Linnéuniversitetet
Kalmar Växjö

Linnaeus University
1DV503 – Database Technology
Programming Assignment 2

Student(s): Ryustem Shaban – rs223fx@student.lnu.se
Ahmad Obay Khairallah – ak224vi@studentt.lnu.se

Programming Assignment 2 Report

1. Project Idea

Since the crypto is seeing great interest, governments have already decided to take some actions regarding that. Recently laws for taxation of cryptocurrencies were adopted. However, since blockchains are decentralized financial exchange governments can not really chase and check one user's profit from the crypto trading, but still it is visible from where the money comes to the debt/credit card so what government asks for is each user to come with complete list of all transactions that were made with all the details included (date, profit amount, method, buy amount, held period) which will affect the taxation percentage. However, for someone who is new to crypto it may be hard to retrieve all this information since the user may need to retrieve transactions from multiple exchanges and, or there may be case when the user is busy person and have no time to check everything and gather information for the tax office. And since the adoption of cryptocurrencies continues with full speed we wanted to introduce our original idea and maybe business in the near future.

The main idea of the project is to show how the crypto is being prepared for taxation by private agency and further forwarded to governmental tax department. What agency offers to its customers is retrieving and calculation of all the transactions for given wallet and prepare them to be taxed while charging them only 1% of the profit. The agency's employees will retrieve the tax details from various exchanges and forward them to the governmental tax department with the personal number of the customer.
One important thing to state once more is that the agency itself does not tax the customers, it does check the transactions and retrieve the necessary information from exchanges with the help of professional agents and forwarding the expected tax amount and all the other details to the tax office where rest of the work will be handled by the authorities. This project will represent a sort of automated taxable feature. This feature can be helpful and can be very huge factor to fight and stop tax evasion in the digital world and make taxation of crypto easier for both sides (citizens and tax offices).

Data source was random and was retrieved from: mockaroo.com

2. Schema Design

The following ER model will express the database for ***yourCryptoExperts AB*** agency.

Our assumptions for the agency are:
Briefly, the database contains 10 offices that are part of the agency. For each office, there are five working employees. For each employee, manages two customers. Each customer has one wallet, and for each wallet, the customer can make four transactions. After the tax details are being successfully retrieved from the exchange by the employee, which will then be forwarded to the governmental tax department. The taxing process is out of the agency's hands.

Cardinality and Relationships:

- One office has many employees. *M: 1 relationship*
- One employee exposits many transactions. *M: 1 relationship*
- One employee reports many tax details. *M: 1 relationship*
- One office has many customers. *M: 1 relationship*
- One customer makes many transactions. *M: 1 relationship*
- One customer has one wallet. *1: 1 relationship*

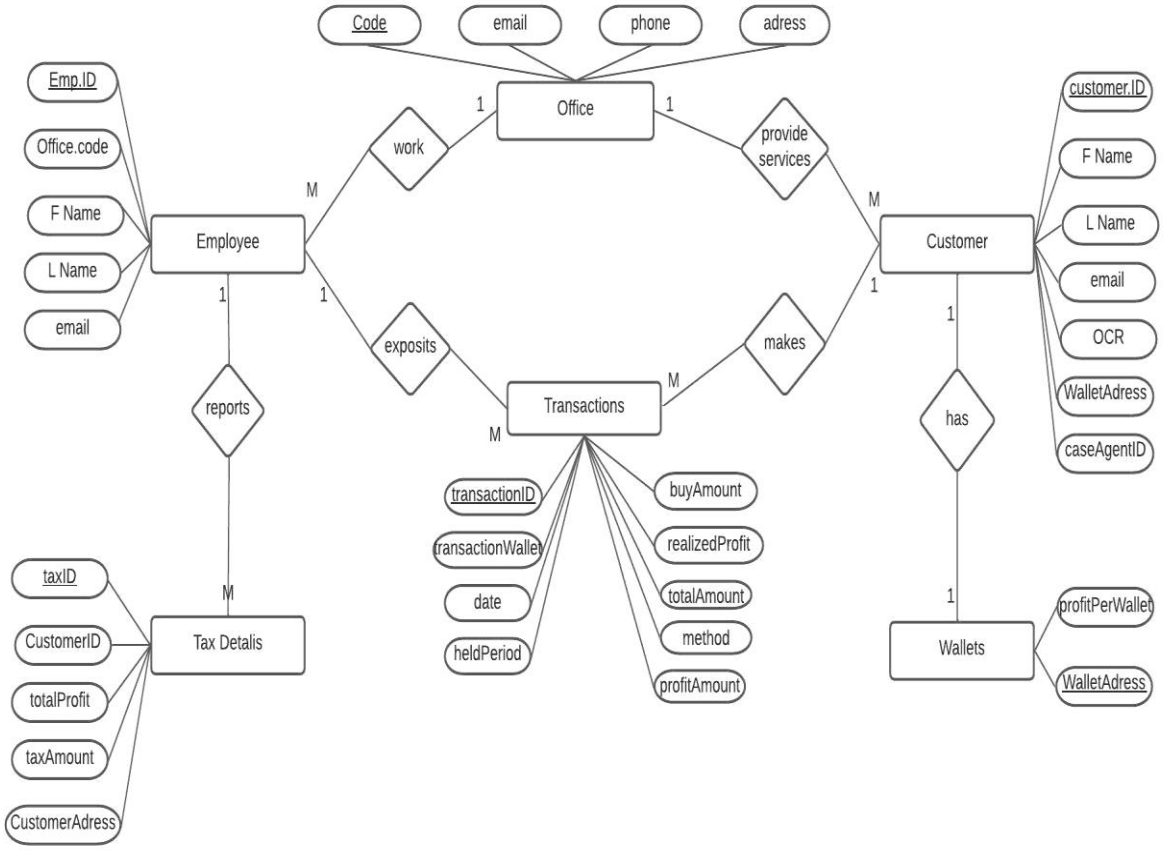

Figure 1

Below are examples of data that is created for the tables:

→10 offices

Office(Code(primary key), Email, Address, Phone number)

18802	storgatan@yourCryptoExperts.com	Storgatan 17, 352 30 Vaxjo	46722629784
18808	marconigatan@yourCryptoExperts.com	Marconigatan 5, 421 42 Vastra Frolunda	46789727947
18871	skattekontor@yourCryptoExperts.com	Skattekontor 2, 171 94 Solna	46705152151

→100 customers

Customer(Customer.ID(Primary key), First_name, Last_name, CaseAgentID(Foreign key), WalletAdress, OCR, Email)

9582113310	Nert	Sherborne	1567	0xd43e20f609d1f5a2507585a0191f2721a9c9152a	1220041865	nsherborne0@deliciousdays.com
9681272004	Aldo	McCluin	1581	0x574ff1245e6e8d396b473a687a13c32fdcda5dab	1220041879	amccluine@acquirethisname.com
9598986102	Isac	Roughley	1589	0x84b49722bc3e392d4eb939c49884191554e30865	1220041887	iroughleym@google.com.br

→50 employees

Employee(Emp.ID(Primary key), Office.code(Foreign key) ,First_name, Last_name, Email)

1567	18802	Jonell	Dani	JonellDani@yourCryptoExperts.com
1568	18802	Veronique	Killbey	VeroniqueKillbey@yourCryptoExperts.com
1569	18802	Damian	Portt	DamianPortt@yourCryptoExperts.com

→100 taxDetails

taxDetails(TaxID(Primary Key), CustomerID(Foreign key) CustomerAdress, TotalProfit, TaxAmount)

147925	9582113310	260 34 Laktargatan 77	3,239.63	647.926
147926	9402705794	241 79 Klappinge	43,7079.91	2477.96
147927	9753718495	312 13 Kantorsvagen	39,4383.69	876.738

→100 wallets

wallet(walletAdress(Primary Key), profitPerWallet)

0xd43e20f609d1f5a2507585a0191f2721a9c9152a	3239.63
0x0d01276f47bba3f22f9e487e79dab21d3e42ad9f	7079.91
0x5f4656e48fe1dc0b37abd5e8a0e3288263206637	4383.69

→400 transactions

Transactions(TransactionID(Primary Key), TransactionWallet(Foreign Key), Date, HeldPeriod, BuyAmount, RealizedProfit, TotalAmount, ProfitAmount, Method)

oo7LCUqrFcjU84NWNYvpgA4JJF2hFeX9g4rPX8qk9yiGNAHhZpe	0xd43e20f609d1f5a2507585a0191f2721a9c9152a	2/10/2021	12	180.57	FALSE	-180.57	0	spot
oobii7iMALfEG6VGBrJHxdKpcf2YzhtMnmzVs4KNXLHpXVR4Kbj	0xd43e20f609d1f5a2507585a0191f2721a9c9152a	4/11/2021	12	1037.44	TRUE	1233.64	196	margin
onnKvHFt9UK7xzsM17hnp6GEiPCgWgtKWaSorC19Z5eFYVYGfWS	0xd43e20f609d1f5a2507585a0191f2721a9c9152a	7/22/2021	10	978.69	FALSE	-978.69	0	spot

3. SQL Queries

Queries below will be important for futuristic: income and tax statistics, observational procedures that helps to keep track of all activities within the agency, and

Q: **Project the average of expected to be paid tax amount in 2020-2021.**
The following query is using the aggregate function AVG to retrieve important information.
Even though it is so monotonous and simple query still wanted to add the Average tax paid, because for the company it is important to have that for the statistics and analytics.

```
SELECT avg(taxAmount)
FROM taxdetails;
```

Q: **Show the number of employee for given office code.**
The following query is multirelational and uses **JOIN** matching the offices.officeCode with the foreign key for employees employees.office. To the **WHERE** clause office code is passed as reference. The query should return us number of employees working in the given office.

```
SELECT offices.adress, count(employeeID)
FROM offices
JOIN employees ON offices.officeCode = employees.office
WHERE officeCode = 18802;
```

Q: **If we want to check how many transactions given customer has made**
The following query is multirelational and uses **JOIN,** joining transactions to the customers by matching customers.walletAdress with transactions.transactionWallet which is foreign key for the transactions**.** Parameter used to satisfy the **WHERE** clause is the customers.ID
It is expected to return us numeric value for number of transactions that were made.
this is important because, we may want to double check the total profit made by wallet, in that case we may need to sum up the transactions and instead of checking the wallet address for each we may just count it.

```
SELECT customers.first_name as Customer, count(transactionID) as
transactionsMade,customers.walletAdress as Wallet
FROM customers
JOIN transactions on customers.walletAdress = transactions.transactionWallet
WHERE customers.ID = 9582113310;
```

Q: **Project the maximum tax paid by customer in 2021.**
The following query is multirelational and uses two **JOIN** clauses in order to retrieve information from two different table that are not connected with each other directly. Firstly joining **customers** to the **taxdetails** matching the **customers.ID** with **taxdetails.citizenID** followed by joining the transactions to the customers where we matched **customers.walletAdress** with **transactions.transactionWallet**. **WHERE** clause is filled with "**%2021"**, because in that query we wanted to retrieve only information for 2021. This may be important query because these details are also supposed to be logged. Of course, this query we understand who made most profit within 2021 also.

```
SELECT concat(customers.first_name," ", customers.last_name) as Customer,customers.ID as ID,
max(taxAmount) as MaximumTaxPaidFor2021
FROM taxdetails
JOIN customers on taxdetails.citizenID = customers.ID
JOIN transactions on customers.walletAdress = transactions.transactionWallet
WHERE dateWirthdraw like '%2021';
```

Q: **Project how many customers there are who held their tokens for less than 4 months and managed to made profit.**
It is not very likely to hold for less than 4 months and still withdraw money profitably. So, this was interesting query and wanted to add.

```
SELECT COUNT(transactions.transactionID) as transcationCount
FROM transactions
WHERE realizedProfit = 1 and heldMonths < 4;
```

Q: **Project the customers who are paying tax over 20%.**
The following query is one multi relational which uses view and additionally has a join and uses grouping. Firstly created **VIEW** in order to be able to use it later on as well directly as a table. This **VIEW** contains mutltirelational query, using **JOIN** to match transactions.transactionWallet on **customers.walletAdress** and putting **WHERE** clause that the hold period should be over 12Months to ensure that the tax percentage is over 20%.
Wanted to Create view and list all the customers who are paying tax over 20% taxation within the crypto is respecting the profit and held months as well. After holding for more than year one crypto the tax is fixed to 25% of the profit otherwise depending on the amount 15-20-35%

```
CREATE VIEW CustomersWithTaxOver20Percent as
SELECT CONCAT (customers.first_name, " ", customers.last_name) as
CustomersWithTaxOver20Percent, customers.ID as ID, avg(heldMonths) as
AverageHoldPeriodPerWallet
FROM customers
JOIN transactions on customers.walletAdress = transactions.transactionWallet
WHERE heldMonths > 12
GROUP BY customers.ID
ORDER BY customers.ID desc;
SELECT * FROM CustomersWithTaxOver20Percent;
```

4. Discussion and Resources

Decided to start this kind of project instead of writing for dorm renter agencies just because wanted to cover a real topic. The idea is to solve one problem that people will have when mass crypto adoption begins. Realistically gathering all these crypto documents and bringing them to the tax office may be so much time consuming and nightmare. So, what we did here is to provide real solution which hopefully will be implemented to the real world also. Another thing to mention is that within the database the Address of the offices are from Swedish cities and are written by hand. Phone numbers are Swedish and the email addresses belonging to the offices and workers are with additional details for the sake of realism. Multiple queries run with the help of **TRANSLATIONS**, however for taxing details and all the information that table is our core table where all other data can be retrieved from that table.
The agency's workflow is as followed:

- Customer goes to the agency and provides them with required information and is being saved to the database with an unique OCR number.
- Regarding the office address one employee is assigned to work for the case.
- Employee checks the given information and retrieves all the information necessary for the tax agencies to tax it
- Additionally, employees update another table called customerWallets as well as a smaller table in order to access some information faster if something not goes as if it was supposed to.
- Finally, the employee fills the tax details and forwarding them to the tax office where authorities are handling the rest.

The project uses the following libraries for python:
mysql.connector
errorcode from mysql.connector
os
msvcrt
csv
tkinter

Source code: on GitHub
[https://github.com/Plann1ng/1DV503DatabasaTehnology/blob/main/yourCryptoExperts.py

Video demonstration on YouTube:
https://www.youtube.com/watch?v=0jo9RrwDSjI&ab_channel=RustemDuran

Changelog

2022-03-08

Person	Task	Date
Rustom & Obay	Designing project	2022-03-01
Rustom	Created data	2022-03-04
Obay	Importing & Creating queries on workbench	2022-03-10
Rustom	Creating queries on python	2022-03-10
Rustom	Python implementations	2022-03-10
Obay	Constructed ER diagram	2022-03-13
Rustom & Obay	Report writing	2022-03-15
Rustom & Obay	Report writing	2022-03-16