Aidan Premeau, Johnny Mai, Caleb Newman
Team 7- Project Report
Cryptocurrency Database

Introduction:

Project Objective: The objective of this database project is to produce a database that is both scalable and accurate to the current values of the cryptocurrency. Our expected outcomes are created to easily locate an individual's personal investments, current coin values and numerical identifiers. The database will be capable of retrieving all of this data and one could update this data by either selling or buying more of the currency chosen.

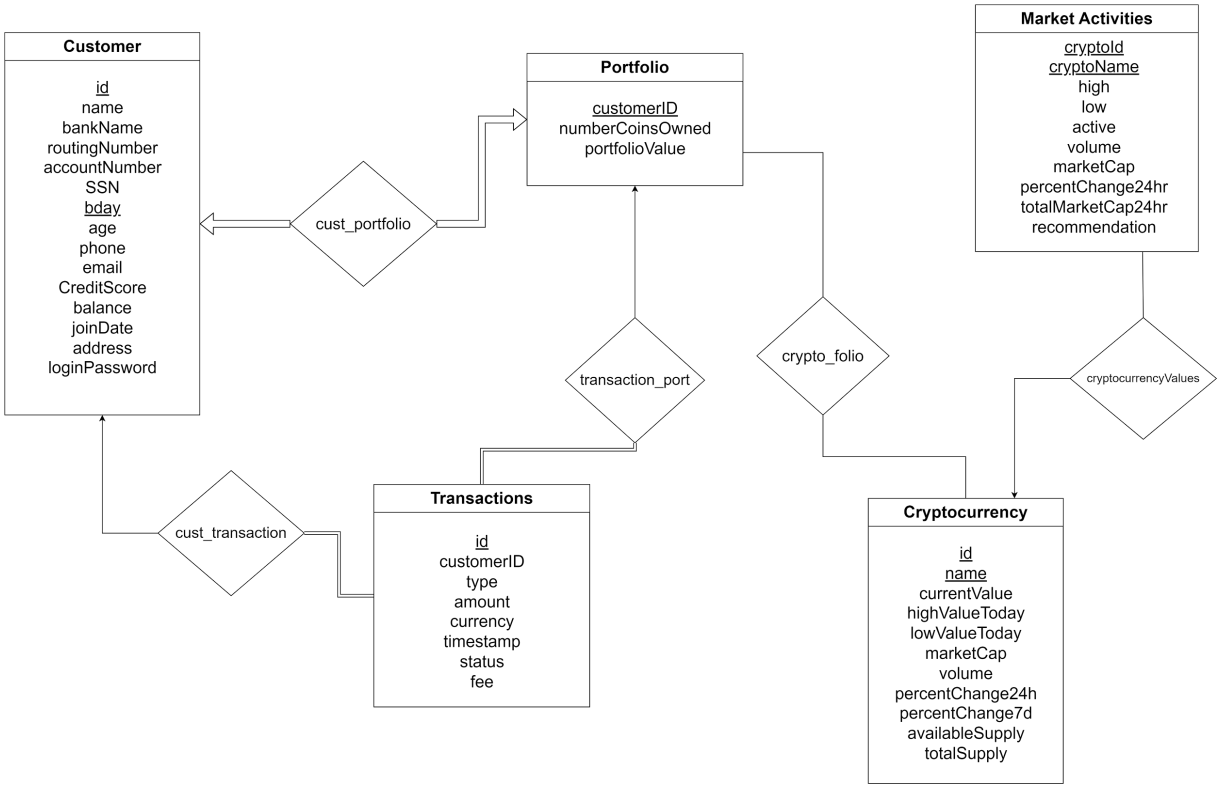
Scope of the Project: This project's scope covers the aspects of an individual's crypto portfolio:

- Numerical identification numbers(User ID, Receipt number, etc.)
- Some string values to identify the currency and the date purchased
- Floats to denote the value of the currencies by the desired denomination
- Integer or float to represent how much of the selected currency the individual owns

Dataset Description: The dataset is fairly standard for identifying an individual's purchases and stock holdings. Key Entities:

- Consumers: Each person who has purchased some form of cryptocurrency will have both an ID and some form of receipt number to refer to the individual purchase of a currency.
- Cryptocurrency: Each individual type of currency has an associated value that is reflected as the selected denomination.
- Purchases: Every purchase made is referred to by a receipt value and an amount of the currency that was purchased, with what denomination and how much of that denomination.

Entity-Relationship Diagram:



Relational Model:

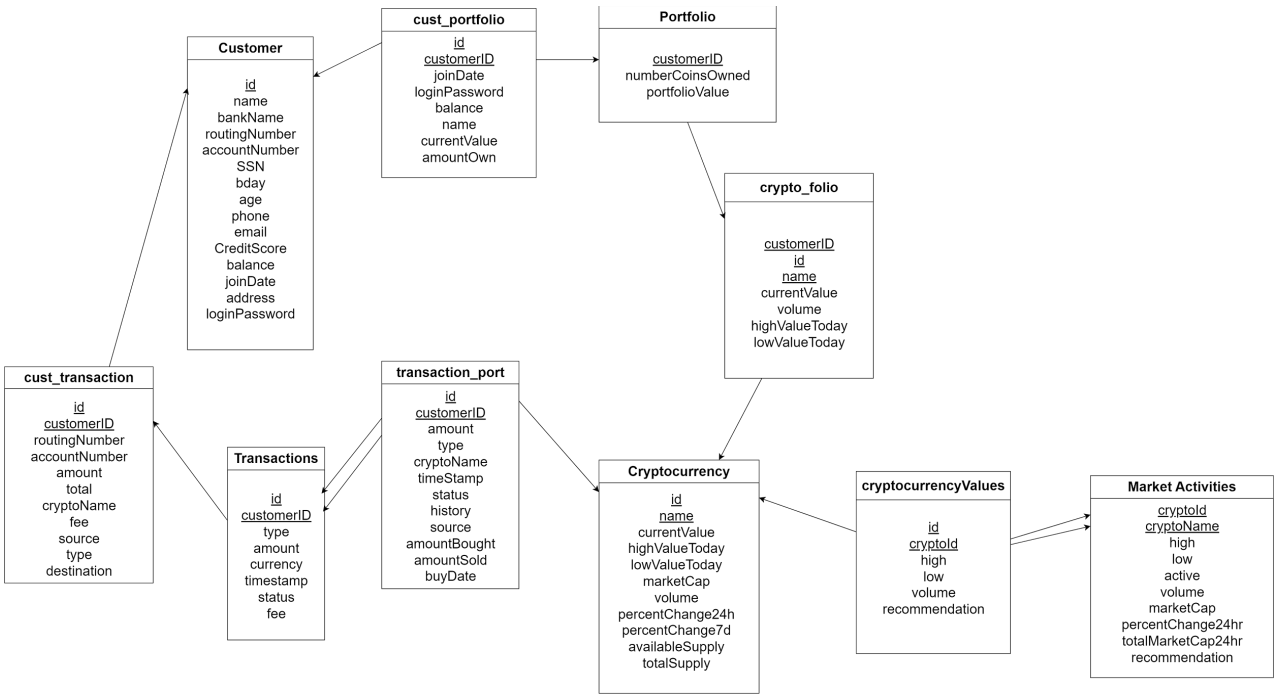


Table Creation: Attached is the SQL file for creating tables for the cryptocurrency database.

Relation	Attribute	Domain
Customer	name	varchar(25)
	id	numeric(7)
Primary Key: (id)	bankName	varchar(25)
	routingNumber	numeric(9,0)
	accountNumber	numeric(15)
	SSN	numeric(9)
	bday	numeric(8)
	age	numeric(3,0)
	phone	numeric(10,0)

	email	varchar(30)
	CreditScore	numeric(3,0)
	balance	numeric(10,2)
	joinDate	numeric(8)
	address	varchar(50)
	loginPassword	varchar(30)
Market Activities PRIMARY KEY (cryptoId) FOREIGN KEY (cryptoName) REFERENCES cryptocurrency(name)	cryptoId	numeric(5)
	high	numeric(12,2)
	low	numeric(12,2)
	cryptoName	varchar(25)
	active	numeric(12,2)
	volume	numeric(14,0)
	marketCap	numeric(14,0)
	percentChange24hr	numeric(5,2)
	totalMarketCap24hr	numeric(14)
	recommendation	varchar(15)
Portfolio PRIMARY KEY (ID)	customerId	numeric(7)
	numberCoinsOwned	numeric(3)
	portfolioValue	numeric(12,2)
Cryptocurrency PRIMARY KEY (id, name)	currentValue	numeric(6,2)
	id	numeric(5)
	name	varchar(35) UNIQUE

	highValueToday	numeric(7,2)
	lowValueToday	numeric(7,2)
	marketCap	numeric(15,2)
	volume	numeric(15,0)
	percentChange24h	numeric(6,4)
	percentChange7d	numeric(6,4)
	availableSupply	numeric(15,0)
	totalSupply	numeric(15,0)
Transactions PRIMARY KEY (id) FOREIGN KEY (customerId) REFERENCES customer (id)	id	numeric(15)
	customerId	numeric(7)
	type	varchar(15)
	amount	numeric(12,2)
	currency	varchar(20)
	timestamp	numeric(14)
	status	varchar(15)
	fee	Numeric (6,2)

Data Population: Attached is the SQL file for populating the database. Data was generated randomly using ChatGPT to fill in relevant information in our tables.

SQL Queries:

Question 1: Retrieve the names and email addresses of customers who have joined the platform after a specific date.

Query 1:

-- Retrieve names and emails of customers who have joined after a specific date

```
SELECT name, email
FROM Customer
WHERE joinDate > 20210405;
```

Jupyter code:

```
query = "SELECT id, name, email FROM customer WHERE joinDate > 20150505"
df = pd.read_sql(query, engine)
print(df)
```

Question 2: Display the total volume of transactions made by each customer.

Query 2:

-- Display the total volume of transactions made by each customer

```
SELECT customerId, SUM(amount) AS total_volume
FROM transactions
GROUP BY customerId
ORDER BY total_volume DESC;
```

Jupyter Code:

```
query = "SELECT customerID, SUM(amount) AS total_volume FROM Transactions
GROUP BY customerID"
df = pd.read_sql(query, engine)
print(df)
```

Question 3: List all transactions where the amount exceeds a certain threshold and the status is 'completed'.

Query 3:

-- List all transactions where the amount exceeds a threshold.

```
SELECT *
FROM transactions
WHERE amount > 22000 AND status = ‘Successful’;
```

Jupyter Code:

```
query = "SELECT * FROM Transactions WHERE amount > 750 AND status =
'Completed'"
df = pd.read_sql(query, engine)
print(df)
```

Question 4: Calculate the average percentage change in cryptocurrency value over the last 7 days.

Query 4:

```
-- Calculate the average percentage change in crypto value over 7 days.

SELECT AVG(percentChange7d) AS average_change_7_days
FROM Cryptocurrency;
```

Jupyter Code:

```
query = "SELECT AVG(percentChange7d) AS average_change_7_days FROM
Cryptocurrency"
df = pd.read_sql(query, engine)
print(df)
```

Question 5: Identify customers who have bought and sold the same cryptocurrency within a specified time frame.

Query 5:

```
-- Identify customers who have executed a transaction between some time frame.

SELECT c.id, c.name
FROM customer AS c JOIN transactions AS t ON c.id = t.customerid
WHERE t.timestamp > 20240408120001 AND t.timestamp < 20240408120026;
```

Jupyter Code:

```
query = "SELECT c.id, c.name FROM customer AS c JOIN transactions AS t ON c.id =
t.customerid WHERE t.timestamp < 20240113235959 AND t.timestamp >
20240111000000"
df = pd.read_sql(query, engine)
print(df)
```

Question 6: Determine the total amount of cryptocurrency bought by customers from a specific bank.

Query 6:

```
-- Determine the total amount of cryptocurrency bought by customers from a
--specific bank.

SELECT c.bankName, SUM(t.amount) AS total_bought
FROM Customer c
JOIN Transactions t ON c.id = t.customerID
WHERE t.type = 'Buy' AND c.bankName = 'Wells Fargo'
GROUP BY c.bankName;
```

Jupyter Code:

```
query = "SELECT c.bankname, SUM(t.amount) AS total_bought FROM customer c
JOIN transactions t ON c.id = t.customerid WHERE t.type = 'Deposit' GROUP BY
c.bankname"
df = pd.read_sql(query, engine)
print(df)
```

Question 7: List all customers who have a credit score above a certain value and have made transactions in the last month.

Query 7:

```
-- List all customers who have credit score above a certain value and joined before a
specific date.

SELECT *
FROM customer
WHERE creditscore > 750 AND joindate < 20220211;
```

Jupyter Code:

```
query = "SELECT * FROM Customer WHERE CreditScore > 670 AND joinDate > 20150101"
df = pd.read_sql(query, engine)
print(df)
```

Question 8: Find the total number of transactions of each type ('buy' or 'sell') for a given cryptocurrency.

Query 8:

```
-- Find the total number of transaction of each type(buy or sell) for a given crypto.

SELECT currency, type, COUNT(*) AS transaction_count
FROM transactions
GROUP BY currency, type;
```

Jupyter Code:

```
query = "SELECT currency, type, COUNT(*) AS transaction_count FROM transactions GROUP BY currency, type"
df = pd.read_sql(query, engine)
print(df)
```

Question 9: Retrieve the names of customers along with the current value of the cryptocurrencies they own, sorted in descending order of value.

Query 9:

```
-- Rank customers by order of the highest portfolio value, display their name, Id, and value.

SELECT p.customerId, c.name, p.portfolioValue
FROM portfolio AS p JOIN customer AS c ON p.customerId = c.id
ORDER BY portfolioValue DESC;
```

Jupyter Code:

```
query = "SELECT name, (currentvalue * amountown) as portfolio_value FROM portfolio
ORDER BY portfolio_value DESC"
df = pd.read_sql(query, engine)
print(df)
```

Question 10: Retrieve the names of customers along with the cryptocurrencies they own and their corresponding current values.

Query 10:

```
-- Retrieve the names of customers along with the number of crypto coins they own.

SELECT p.customerId, c.name, p.numberCoinsOwned
FROM portfolio AS p JOIN customer AS c ON p.customerId = c.id
ORDER BY p.numberCoinsOwned DESC;
```

Jupyter Code:

```
query = "SELECT name, cryptoname, currentvalue FROM portfolio ORDER BY
currentvalue DESC"
df = pd.read_sql(query, engine)
print(df)
```

Schemas derived from the relationship sets in the E-R diagram:

Customer(customer.name, customer.id, customer.bankName, customer.routingNumber, customer.sos, customer.bday, customer.age, customer.phone, customer.email, customer.CreditScore, customer.balance, customer.joinDate, customer.address, customer.loginPassword)

MarketActivities(cryptoId, crypto.high, crypto.low, crypto.active, market.volume, market.marketCap, market/percentChange24hr, market.totalMarketCap24hr, market.recommendation)

Portfolio(portfolio.buyValue, portfolio.sellValue, portfolio.sellDate, portfolio.buyDate, portfolio.name, portfolio.currentValue, portfolio.amountOwn, portfolio.amountSold, portfolio.amountBought)

Cryptocurrency(crypto.currentValue, crypto.name, crypto.highValuetoday, crypto.lowValuetoday, crypto.marketCap, crypto.volume, crypto.percentChange24hr, crypto.percentChange7d, crypto.availableSupply, crypto.totalSupply)

Transaction(transaction.id, transaction.customerID, transaction.type, transaction.amount, transaction.currency, transaction.timestamp, transaction. Destination, transaction.status, transaction.fee, transaction.source,transaction.fee, Transaction.source, transaction. history)

Conclusion:

Our primary takeaways from this project were most certainly the construction of a database using SQL and the process by which one arrives at a final product. We encountered several challenges along the way leading to inconsistencies between SQL files, incorrect tables, diagrams and conflicting data. The goal of our project was to develop a database capable of storing both information and transactions made regarding various types of cryptocurrency much like modern day applications such as Coinbase, Crypto.com or Kraken. Future plans for the project could be the implementation of something akin to a modern day Blockchain which manages these crypto transactions through a distributed ledger. This ledger grows in size with every record or ‘block’ linked together by cryptographic hashes. But overall the database and interface developed in python work as intended with definite room for growth and improvement.