

# מודל נתונים עבור אגף פיננסים בבנק

שמות מגישות: דוד אוהב ציון ומרקוס צ'אמה.

המערכת: בנק

היחידה הנבחרת: ישנם שלושה אגפים בבנק: לקוחות, פיננסים, משאבי אנוש. אנו נתמקד באגף פיננסים.

## 1. מבוא

מודל נתונים זה מתאר את המבנה של בסיס נתונים עבור אגף פיננסים בבנק. המודל כולל מספר ישויות, ביניהן חשבונות, טרנזקציות, הלוואות, כרטיסי אשראי, שיקים ופקדונות. הקשרים בין הישויות מוגדרים כ-1:Many.

מטרות המערכת:

- ניהול נתוני חשבונות
- מעקב אחר טרנזקציות פיננסיות
- ניהול הלוואות וכרטיסי אשראי
- מעקב אחר תשלומים באמצעות שיקים
- ניהול פקדונות

## ישויות:

- חשבונות (Accounts):
  - תכונות:
    - מזהה חשבון (Account ID): מפתח ראשוני ייחודי לכל חשבון.
    - מזהה לקוח (Customer ID): מפתח זר המקשר לחשבון ללקוח שלו.
    - סוג חשבון (Account Type): סוג החשבון (חיסכון, עובר ושב, פקדון).
    - יתרת חשבון (Balance): סכום הכסף הזמין בחשבון.
    - תאריך פתיחת חשבון (Account Opening Date): התאריך שבו נפתח החשבון.
    - מפתח זר : Account ID
  - טרנזקציות (Transactions):
    - תכונות:
      - מזהה טרנזקציה (Transaction ID): מפתח ראשוני ייחודי לכל טרנזקציה.
      - מזהה חשבון (Account ID): מפתח זר המקשר את הטרנזקציה לחשבון.
      - תאריך טרנזקציה (Transaction Date): התאריך שבו בוצעה הטרנזקציה.
      - סכום (Amount): סכום הכסף שהועבר בטרנזקציה.
      - מפתח זר : Account ID

- הלוואות (Loans):

- תכונות:

- **מזהה הלוואה (Loan ID):** מפתח ראשוני ייחודי לכל הלוואה.
- סכום הלוואה (Loan Amount): סכום הכסף שהלווה הבנק ללקוח.
- ריבית (Interest Rate): שיעור הריבית על ההלוואה.
- תאריך התחלה (Start Date): התאריך שבו החלה ההלוואה.
- תאריך סיום (End Date): התאריך שבו תסתיים ההלוואה.
- מפתח זר : Account ID

- כרטיסי אשראי (Credit Cards):

- תכונות:

- **מזהה כרטיס (Card ID):** מפתח ראשוני ייחודי לכל כרטיס.
- מספר כרטיס (Card Number): מספר הכרטיס הייחודי.
- סוג כרטיס (Card Type): סוג הכרטיס (Debit, Regular).
- תאריך תפוגה (Expiration Date): התאריך שבו הכרטיס פג תוקפו.
- מסגרת אשראי (Credit Limit): סכום הכסף המקסימלי שניתן להוציא באמצעות הכרטיס.
- מפתח זר : Account ID

- שיקים (Checks):

- תכונות:

- **מזהה שיק (Check ID):** מפתח ראשוני ייחודי לכל שיק.
- מזהה חשבון (Account ID): מפתח זר המקשר את השיק לחשבון.
- מספר שיק (Check Number): מספר השיק הייחודי.
- סכום (Amount): סכום הכסף ששולם באמצעות השיק.
- תאריך הנפקה (Issue Date): התאריך שבו הונפק השיק.
- תאריך פרעון (Clearing Date): התאריך שבו השיק אמור להיפרע.
- מפתח זר : Account ID

- פקדונות (Deposits):

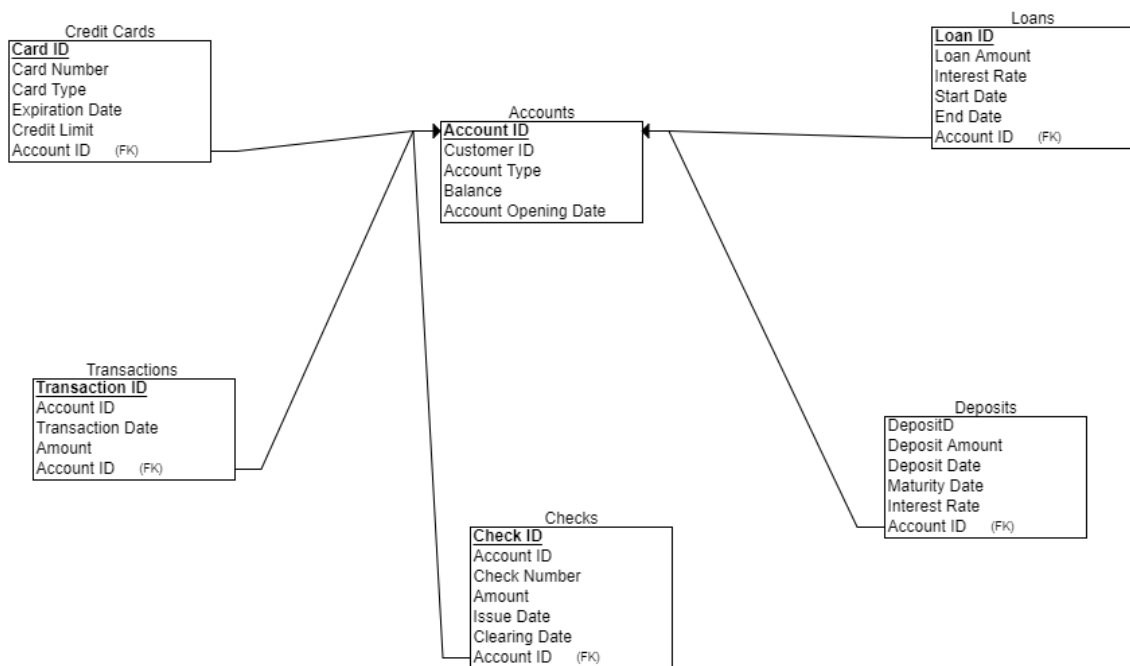
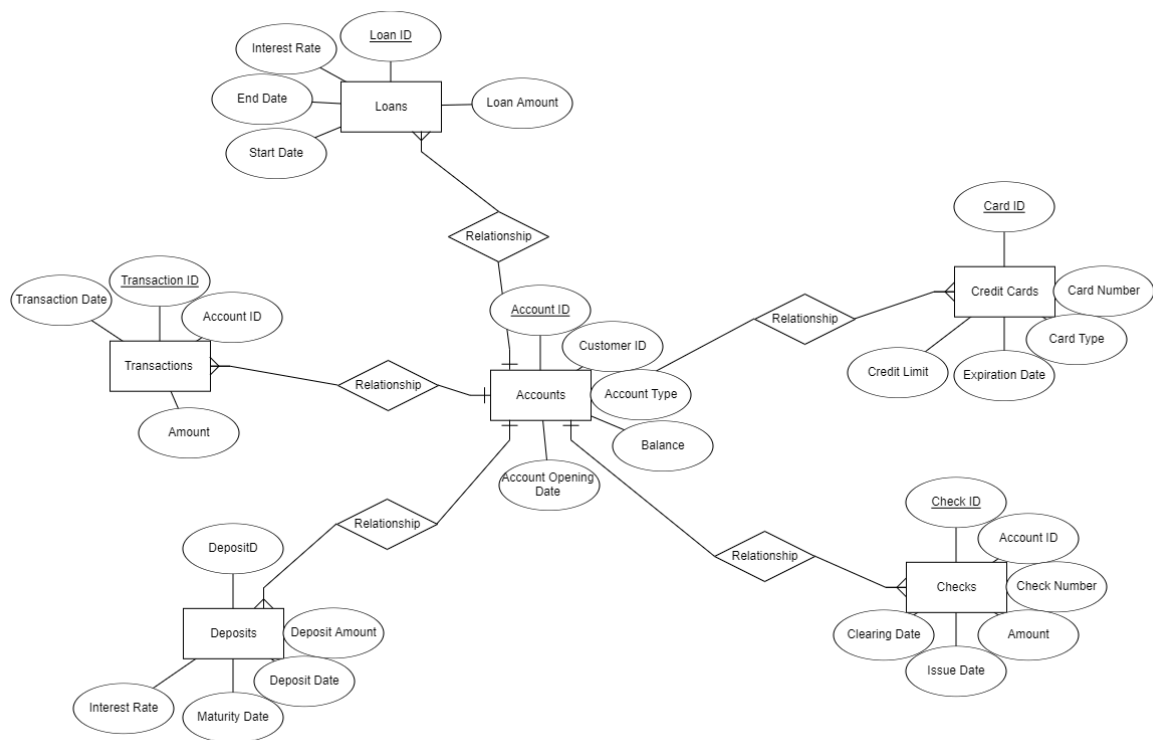
- תכונות:

- **מזהה פקדון (DepositID):** מפתח ראשוני ייחודי לכל פקדון.
- סכום פקדון (Deposit Amount): סכום הכסף שהופקד בחשבון.
- תאריך פקדון (Deposit Date): התאריך שבו בוצע הפקדון.
- תאריך פדיון (Maturity Date): התאריך שבו ניתן לפדות את הפקדון.
- ריבית (Interest Rate): שיעור הריבית על הפקדון.
- מפתח זר : Account ID

## קשרים: כל הקשרים מוגדרים כ-Many:1

- לקוחות-חשבונות: לקוח אחד יכול להחזיק מספר חשבונות. אבל לכל חשבון לקוח אחד
- חשבונות-טרנזקציות: חשבון אחד יכול להכיל מספר טרנזקציות. אך כל טרנזקציה משויכת לחשבון אחד.
- לקוחות-הלוואות: לקוח אחד יכול לקחת מספר הלוואות. אך כל הלוואה שייכת לחשבון אחד

- לקוחות-כרטיסי אשראי: לקוח אחד יכול להחזיק מספר כרטיסי אשראי. אך כל כרטיס אשראי שייך לחשבון אחד
- חשבונות-שיקים: חשבון אחד יכול להוציא מספר שיקים. אך על שייך שייך לחשבון אחד.
- לקוחות-פקדונות: לקוח אחד יכול להחזיק מספר פקדונות. אך כל פקדון שייך לחשבון אחד



כל הטבלאות הינם ברמה של 3NF ואין צורך בנרמול, נוכיח זאת:

כל הטבלאות עומדות ביחס של 1NF מפני שכל השדות הם אטומיים.

כל הטבלאות עומדות ביחס של 2NF בכל הטבלאות המפתח הינו רק שדה אחד, ולכן לא תתכן תלות בחלק מן המפתח אלא בכולו.

כל הטבלאות עומדות ביחס של 3NF : אין קשר בין השדות השונים, הקשר היחיד הוא ע"י שדה המפתח.

**יצירת הטבלאות:**



createTables.sql

dropTables.sql

selectAll.sql

main

insertTables.s

SQL

Output

Statistics

```
CREATE TABLE Accounts
(
    Account_ID INT NOT NULL,
    Customer_ID INT NOT NULL,
    Account_Type varchar2(10),
    Balance INT NOT NULL,
    Account_Opening_Date DATE NOT NULL,
    PRIMARY KEY (Account_ID)
);

CREATE TABLE Transactions
(
    Transaction_ID INT NOT NULL,
    Amount DATE NOT NULL,
    Account_ID INT NOT NULL,
    Transaction_Date DATE,
    PRIMARY KEY (Transaction_ID),
    FOREIGN KEY (Account_ID) REFERENCES Accounts (Account_ID)
);

CREATE TABLE Loans
(
    Loan_ID INT NOT NULL,
    Customer_ID INT NOT NULL,
    Loan_Amount INT NOT NULL,
    Interest_Rate INT NOT NULL,
    Start_Date DATE NOT NULL,
    End_Date DATE NOT NULL,
    Account ID INT NOT NULL,
```

	CHECK_ID	CHECK_NUMBER	AMOUNT	ISSUE_DATE	CLEARING_DATE	ACCOUNT_ID
1	401	98765	200	18/05/2024	21/05/2024	1
2	402	12345	150	17/05/2024	20/05/2024	2
3	403	67890	250	19/05/2024	22/05/2024	3
4	404	23456	300	20/05/2024	23/05/2024	4
5	405	34567	450	21/05/2024	24/05/2024	5
6	406	45678	500	22/05/2024	25/05/2024	6
7	407	56789	550	23/05/2024	26/05/2024	7
8	408	67891	600	24/05/2024	27/05/2024	8
9	409	78912	650	25/05/2024	28/05/2024	9
10	410	89012	700	26/05/2024	29/05/2024	10

	DEPOSITID	CUSTOMER_ID	DEPOSIT_AMOUNT	DEPOSIT_DATE	MATURITY_DATE	INTEREST_RATE	ACCOUNT_ID
1	501	10	3000	15/05/2024	20/05/2025	3	1
2	504	40	4000	17/05/2024	22/05/2025	4	4
3	505	50	4500	18/05/2024	23/05/2025	4	5
4	506	60	5000	19/05/2024	24/05/2025	4	6
5	507	70	5500	20/05/2024	25/05/2025	4	7
6	508	80	6000	21/05/2024	26/05/2025	4	8
7	509	90	6500	22/05/2024	27/05/2025	5	9
8	510	100	7000	23/05/2024	28/05/2025	5	10
9	502	20	2000	10/05/2024	10/05/2025	3	2
10	503	30	3500	16/05/2024	21/05/2025	4	3

	CARD_ID	CUSTOMER_ID	CARD_NUMBER	CARD_TYPE	EXPIRATION_DATE	CREDIT_LIMIT	ACCOUNT_ID
1	301	10	123456789012	Visa	20/05/2027	15000	1
2	302	20	987654321012	Mastercard	20/05/2026	10000	2
3	303	30	111122223333	Visa	30/06/2028	12000	3
4	304	40	444455556666	Mastercard	15/04/2025	8000	4
5	305	50	777788889999	Visa	22/08/2027	9000	5
6	306	60	123443215678	Mastercard	31/12/2026	11000	6
7	307	70	876543219876	Visa	01/01/2029	14000	7
8	308	80	112233445566	Mastercard	20/07/2026	13000	8
9	309	90	998877665544	Visa	25/11/2027	16000	9
10	310	100	334455667788	Mastercard	30/05/2025	7000	10

	LOAN_ID	CUSTOMER_ID	LOAN_AMOUNT	INTEREST_RATE	START_DATE	END_DATE	ACCOUNT_ID
1	201	10	10000	7	20/05/2023	20/05/2024	1
2	202	20	5000	8	20/05/2022	20/05/2023	2
3	203	30	10000	7	20/05/2023	20/05/2024	3
4	204	40	5000	8	20/05/2022	20/05/2023	4
5	205	50	10000	7	20/05/2023	20/05/2024	5
6	206	60	5000	8	20/05/2022	20/05/2023	6
7	207	70	10000	7	20/05/2023	20/05/2024	7
8	208	80	5000	8	20/05/2022	20/05/2023	8
9	209	90	10000	7	20/05/2023	20/05/2024	9
10	210	100	5000	8	20/05/2022	20/05/2023	10



	TRANSACTION_ID	AMOUNT	ACCOUNT_ID	TRANSACTION_DATE	
▶	1	101	500	1	18/05/2024
	2	102	410	2	18/05/2024
	3	103	487	3	18/05/2024
	4	104	250	4	18/05/2024
	5	105	452	5	18/05/2024
	6	106	250	6	18/05/2024
	7	107	553	7	18/05/2024
	8	108	255	8	18/05/2024
	9	109	550	9	18/05/2024
	10	110	111	10	18/05/2024

נתונים אלו הוכנסו ידנית על ידי הקובץ InsertTable.sql

```

createTables.sql dropTables.sql selectAll.sql main insertTables.sql X
SQL Output Statistics
INSERT INTO Accounts (Account_ID, Customer_ID, Account_Type, Balance, Account_Opening_Date)
VALUES (1, 10, 'Savings', 1000, DATE '2024-05-20');

INSERT INTO Accounts (Account_ID, Customer_ID, Account_Type, Balance, Account_Opening_Date)
VALUES (2, 20, 'Checking', 2000, DATE '2024-05-15');

INSERT INTO Accounts (Account_ID, Customer_ID, Account_Type, Balance, Account_Opening_Date)
VALUES (3, 30, 'Checking', 2555, DATE '2024-05-16');

INSERT INTO Accounts (Account_ID, Customer_ID, Account_Type, Balance, Account_Opening_Date)
VALUES (4, 40, 'Checking', 2040, DATE '2024-05-19');
INSERT INTO Accounts (Account_ID, Customer_ID, Account_Type, Balance, Account_Opening_Date)
VALUES (5, 50, 'Savings', 2900, DATE '2024-05-18');
INSERT INTO Accounts (Account_ID, Customer_ID, Account_Type, Balance, Account_Opening_Date)
VALUES (6, 60, 'Checking', 9999, DATE '2024-05-12');
INSERT INTO Accounts (Account_ID, Customer_ID, Account_Type, Balance, Account_Opening_Date)
VALUES (7, 70, 'Savings', 1542, DATE '2024-05-07');
INSERT INTO Accounts (Account_ID, Customer_ID, Account_Type, Balance, Account_Opening_Date)
VALUES (8, 80, 'Checking', 8521, DATE '2024-05-06');
INSERT INTO Accounts (Account_ID, Customer_ID, Account_Type, Balance, Account_Opening_Date)
VALUES (9, 90, 'Savings', 4523, DATE '2024-05-05');
INSERT INTO Accounts (Account_ID, Customer_ID, Account_Type, Balance, Account_Opening_Date)
VALUES (10, 100, 'Checking', 4658, DATE '2024-05-01');

INSERT INTO Transactions (Transaction_ID, Amount, Account_ID, Transaction_Date)
VALUES (101, 500, 1, DATE '2024-05-18');

INSERT INTO Transactions (Transaction_ID, Amount, Account_ID, Transaction_Date)
VALUES (102, 410, 2, DATE '2024-05-18');

```

נראה שימוש ב-Data Generator להכנסת מידע באופן אוטומטי:

depositdg.gd

DEPOSITS

Owner: SYS Table: DEPOSITS Number of records: 500

Name	Type	Size	Data	Master
DEPOSITID	NUMBER		Sequence(999, [Inc], [WithinParent])	...
DEPOSIT_AMOUNT	NUMBER		Random(20, 500)	...
DEPOSIT_DATE	DATE		Random(01/01/18, 01/01/21)	...
✓ MATURITY_DATE	DATE		Random(01/01/21, 01/01/25)	...
INTEREST_RATE	NUMBER		Random(1, 7)	...
ACCOUNT_ID	NUMBER		List(select account_id from accounts)	...
*				...

Definition Options Result

loansgd.gd

LOANS

Owner: SYS Table: LOANS Number of records: 400

Name	Type	Size	Data	Master
LOAN_ID	NUMBER		Sequence(99999, [Inc], [WithinParent])	...
LOAN_AMOUNT	NUMBER		Random(100, 100000)	...
INTEREST_RATE	NUMBER		Random(1, 7)	...
START_DATE	DATE		Random(01/01/18, 01/01/20)	...
✓ END_DATE	DATE		Random(01/01/20, 01/01/25)	...
ACCOUNT_ID	NUMBER		List(select account_ID from accounts)	...
*				...

Close tab

ACCOUNTS

Owner: SYS Table: ACCOUNTS Number of records: 500

Name	Type	Size	Data	Master
ACCOUNT_ID	NUMBER		Sequence(10, [Inc], [WithinParent])	...
CUSTOMER_ID	NUMBER		Random(100, 10000)	...
ACCOUNT_TYPE	VARCHAR2	10	List('saving', 'chaking', 'loan', 'current')	...
BALANCE	NUMBER		Random(-100, 12000)	...
▶ ACCOUNT_OPENING_C DATE	DATE		Random(01/01/13, 01/01/18)	...
*				...

CHECKS

<

Owner

Table

Number of records

>

SYS

CHECKS

500

...

Name	Type	Size	Data	Master
✓ CHECK_ID	NUMBER		Sequence(999999, [Inc], [WithinParent])	***
CHECK_NUMBER	NUMBER		Random(100000, 10000000)	***
AMOUNT	NUMBER		Random(100,10000)	***
ISSUE_DATE	DATE		Random(01/01/18, 01/01/24)	***
CLEARING_DATE	DATE		Random(01/01/24, 01/01/25)	***
ACCOUNT_ID	NUMBER		List(select account_ID from accounts)	***
*				***

trangd.gd

</

CREDIT\_CARDS

< Owner

Table

Number of records

> SYS

CREDIT\_CARDS

500

...

Name	Type	Size	Data	Master
CARD_ID	NUMBER		Sequence(9999, [Inc], [WithinParent])	***
CARD_NUMBER	NUMBER		Random(1000000, 9999999)	***
CARD_TYPE	VARCHAR2	10	List('visa', 'mastercard', 'debit')	***
▶ EXPIRATION_DATE	DATE		Random(01/01/21, 01/01/26)	***
CREDIT_LIMIT	NUMBER		Signal(2000, 10000, 500,1)	***
ACCOUNT_ID	NUMBER		List(select account_id from accounts)	***
✱				***

## בונה סקריפט בפייתון שיכניס נתונים לטבלאות:

```
C:\Users\ohevd\Downloads\DataImporterFiles > pythonSql.py
1 import random
2 from faker import Faker
3
4 fake = Faker()
5
6
7 # Function to generate SQL insert statements
8 def generate_sql():
9     with open('insert_records.sql', 'w') as f:
10         # Generate 400 records for Accounts table
11         for account_id in range(1000, 1401):
12             customer_id = account_id
13             balance = random.randint(1000, 50000)
14             account_opening_date = fake.date_this_decade()
15             f.write(
16                 f"INSERT INTO Accounts (Account_ID, Customer_ID, Balance, Account_Opening_Date) VALUES ({account_id}, {customer_id}, {balance}, TO_DATE('{account_opening_
17
18         # Generate 400 records for Transactions table
19         for transaction_id in range(1000, 1401):
20             amount = random.randint(100, 10000)
21             account_id = random.randint(1, 400)
22             transaction_date = fake.date_this_decade()
23             f.write(
24                 f"INSERT INTO Transactions (Transaction_ID, Amount, Account_ID, Transaction_Date) VALUES ({transaction_id}, {amount}, {account_id}, TO_DATE('{transaction_
25
26         # Generate 400 records for Loans table
27         for loan_id in range(1000, 1401):
28             loan_amount = random.randint(5000, 100000)
29             interest_rate = random.randint(1, 10)
30             start_date = fake.date_this_decade()
31             end_date = fake.date_between(start_date=start_date, end_date='+5y')
32             account_id = random.randint(1, 400)
33             f.write(
34                 f"INSERT INTO Loans (Loan_ID, Loan_Amount, Interest_Rate, Start_Date, End_Date, Account_ID) VALUES ({loan_id}, {customer_id}, {loan_amount}, {interest_rat
35
```

```
> Users\ohevd\Downloads\DataImporterFiles > pythonSql.py
8 def generate_sql():
9
10     for card_id in range(1000, 1401):
11         card_number = fake.credit_card_number(card_type=None)
12         card_type = fake.credit_card_provider()
13         expiration_date = fake.date_this_decade()
14         credit_limit = random.randint(5000, 20000)
15         account_id = random.randint(1, 400)
16         f.write(
17             f"INSERT INTO Credit_Cards (Card_ID, Card_Number, Card_Type, Expiration_Date, Credit_Limit, Account_ID) VALUES ({card_id}, {customer_id}, {card_number}, '
18
19     # Generate 400 records for Checks table
20     for check_id in range(1000, 1401):
21         check_number = random.randint(100000, 999999)
22         amount = random.randint(100, 10000)
23         issue_date = fake.date_this_decade()
24         clearing_date = fake.date_between(start_date=issue_date, end_date='+1y')
25         account_id = random.randint(1, 400)
26         f.write(
27             f"INSERT INTO Checks (Check_ID, Check_Number, Amount, Issue_Date, Clearing_Date, Account_ID) VALUES ({check_id}, {check_number}, {amount}, TO_DATE('{issue
28
29     # Generate 400 records for Deposits table
30     for deposit_id in range(1000, 1401):
31         deposit_amount = random.randint(1000, 50000)
32         deposit_date = fake.date_this_decade()
33         maturity_date = fake.date_between(start_date=deposit_date, end_date='+5y')
34         interest_rate = random.randint(1, 10)
35         account_id = random.randint(1, 400)
36         f.write(
37             f"INSERT INTO Deposits (DepositID, Deposit_Amount, Deposit_Date, Maturity_Date, Interest_Rate, Account_ID) VALUES ({deposit_id}, {customer_id}, {deposit_a
38
39 if __name__ == "__main__":
40     generate_sql()
41
```

נראה הכנסה של מידע מקובץ טקסט לטבלה accounts:

```
*C:\Users\ohevd\Downloads\מסמך טקסט חדש.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
createTables.sql | dropTables.sql | insertTables.sql | selectAll.sql | מסמך טקסט חדש.txt
1 500,5000, Savings, 1000, 20/05/2024
2 600,6000, Savings, 2000, 21/05/2024
3 700,7000, Savings, 3000, 22/05/2024
4 800,8000, Savings, 4000, 23/05/2024
```

createTables.sql | dropTables.sql | selectAll.sql | main | insertTables.sql | text1.txt

Data from Textfile | Data to Oracle

File Data	Encoding
500,5000, "Savings", 1000, 20/05/2024	
600,6000, "Savings", 2000, 21/05/2024	
700,7000, "Savings", 3000, 22/05/2024	
800,8000, "Savings", 4000, 23/05/2024	

**Configuration**

General

Fieldcount: 5

☒ End at line-end

☐ Name in header

☒ Skip empty lines

Quote character: "

Comment line: --

Import lines: 1 ..

Field1 (+0 .. ",")

Field2 (+0 .. ",")

Field3 (+0 .. ",")

Field4 (+0 .. ",")

Field5 (+0 .. ",")

Field Start:

☒ Relative position

☐ Absolute position

☐ Character

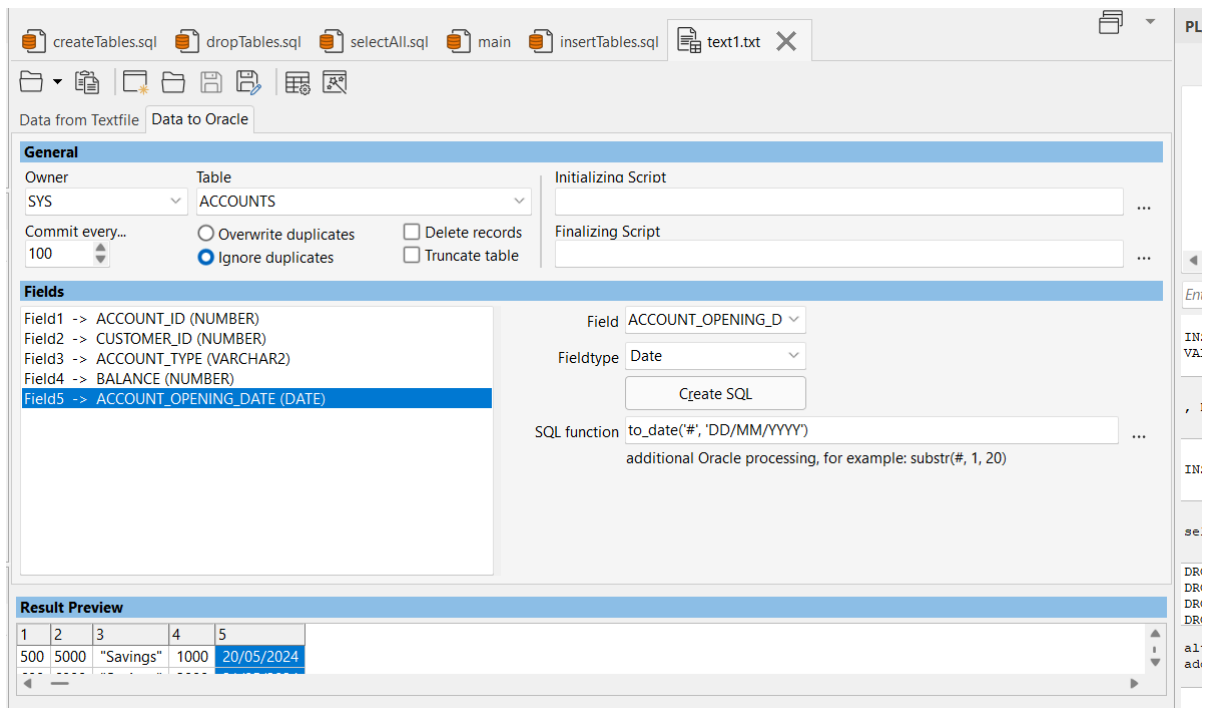
Field End:

☐ Length

☒ Character

Filter:

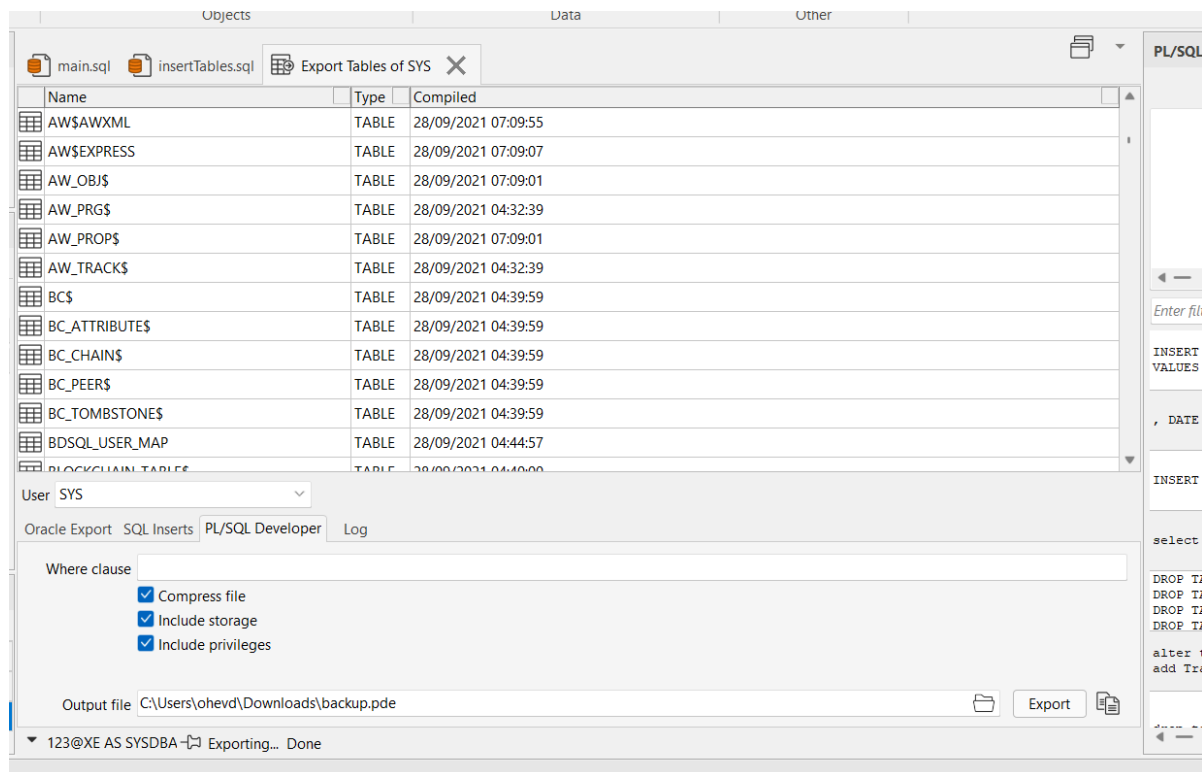
Apply



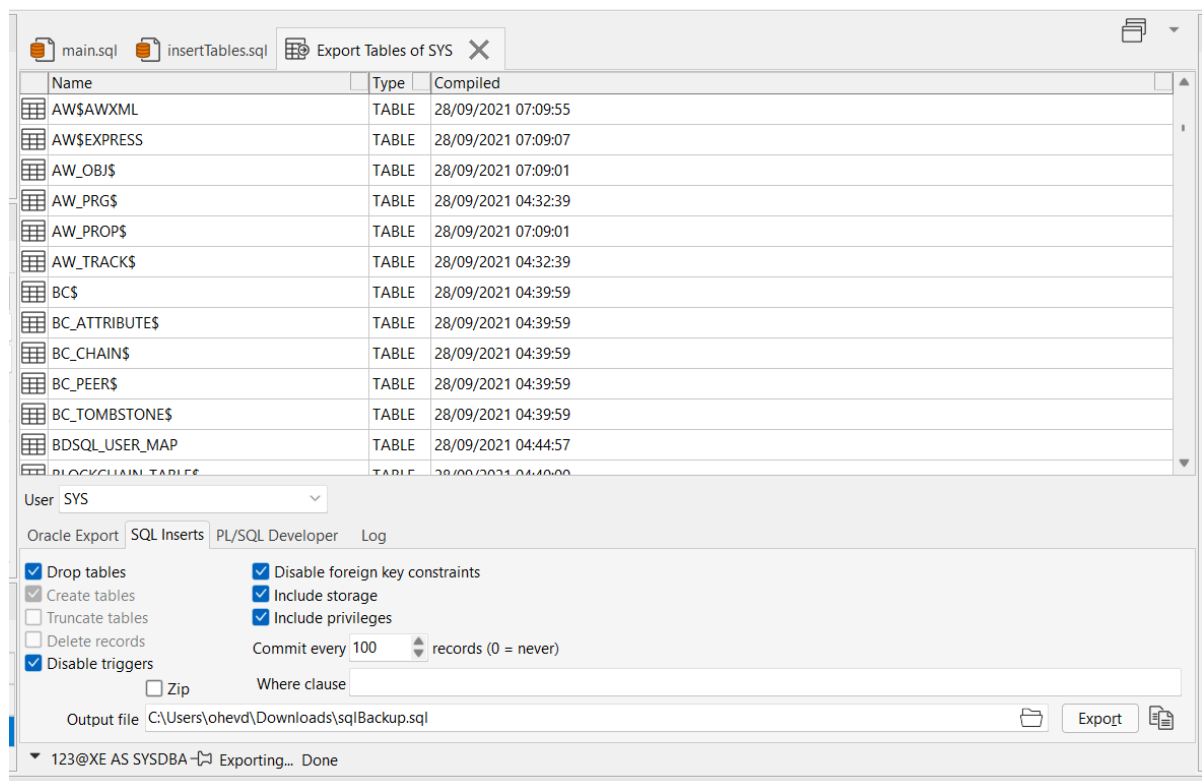
מדהים, ניתן גם בקלות לעשות סקריפט בפייתון שימלא לנו את הקובץ, ואז נמשוך את המידע מהקובץ בצורה זו.

בעצם הראנו ארבע אופציות איך להכניס מידע לטבלה: ידני, סקריפט, קובץ טקסט, Data Generator.

נבצע גיבוי למסד הנתונים בכדי שנוכל לפתוח אותו מכל מחשב.  
נבצע export :



כמו כן ניתן לייצא את הגיבוי לקובץ SQL



קובץ הSQL נראה ככה:

```
sqlBackup21.5.24.sql X
C: > Users > ohevd > Downloads > sqlBackup21.5.24.sql
1 prompt PL/SQL Developer Export Tables for user SYS@XE
2 prompt Created by ohevd on יום שני 21 יולי 2024
3 set feedback off
4 set define off
5
6 prompt Dropping ACCOUNTS...
7 drop table ACCOUNTS cascade constraints;
8 prompt Dropping CHECKS...
9 drop table CHECKS cascade constraints;
10 prompt Dropping CREDIT_CARDS...
11 drop table CREDIT_CARDS cascade constraints;
12 prompt Dropping DEPOSITS...
13 drop table DEPOSITS cascade constraints;
14 prompt Dropping LOANS...
15 drop table LOANS cascade constraints;
16 prompt Dropping TRANSACTIONS...
17 drop table TRANSACTIONS cascade constraints;
18 prompt Creating ACCOUNTS...
19 create table ACCOUNTS
20 (
21     account_id          INTEGER not null,
22     customer_id          INTEGER not null,
23     account_type         VARCHAR2(10),
24     balance              INTEGER not null,
25     account_opening_date DATE not null
26 )
27 tablespace SYSTEM
28 pctfree 10
29 pctused 40
30 initrans 1
31 maxtrans 255
32 storage
33 (
34     initial 64K
35     next 1M
36     minextents 1
```

נדגים מחיקה של הטבלאות והחזרתן:  
נריץ את הקובץ דרופ אול:



main.sql

insertTables.sql

Export Tables of SYS

dropTables.sql

SQL

Output

Statistics

DROP TABLE Transactions;

DROP TABLE Loans;

DROP TABLE Credit\_Cards;

DROP TABLE Deposits;

DROP TABLE Checks;

DROP TABLE Accounts;

Drop transactions

Drop loans

Drop credit\_cards

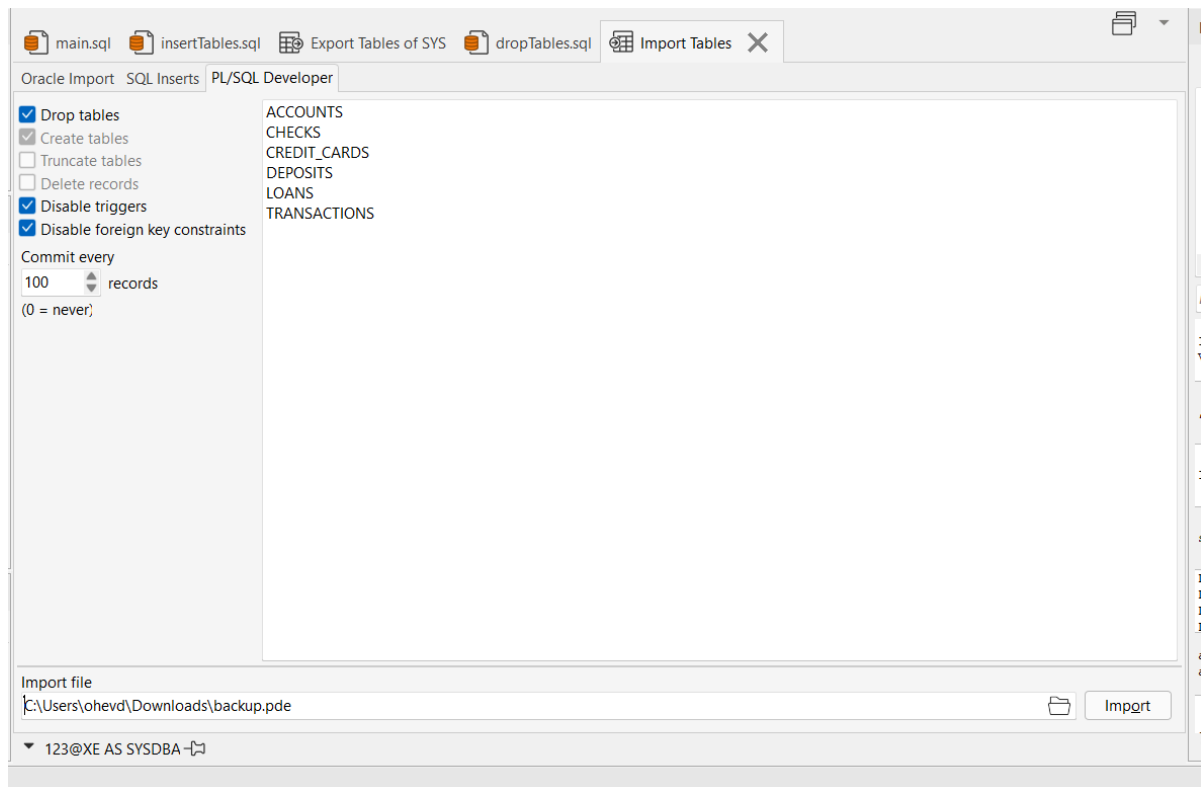
Drop deposits

Drop checks

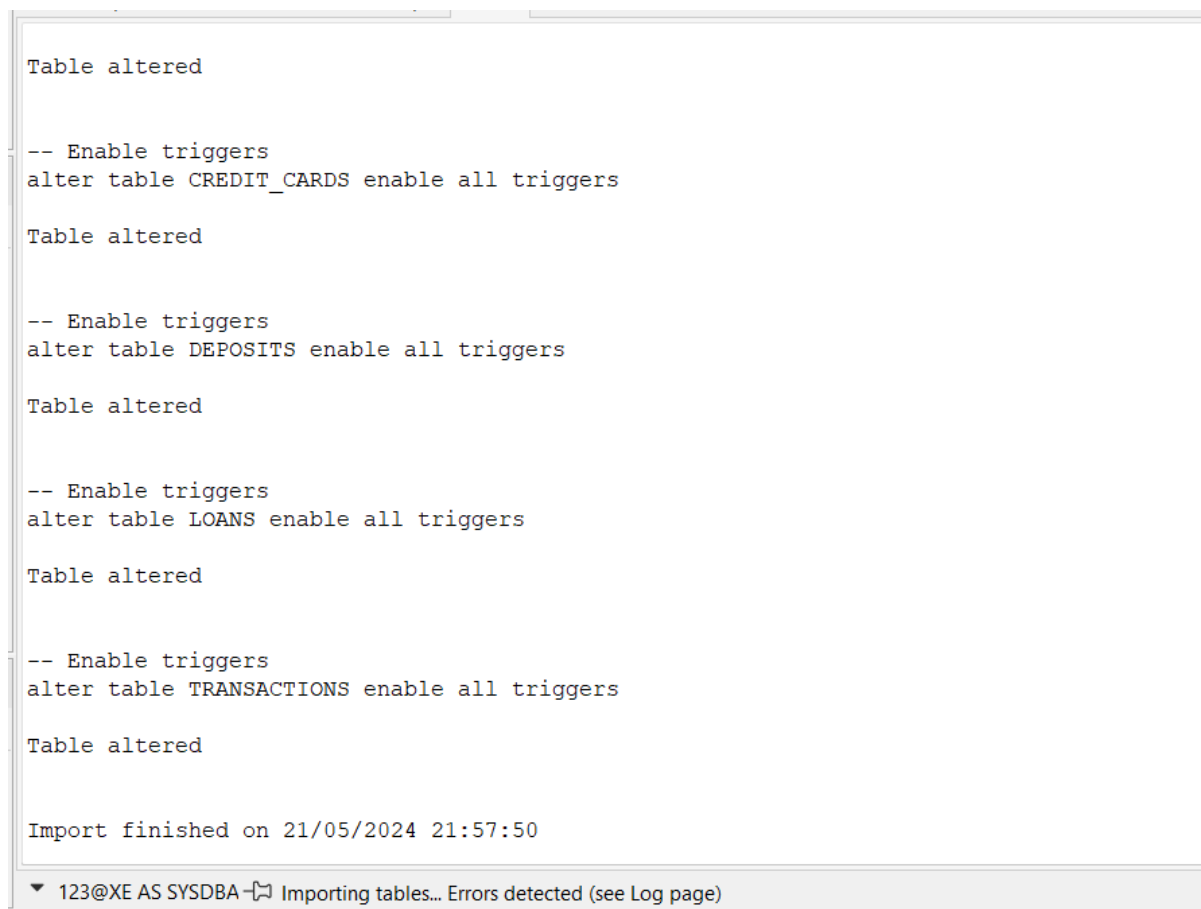
Drop accounts

(no result set)

נבצע שיחזור:



השיחזור הושלם:



נרץ את הקובץ selectAll בכדי לראות שהנתונים חזרו:

The screenshot shows a SQL IDE interface. The top toolbar includes buttons for opening files, saving, and running queries. The main window is titled 'selectAll.sql' and contains the following SQL code:

```
select * from Transactions;  
select * from Loans;  
select * from Credit_Cards;  
select * from Deposits;  
select * from Checks;  
select * from Accounts;
```

Below the code editor, there are tabs for 'Select transactions', 'Select loans', 'Select credit\_cards', 'Select deposits', 'Select checks', and 'Select accounts'. The 'Select transactions' tab is active, showing a table with the following data:

	TRANSACTION_ID	AMOUNT	ACCOUNT_ID	TRANSACTION_DATE	
1	101	500	1	18/05/2024	...
2	102	410	2	18/05/2024	...
3	103	487	3	18/05/2024	...
4	104	250	4	18/05/2024	...
5	105	452	5	18/05/2024	...
6	106	250	6	18/05/2024	...
7	107	553	7	18/05/2024	...
8	108	255	8	18/05/2024	...
9	109	550	9	18/05/2024	...
10	110	111	10	18/05/2024	...
11	7955	1923	85	15/05/2024	...
12	3562	1061	333	05/05/2024	...
13	6448	1421	697	06/05/2024	...

The status bar at the bottom indicates '123@XE AS SYSDBA [21:59:12] 13 rows selected in 0.088 seconds (more...)'. The status bar also shows '000' and '1 of 13'.

טא דא! הכל שב אלינו.

עד כאן שלב 1.

## שלב 2:

ניצור שאילתות מורכבות:

The screenshot shows a SQL IDE interface. The main window is titled 'SQL' and contains the following SQL code:

```
SELECT Customer_ID, COUNT(Account_ID) AS Num_Accounts, AVG(Balance) AS Avg_Balance  
FROM Accounts  
GROUP BY Customer_ID  
ORDER BY Num_Accounts DESC;
```

בשאלתה הנ"ל ניתן לראות שאנו מחפשים את כל הלקוחות שיש להם כמה חשבונות, ומחזירים לכל לקוח את כמות החשבונות שלו, ביחד עם הממוצע בכל החשבונות, וכמובן את מספר הלקוח. הנה דוגמא לפלט:

		CUSTOMER_ID	NUM_ACCOUNTS	AVG_BALANCE
▶	1	5073	3	5577.666666666667
	2	2632	2	6009
	3	6577	2	3233
	4	3358	2	2987
	5	9725	2	6596
	6	5753	2	1326.5
	7	50	1	2900
	8	3772	1	11083
	9	4487	1	3840
	10	8356	1	9295
	11	9206	1	11446
	12	2110	1	4814
	13	424	1	10734
	14	6871	1	5898

הנה עוד שאלתה:

```
SELECT Account_ID, Account_Type, Balance
FROM Accounts
WHERE Account_Opening_Date < TO_DATE('2024-05-20', 'YYYY-MM-DD')
AND Account_ID NOT IN (
    SELECT DISTINCT Account_ID FROM Transactions);
```

בדוגמא הנ"ל ניתן לראות שאנו מחפשים את כל החשבונות שלא היו פעילים בשבוע האחרון, כלומר לא ביצעו של טרנזקציה בשבוע שחלף. אבל גם נוודא שהם לא חדשים בבנק ואכן החשבון שלהם נפתח לפני שבוע ויותר. נייצג את הנתונים בעזרת מספר חשבון, יתרת חשבון, וסוג חשבון. דוגמא לפלט:

	ACCOUNT_ID	ACCOUNT_TYPE	BALANCE
▶	1	165 Checking	975649
	2	163 Checking	915437
	3	153 Savings	783778
	4	149 Checking	892744
	5	147 Savings	48356
	6	143 Checking	380458
	7	142 Savings	174840
	8	135 Checking	696326
	9	123 Savings	677440

שאלתה 3:

```

select A.account_Id,
       max(D.Deposit_Amount) as maxDeposit,
       max(L.Loan_Amount) as maxLoan
from accounts A
Join Loans L on A.ACCOUNT_ID = L.Account_id
Join Deposits D on A.ACCOUNT_ID = D.Account_id
Group BY A.ACCOUNT_ID;

```

כאן אנו מחפשים עבור כל חשבון מה הסכום הגבוהה ביותר שהוא לקח להלוואה ומה הסכום הגבוה ביותר שהוא הפקיד לחשבון. את נתונים אלו נציג ביחד עם מספר חשבון:

	ACCOUNT_ID	MAXDEPOSIT	MAXLOAN
1	6	5000	5000
2	34293	225	71996
3	554	451	81157
4	124	172	9369
5	69493	85	47970
6	161	138	32841
7	66260	331	297
8	63305	474	78390
9	770	335	44754

#### שאלתה 4:

```

SELECT A.Account_ID,
       CC.Card_Number,
       CC.Expiration_Date AS Card_Expiration_Date,
       L.Loan_ID,
       L.Interest_Rate AS Loan_Interest_Rate
FROM Accounts A
LEFT JOIN Credit_Cards CC ON A.Account_ID = CC.Account_ID AND
       CC.Expiration_Date <= ADD_MONTHS(SYSDATE, 1)
LEFT JOIN Loans L ON A.Account_ID = L.Account_ID AND
       L.End_Date <= ADD_MONTHS(SYSDATE, 1);

```

כאן אנו בעצם מבקשים לחפש עבור כל מספר חשבון אם יש לו הלוואה שעומדת להגמר בקרוב וכן אם יש לו כרטיס אשראי שעומד לפוג תוקפו לא עלינו. לגבי ההלוואה נחזיר גם את מידת הריבית שהוא משלם על ההלוואה, בכדי שהלקוח יהיה מרוצה ויראה לפניו כמה ריבית הוא משלם ועכשיו ההלוואה עומדת להגמר ומה טוב ומה נעים שלא יצטרך להמשיך לשלם אותה ריבית.

	ACCOUNT_ID	CARD_NUMBER	CARD_EXPIRATION_DATE	LOAN_ID	LOAN_INTEREST_RATE
1	1		***	201	7
2	1		***	8893	1
3	2	5776057	01/05/2024	202	8
4	3	5029408	18/05/2024	203	7
5	3	5029408	18/05/2024	4176	3
6	3	4041855	15/05/2024	203	7
7	3	4041855	15/05/2024	4176	3
8	4		***	204	8
9	4		***	6916	4

ניתן לראות שבהתאם לאופי השאלתה לא כל הטבלאות מלאות, וזאת משום שלא כל חשבון יש לו גם את הנתונים האלו וגם את אלו, אל ישנם חלק שיש להם הלוואה שעומדת להגמר וחלק שכרטיס האשראי עומד לפוג.

שאלתת מחיקה:

```
DELETE FROM Credit_Cards
WHERE Expiration_Date < SYSDATE;
```

```
select * from Credit_Cards
```

נראה את התוצאה:

CARD_ID	CARD_NUMBER	CARD_TYPE	EXPIRATION_DATE	CREDIT_LIMIT	ACCOUNT_ID
1	301	123456789012 Visa	20/05/2027	15000	1
2	302	987654321012 Mastercard	20/05/2026	10000	2
3	303	111122233333 Visa	30/06/2028	12000	3
4	304	444455566666 Mastercard	15/04/2025	8000	4
5	305	777788889999 Visa	22/08/2027	9000	5
6	306	123443215678 Mastercard	31/12/2026	11000	6
7	307	876543219876 Visa	01/01/2029	14000	7
8	308	112233445566 Mastercard	20/07/2026	13000	8
9	309	998877665544 Visa	25/11/2027	16000	9

נריך ROLLBACK בכדי לא לאבד את הנתונים.

הוכחה לאי איבוד מידע

```
DELETE FROM Credit_Cards
WHERE Expiration_Date < SYSDATE;
```

```
ROLLBACK;
```

```
select * from Credit_Cards
where Expiration_Date < SYSDATE;
```

CARD_ID	CARD_NUMBER	CARD_TYPE	EXPIRATION_DATE	CREDIT_LIMIT	ACCOUNT_ID
1	76249	9427907 mastercard	01/05/2024	6290	65893
2	19012	9252370 debit	01/05/2024	6262	14577
3	33795	1634604 mastercard	12/05/2024	6157	163
4	75548	1953775 mastercard	07/05/2024	5938	500
5	64505	7537240 visa	06/05/2024	5593	22252
6	92580	5919936 mastercard	07/05/2024	5157	421
7	56128	3997988 debit	15/05/2024	4721	69603
8	40997	4120698 mastercard	20/05/2024	4375	91608
9	98262	3428629 debit	07/05/2024	4158	46450

עוד שאילתת מחיקה

```
DELETE FROM Loans
WHERE End_Date < SYSDATE;
```

```
ROLLBACK;
```

נמחק את כל ההלוואות שזמן הסיום שלהם קטן מהיום, כלומר נגמרו.

שאילתת עידכון:

```
UPDATE Credit_Cards
SET Expiration_Date = ADD_MONTHS(Expiration_Date, 12);
```

נעדכן את תאריך התפוגה של כרטיס הארשאי בעוד שנה.

```
UPDATE Credit_Cards
SET Expiration_Date = ADD_MONTHS(Expiration_Date, 12);

select * from Credit_Cards
WHERE End_Date < SYSDATE;
```

Select accounts Select accounts Select accounts Select accounts Delete credit\_cards Rollback Delete loans Rollback Update credit\_cards Select credit\_cards

(no result set)

כאן ניתן לראות שאחרי שהוספנו לכולם שנה, אין תוצאות, לפני זה ביקשנו רק את מי שקטן מהיום וכולם היו בתאריכים קטנים להיום אך קטנים ממנו, ברגע שהוספנו לכולם שנה, לא נותר את מי לבחור.

שאלת עידכון 2:

```
UPDATE Loans
SET Interest_Rate = Interest_Rate + 0.5;
```

Select accounts Select accounts Select accounts Select accounts Delete credit\_cards Rollback Delete loans Rollback Rollback Select credit\_cards Update loans

נעדכן את כל הריביות על הלוואות בעוד חצי אחוז. בכל זאת, הריבית עולה, אינפלציה בשמיים. בואו, זה לא מה שהיה פעם, 2 אחוז ריבית.

שאלת עם פרמטרים 1:

```
DECLARE
p_Balance NUMBER := 10000;
CURSOR c_Accounts IS
SELECT Account_ID, Customer_ID, Account_Type, Balance, Account_Opening_Date
FROM Accounts
WHERE Balance > p_Balance;
BEGIN
FOR rec IN c_Accounts LOOP
DBMS_OUTPUT.PUT_LINE('Account ID: ' || rec.Account_ID || ', Customer ID: ' || rec.Customer_ID ||
', Account Type: ' || rec.Account_Type || ', Balance: ' || rec.Balance ||
', Opening Date: ' || rec.Account_Opening_Date);
END LOOP;
END;
/
```

נחפש את המידע על החשבון שהיתרה שלו גדולה מהפרמטר :  
פלט:

```

Account ID: 27179, Customer ID: 371, Account Type: Savings, Balance: 775809, Opening Date: 20-MAY-24
Account ID: 45206, Customer ID: 368, Account Type: Savings, Balance: 184913, Opening Date: 07-MAY-24
Account ID: 31951, Customer ID: 363, Account Type: Savings, Balance: 81168, Opening Date: 15-MAY-24
Account ID: 21059, Customer ID: 357, Account Type: Checking, Balance: 387989, Opening Date: 15-MAY-24
Account ID: 46901, Customer ID: 344, Account Type: Checking, Balance: 907853, Opening Date: 20-MAY-24
Account ID: 73766, Customer ID: 337, Account Type: Savings, Balance: 579008, Opening Date: 06-MAY-24
Account ID: 12055, Customer ID: 329, Account Type: Savings, Balance: 879697, Opening Date: 18-MAY-24
Account ID: 43050, Customer ID: 318, Account Type: Savings, Balance: 75870, Opening Date: 06-MAY-24
Account ID: 94436, Customer ID: 310, Account Type: Checking, Balance: 258853, Opening Date: 12-MAY-24
Account ID: 51745, Customer ID: 300, Account Type: Savings, Balance: 397349, Opening Date: 20-MAY-24
Account ID: 57740, Customer ID: 289, Account Type: Checking, Balance: 905764, Opening Date: 15-MAY-24
Account ID: 44032, Customer ID: 279, Account Type: Savings, Balance: 818644, Opening Date: 16-MAY-24
Account ID: 24487, Customer ID: 268, Account Type: Checking, Balance: 712644, Opening Date: 18-MAY-24
Account ID: 21541, Customer ID: 257, Account Type: Savings, Balance: 696039, Opening Date: 07-MAY-24
Account ID: 80906, Customer ID: 247, Account Type: Savings, Balance: 64451, Opening Date: 05-MAY-24
Account ID: 45906, Customer ID: 237, Account Type: Checking, Balance: 879812, Opening Date: 15-MAY-24
Account ID: 13940, Customer ID: 228, Account Type: Checking, Balance: 752056, Opening Date: 18-MAY-24
Account ID: 81105, Customer ID: 218, Account Type: Checking, Balance: 573483, Opening Date: 19-MAY-24
Account ID: 43236, Customer ID: 211, Account Type: Checking, Balance: 679369, Opening Date: 18-MAY-24
Account ID: 13271, Customer ID: 202, Account Type: Checking, Balance: 918042, Opening Date: 07-MAY-24
Account ID: 48175, Customer ID: 196, Account Type: Checking, Balance: 713675, Opening Date: 15-MAY-24
Account ID: 36975, Customer ID: 190, Account Type: Savings, Balance: 692597, Opening Date: 06-MAY-24
Account ID: 6748, Customer ID: 184, Account Type: Checking, Balance: 674222, Opening Date: 19-MAY-24
Account ID: 13222, Customer ID: 179, Account Type: Savings, Balance: 633371, Opening Date: 12-MAY-24
Account ID: 17512, Customer ID: 176, Account Type: Savings, Balance: 919536, Opening Date: 20-MAY-24
Account ID: 66260, Customer ID: 171, Account Type: Savings, Balance: 663978, Opening Date: 06-MAY-24
Account ID: 72682, Customer ID: 168, Account Type: Checking, Balance: 235513, Opening Date: 20-MAY-24

```

000 & 638:1 123@XE AS SYSDBA [23:04:29] Done in 0.011 seconds

## שאלתא עם פרמטרים 2:

```

DECLARE
p_Start_Date DATE := TO_DATE('2022-01-01', 'YYYY-MM-DD');
p_End_Date DATE := TO_DATE('2026-12-31', 'YYYY-MM-DD');
CURSOR c_Loans IS
    SELECT Loan_ID, Loan_Amount, Interest_Rate, Start_Date, End_Date, Account_ID
    FROM Loans
    WHERE Start_Date BETWEEN p_Start_Date AND p_End_Date;
BEGIN
    FOR rec IN c_Loans LOOP
        DBMS_OUTPUT.PUT_LINE('Loan ID: ' || rec.Loan_ID || ', Loan Amount: ' ||
            rec.Loan_Amount || ', Interest Rate: ' || rec.Interest_Rate ||
            ', Start Date: ' || rec.Start_Date || ', End Date: ' || rec.End_Date ||
            ', Account ID: ' || rec.Account_ID);
    END LOOP;
END;
/

```

בודקת איזה הלוואות נלקחו בין תאריכים מסויימים:  
פלט:



SQL	Output	Statistics
Clear	Buffer size 3200000	<input checked="" type="checkbox"/> Enabled
Loan ID: 9002, Loan Amount: 86087, Interest Rate: 1, Start Date: 20-MAY-24, End Date: 06-MAY-24, Account ID: 2651		
Loan ID: 3903, Loan Amount: 63723, Interest Rate: 1, Start Date: 20-MAY-24, End Date: 01-MAY-24, Account ID: 47		
Loan ID: 5357, Loan Amount: 19896, Interest Rate: 1, Start Date: 07-MAY-24, End Date: 06-MAY-24, Account ID: 86		
Loan ID: 9705, Loan Amount: 42304, Interest Rate: 1, Start Date: 12-MAY-24, End Date: 19-MAY-24, Account ID: 585		
Loan ID: 5020, Loan Amount: 42617, Interest Rate: 1, Start Date: 07-MAY-24, End Date: 19-MAY-24, Account ID: 331		
Loan ID: 1902, Loan Amount: 73929, Interest Rate: 1, Start Date: 15-MAY-24, End Date: 20-MAY-24, Account ID: 190		
Loan ID: 7081, Loan Amount: 5156, Interest Rate: 1, Start Date: 15-MAY-24, End Date: 07-MAY-24, Account ID: 67		
Loan ID: 8405, Loan Amount: 1194, Interest Rate: 1, Start Date: 18-MAY-24, End Date: 15-MAY-24, Account ID: 1904		
Loan ID: 867, Loan Amount: 46430, Interest Rate: 1, Start Date: 18-MAY-24, End Date: 01-MAY-24, Account ID: 7617		
Loan ID: 9824, Loan Amount: 72275, Interest Rate: 1, Start Date: 01-MAY-24, End Date: 01-MAY-24, Account ID: 969		
Loan ID: 5056, Loan Amount: 47268, Interest Rate: 1, Start Date: 20-MAY-24, End Date: 07-MAY-24, Account ID: 945		
Loan ID: 8899, Loan Amount: 61030, Interest Rate: 1, Start Date: 07-MAY-24, End Date: 18-MAY-24, Account ID: 973		
Loan ID: 9586, Loan Amount: 73215, Interest Rate: 1, Start Date: 18-MAY-24, End Date: 01-MAY-24, Account ID: 218		
Loan ID: 2300, Loan Amount: 85881, Interest Rate: 1, Start Date: 20-MAY-24, End Date: 20-MAY-24, Account ID: 409		
Loan ID: 5013, Loan Amount: 58982, Interest Rate: 1, Start Date: 01-MAY-24, End Date: 07-MAY-24, Account ID: 964		
Loan ID: 9894, Loan Amount: 95572, Interest Rate: 1, Start Date: 06-MAY-24, End Date: 18-MAY-24, Account ID: 994		
Loan ID: 9101, Loan Amount: 78390, Interest Rate: 1, Start Date: 06-MAY-24, End Date: 20-MAY-24, Account ID: 633		
Loan ID: 2937, Loan Amount: 34829, Interest Rate: 1, Start Date: 19-MAY-24, End Date: 07-MAY-24, Account ID: 633		
Loan ID: 7070, Loan Amount: 53907, Interest Rate: 1, Start Date: 20-MAY-24, End Date: 19-MAY-24, Account ID: 9		
Loan ID: 6391, Loan Amount: 75089, Interest Rate: 1, Start Date: 16-MAY-24, End Date: 01-MAY-24, Account ID: 144		
Loan ID: 7550, Loan Amount: 47933, Interest Rate: 1, Start Date: 15-MAY-24, End Date: 01-MAY-24, Account ID: 812		
Loan ID: 2707, Loan Amount: 43971, Interest Rate: 1, Start Date: 07-MAY-24, End Date: 15-MAY-24, Account ID: 846		
Loan ID: 3498, Loan Amount: 40427, Interest Rate: 1, Start Date: 19-MAY-24, End Date: 19-MAY-24, Account ID: 487		
Loan ID: 7877, Loan Amount: 64869, Interest Rate: 1, Start Date: 18-MAY-24, End Date: 15-MAY-24, Account ID: 284		

שאלת טרנזקציה לפי סוג חשבון:

SQL	Output	Statistics
<pre> DECLARE p_Account_Type VARCHAR2(10) := 'Savings'; -- ערך לדוגמה לפרמטר CURSOR c_Transactions IS SELECT t.Transaction_ID, t.Amount, t.Account_ID, t.Transaction_Date FROM Transactions t JOIN Accounts a ON t.Account_ID = a.Account_ID WHERE a.Account_Type = p_Account_Type; BEGIN FOR rec IN c_Transactions LOOP DBMS_OUTPUT.PUT_LINE('Transaction ID: '    rec.Transaction_ID); DBMS_OUTPUT.PUT_LINE('Amount: '    rec.Amount); DBMS_OUTPUT.PUT_LINE('Account ID: '    rec.Account_ID); DBMS_OUTPUT.PUT_LINE('Transaction Date: '    rec.Transaction_Date); END LOOP; END; / </pre>		

פלט:

```
Transaction ID: 235
Amount: 2519
Account ID: 31429
Transaction Date: 01-MAY-24
Transaction ID: 64
Amount: 3717
Account ID: 55149
Transaction Date: 05-MAY-24
Transaction ID: 233
Amount: 3369
Account ID: 550
Transaction Date: 19-MAY-24
Transaction ID: 246
Amount: 4044
Account ID: 39364
Transaction Date: 16-MAY-24
Transaction ID: 122
Amount: 2549
Account ID: 243
Transaction Date: 06-MAY-24
Transaction ID: 802
Amount: 2834
Account ID: 94
Transaction Date: 12-MAY-24
|
```

1117:1 123@XF AS SYSDBA 17:21:49 Done in 0.018 seconds

#### 4. שאילתת כרטיסי אשראי עם תאריך תפוגה בטווח מסוים:

```
DECLARE
p_Start_Expiration DATE := TO_DATE('2022-01-01', 'YYYY-MM-DD');
p_End_Expiration DATE := TO_DATE('2024-12-31', 'YYYY-MM-DD');
CURSOR c_Credit_Cards IS
SELECT Card_ID, Card_Number, Card_Type, Expiration_Date, Credit_Limit, Account_ID
FROM Credit_Cards
WHERE Expiration_Date BETWEEN p_Start_Expiration AND p_End_Expiration;
BEGIN
FOR rec IN c_Credit_Cards LOOP
DBMS_OUTPUT.PUT_LINE('Card ID: ' || rec.Card_ID);
DBMS_OUTPUT.PUT_LINE('Card Number: ' || rec.Card_Number);
DBMS_OUTPUT.PUT_LINE('Card Type: ' || rec.Card_Type);
DBMS_OUTPUT.PUT_LINE('Expiration Date: ' || rec.Expiration_Date);
DBMS_OUTPUT.PUT_LINE('Credit Limit: ' || rec.Credit_Limit);
DBMS_OUTPUT.PUT_LINE('Account ID: ' || rec.Account_ID);
END LOOP;
END;
/
```

```
Clear Buffer size 5200000 [X] enabled
Account ID: 97
Card ID: 17525
Card Number: 8139362
Card Type: visa
Expiration Date: 07-MAY-24
Credit Limit: 6875
Account ID: 833
Card ID: 31661
Card Number: 1756831
Card Type: mastercard
Expiration Date: 07-MAY-24
Credit Limit: 6719
Account ID: 916
Card ID: 71043
Card Number: 1773869
Card Type: mastercard
Expiration Date: 18-MAY-24
Credit Limit: 6467
Account ID: 916
Card ID: 26892
Card Number: 2920466
Card Type: visa
Expiration Date: 06-MAY-24
Credit Limit: 6121
Account ID: 38851
|
```

נדגים את קובץ האילוצים:

```
SQL Output Statistics
ALTER TABLE Loans
MODIFY (Interest_Rate NOT NULL);

ALTER TABLE Credit_Card DECLARE
ADD CONSTRAINT chk_Credit_Limit CHECK (Credit_Limit > 0);

ALTER TABLE Deposits
MODIFY (Interest_Rate DEFAULT 5);

Alter loans Alter credit_cards Alter deposits
```

שמנו אילוץ שבכום הריבית בהלוואה יהיה לא null  
הגבלת אשראי גדולה מ0  
ברירת המחדל של ריבית על הפקדות היא 5

נראה דוגמא של הכנסה לטבלה בלי ערך ריבית 👍

SQL Output Statistics

```
INSERT INTO Deposits (DepositID, Deposit_Amount, Deposit_Date, Maturity_Date, Account_ID)
VALUES (51510, 7000, DATE '2024-05-23', DATE '2025-05-28', 10);

select * from deposits
where depositid = 51510
```

Insert deposits Select deposits

	DEPOSITID	DEPOSIT_AMOUNT	DEPOSIT_DATE	MATURITY_DATE	INTEREST_RATE	ACCOUNT_ID
1	51510	7000	23/05/2024	28/05/2025	5	10

באופן אוטומטי הערך הוא 5.

נראה דוגמא של הכנסת מגבלת אשראי קטנה מ:0:

main.sql X insertTables.sql selectAll.sql Queries createTables.sql Constraints

SQL Output Statistics

```
INSERT INTO Credit_Cards (Card_ID, Card_Number, Card_Type, Expiration_Date, Credit_Limit, Account_ID)
VALUES (3154201, 334455667788, 'Mastercard', DATE '2025-05-30', -7000, 10);

select * from Credit_Cards
where Card_ID = 3154201
```

Error

ORA-02290: check constraint (SYS.CHK\_CREDIT\_LIMIT) violated

OK Cancel Help

Insert credit\_cards Select credit\_cards

ובכן קיבלנו הודעת שגיאה כמצופה.

חזק וברוך עד כאן שלב 2.

שלב 3

דוד אוהב ציון ומרקוס צ'אמה.

בעקבות האינפלציה במשק, הוחלט בבנק על מתן הקלות ללקוחות פעילים בבנק. עבור לקוחות פעילים אלו הבנק מוכן לקצץ עד 5% מהריבית על ההלוואות. ובכל מקרה ריבית על הלוואה לא תפחת מ-2%.

קרטונים ללקוח פעיל:

- ביצע לפחות 5 העברות בשנתיים האחרונות.
- פדה לפחות 5 שיקים בשנה האחרונה.
- ביצע לפחות 5 הפקדות מאז פתיחת החשבון.

בהתאם לזה, הבנק מוכן לקצץ את אחוזי הריבית ע"פ הנוסחה הנ"ל:  
(סכום השקים שנפדו בשנה האחרונה + סכום ההפקדות בשנה האחרונה) \ מספר החודשים  
שהחשבון קיים) \* 0.01

ובכל מקרה סכום ההנחה לא יעלה על 5% - כלומר לקוח עם הלוואה בגובה ריבית 7% וזכאות להנחה בגובה 6% יזכה להנחה רק בגובה 5% ובהתאם לזה גובה הריבית על הלוואה שלו יעודכן ל-2%.

נבנה פונקציה שבודקת לאיזה לקוח מגיע הנחה ועוד פונקציה שמחזירה את גובה ההנחה. כמו כן נבנה פרוצדורה שמעדכנת את בסיס הנתונים. נבנה פונקציה Main שתבצע את כל התהליך.

פונקציה שבודקת האם ללקוח מגיע הנחה:

```

create or replace noneditionable function is_active5(p_account_id IN NUMBER) return boolean is
FunctionResult boolean;
v_transfers_count integer;
v_checks_count integer;
v_deposits_count integer;
begin
SELECT COUNT(*) INTO v_transfers_count
FROM Transactions t
WHERE account_id = p_account_id
AND t.transaction_date >= ADD_MONTHS(SYSDATE, -24);

SELECT COUNT(*) INTO v_checks_count
FROM Checks ch
WHERE ch.account_id = p_account_id
AND ch.clearing_date >= ADD_MONTHS(SYSDATE, -12);

SELECT COUNT(*) INTO v_deposits_count
FROM Deposits dp
WHERE dp.account_id = p_account_id;

IF v_transfers_count >= 5 AND v_checks_count >= 5 AND v_deposits_count >= 5 THEN
FunctionResult := TRUE;
ELSE
FunctionResult := FALSE;
END IF;

begin
SELECT COUNT(*) INTO v_transfers_count
FROM Transactions t
WHERE account_id = p_account_id
AND t.transaction_date >= ADD_MONTHS(SYSDATE, -24);

SELECT COUNT(*) INTO v_checks_count
FROM Checks ch
WHERE ch.account_id = p_account_id
AND ch.clearing_date >= ADD_MONTHS(SYSDATE, -12);

SELECT COUNT(*) INTO v_deposits_count
FROM Deposits dp
WHERE dp.account_id = p_account_id;

IF v_transfers_count >= 5 AND v_checks_count >= 5 AND v_deposits_count >= 5 THEN
FunctionResult := TRUE;
ELSE
FunctionResult := FALSE;
END IF;
return(FunctionResult);
end is_active5;

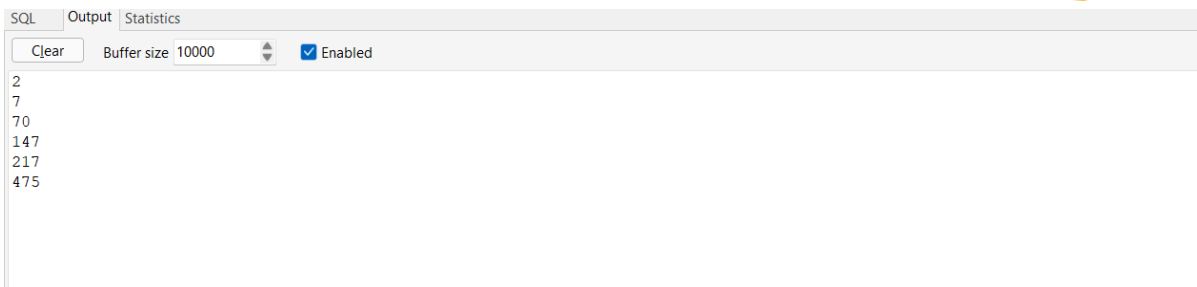
```

נראה את מספרי הלקוחות שעבורם הפונקציה תחזיר אמת:

נריץ פונקציה שתבדוק :

```
---main
DECLARE
    CURSOR customer_cursor IS
        SELECT account_ID FROM Accounts;
BEGIN
    FOR customer_record IN customer_cursor LOOP
        if is_active5(customer_record.account_id) then
            dbms_output.put_line(customer_record.account_id);
        end if;
    END LOOP;
END;
```

והתוצאה 👍



SQL Output Statistics

Clear Buffer size 10000 Enabled

2  
7  
70  
147  
217  
475

כלומר מצאנו 6 לקוחות שמגיע להם הנחה בגובה הריבית בהלוואות.  
פונקציה שמחזירה את גובה ההנחה עבור כל לקוח שנמצא זכאי:

```

1 CREATE OR REPLACE FUNCTION get_discount(p_account_id IN NUMBER)
2 RETURN NUMBER IS
3     v_checks_sum NUMBER;
4     v_deposits_sum NUMBER;
5     v_account_open_date DATE;
6     v_months_open NUMBER;
7     func_result NUMBER;
8 BEGIN
9
10    SELECT NVL(SUM(ch.amount),0) INTO v_checks_sum
11    FROM Checks ch
12    WHERE ch.account_id = p_account_id
13          AND ch.clearing_date >= ADD_MONTHS(SYSDATE, -12);
14
15
16    SELECT NVL(SUM(dep.Deposit_Amount),0) INTO v_deposits_sum
17    FROM Deposits dep
18    WHERE dep.account_id = p_account_id
19          AND dep.deposit_date >= ADD_MONTHS(SYSDATE, -12);
20
21
22    select a.account_opening_date
23    INTO v_account_open_date
24    from accounts a
25
26
27    FROM Deposits dep
28    WHERE dep.account_id = p_account_id
29          AND dep.deposit_date >= ADD_MONTHS(SYSDATE, -12);
30
31
32    select a.account_opening_date
33    INTO v_account_open_date
34    from accounts a
35    where a.account_id = p_account_id;
36
37
38    v_months_open := MONTHS_BETWEEN(SYSDATE, v_account_open_date);
39
40
41    func_result := ((v_deposits_sum + v_checks_sum) / v_months_open) *0.01;
42
43
44    IF func_result > 5 THEN
45        func_result := 5;
46    END IF;
47
48    RETURN ROUND (NVL(func_result,0),2);
49
50 END get_discount;

```

נבצע הריצת בדיקה: הדפס עבור כל מי שזכאי את גובה ההנחה לה הוא זכאי:

```

DECLARE
    CURSOR customer_cursor IS
        SELECT account_ID FROM Accounts;
BEGIN
    FOR customer_record IN customer_cursor LOOP
        if is_active5(customer_record.account_id) then
            dbms_output.put_line(get_discount(customer_record.account_id));
        end if;
    END LOOP;
END;

```



```
5
5
3.37
1.76
2.07
2.45
```

כלומר מצאנו שכל הלקוחות הזכאים להנחה מסויימת בריבית על ההלוואה, ושני לקוחות זכאים לגובהה המקסימלי של ההנחה.  
עכשיו רק נותר ליצור פרוצדורה שבודקת האם יש ללקוח הלוואות, מכל אותן הלאות היא תבחר את ההלוואה עם הריבית הגבוהה ביותר, ותתן ללקוח את ההנחה ( תשנה את הנתונים )

```
CREATE OR REPLACE PROCEDURE apply_discount(p_account_id IN NUMBER,p_discount IN NUMBER
IS
    v_loan_id NUMBER;
    v_max_interest NUMBER;
    v_new_interest NUMBER;
BEGIN
    BEGIN
        SELECT loan_id, interest_rate
        INTO v_loan_id, v_max_interest
        FROM (
            SELECT loan_id, interest_rate
            FROM Loans
            WHERE account_id = p_account_id
            ORDER BY interest_rate DESC
        )
        WHERE ROWNUM = 1;

        v_new_interest := v_max_interest - p_discount;

        IF v_new_interest < 2 THEN
            v_new_interest := 2;

            v_new_interest := v_max_interest - p_discount;

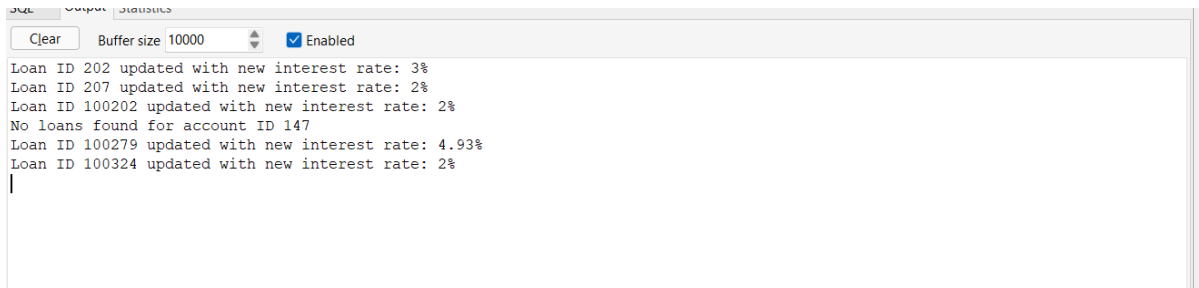
            IF v_new_interest < 2 THEN
                v_new_interest := 2;
            END IF;

            UPDATE Loans
            SET interest_rate = v_new_interest
            WHERE loan_id = v_loan_id;

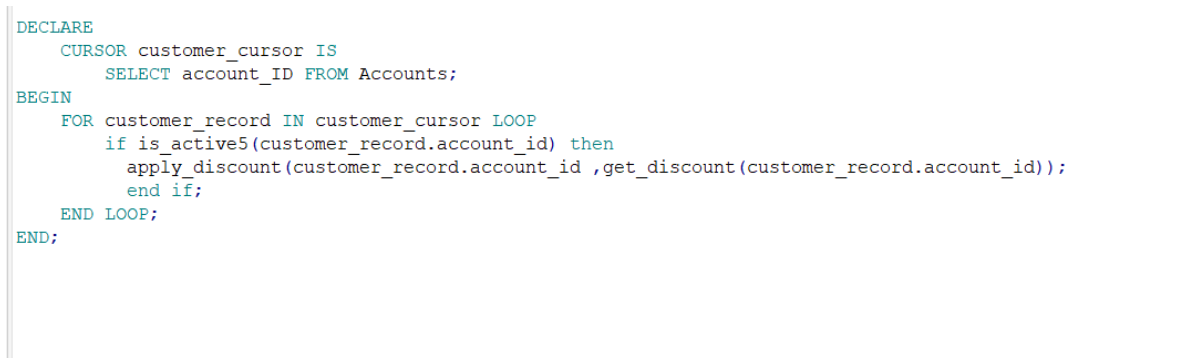
            DBMS_OUTPUT.PUT_LINE('Loan ID ' || v_loan_id || ' updated with new interest rate: ' || v_new_

        EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('No loans found for account ID ' || p_account_id);
    END;
END;
```

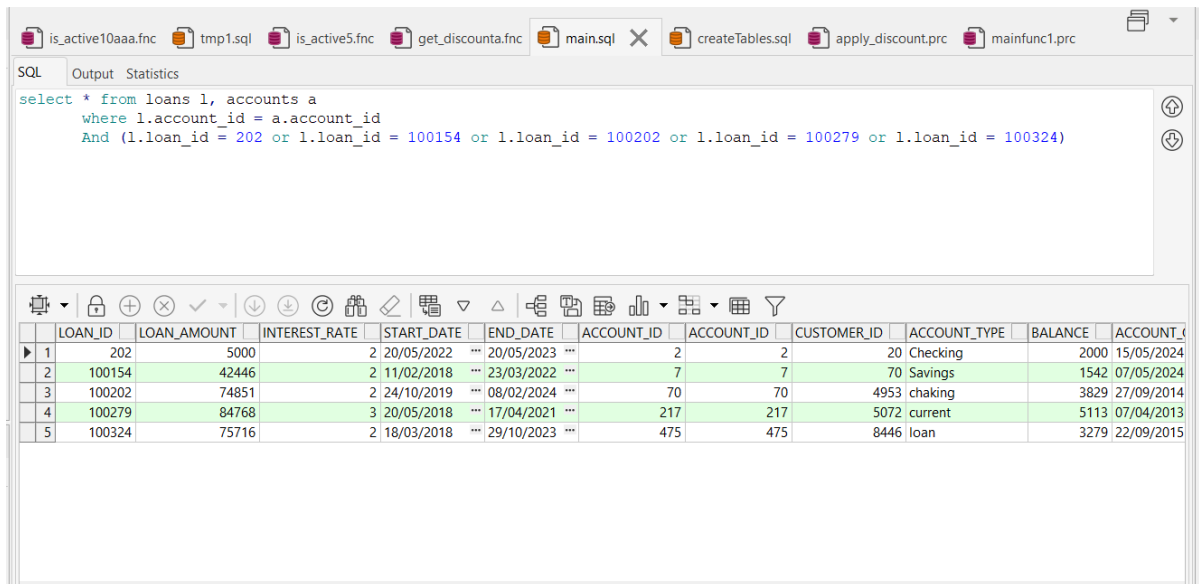
נריץ בדיקה:



נגדיר פונקציות מיון שתבדוק עבור כל לקוח האם מגיעה לו הנחה, אם כן, מה גובה ההנחה ועידכון DB!



נראה שאכן הטבלאות מתעדכנות:



וואוואלה:  
תוכנית 2:

בעקבות המלחמה הבנק נכנס לתקופה קשה ועומד לפני פשיטת רגל, בשביל להציל את המצב, ולמנוע מכל לקוחות הבנק לאבד את כספם במקרה של פשיטת רגל, הוחלט בבנק על צעדים משמעותיים שיפגעו במספר לקוחות אך יצילו את הבנק מקריסה! תוכנית הפעולה של הבנק היא לגבות את כל ההלוואות שתאריך הפידיון שלהם עוד לא הגיע, בתנאי שללקוח יש מספיק כסף נזיל בחשבון. במקרה שבו ללקוח יש מספר הלוואות, הבנק יפדה את ההלוואה הכי קטנה (סכום ההלוואה הנמוך ביותר). לדוגמא מנחם הנחומי לקח הלוואה של 10 אש"ח וצריך להחזיר אותה רק בעוד שנה, אם למנחם יש יותר מ10 אש"ח בחשבון הבנק באופן אוטומטי יקח את הכסף, ויידע על כך את מנחם. יש לדבר היתכנות משפטית, וזכויות הפרט אינם נכללים בשיקול זה. אלה שיקול הכללי. ניצור פונקציה שבודקת האם ללקוח יש הלוואה וקיים מספיק כסף בחשבון בשביל לכסות אותה, ופרוצדורה שתעדכן את הבסיס נתונים בכסף שנלקח. בנוסף נכתוב פונקציה מיין שתתפעל את הפונקציות שיצרנו.

פונקציה בוליאנית שמחזירה אמת אם הלקוח יכול לכסות הלוואה באופן מיידי:

```
CREATE OR REPLACE FUNCTION can_cover(p_account_id IN NUMBER)
RETURN BOOLEAN IS
    v_loan_amount NUMBER;
    v_account_balance NUMBER;
BEGIN

    SELECT MIN(Loan_Amount)
    INTO v_loan_amount
    FROM Loans
    WHERE account_id = p_account_id
    AND End_Date > SYSDATE;

    SELECT Balance
    INTO v_account_balance
    FROM Accounts
    WHERE account_id = p_account_id;

    IF v_account_balance >= v_loan_amount THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;

    SELECT Balance
    INTO v_account_balance
    FROM Accounts
    WHERE account_id = p_account_id;

    IF v_account_balance >= v_loan_amount THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN FALSE;
END can_cover;
```

נריץ בדיקה:

```

DECLARE
    CURSOR customer_cursor IS
        SELECT account_ID FROM Accounts;
BEGIN
    FOR customer_record IN customer_cursor LOOP
        if can_cover(customer_record.account_id) then
            dbms_output.put_line(customer_record.account_id);
        end if;
    END LOOP;
END;

```

```

95
104
238
336

```

ב"ה ניתן לראות שרק ארבעה לקוחות נפלו ברשת של הבנק ויצטרכו לכסות את ההלוואה באופן מידוי.  
ניצור פרוצדורה שמעדכנת את הנ"ל בבסיס נתונים.

```

1 create or replace procedure cover_loan_and_update(p_account_id in Number) is
2     v_loan_amount NUMBER;
3     v_loan_id NUMBER;
4
5 begin
6
7     update accounts a
8     set a.balance = a.balance - ( SELECT MIN(Loan_Amount)
9     FROM Loans
10    WHERE account_id = p_account_id
11        AND End_Date > SYSDATE)
12    where a.account_id = p_account_id;
13
14
15    --find loan_id of the minimal loan
16    SELECT loan_id, Loan_Amount
17    INTO v_loan_id, v_loan_amount
18    FROM Loans
19    WHERE account_id = p_account_id
20        AND End_Date > SYSDATE
21    ORDER BY Loan_Amount ASC
22    FETCH FIRST 1 ROWS ONLY;
23
24    update loans l
25    set loan_amount = 0

```

```

23
24    update loans l
25    set loan_amount = 0,
26    l.end_date = SYSDATE
27    where l.account_id = p_account_id
28    And l.loan_id = v_loan_id;
29
30    commit;
31    DBMS_OUTPUT.PUT_LINE('Loan covered and database updated successfully. ');
32    EXCEPTION
33    WHEN NO_DATA_FOUND THEN
34        DBMS_OUTPUT.PUT_LINE('No eligible loan found or insufficient balance. ');
35
36 end cover_loan_and_update;

```

בעצם אנו מעדכנים את היתרת חשבון ע"י הפחתה של סכום ההלוואה ע"י שאילתא מקוננת, ואח"כ מוצאים את מספר הלוואה בכדי לדעת איזה אחת לאפס ומאפסים את ההלוואה ומעדכנים את התאריך פדיון.

```
DECLARE
    CURSOR customer_cursor IS
        SELECT account_ID FROM Accounts;
BEGIN
    FOR customer_record IN customer_cursor LOOP
        if can_cover(customer_record.account_id) then
            cover_loan_and_update(customer_record.account_id);
        end if;
    END LOOP;
END;
```

```
Loan covered and database updated successfully.
Loan covered and database updated successfully.
Loan covered and database updated successfully.
Loan covered and database updated successfully.
```

```
SQL Output Statistics
select * from loans l
where (l.account id = 95 or l.account id = 104 or l.account id = 336 or l.account id = 238)
```


ניתן לראות שהתארכי סיום עודכנו להיום, הסכום עודכן ל0 חוץ ממספר חשבון 336 שהיו לו שתי הלוואות ורק ההלוואה הנמוכה כוסתה. ניתן לראות בוודאות שהנתונים שונים, כמו כן הנה לפני ואחרי של מצב החשבון, ניתן לראות שהסכום הנזיל השתנה בהתאם לסכום ההלוואה:  
לפני:

SQL

Output

Statistics

```
select * from accounts a
  where (a.account_id = 95 or a.account_id = 104 or a.account_id = 336 or a.account_id = 238)
```




	ACCOUNT_ID	CUSTOMER_ID	ACCOUNT_TYPE	BALANCE	ACCOUNT_OPENING_DATE	
▶ 1	95	9721	loan	8222	12/04/2014	...
2	104	5693	saving	6802	15/07/2015	...
3	238	9686	saving	5479	17/03/2017	...
4	336	4644	current	4294967210	13/03/2014	...

אחרי:

SQL

Output Statistics

```
select * from accounts l
  where (l.account_id = 95 or l.account_id = 104 or l.account_id = 336 or l.account_id = 238)
```



	ACCOUNT_ID	CUSTOMER_ID	ACCOUNT_TYPE	BALANCE	ACCOUNT_OPENING_DATE	
▶	1	95	9721 loan	7642	12/04/2014	...
	2	104	5693 saving	2723	15/07/2015	...
	3	238	9686 saving	4680	17/03/2017	...
	4	336	4644 current	4294961219	13/03/2014	...

בשעה טובה, בזכות לקוחות פרטיים אלו הבנק ניצל מפשיטת רגל!

עד כאן שלב 3  
דוד אוהב ציון ומרקוס צ'אמה.

## שלב 4 דוד אוהב ומרקוס צ'אמה

נבצע אינטגרציה בין חברנו ממחלקת לקוחות.

דבר ראשון ניצור גיבוי נוסף לנתונים שלנו לפני שמתחילים את הבאלגן.  
בגלל שיש לנו שמות חופפים של טבלאות, נשנה את השמות של הטבלאות שלנו לסיומת 1, לסימון צוות 1.

```

---stage 4
--rename our table

rename accounts to accounts1;
rename checks to checks1;
rename credit_cards to credit_cards1;
rename deposits to deposits1;
rename transactions to transactions1;
rename loans to loans1;

```

Rename Rename **Rename** Rename Rename

(no result set)

נריץ בחירה של הכל נראה שהכל עבד:

```

select * from accounts1;
select * from checks1;
select * from credit_cards1;
select * from deposits1;
select * from transactions1;
select * from loans1;

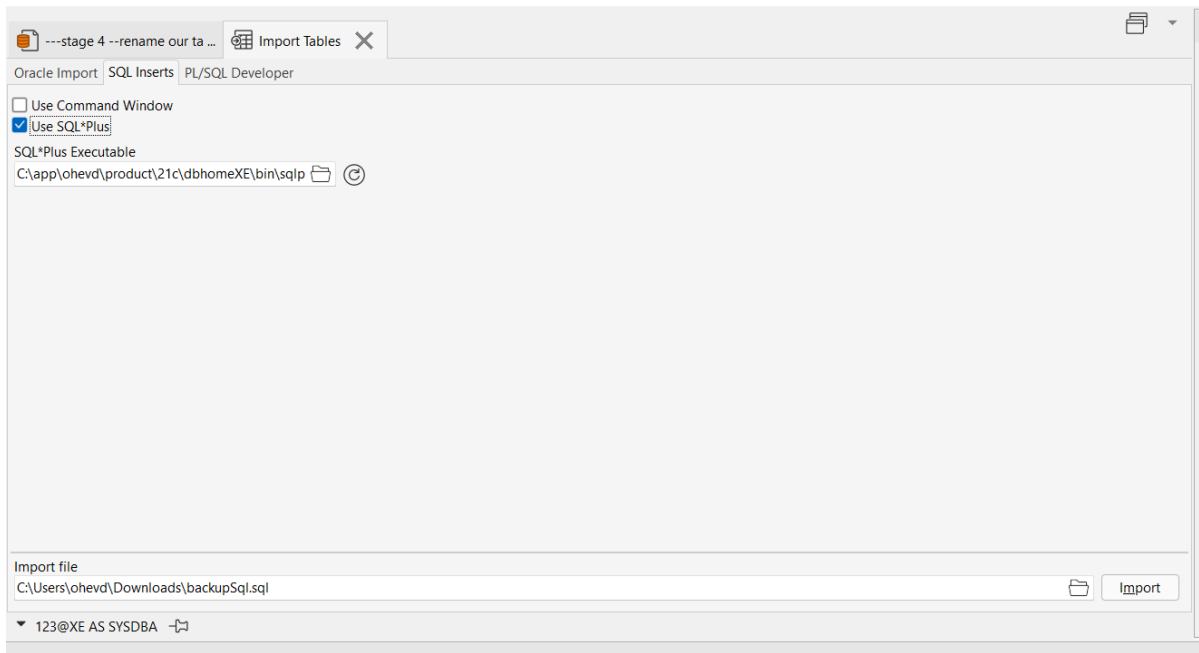
```

Select accounts1 Select checks1 Select credit\_cards1 Select deposits1 Select transactions1 Select loans1

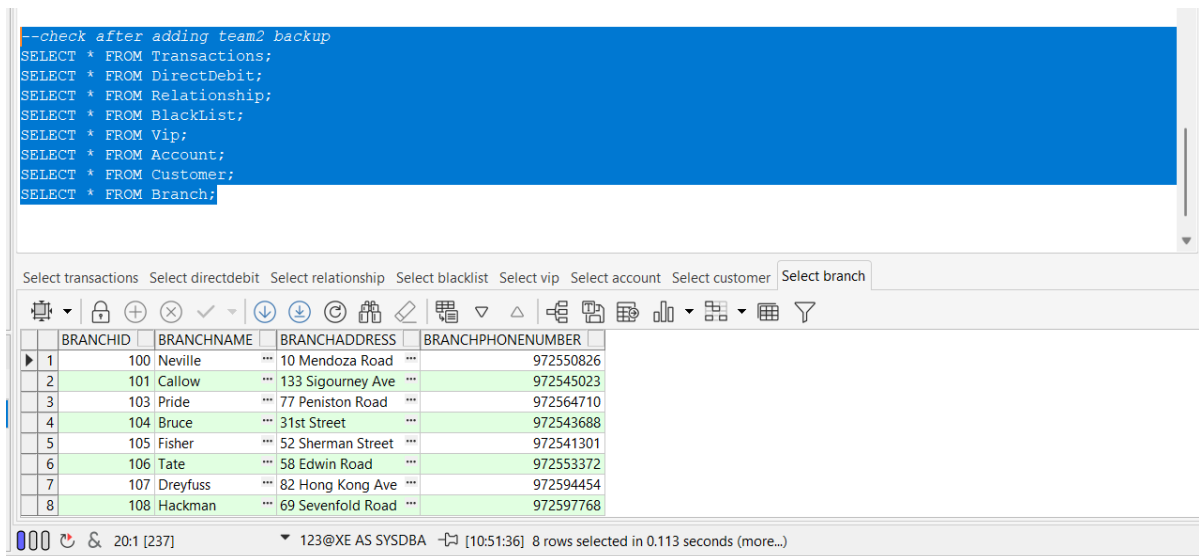
	ACCOUNT_ID	CUSTOMER_ID	ACCOUNT_TYPE	BALANCE	ACCOUNT_OPENING_DATE	
1	1	10	Savings	4000	20/05/2024	...
2	2	20	Checking	2000	15/05/2024	...
3	3	30	Checking	2555	16/05/2024	...
4	4	40	Checking	2040	19/05/2024	...
5	5	50	Savings	2900	18/05/2024	...
6	6	60	Checking	9999	12/05/2024	...
7	7	70	Savings	1542	07/05/2024	...
8	8	80	Checking	8521	06/05/2024	...

1 of 8 123@XE AS SYSDBA 122:19:27 8 rows selected in 0.188 seconds (more...)

ניקח את הגיבוי של צוות 2 ונעלה אותו למערכת שלנו:



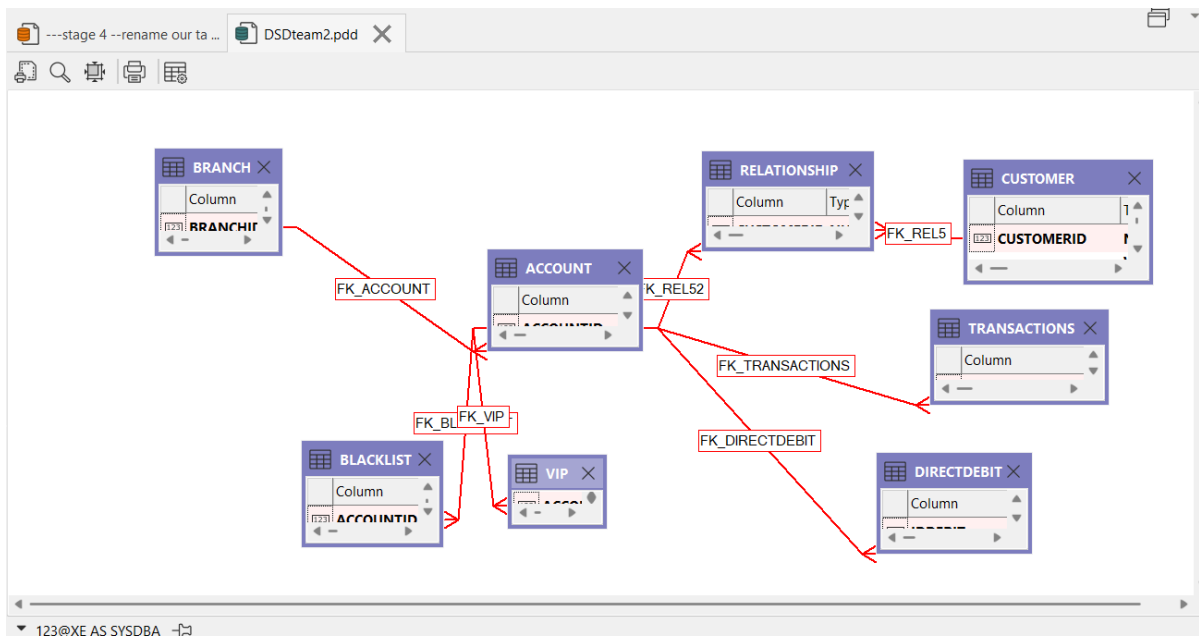
נריץ בחירה על טבלאות צוות 2 לראות שהכל עבר.



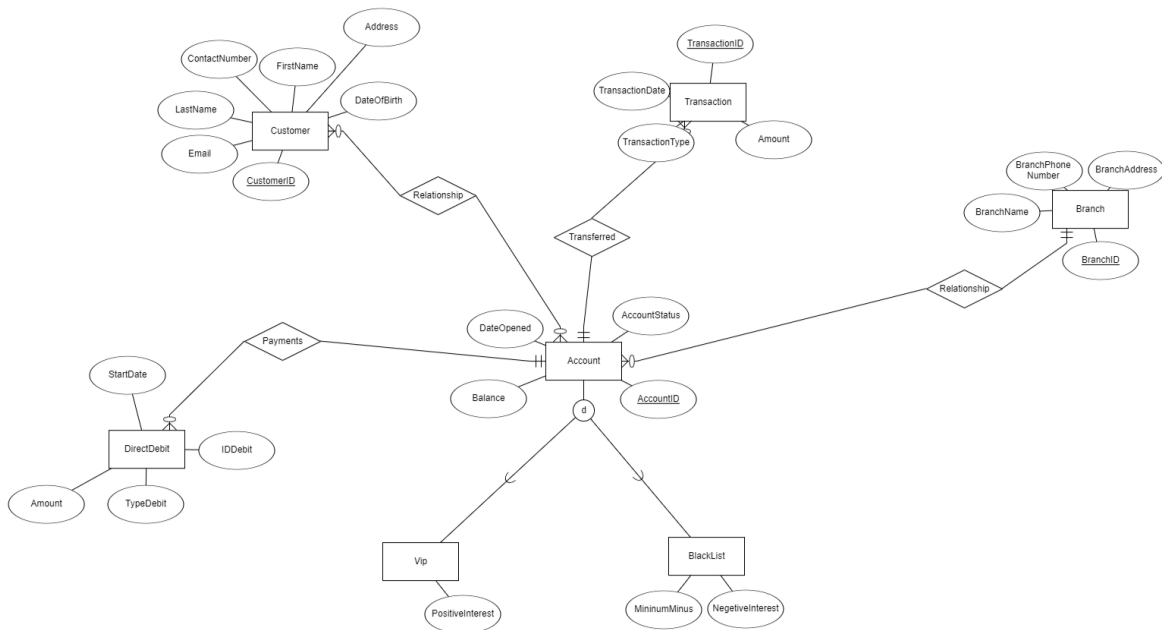
יפה הכל עבד כשורה!

נייצר תרשים DSD של צוות 2:



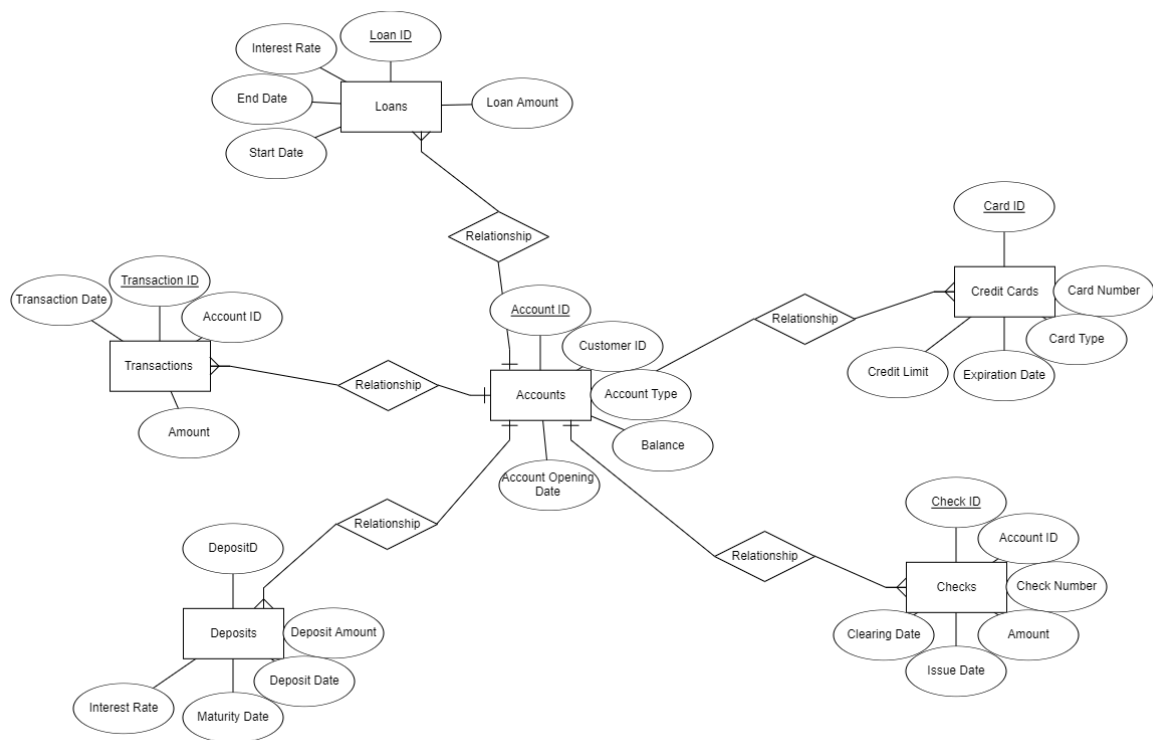


נשחזר את ה ERD ע"י ניתוח והבנה מעמיקה של המפתחות הזרים בכדי לגלות קשרים:



עד כאן מבוא לרברסינג .

את ה ERD שלנו כבר יש לנו להלן:



עכשיו נצטרך לחשוב, איך אנחנו מחברים ביניהם לכדי integrationERD.

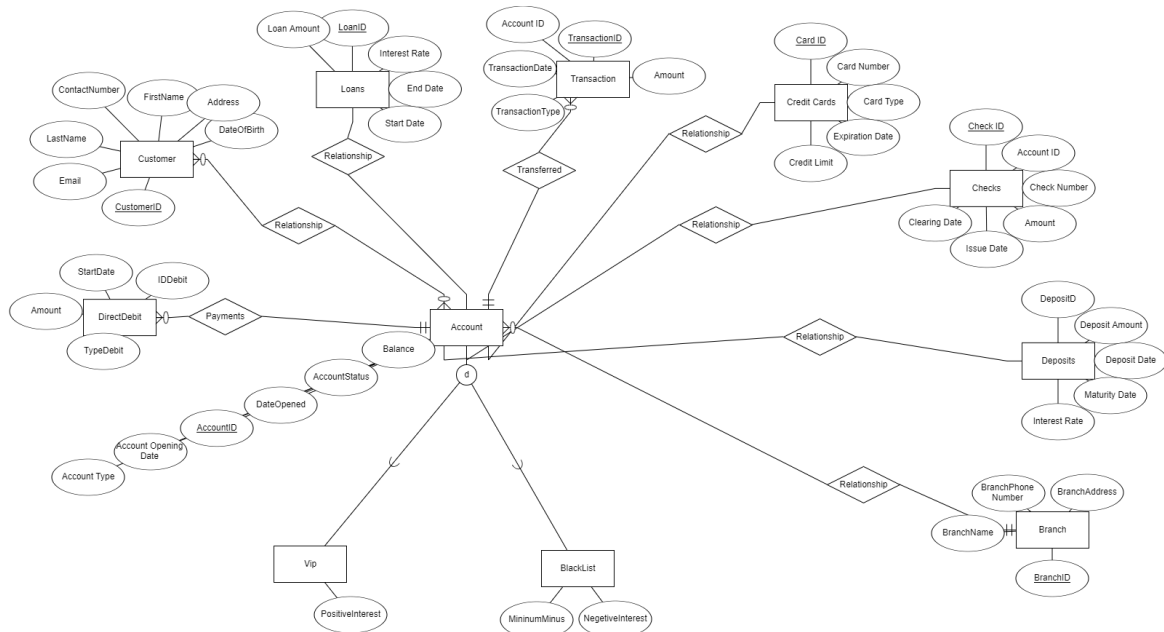
הדבר כרוך בתחבולות ועיקושים אך נציג את התוצאות להלן ולאחר מכן נסביר:

עבור סנכרון עם BRANCH נצטרך להוסיף את השדה brachId לטבלה accounts.

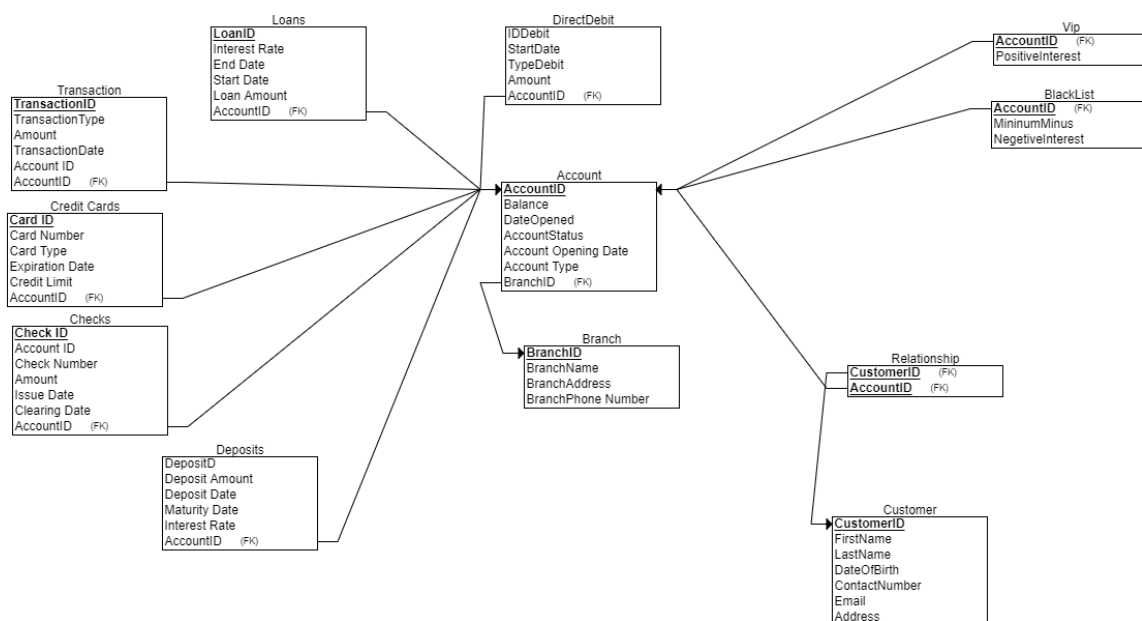
הטבלה transaction תואמת לשלנו נבצע איחוד - ונוסיף את העמודה transactionType לנתונים שלנו.

בaccounts נבצע איחוד שידרוש מאיתנו מעט התאמות, רוב המידע זהה אך בשדות שונים וכדומה נטפל.

ולהלן התרשים ERD המאוחד:



נתחיל להריץ פקודות איחוד ואינטגרציה, ושהשם הטוב יהיה בעזרנו. אנו שואפים למשהו כזה:



הבעיה הכי גדולה שלנו, שצוות 2 הגדיר את היחס לקוח חשבון כרבים לרבים, ואנו הגדרנו את היחס יחיד לרבים, כלומר שורש המחלוקת, האם ישנם מספר אנשים שיכולים לנהל חשבון בנק במשותף? אכן סהדי דלכל חשבון יש בעל אחד ויחיד, צוות 2 בא לטעון שיכול להיות מצב של חשבון זוגי, לבני זוג, או שמא לשותפים לדירה וכו' (רח"ל).

2 אופציות בפנינו, או להוריד את העמודה customerID מהטבלה accounts וללכת בשיטת צוות 2, או לוותר על הטבלה של הקשר ובכך בעצם לקבע את העובדה שלכל חשבון יכול להיות רק בעל אחד.

משום שאנו מאמינים בחיי זוגיות, ובמוסד הנישואין, ומעל הכל בשלום ואחדות בין חלקי העם והארץ, לא נותר לנו אלא לבחור בגישת צוות מס' 2. בהרכנת ראש עמוקה נצטרך להוריד את העמודה customerID מהטבלה accounts ולקוות זה לא יעשה בעיות, שהרי אין לנו ברירה שני הבדלים אלו הם משמעותיים, ועלינו לבחור בגישה אחת, כל בחירה תפגע בשלמות הטבלאות, וכנ"ל וכו'.

נתחיל לאט לאט, הטבלה "העברות" דומה מאוד לשלנו, חסר לנו עמודה אחת, פשוט נוסיף לטבלה של צוות 2 את כל מה שחסר אצלם

```
--integration command
--transaction
ALTER TABLE transactions1 ADD transactiontype VARCHAR(20);

UPDATE transactions1 t1
SET t1.transactiontype = (
  SELECT t.transactiontype
  FROM transactions t
  ORDER BY DBMS_RANDOM.VALUE
  FETCH FIRST 1 ROWS ONLY
);

INSERT INTO transactions (transactionid, transactiontype, amount, transactiondate, accountid)
SELECT t1.transaction_id, t1.transactiontype, t1.amount, t1.transaction_date, t1.account_id
FROM transactions1 t1
WHERE t1.transaction_id not in (select t2.transactionid from Transactions t2);
```

32:1 [235] 123@XE AS SYSDBA [11:23:47] 2610 rows updated in 0.223 seconds

דבר ראשון הוספנו לטבלת ההעברות שלנו את העמודה החסרה לעומת הטבלה של צוות 2, דבר שני מילאנו אותה בערכים רנדומליים ע"פ הערכים שנמצאים בטבלה שלהם.

נבצע איחוד בין שני הטבלאות:

ננסה להכניס את מי שלא נמצא בטבלה אחת לטבלה 2 ונקבל שגיאה צפויה מראש(הסבר למטה):

```
INSERT INTO transactions (transactionid, transactiontype, amount, transactiondate, accountid)
SELECT t1.transaction_id, t1.transactiontype, t1.amount, t1.transaction_date, t1.account_id
FROM transactions1 t1
WHERE t1.transaction_id not in (select t2.transactionid from Transactions t2);
```

41:1 [289] 123@XE AS SYSDBA ORA-02291: integrity constraint (SYS.FK\_TRANSACTIONS) violated - parent key not found

זאת משום שיש מספרי חשבון שלא נמצאים בטבלת האבא והם מפתחות זרים שמצביעים עליו. לכן צריך קודם לטפל בטבלת חשבונות, אח"כ נמשיך.

```
integration Commands.sql X
SQL Output Statistics

--account integratin
ALTER TABLE accounts1 ADD branchid NUMBER(38);
ALTER TABLE accounts1 ADD AccountStatus VARCHAR2(20);
ALTER TABLE account ADD account_type varchar2(10);

UPDATE account a
SET a.account_type = (
    SELECT a1.account_type
    FROM accounts1 a1
    ORDER BY DBMS_RANDOM.VALUE
    FETCH FIRST 1 ROWS ONLY
);

UPDATE accounts1 a1
SET a1.branchid = (
    SELECT a.branchid
    FROM account a
    ORDER BY DBMS_RANDOM.VALUE
    FETCH FIRST 1 ROWS ONLY
);
```

נסביר מה עשינו, דבר ראשון הוספנו לכל העמודות החסרות את מה שחסר להן, לאחר מכן עידכנו את העמודות החסרות בערכים.

```
UPDATE accounts1 a1
SET a1.AccountStatus = (
    SELECT a.AccountStatus
    FROM account a
    ORDER BY DBMS_RANDOM.VALUE
    FETCH FIRST 1 ROWS ONLY
);

INSERT INTO account (accountid, balance, dateopened, accountstatus, branchid, account_type)
SELECT a1.account_id, a1.balance, a1.account_opening_date, a1.accountstatus, a1.branchid, a1.account_type
FROM accounts1 a1
WHERE NOT EXISTS (
    SELECT 1
    FROM account a
    WHERE a.accountid = a1.account_id
);
```

77:38 123@XE AS SYSDBA [12:01:49] 509 rows inserted in 0.011 seconds

הבעיה מתחילה בכך שמספרי החשבון לא מסונכרנים, ולכן בטבלה של ההעברות לא מוגדרים המפתחות הזרים.

הפתרון יהיה לבצע את סנכרון טבלאות החשבונות בתחילה, ורק לאחר הסנכרון לאחד את טבלאות ההעברות.

הכנסנו לתוך הטבלה account של צוות 2 את כל הרשומות מהטבלה שלנו, שלא נמצאים בטבלה שלהם.

ניתן לראות שהוספנו 509 רשומות כלומר רוב הרשומות היו חופפות מבחינת מספר חשבון

נריץ בדיקה לראות את התוצאה:

```
select * from account;
select count(*) from account
```

```
--end of transaction integration
--INSERT INTO transactions (transactionid, transactiontype, amount, transactiondate, accountid)
--SELECT t1.transaction_id, t1.transactiontype, t1.amount, t1.transaction_date, t1.account_id
```

Select account Select account

	ACCOUNTID	BALANCE	DATEOPENED	ACCOUNTSTATUS	BRANCHID	ACCOUNT_TYPE
1	100000	18139	17/11/2015 06:22:32	soldier	190	chaking
2	100001	84707	28/07/2013 02:58:28	elderly	307	chaking
3	100002	-15110	06/03/2015 15:07:32	soldier	315	chaking
4	100003	49741	10/11/2012 23:39:01	student	236	chaking
5	100004	49492	01/10/2002 11:14:12	soldier	497	chaking
6	100005	30700	01/10/2003 20:20:21	regular	475	chaking
7	100006	58521	15/01/2015 04:10:32	student	326	chaking
8	100007	51206	23/12/2001 15:50:31	foreign	245	chaking

1 of 8 123@XE AS SYSDBA [12:06:19] 8 rows selected in 0.034 seconds (more...)

הנתונים טובים, כמו כן יצאו לנו 2500 רשומות. נחמד סה"כ

נחזור לסיים את מה שהתחלנו בטבלת העברות.

```
--end of transaction integration
INSERT INTO transactions (transactionid, transactiontype, amount, transactiondate, accountid)
SELECT t1.transaction_id, t1.transactiontype, t1.amount, t1.transaction_date, t1.account_id
FROM transactions1 t1
WHERE t1.transaction_id not in (select t2.transactionid from Transactions t2);
```

85:1 [287] 123@XE AS SYSDBA [12:14:16] 2610 rows inserted in 0.027 seconds

עבד חלק כמו טוסיק של תינוק.

וכהרגלנו בדיקה:

```
select * from transactions;
```

	TRANSACTIONID	TRANSACTIONTYPE	AMOUNT	TRANSACTIONDATE	ACCOUNTID
1	10000000	Transfer	3519	14/02/2015 22:55:08	101213
2	10000001	Bill payment	2221	30/07/2018 04:11:36	100088
3	10000002	Check	5920	22/01/2006 12:34:17	101658
4	10000003	Transactions	9896	14/03/2018 10:38:46	100946
5	10000004	Check	8221	24/09/2014 22:36:19	101979
6	10000005	ATM	5846	30/05/2012 23:22:59	100466
7	10000006	Bill payment	4567	22/04/2017 04:53:16	101920
8	10000007	Check	8893	13/02/2019 08:11:35	100150
9	10000008	Check	2996	08/06/2014 04:20:23	100985

91:1 [29] 123@XE AS SYSDBA [12:18:06] 9 rows selected in 0.027 seconds (more...)

לסיכום : יש לנו כרגע את טבלת חשבונות והעברות ביד (account, transaction).

בטבלאות הלואות אין לנו התנגשות, אך נצטרך לשנות את המצביע למפתח הזר

ראשית נבצע את המעבר של המפתח הזר - כרגע הוא מצביע לaccount שלנו ונרציה שיצביע לטבלת חשבונות של צוות 2

```
--loans integration
ALTER TABLE loans1 DROP CONSTRAINT SYS_C008666;

ALTER TABLE loans1
ADD CONSTRAINT SYS_C008666
FOREIGN KEY (account_id)
REFERENCES account (accountid);
```

הורדנו את המפתח הזר הישן שהצביע על הטבלה accounts1 ועכשיו הוא יצביע על הטבלה account

מעולה סיימנו עם הטבלה loans1

נעשה בדיוק אותו דבר עם credit\_cards1

```
---credit_card integration
SELECT constraint_name
FROM user_constraints
WHERE table_name = 'CREDIT_CARDS1' AND constraint_type = 'R';

ALTER TABLE credit_cards1 DROP CONSTRAINT SYS_C008673;

ALTER TABLE credit_cards1
ADD CONSTRAINT SYS_C008673
FOREIGN KEY (account_id)
REFERENCES account (accountid);
```

נבצע חיפוש של שם הconstraint מתוך user\_constraint איפה שהטבלה שווה לכרטיסי אשראי, ובנוסף אילוץ מסוג referential constraints

נבטל אותו וניצור חדש לטבלה החדשה.

סיימנו עם credit\_cards1

כנל לצקים והפקדות:

```
--check integration
SELECT constraint_name
FROM user_constraints
WHERE table_name = 'CHECKS1' AND constraint_type = 'R';

ALTER TABLE CHECKS1 DROP CONSTRAINT SYS_C008681;

ALTER TABLE CHECKS1
ADD CONSTRAINT SYS_C008681
FOREIGN KEY (account_id)
REFERENCES account (accountid);
```

Alter checks1 Alter checks1

(no result set)

128:20 123@XE AS SYSDBA [13:10:45] Done in 0.016 seconds

```
--deposits integration
SELECT constraint_name
FROM user_constraints
WHERE table_name = 'DEPOSITS1' AND constraint_type = 'R';

ALTER TABLE DEPOSITS1 DROP CONSTRAINT SYS_C008689;

ALTER TABLE DEPOSITS1
ADD CONSTRAINT SYS_C008689
FOREIGN KEY (account_id)
REFERENCES account (accountid);
```

Alter deposits1 Alter deposits1

(no result set)

145:1 123@XE AS SYSDBA [13:11:56] Done in 0.016 seconds

סיימנו עם **DEPOSITS1** וגם **CHECKS1**

נבצע איחוד לdirect debit בגלל שהוא בא מצוות 2 אין צורך להוריד מפתח זר וכו.

למעשה כל מה שהגיע מצוות 2 תקין ואכמ"ל, בגלל שחיברנו את הנתונים שלנו לנתונים של צוות 2, ההינו צריכים לבצע שינויים אצלנו. הנתונים שלהם תקינים.

כל מה שנותר לעשות הוא להחזיר את השמות של הטבלאות שלנו ששינו לשמות המקוריים:



```
--renaming to original names
rename checks1 to checks;
rename loans1 to loans;
rename credit_cards1 to credit_cards;
rename deposits1 to deposits;

--delete the unneccery tabels
drop table transactions1 ;
drop table accounts1 ;
```

154:27 123@XE AS SYSDBA [13:29:08] Done in 0.087 seconds

נסביר מה עשינו, כל הטבלאות ששינו את שמן צריכות לחזור למקור, הטבלה העברות התאחדה עם העברות שהגיע מצוות 2 ולכן אין צורך בה ומוחקים אותה, כנל על הטבלה חשבונות.

נראה שסיימנו את שלב האינטגרציה.

## טבלאות ועמודות במסד הנתונים החדש - שנראה בערך ככה:

### 1. Branch

טבלת הסניפים, מייצגת את הסניפים של הבנק בהם הלקוחות יכולים לגשת לשירותי בנקאות.

- **BranchID**: מספר מזהה ייחודי של הסניף (Primary Key)
- **BranchName**: שם הסניף
- **BranchAddress**: כתובת הסניף
- **BranchPhoneNumber**: מספר טלפון של הסניף

### 2. Customer

טבלת הלקוחות, מייצגת את הלקוחות של הבנק שמחזיקים בחשבונות בנק.

- **CustomerID**: מספר מזהה ייחודי של הלקוח (Primary Key)
- **FirstName**: שם פרטי
- **LastName**: שם משפחה
- **DateOfBirth**: תאריך לידה
- **Address**: כתובת
- **ContactNumber**: מספר טלפון
- **Email**: דוא"ל

### 3. Account

טבלת החשבונות, מייצגת את החשבונות של הבנק שמוחזקים על ידי הלקוחות.

- **AccountID**: מספר מזהה ייחודי של החשבון (Primary Key)
- **Balance**: יתרת חשבון
- **DateOpened**: תאריך פתיחת החשבון
- **AccountStatus**: מצב החשבון
- **BranchID**: מזהה הסניף (Foreign Key המצביע על BranchID בטבלה Branch)

- **AccountType**: סוג החשבון

#### 4. Vip

טבלת הלקוחות ה-VIP, מייצגת לקוחות VIP שמחזיקים בחשבונות מיוחדים עם הטבות ייחודיות.

- **AccountID**: מספר מזהה ייחודי של החשבון (Primary Key ו-Foreign Key המצביע על AccountID בטבלה Account)
- **PositiveInterest**: ריבית חיובית

#### 5. BlackList

טבלת הרשימה השחורה, מייצגת לקוחות שנכללו ברשימה השחורה מסיבות מסוימות.

- **AccountID**: מספר מזהה ייחודי של החשבון (Primary Key ו-Foreign Key המצביע על AccountID בטבלה Account)
- **NegetiveInterest**: ריבית שלילית
- **MinimumMinus**: מינוס מינימלי

#### 6. Relationship

טבלת הקשר בין לקוחות לחשבונות, מייצגת את הקשרים בין לקוחות לחשבונות שלהם.

- **CustomerID**: מספר מזהה ייחודי של הלקוח (Primary Key ו-Foreign Key המצביע על CustomerID בטבלה Customer)
- **AccountID**: מספר מזהה ייחודי של החשבון (Primary Key ו-Foreign Key המצביע על AccountID בטבלה Account)

#### 7. DirectDebit

טבלת החיובים הישירים, מייצגת את ההסדרים לחיוב ישיר שמוקמים על ידי הלקוחות לתשלומים חוזרים.

- **IDDebit**: מספר מזהה ייחודי של החיוב (Primary Key)
- **StartDate**: תאריך התחלה
- **TypeDebit**: סוג החיוב
- **Amount**: סכום
- **AccountID**: מספר מזהה של החשבון (Foreign Key המצביע על AccountID בטבלה Account)

#### 8. Transactions

טבלת הטרנזקציות, מייצגת את הטרנזקציות הקשורות לחשבונות הלקוחות.

- **TransactionID**: מספר מזהה ייחודי של הטרנזקציה (Primary Key)
- **TransactionType**: סוג הטרנזקציה
- **Amount**: סכום הטרנזקציה
- **TransactionDate**: תאריך הטרנזקציה

- **AccountID**: מספר מזהה של החשבון (Foreign Key המצביע על AccountID בטבלה (Account

## 9. CreditCards

טבלת כרטיסי האשראי, מייצגת את כרטיסי האשראי שמונפקים ללקוחות.

- **CardID**: מספר מזהה ייחודי של כרטיס האשראי (Primary Key)
- **CardNumber**: מספר הכרטיס
- **CardType**: סוג הכרטיס (Debit, Regular)
- **ExpirationDate**: תאריך פקיעה
- **CreditLimit**: מסגרת אשראי
- **AccountID**: מספר מזהה של החשבון (Foreign Key המצביע על AccountID בטבלה (Account

## 10. Loans

טבלת ההלוואות, מייצגת את ההלוואות שניתנו ללקוחות.

- **LoanID**: מספר מזהה ייחודי של ההלוואה (Primary Key)
- **LoanAmount**: סכום ההלוואה
- **InterestRate**: ריבית
- **StartDate**: תאריך התחלה
- **EndDate**: תאריך סיום
- **AccountID**: מספר מזהה של החשבון (Foreign Key המצביע על AccountID בטבלה (Account

## 11. Checks

טבלת השיקים, מייצגת את השיקים שנמשכו על ידי הלקוחות.

- **CheckID**: מספר מזהה ייחודי של השיק (Primary Key)
- **AccountID**: מספר מזהה של החשבון (Foreign Key המצביע על AccountID בטבלה (Account
- **CheckNumber**: מספר השיק
- **Amount**: סכום השיק
- **IssueDate**: תאריך הנפקה
- **ClearingDate**: תאריך פרעון

## 12. Deposits

טבלת ההפקדות, מייצגת את ההפקדות שנעשו על ידי הלקוחות.

- **DepositID**: מספר מזהה ייחודי של ההפקדה (Primary Key)
- **DepositAmount**: סכום ההפקדה
- **DepositDate**: תאריך ההפקדה
- **MaturityDate**: תאריך פדיון

- **InterestRate**: ריבית
- **AccountID**: מספר מזהה של החשבון (Foreign Key המצביע על AccountID בטבלה (Account

ניצור view שיציג לנו את הנתונים שלנו לפני האינטגרציה.

```

SQL | Output | Statistics
CREATE VIEW OurDBView AS
SELECT a.AccountID AS AccountID,
       a.Balance,
       a.DateOpened ,
       a.AccountStatus,
       a.BranchID,
       l.Loan_ID,
       l.Loan_Amount,
       l.Interest_Rate as L_Interest_Rate,
       l.Start_Date ,
       l.End_Date ,
       cc.Card_ID ,
       cc.Card_Number ,
       cc.Card_Type,
       cc.Expiration_Date ,
       cc.Credit_Limit ,
       t.TransactionID ,
       t.TransactionType ,
       t.Amount as t_amount,
       t.TransactionDate ,
       t.AccountID AS AccountID_t,
       d.DepositID ,
       d.Deposit_Amount ,
       d.Deposit_Date,
       d.Maturity_Date ,

```

49:1 123@XE AS SYSDBA [22:56:16] 8 rows selected in 0.086 seconds (ms)

נבחר פשוט בכל המשתנים, לאחר שעשינו leftJoin

```

t.AccountID AS AccountID_t,
d.DepositID ,
d.Deposit_Amount ,
d.Deposit_Date,
d.Maturity_Date ,
d.Interest_Rate AS InterestRate_d,
ch.Check_ID ,
ch.Check_Number ,
ch.Amount AS Amount_ch,
ch.Issue_Date ,
ch.Clearing_Date
FROM Account a
LEFT JOIN Loans l ON a.AccountID = l.Account_ID
LEFT JOIN Credit_Cards cc ON a.AccountID = cc.Account_ID
LEFT JOIN transactions t ON a.AccountID = t.AccountID
LEFT JOIN deposits d ON a.AccountID = d.Account_ID
LEFT JOIN checks ch ON a.AccountID = ch.Account_ID;

```

עשינו דווקא left join בכדי שנקבל גם שורות שבהם יש null ולא דווקא יש התאמה בנתונים.

נריץ שאילתא פשוטה. שתיתן לנו מידע על כל מספר חשבון, מהו הסכום שיש לו בחשבון. סכום ההלוואות שלקח, ממוצע הריבית על ההלוואות האלו, וכמה העברות ביצע.

SQL Output Statistics

```
SELECT
  AccountID,
  SUM(Loan_Amount) AS TotalLoanAmount,
  AVG(L_Interest_Rate) AS AvgLoanInterestRate,
  SUM(Balance) AS TotalBalance,
  COUNT(transactionid) AS total_transaction
FROM
  OurDBView
GROUP BY
  AccountID;
```

	ACCOUNTID	TOTALLOANAMOUNT	AVGLOANINTERESTRATE	TOTALBALANCE	TOTAL_TRANSACTION
1	100013			166584	4
2	100017			1284528	21
3	100019			10734	3
4	100027	1587530	3	105329	7
5	100044	1830750	4	76680	5
6	100047			238359	3
7	100049			76616	1
8	100051			-188115	5

100 44:1 [250] 123@XE AS SYSDBA [23:05:30] 8 rows selected in 0.109 seconds (more...)

נכתוב עוד שאילתא: שמחשבת את סך כל ההלוואות לכל סניף. ומחשבת את הריבית הממוצעת על ההלוואות לכל סניף. התוצאות מסודרות לפי מזהה הסניף בסדר עולה.

```
SELECT
  BranchID,
  SUM(Loan_Amount) AS TotalLoanAmount,
  AVG(L_Interest_Rate) AS AvgInterestRate
FROM OurDBView
GROUP BY BranchID
ORDER BY BranchID;
```

	BRANCHID	TOTALLOANAMOUNT	AVGINTERESTRATE
1	100	6390027	4.8
2	101		
3	103	3329250	5.5
4	104	6720148	3.04545454545455
5	105	697881	2
6	106	3709704	5
7	107		
8	108	4730944	3.33333333333333
9	109	5393058	4.8

100 52:19 123@XE AS SYSDBA [23:19:15] 9 rows selected in 0.155 seconds (more...)

עכשיו ניצור view שני לנתונים שבאו מהצוות השני:

```

SQL Output Statistics

CREATE OR REPLACE VIEW Team2DBView AS
SELECT
  c.CustomerID,
  c.FirstName,
  c.LastName,
  c.DateOfBirth,
  c.Address AS CustomerAddress,
  c.ContactNumber,
  c.Email,
  a.AccountID,
  a.Balance,
  a.DateOpened,
  a.AccountStatus,
  a.Account_Type,
  b.BranchID,
  b.BranchName,
  b.BranchAddress,
  b.BranchPhoneNumber,
  CASE
    WHEN v.AccountID IS NOT NULL THEN 'VIP'
    WHEN bl.AccountID IS NOT NULL THEN 'BlackList'
    ELSE 'Regular'
  END AS CustomerStatus,
  v.PositiveInterest,

  WHEN bl.AccountID IS NOT NULL THEN 'BlackList'
  ELSE 'Regular'
END AS CustomerStatus,
v.PositiveInterest,
bl.NegativeInterest,
bl.MinimumMinus,
dd.IDDebit,
dd.StartDate AS DirectDebitStartDate,
dd.TypeDebit,
dd.Amount AS DirectDebitAmount,
t.TransactionID,
t.TransactionType,
t.Amount AS TransactionAmount,
t.TransactionDate
FROM
  Customer c
JOIN Relationship r ON c.CustomerID = r.CustomerID
JOIN Account a ON r.AccountID = a.AccountID
JOIN Branch b ON a.BranchID = b.BranchID
LEFT JOIN Vip v ON a.AccountID = v.AccountID
LEFT JOIN BlackList bl ON a.AccountID = bl.AccountID
LEFT JOIN DirectDebit dd ON a.AccountID = dd.AccountID
LEFT JOIN Transactions t ON a.AccountID = t.AccountID;

```

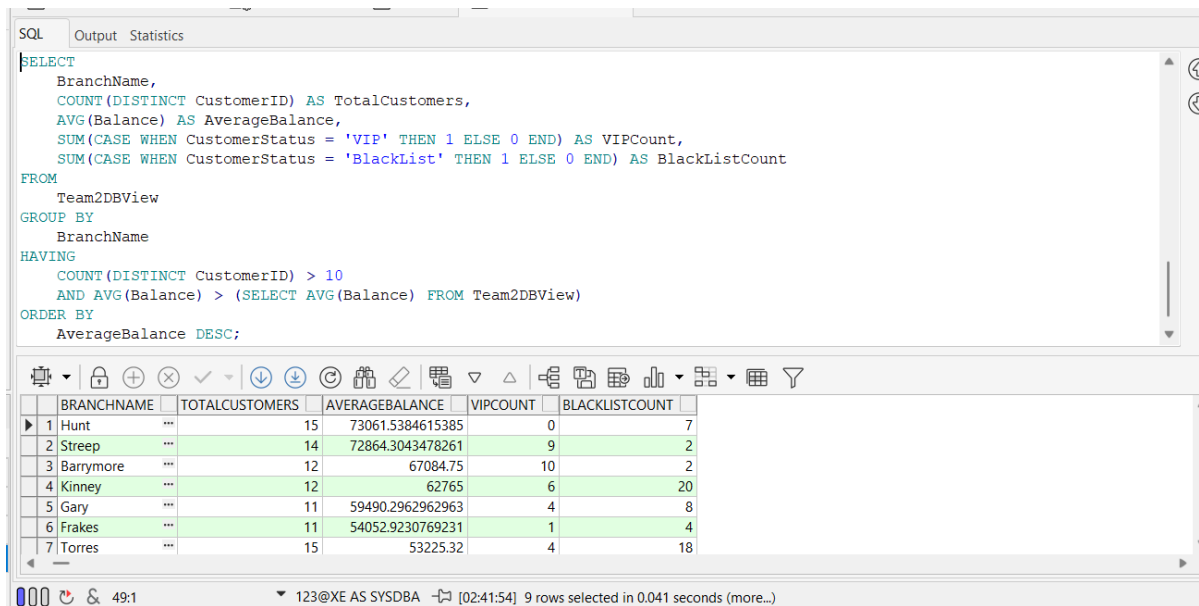
ביצענו איחוד join של כל הטבלאות, נראה שזה עובד ע"י שאילתא select all

```
SELECT * FROM Team2DBView FETCH FIRST 5000 ROWS ONLY;
```

	CUSTOMERID	FIRSTNAME	LASTNAME	DATEOFBIRTH	CUSTOMERADDRESS	CONTACTNUMBER	EMAIL	ACCOUNTID	BA
1	1001385	Stewart	Checker	10/09/2005 17:07:07	544 Dartmouth Ave	972585187	stewart.checker@trekequipment.com	100000	
2	1001385	Stewart	Checker	10/09/2005 17:07:07	544 Dartmouth Ave	972585187	stewart.checker@trekequipment.com	100000	
3	1001385	Stewart	Checker	10/09/2005 17:07:07	544 Dartmouth Ave	972585187	stewart.checker@trekequipment.com	100000	
4	1001385	Stewart	Checker	10/09/2005 17:07:07	544 Dartmouth Ave	972585187	stewart.checker@trekequipment.com	100000	
5	1001385	Stewart	Checker	10/09/2005 17:07:07	544 Dartmouth Ave	972585187	stewart.checker@trekequipment.com	100000	
6	1000502	Jose	Rudd	30/12/2022 10:19:19	1 Tyson Drive	972568305	jrudd@circuitcitystores.de	100001	
7	1000502	Jose	Rudd	30/12/2022 10:19:19	1 Tyson Drive	972568305	jrudd@circuitcitystores.de	100001	
8	1000502	Jose	Rudd	30/12/2022 10:19:19	1 Tyson Drive	972568305	jrudd@circuitcitystores.de	100001	

123@XE AS SYSDBA [02:35:31] 10 rows selected in 0.090 seconds (more...)

## ניכתוב שאילתא על הנתונים:



The screenshot shows the SQL Developer interface with the following SQL query in the SQL window:

```
SELECT
  BranchName,
  COUNT(DISTINCT CustomerID) AS TotalCustomers,
  AVG(Balance) AS AverageBalance,
  SUM(CASE WHEN CustomerStatus = 'VIP' THEN 1 ELSE 0 END) AS VIPCount,
  SUM(CASE WHEN CustomerStatus = 'BlackList' THEN 1 ELSE 0 END) AS BlackListCount
FROM
  Team2DBView
GROUP BY
  BranchName
HAVING
  COUNT(DISTINCT CustomerID) > 10
  AND AVG(Balance) > (SELECT AVG(Balance) FROM Team2DBView)
ORDER BY
  AverageBalance DESC;
```

The Results window displays the following data:

	BRANCHNAME	TOTALCUSTOMERS	AVERAGEBALANCE	VIPCOUNT	BLACKLISTCOUNT
1	Hunt	15	73061.5384615385	0	7
2	Streep	14	72864.3043478261	9	2
3	Barrymore	12	67084.75	10	2
4	Kinney	12	62765	6	20
5	Gary	11	59490.2962962963	4	8
6	Frakes	11	54052.9230769231	1	4
7	Torres	15	53225.32	4	18

The status bar at the bottom indicates: 123@XE AS SYSDBA [02:41:54] 9 rows selected in 0.041 seconds (more...)

אנחנו מקבצים לפי אגף, ומחזירים כמה לקוחות ישנם באגף, את הממוצע יתרת חשבון שלהם, כמות ה-VIP באגף וכמות אלו שנמצאים ב-blackList

כל זה בתנאי שלאגף יש יותר מ-10 לקוחות וגם הממוצע יתרת חשבון של אנשי האגף גבוהה מממוצע יתרת חשבון של כל לקוחות הבנק.

שאילתא 2:

Integration Commands.sql Data Generator views.sql team2View.sql

SQL Output Statistics

```
--query number 2

SELECT
    CustomerID,
    FirstName,
    COUNT(DISTINCT TransactionID) AS TransactionCount,
    SUM(CASE WHEN TransactionDate >= ADD_MONTHS(SYSDATE, -12) THEN TransactionAmount ELSE 0 END) AS TotalTransactionAmount,
    COUNT(DISTINCT IDDebit) AS DirectDebitCount,
    SUM(CASE WHEN DirectDebitStartDate >= ADD_MONTHS(SYSDATE, -12) THEN DirectDebitAmount ELSE 0 END) AS TotalDirectDebitAmount,
    CASE
        WHEN COUNT(DISTINCT TransactionID) > 10 THEN 'High Activity'
        WHEN COUNT(DISTINCT TransactionID) > 5 THEN 'Medium Activity'
        ELSE 'Low Activity'
    END AS ActivityLevel
FROM
    Team2DBView
GROUP BY
    CustomerID, FirstName, LastName
HAVING
    COUNT(DISTINCT TransactionID) > 0 or COUNT(DISTINCT IDDebit) > 0
ORDER BY
    TotalTransactionAmount DESC, TotalDirectDebitAmount DESC;
```

```

        WHEN COUNT(DISTINCT TransactionID) > 5 THEN 'Medium Activity'
        ELSE 'Low Activity'
    END AS ActivityLevel
FROM
    Team2DBView
GROUP BY
    CustomerID, FirstName, LastName
HAVING
    COUNT(DISTINCT TransactionID) > 0 or COUNT(DISTINCT IDDebit) > 0
ORDER BY
    TotalTransactionAmount DESC, TotalDirectDebitAmount DESC;
```

	CUSTOMERID	FIRSTNAME	TRANSACTIONCOUNT	TOTALTRANSACTIONAMOUNT	DIRECTDEBITCOUNT	TOTALDIRECTDEBITAMOUNT	ACTIVITYLEVEL
1	1001359	Ricardo	13	54000	0	0	High Activity
2	1000026	Hazel	13	54000	0	0	High Activity
3	1000091	Blair	11	54000	0	0	High Activity
4	1001289	Juliette	7	20000	0	0	Medium Activity
5	1000525	Rodney	10	20000	0	0	Medium Activity
6	1000777	Jennifer	12	12444	1	0	High Activity
7	1000675	William	14	12321	0	0	High Activity

פה אנחנו בעצם מחפשים את הלקוחות שביצעו יותר מ־10 העברות או שיש להן יותר מאפס הוראות קבע. נחזיר את הסכום הכולל של העברות שביצעו בשנה האחרונה ומספר העברות כמו גם את מספר החיובי קבע בנוסף לסכום החיוב, נוסיף לכל לקוח בהתאם לתנאים האם הוא ניקרא לקוח פעיל או לא.

ברמה התכנית אנחנו פשוט מקבצים לפי מס לקוח שם ושם משפחה, בודקים את התנאי ומסדרים לפי הסכום הכולל של העברות והוראות קבע.