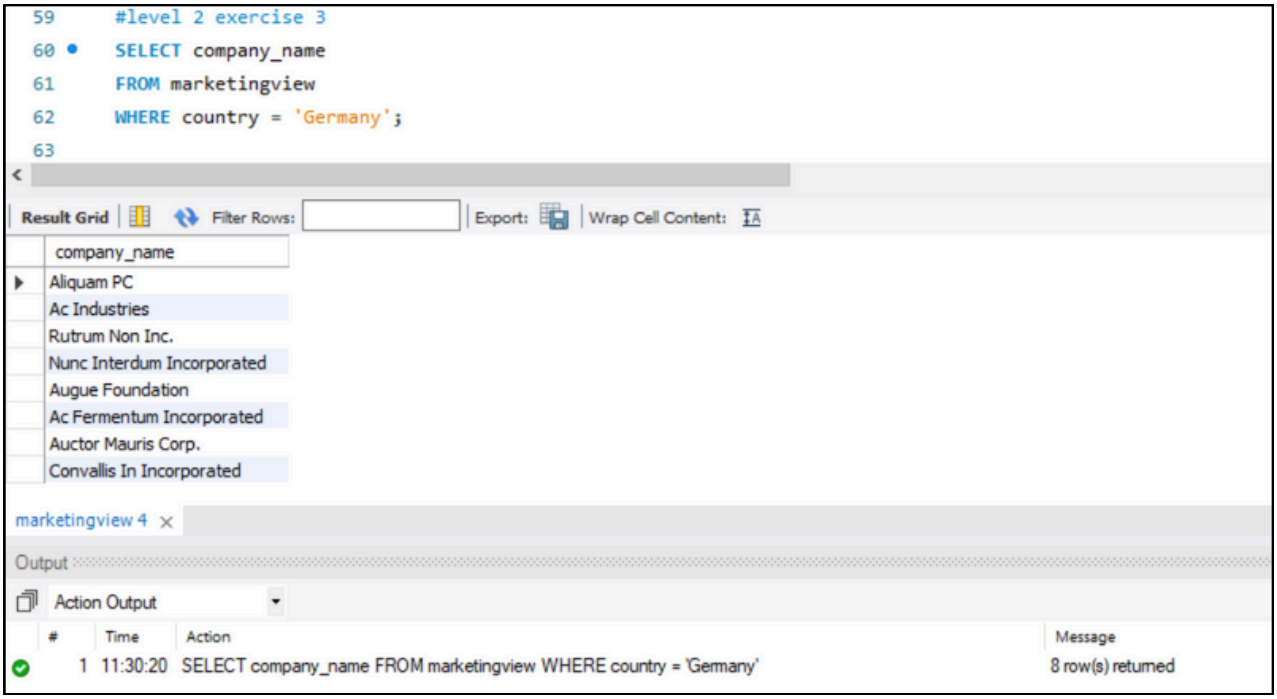


Figure 2.1.1 Relationship

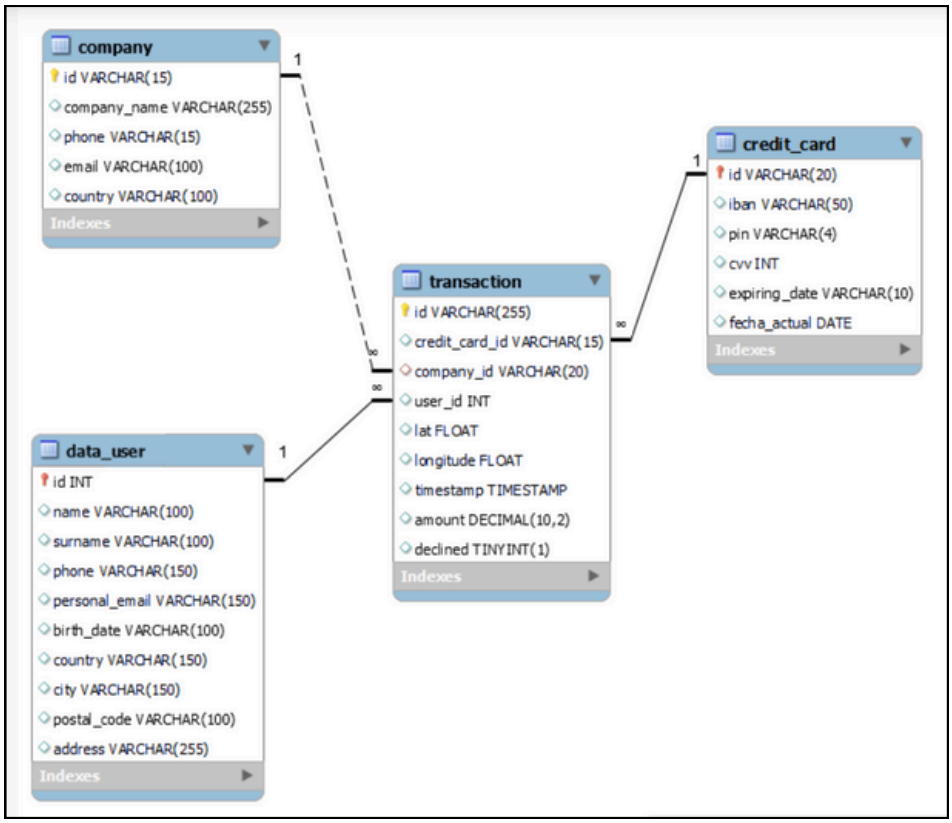
EXPLANATION:

To retrieve all the companies located in Germany from the MarketingView table, you can run a SQL query that selects the name of the company and applies a condition in the WHERE clause to filter for companies based in Germany.

LEVEL 3

Exercise 1 :

GOAL 1: Next week you will have a new meeting with marketing managers. A colleague in your team made modifications to the database, but does not remember how he made them. He asks you to help him leave the executed commands to obtain the following diagram:

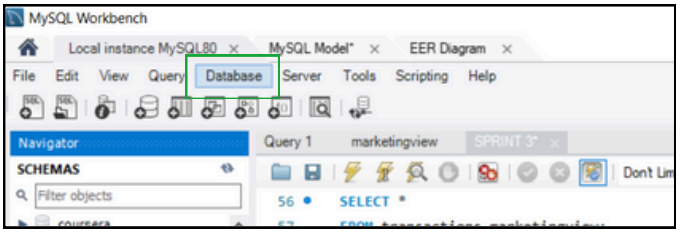


Level 3 Exercise 1

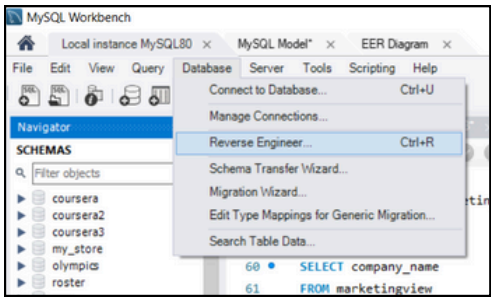
LEVEL 3

Steps to Reverse Engineer a Database in MySQL Workbench

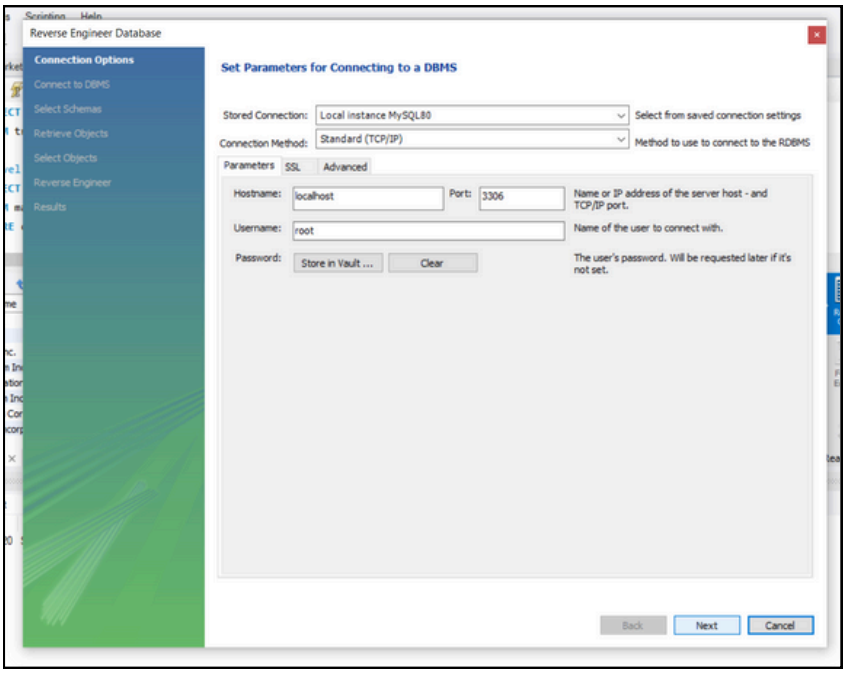
STEP 1: Hover your mouse over the upper right button of the MySQL Workbench, and click **DATABASE** to show more options.



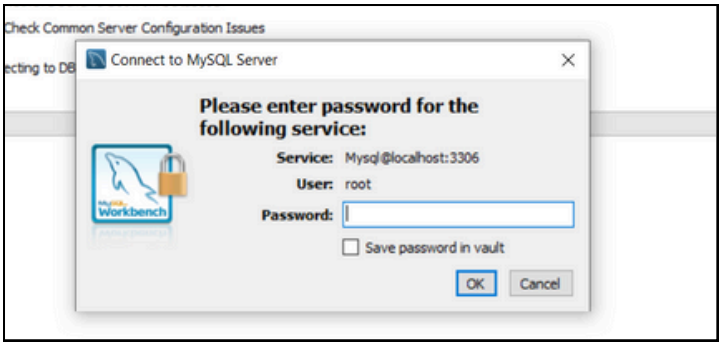
STEP 2: When the options appear, click Reverse Engineer.



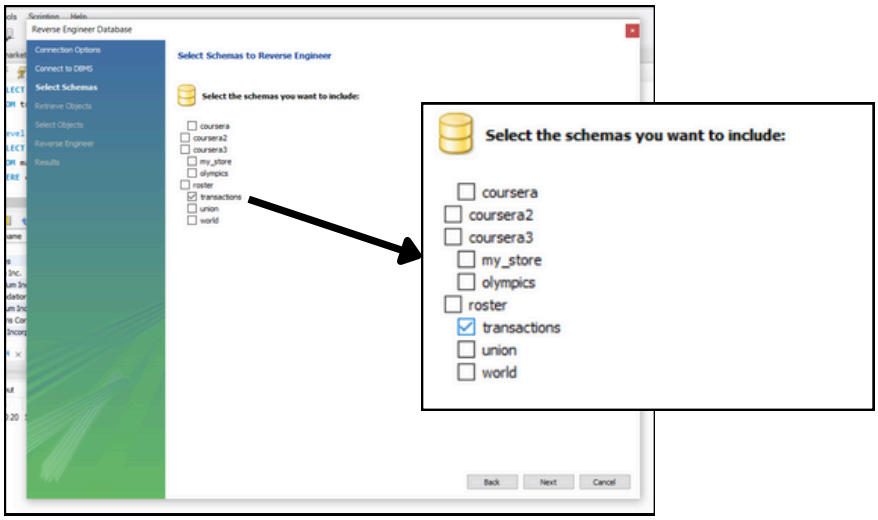
STEP 3: The **Set Parameter for Connecting to a DBMS** dialog box will appear. Click **NEXT.**



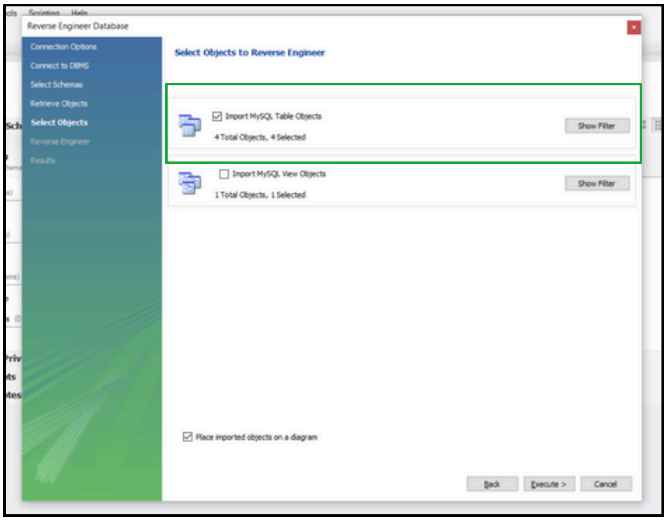
STEP 4: MySQL Workbench will request the password. Enter **1234** and click **OK**. Once the password is accepted, click **NEXT.**



STEP 5: A **Select Schema to Reverse Engineer** dialog box will appear. Tick **transactions,** then click **NEXT.**

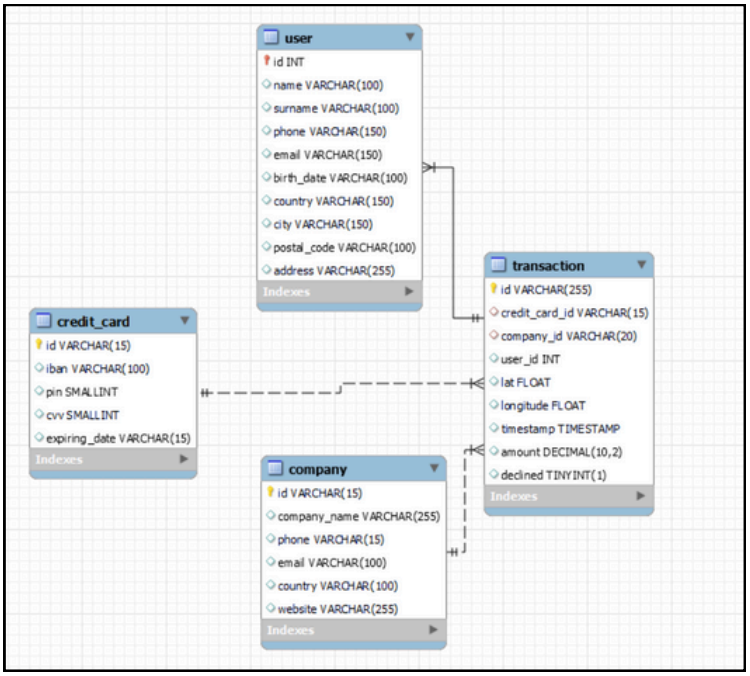


STEP 6: A **Select Objects to Reverse Engineer** dialog box will appear. **Tick** only the four main tables and **untick** the MySQL view table. Then click **EXECUTE.**



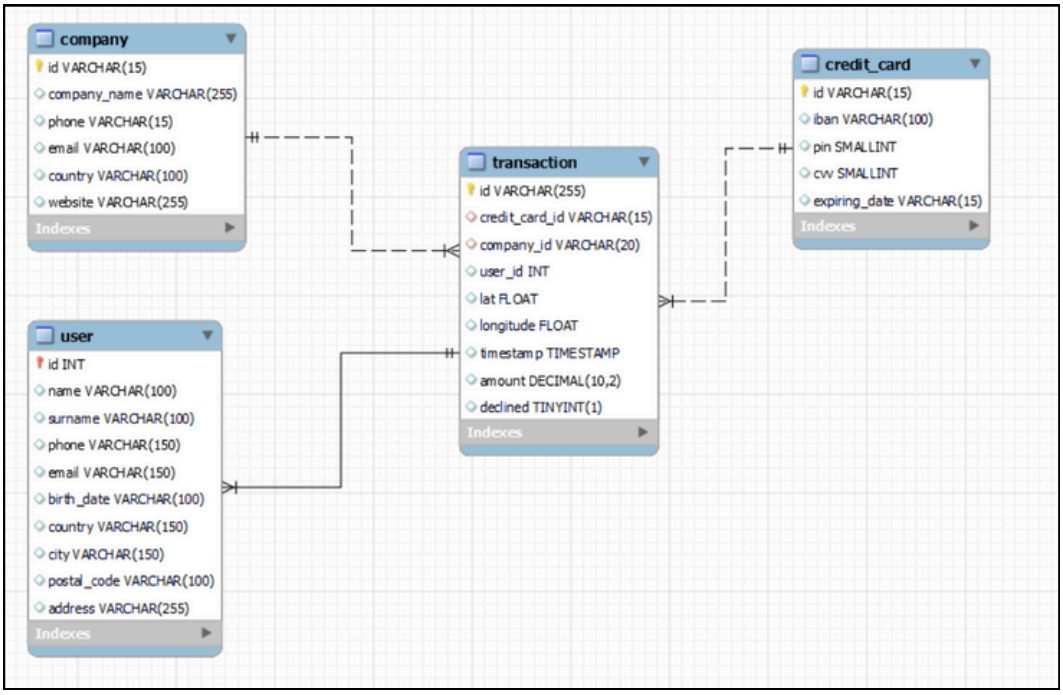
STEP 7: Once the process is completed, click NEXT, then FINISH.

Here's the view of the EER Diagram window.

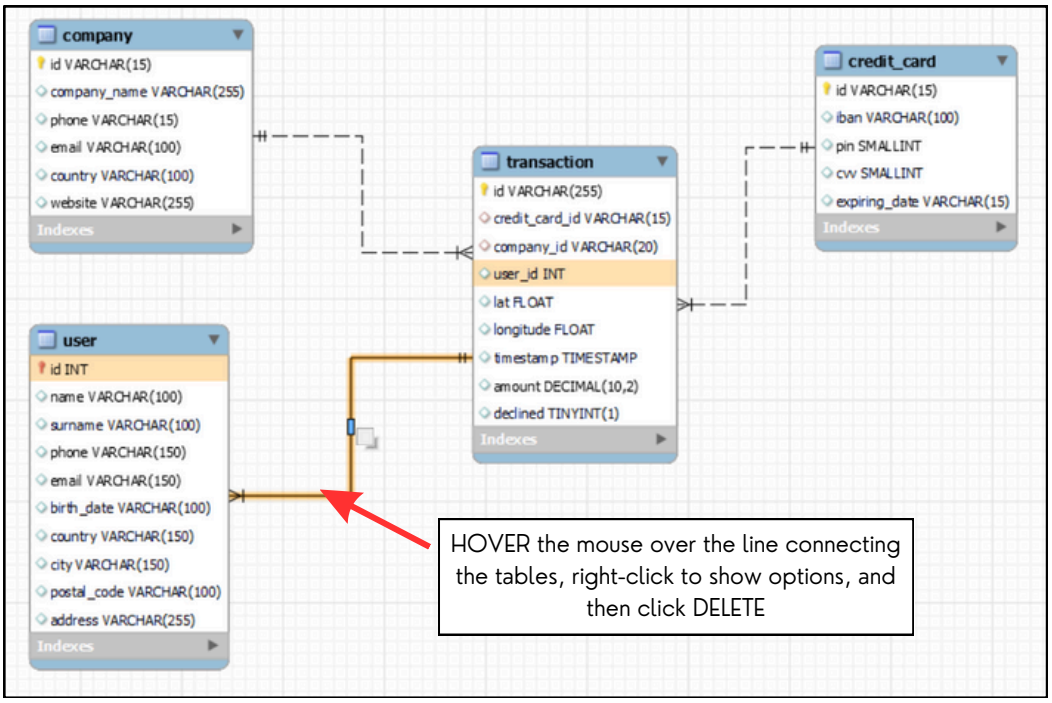


Here's how to obtain the same diagram as in Figure 3.1.1 Level 3 Exercise 1:

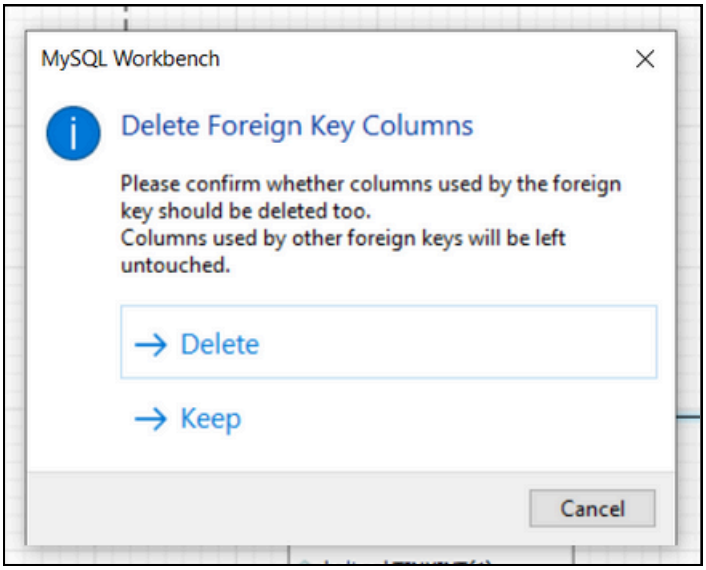
STEP 1: Drag the tables into the **EER Diagram** window. Position the company table at the top left. Drag the user table below the company table. Place the transaction table in the middle of the diagram. Finally, drag the credit_card table to the right of the transaction table.



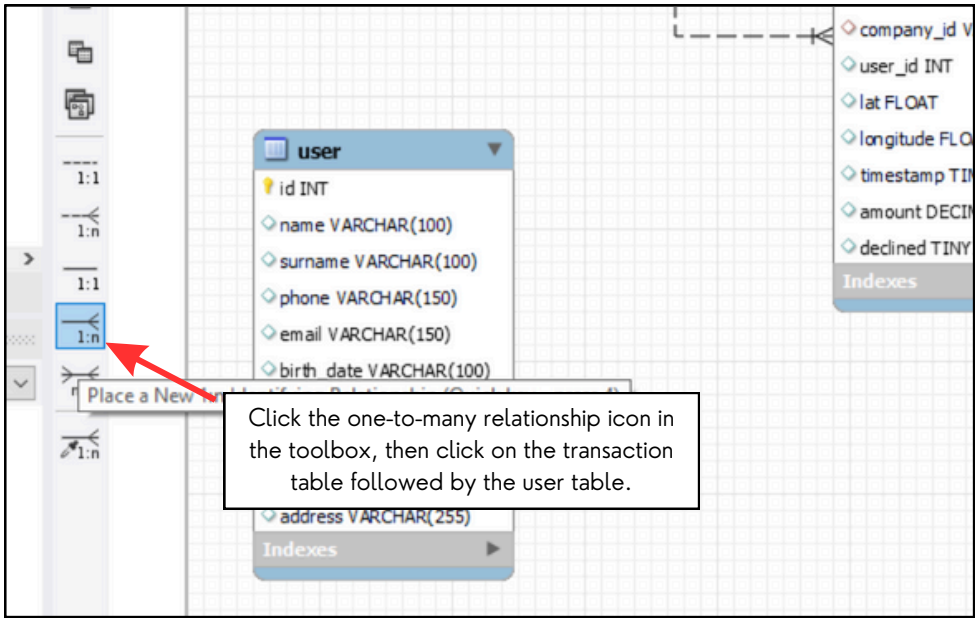
STEP 2: The relationships between the transaction and user tables are not established properly. To correct this, the existing relationship needs to be deleted first. To delete the relationship, **HOVER** the mouse over the line connecting the tables, right-click to show options, and then click **DELETE**



Step 3: A dialogue box will appear asking if you would like to delete or keep the foreign key columns. Here, click Keep because you want to maintain the column.



STEP 4: To establish the relationship between the transaction and user tables, click the one-to-many relationship icon in the toolbox, then click on the transaction table followed by the user table.



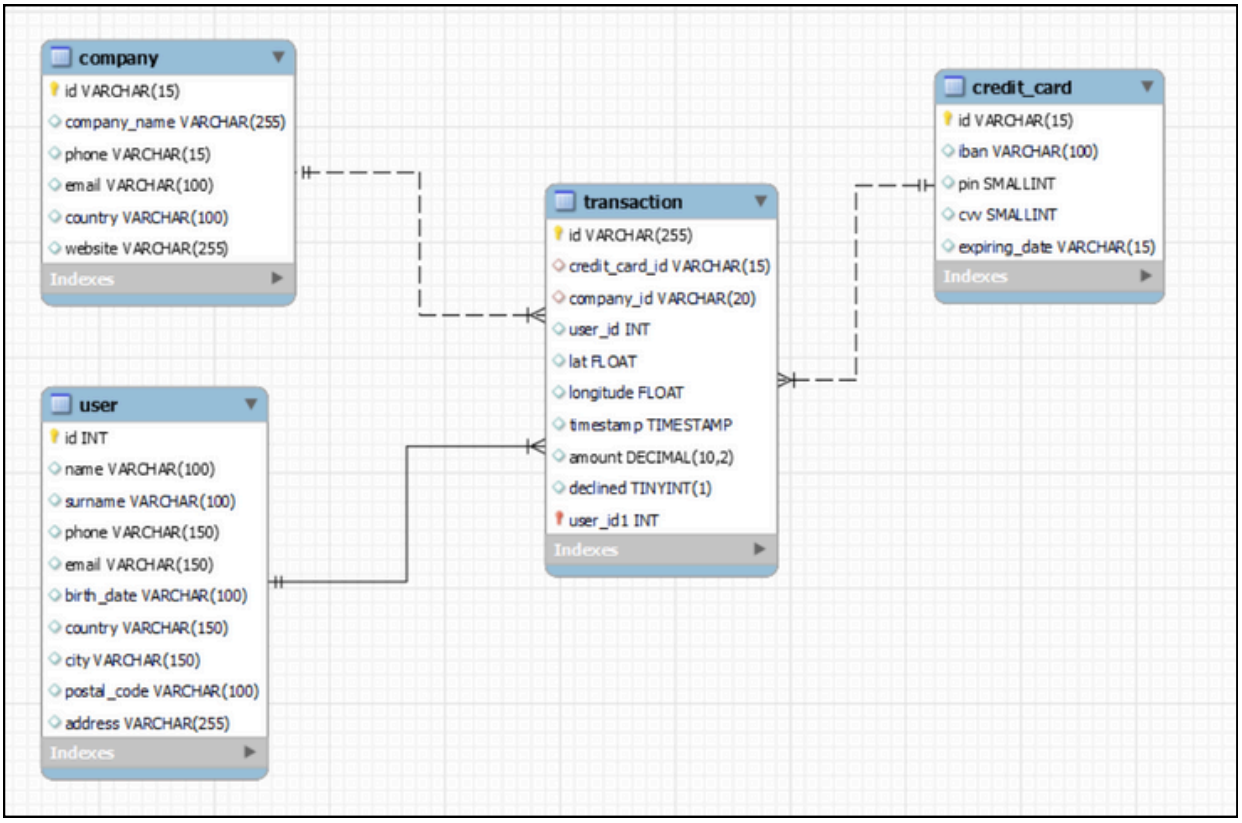
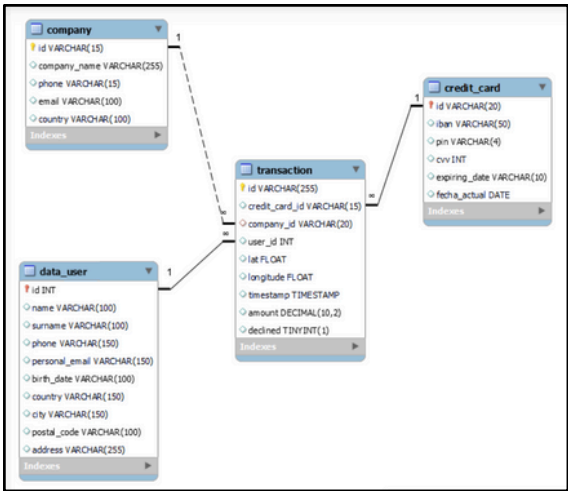
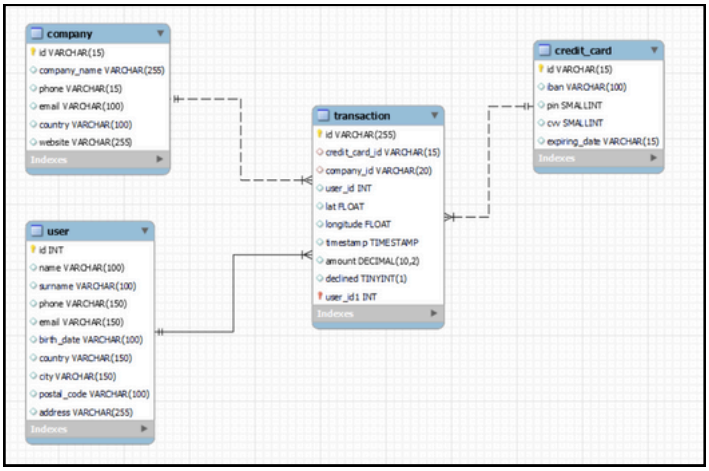


Figure 3.1.1 Relationship

Steps to change the column name and type:



Modified database



Needs to be modified database

company table DROP COLUMN:

```
55    #Level 3 exercise 1
56 •  ALTER TABLE company DROP COLUMN website;
57
```

Output

Action Output

#	Time	Action	Message
1	13:40:03	ALTER TABLE company DROP COLUMN website	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

EXPLANATION:

The modified database includes only the following columns: id, company_name, phone, email, and country. The column website should be eliminated.

credit_card table MODIFY COLUMNS:

```
60 •  ALTER TABLE credit_card
61    MODIFY COLUMN id VARCHAR (20),
62    MODIFY COLUMN iban VARCHAR (50),
63    MODIFY COLUMN pin VARCHAR (4),
64    MODIFY COLUMN cvv INT,
65    MODIFY COLUMN expiring_date VARCHAR (10);
66
```

Output

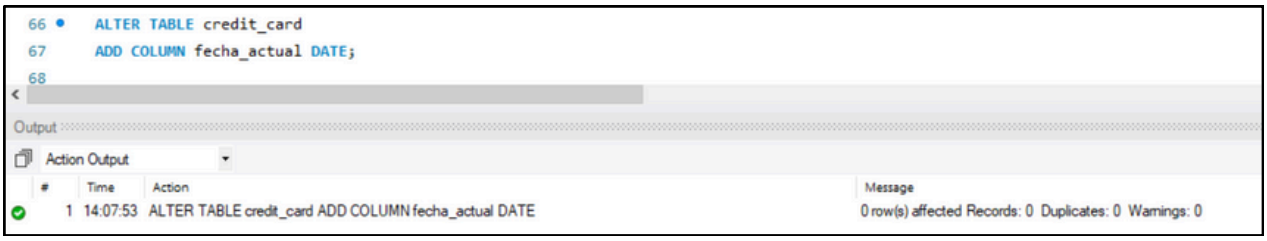
Action Output

#	Time	Action	Message
1	13:53:43	ALTER TABLE credit_card MODIFY COLUMN id VARCHAR (20), MODIFY COLUMN iban VARCHAR (50), MOD...	275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0

EXPLANATION:

The modified database has different data types in the credit_card table compared to the previous version. To align the data types with the required structure, the ALTER TABLE and MODIFY COLUMN commands were used to update the columns' data types accordingly.

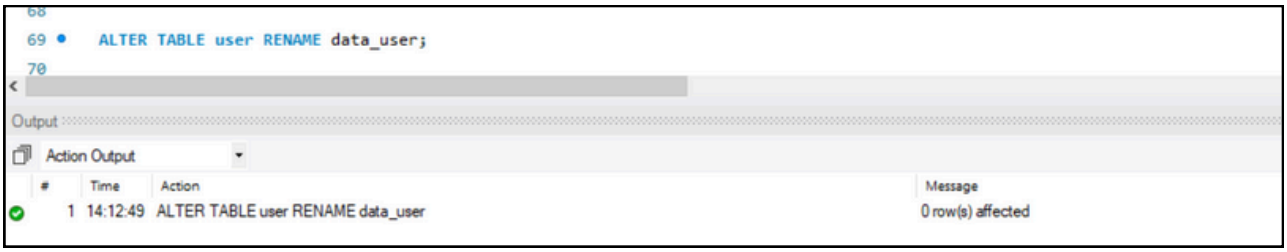
credit_card table ADD COLUMN:



EXPLANATION:

The modified database has a new column in the credit_card table. To add a new column to a table, use the ADD COLUMN command, specifying the name of the new column and its data type.

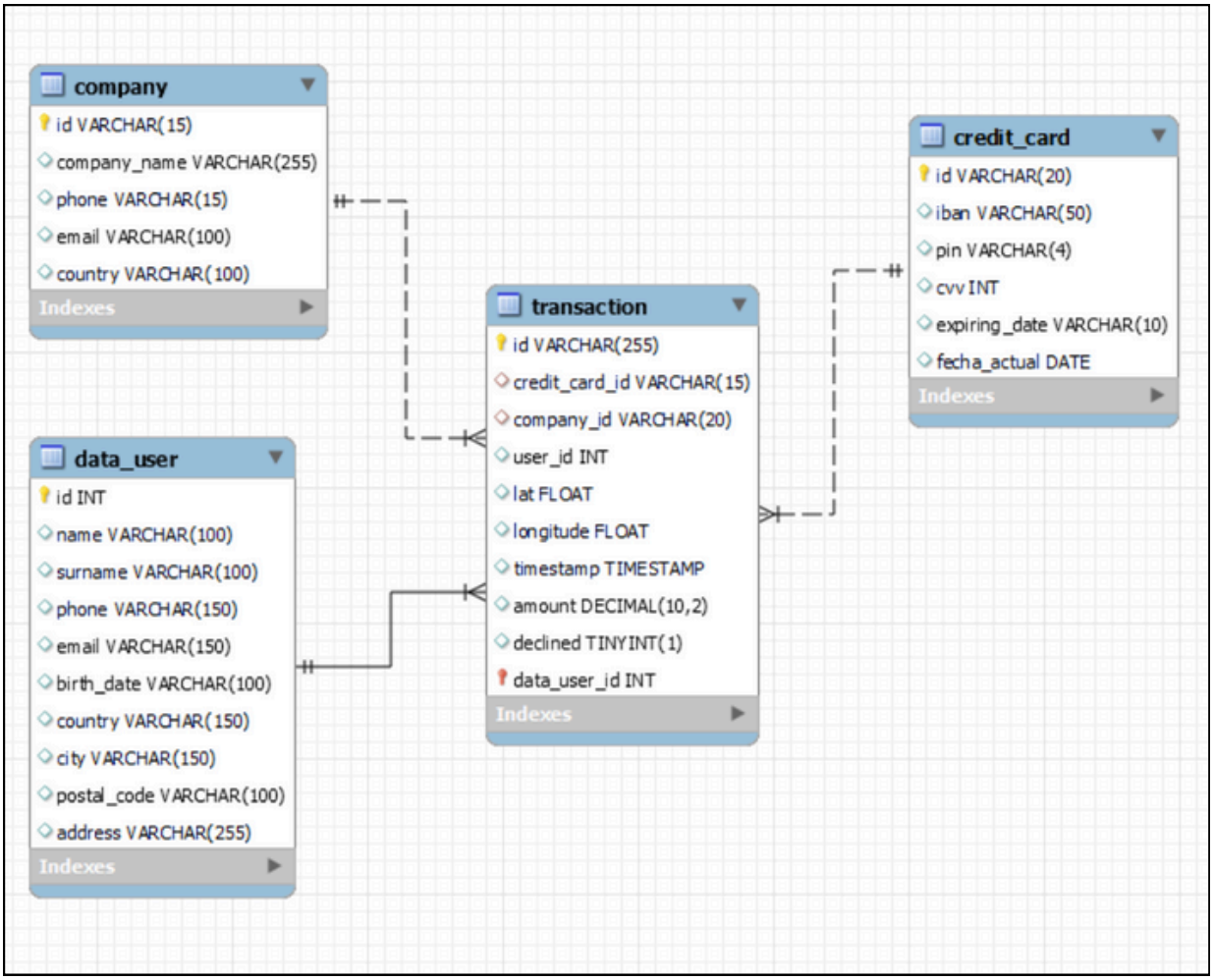
user table RENAME table name:



EXPLANATION:

The modified database has a different name for the user table. To change the name of the table, use the RENAME command. First, specify the table to alter, then write the RENAME command followed by the new name of the table.

MODIFIED DATABASE:



LEVEL 3

Exercise 2 :

GOAL 1: The company also asks you to create a view called "TechReport" containing the following information:

Transaction ID
User's first name
User's last name
IBAN of the credit card used.
Company name of the transaction made.

Be sure to include relevant information from both tables and use aliases to rename columns as needed. Display the results of the view, sort the results in descending order based on the transaction ID variable.

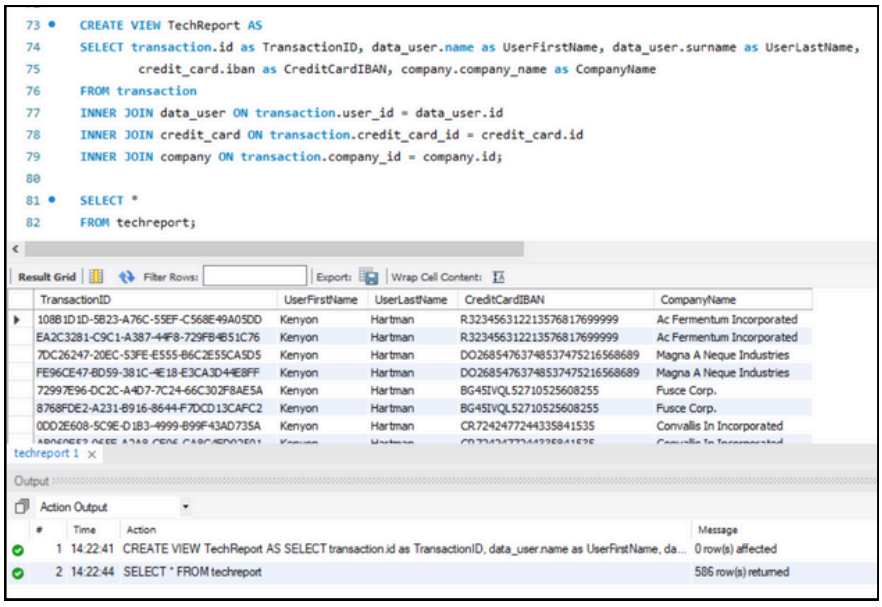


Figure 3.2.1 Relationship

EXPLANATION:
To do this view, first, create a query that fetches the following information:

- Transaction ID
- User's first name
- User's last name
- IBAN of the credit card used
- Company name of the transaction made

Using the INNER JOIN, connect the tables company and transaction where this information is located. These tables are linked by the id column in the id and the user_id column in the transaction table, the id column in the credit_card table and credit_card_id in the transaction table, and the id column in the company table and company_id column in the transaction table.

NOTE: Just be careful in naming the new columns to avoid confusion. It should be a manageable length and should have a different name than the other columns in the other tables.

Once the results meet the required information, create a view by writing the query above the SELECT query and naming it TechReport.