1. Submission located within the doc folder

See doc/BurgerCryptoUML.pdf

2. Having a complete and correct ER/UML diagram

See doc/BurgerCryptoUML.pdf

3. The ER/UML diagram has 5 or more entities

See doc/BurgerCryptoUML.pdf

4. Database Schema is normalized and the process of normalization is clearly stated

Choosing between BCNF (Boyce-Codd Normal Form) and 3NF (Third Normal Form) depends on the trade-offs between normalization strength and dependency preservation. BCNF is chosen over 3NF when the primary goal is to **eliminate redundancy** and ensure that every determinant is a candidate key, which minimizes anomalies in insertion, deletion, and update operations.
However, BCNF can sometimes lead to a **loss of functional dependencies** if the decomposition process to achieve BCNF breaks down a relation in a way that these dependencies cannot be preserved without redundancy.
Therefore, we finally choose to use 3NF since it preserves functional dependencies, which is crucial for the integrity of the database schema, even if it means tolerating a minimal amount of redundancy. 3NF makes it a more balanced choice for practical database design where dependency preservation is as important as reducing redundancy.

****

Function Dependency:

Consider the functional dependencies on
R(AssetID, CryptoName, SymbolURL, TradeID, TradeType, Quantity, TradePrice, TradeTime, Portfolio, PortfolioID, PortfolioName, DisplayColor, IsPinned, Quantity, PostID, PostTitle, ThumbsUpNum, Description, PostContent, UserID, UserName, Password, Email, JoinTime, Balance, BurgerCoins, InvestmentAdviceID, InvestmentAdviceTitle, InvestmentAdviceContent, InvestmentAdviceTime):

We have the relation as follows:
AssetID → CryptoName, SymbolURL
TradeID → TradeType, Quantity, TradePrice, TradeTime, AssetID, PortfolioID
PortfolioID → PortfolioName, DisplayColor, IsPinned, UserID
PortfolioID, AssetID → Quantity
PostID → PostTitle, ThumbsUpNum, Description, PostContent, UserID
UserID → UserName, Password, Email, JoinTime, Balance, BurgerCoins
InvestmentAdviceID → InvestmentAdviceTitle, InvestmentAdviceContent, InvestmentAdviceTime, PortfolioID

1. Get minimal basis for the given FDs: Current FDs are already minimal basis.
2. For each FD in minimal basis construct new relations.

Asset(AssetID, CryptoName, SymbolURL); Trade(TradeID, TradeType, Quantity, TradePrice, TradeTime, AssetID, PortfolioID); Portfolio(PortfolioID, PortfolioName, DisplayColor, IsPinned, UserId); Post(PostID,

PostTitle,ThumbsUpNum, Description, PostContent, UserID); User(UserID, UserName, Password, Email, JoinTime, Balance, BurgerCoins); InvestmentAdvice(InvestmentAdviceID, InvestmentAdviceTitle, InvestmentAdviceContent, InvestmentAdviceTime, PortfolioID);

3. Add another relation whose schema is a key for the original relation (3NF)
   M(TradeID, InvestmentAdviceID, PostID)

5. Having a complete and correctly translated relational schema

- User(UserID:INT [PK], UserName:VARCHAR(X), Password:VARCHAR(X), Email:VARCHAR(X), JoinTime:DATE, Balance:DECIMAL, BurgerCoins:INT)
- Post(PostID:INT [PK], UserID: INT [FK to User.ID], Title:VARCHAR(X), ThumbsUpNum:INT, Description:VARCHAR(X), Content:VARCHAR(X),CreateTime:DATETIME, UpdateTime:DATETIME)
- InvestmentAdvice(AdviceID:INT [PK], PortflioID: INT [FK to Portfolio.ID], Content:VARCHAR(X), Title:VARCHAR(X), Time:DATETIME)
- Asset(AssetID:INT [PK], CryptoName:VARCHAR(X), SymbolURL:VARCHAR(X))
- Portfolio(PortfolioID:INT [PK], UserID: INT [FK to User.ID], Name:VARCHAR(X), DisplayColor:VARCHAR(X), IsPinned:BOOL)
- Trade(TradeID:INT [PK], PortfolioID:INT [PK, FK to Portfolio.ID], AssetID:INT [PK, FK to Asset.ID], TradeType:VARCHAR(X), Quantity:DECIMAL, TradePrice:DECIMAL, TradeTime:DATETIME)

  User has 0..* Portfolio
  User posts 0..* EducationalContent
  Portfolio generates 1..* InvestmentAdvice
  CryptoAsset on 0..* Trade
  Portfolio execute 0..* Trade

6. Assumptions of the ER/UML diagram

- **User:** Represents the platform's users, including both investors and content creators. The inclusion of balance and BurgerCoins directly within the User entity simplifies financial transactions and rewards mechanisms, avoiding the need for a separate Wallet entity. This assumes that each user's financial interactions can be directly managed through their profile for efficiency.
- **Post:** Encapsulates educational materials/blogs provided by users. Linking directly to the User entity assumes that every piece of content can be traced back to its creator, fostering accountability and enabling features like content rating (ThumbsUpNum) and detailed descriptions for user engagement.
- **InvestmentAdvice**: Contains tailored investment advice tied to specific portfolios. This assumes a personalized approach to investment guidance, where advice is directly relevant to the configurations of a user's investment portfolio, ensuring customized support.

- **Portfolio:** Represents a collection of investments held by a user. The direct link to User underscores the assumption that each portfolio is unique to a user, facilitating personalized investment tracking and management.
- **CryptoAsset**: Defines the various cryptocurrencies available for trading. This entity's standalone nature assumes the need for a central repository of crypto assets that all users can access, trade, and analyze, separate from individual user actions or portfolios.
- **Trade:** Records transactions made within a portfolio. By associating trades with both Portfolio and CryptoAsset, it's assumed that tracking the specifics of each trade (quantity, price, and time) is crucial for investment strategy and performance review.

7. A description of each relationship and its cardinality

- A post can be posted by exactly 1 User.
- A Portfolio may have 0 to many InvestmentAdvice while an InvestmentAdvice belongs to one and only one Portfolio.
- A portfolio may hold 0 to many assets and one kind of asset can be held by many portfolios.
- A crypto asset may have many price history records.
- A Portfolio may record 0-many Trades while each Trade is executed on one and only one Portfolio.
- A user may have 0 to many Portfolio while a Portfolio belongs to one and only one User.

8. Create a release with the correct tag for your submission and submit it on PrairieLearn

See Stage.2: https://github.com/cs411-alawini/sp24-cs411-team068-BurgerMaker/tree/stage.2

9. .Fix all suggestions for the previous stages

See in Proposal.md.