


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;

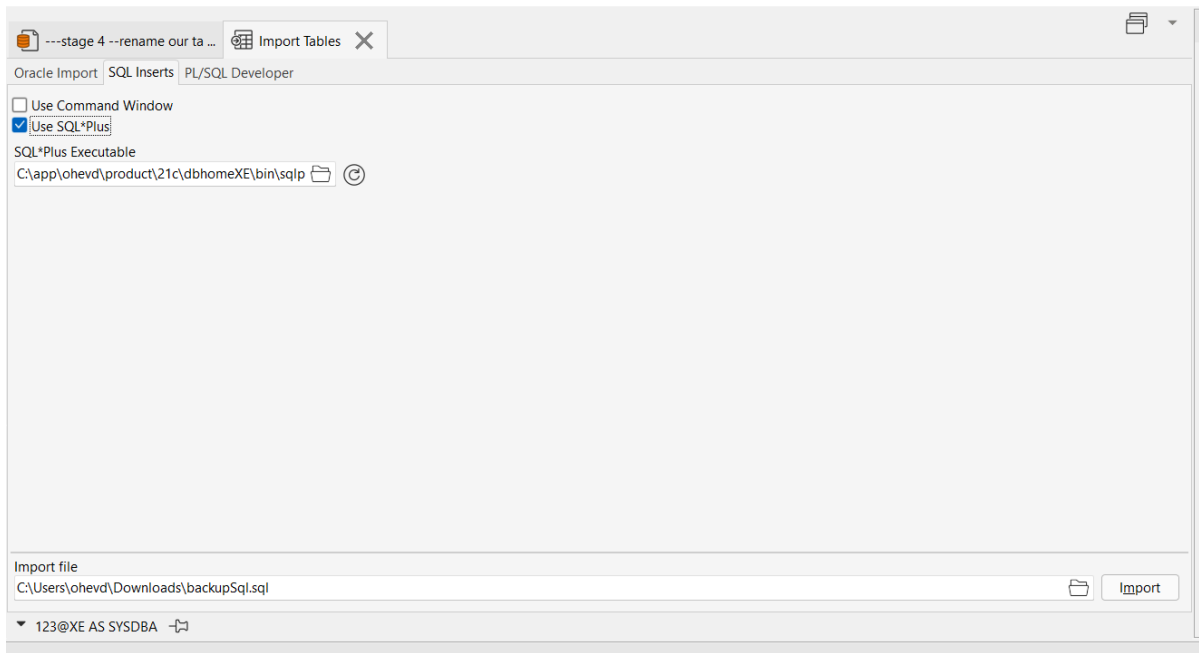
```

Select accounts1 Select checks1 Select credit_cards1 Select deposits1 Select transactions1 Select loans1

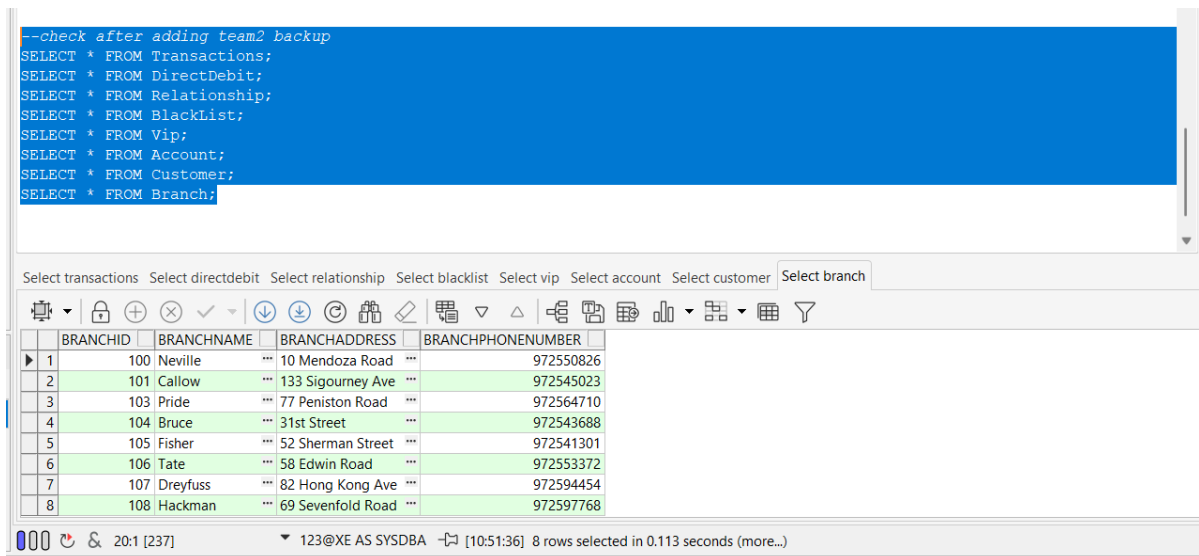
	ACCOUNT_ID	CUSTOMER_ID	ACCOUNT_TYPE	BALANCE	ACCOUNT_OPENING_DATE	
▶ 1	1	10	Savings	4000	20/05/2024	...
2	2	20	Checking	2000	15/05/2024	...
3	3	30	Checking	2555	16/05/2024	...
4	4	40	Checking	2040	19/05/2024	...
5	5	50	Savings	2900	18/05/2024	...
6	6	60	Checking	9999	12/05/2024	...
7	7	70	Savings	1542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

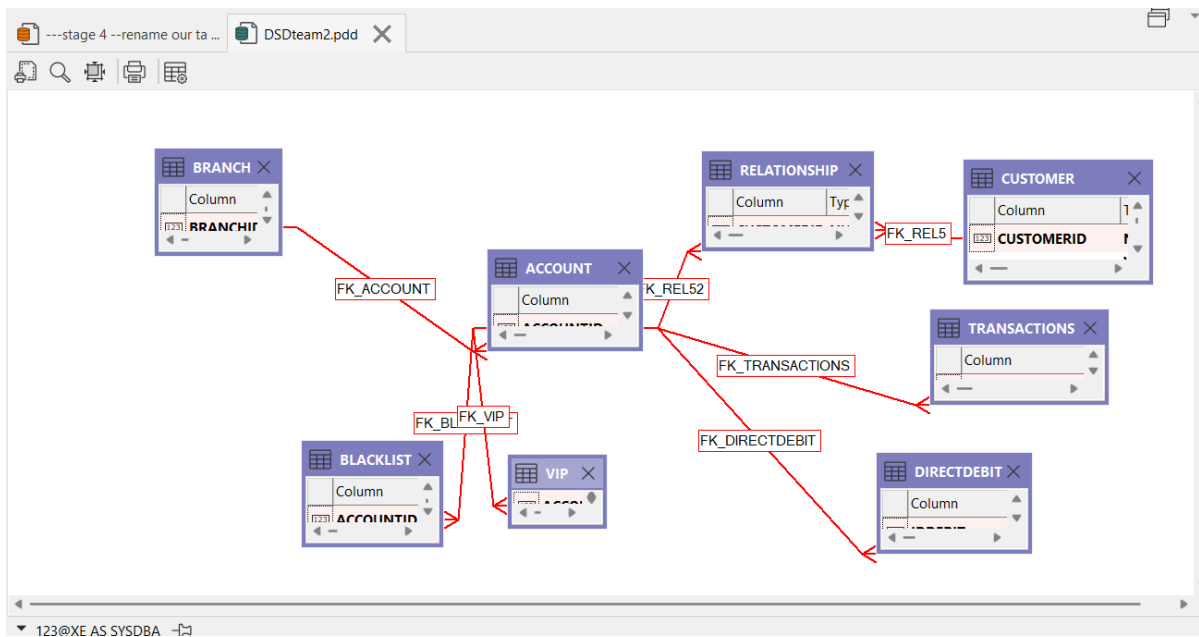


נריץ בחירה על טבלאות צוות 2 לראות שהכל עבר.

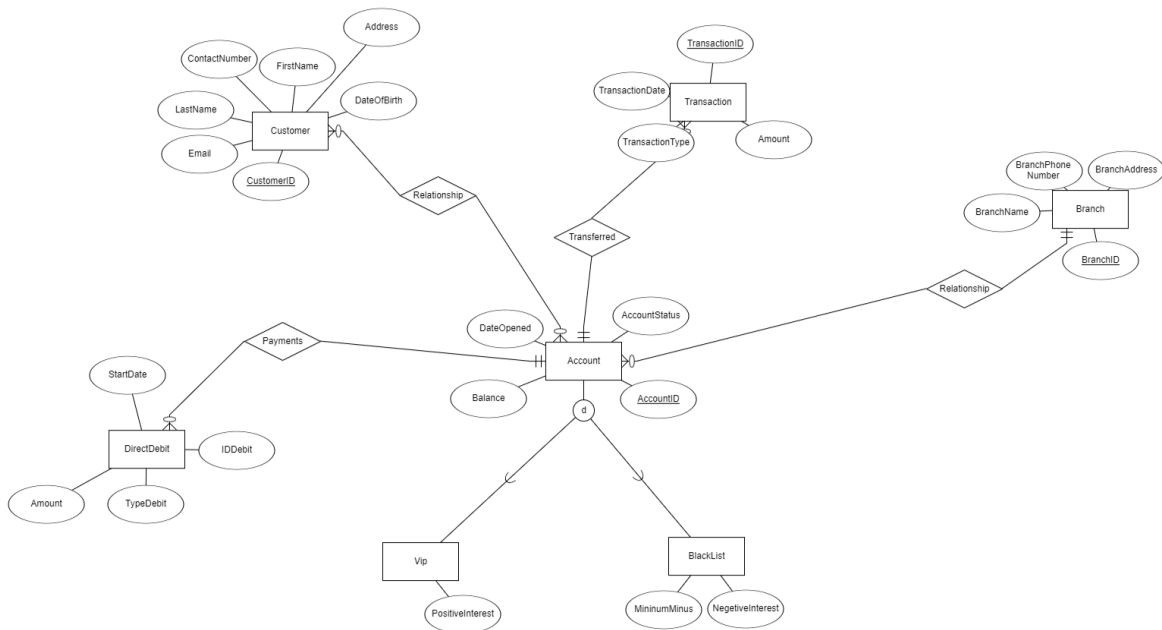


יפה הכל עבד כשורה!

נייצר תרשים DSD של צוות 2:

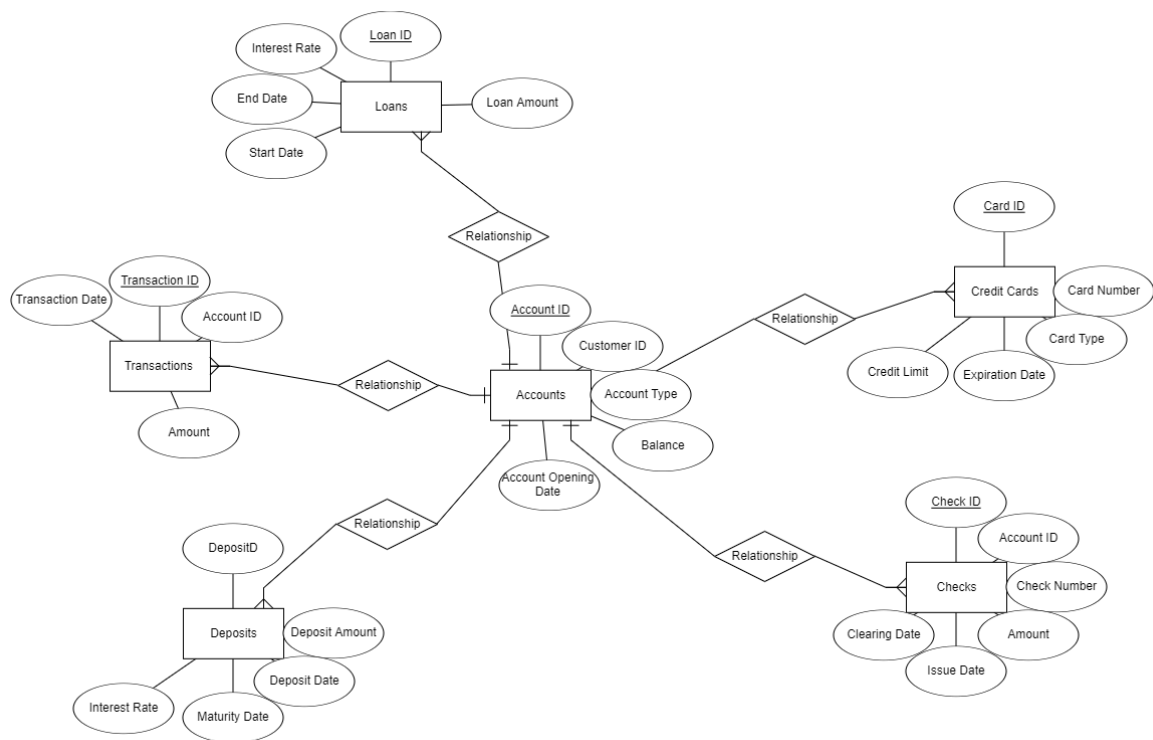


נשחזר את הERD ע"י ניתוח והבנה מעמיקה של המפתחות הזרים בכדי לגלות קשרים:



עד כאן מבוא לרברסינג .

את הERD שלנו כבר יש לנו להלן:



עכשיו נצטרך לחשוב, איך אנחנו מחברים ביניהם לכדי integrationERD.

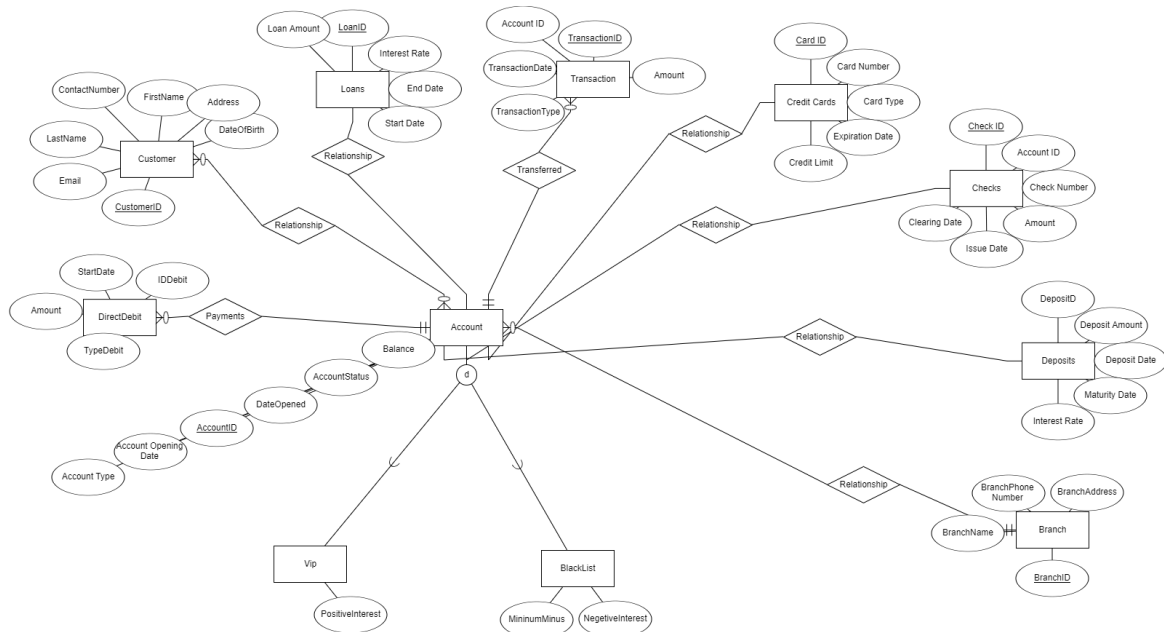
הדבר כרוך בתחבולות ועיקושים אך נציג את התוצאות להלן ולאחר מכן נסביר:

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

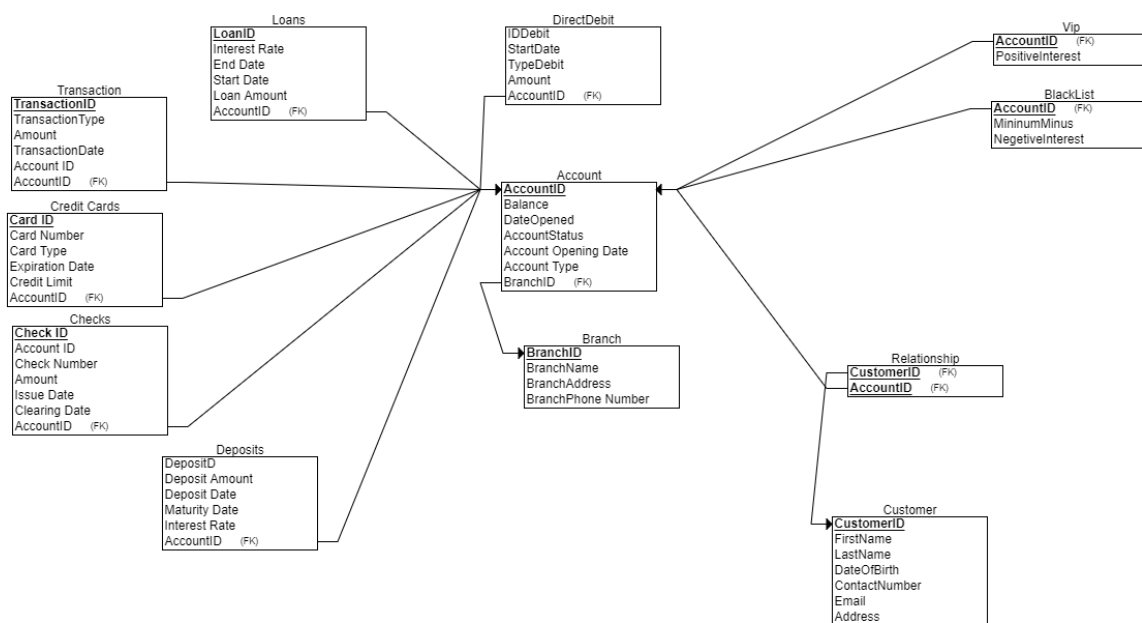
הטבלה transaction תואמת לשלנו נבצע איחוד - ונוסיף את העמודה transactionType לנתונים שלנו.

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

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

UPDATE accountsl 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 accountsl 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 al.account_id, al.balance, al.account_opening_date, al.accountstatus, al.branchid, al.account_type
FROM accountsl 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

	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 ששינונו 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 unnecessery 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 ,

```

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;

```

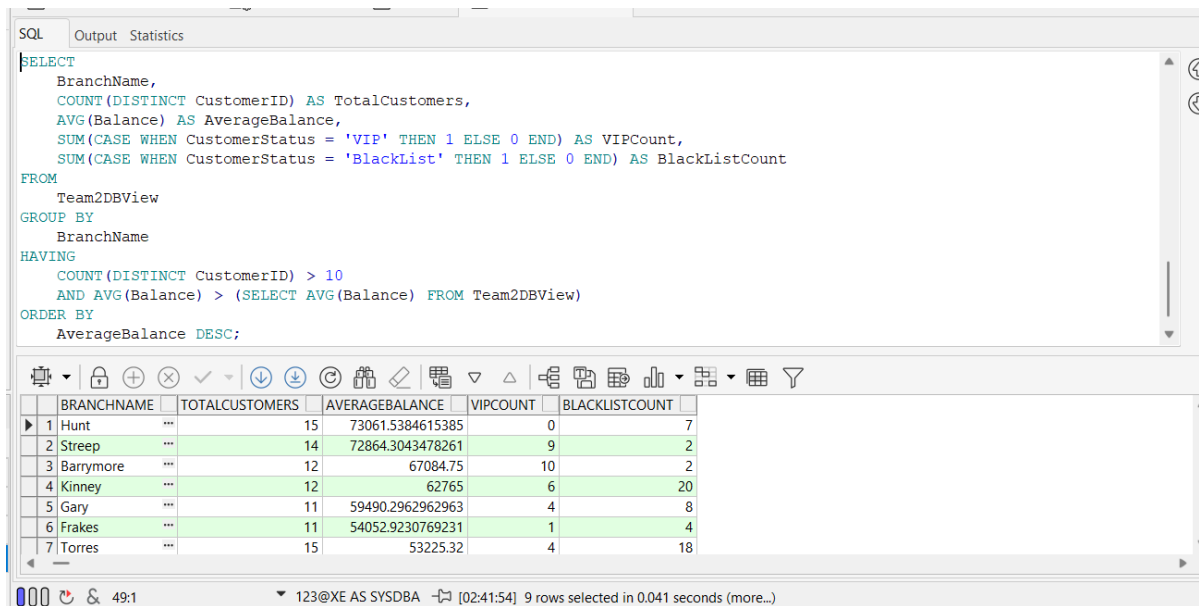
ביצענו איחוד join של כל הטבלאות, נראה שזה עובד ע"י שאילתא select all

```
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 a query window containing the following SQL statement:

```
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;
```

Below the query window, the results are displayed in a table with 7 rows and 6 columns:

	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 מ0 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.