MANSOURA UNIVERSITY
FACULTY OF COMPUTER AND INFORMATION

Payify

By:
Mohamed Saad Ahmed Ali Omar
SWE - 805554525

Under supervision:
Prof: Hazem Elbakry
Eng: Mohamed Eldawansy

1- THE PROJECT DESCRIPTION

- **Payify** is a cutting-edge money transfer and US stock trading service that aims to make financial transactions as seamless as possible. By leveraging the power of email and a user-friendly interface, Payify empowers its users to transfer money and trade stocks effortlessly.
- **Problem:** Traditional money transfer and stock trading platforms often involve complicated processes, multiple intermediaries, and time-consuming procedures.
- **Objective:** To create a simple and efficient platform for users to transfer money and trade stocks using just their email addresses.
- **Scope:** Payify is designed to cater to individual users looking for an easy-to-use, fast, and secure platform to transfer money and trade US stocks.
- **Solution:** Payify utilizes Python, Flask, MySQL, HTML, and CSS to create a seamless platform for money transfers and stock trading. Users can view their transaction history, monitor stock prices, and execute trades with just a few clicks.
- **Related Projects:** Payify is inspired by and related to projects such as PayPal, Venmo, Zelle, and Robinhood, but it distinguishes itself by offering a combined money transfer and stock trading functionality.

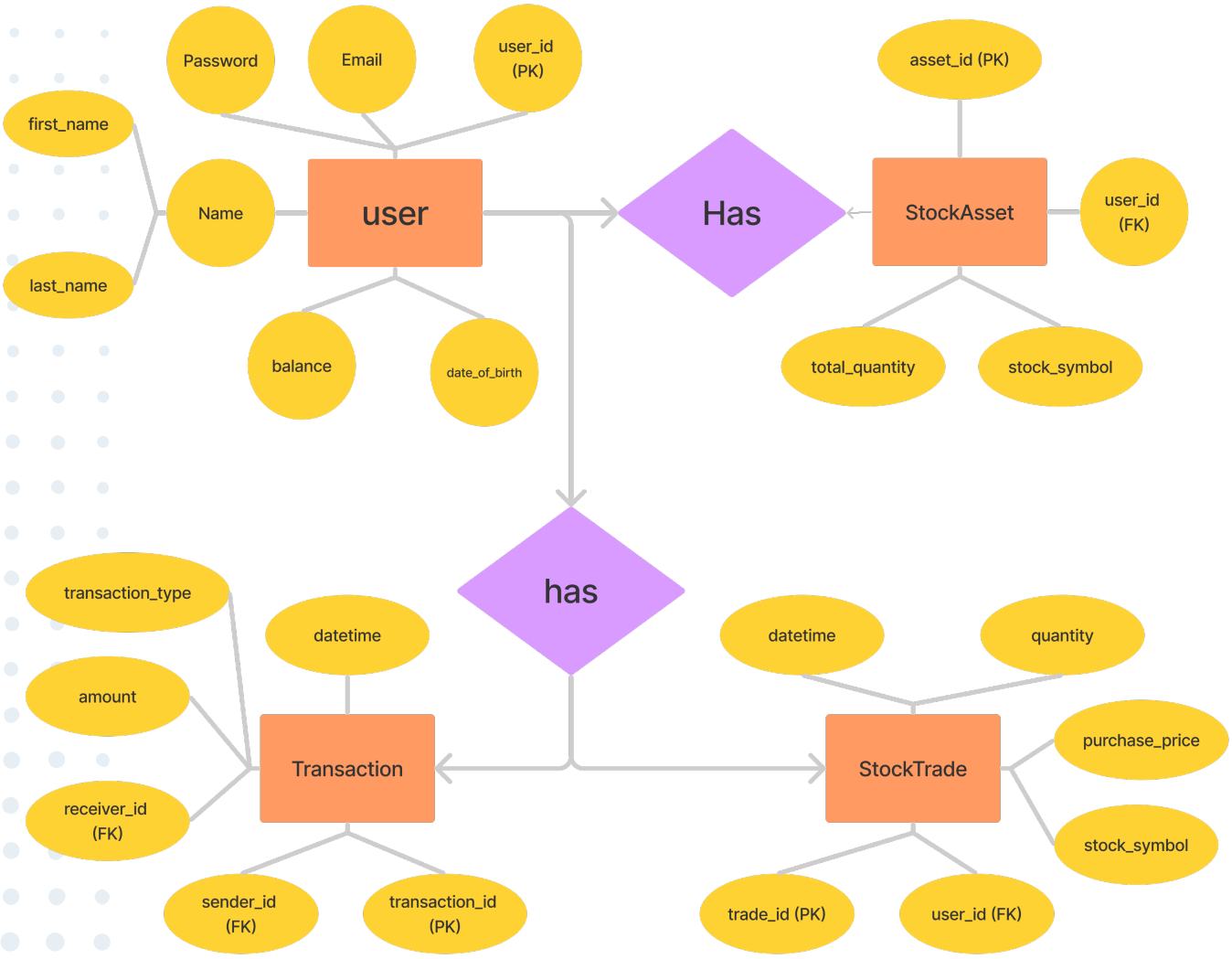
2- THE PROJECT BUSINESS RULES

- Users must register and verify their email addresses before using Payify's services.
- Users are required to provide accurate personal information and be over 18 to register.
- Transfers must be initiated using the recipient's email address.
- Payify will fetch stock prices from a trusted API but will store only the purchase price in the database.
- Stock trades will be executed according to applicable laws and regulations.
- Users will have access to their transaction history, including both money transfers and stock trades.
- Payify will adhere to strict security and data protection measures to ensure user privacy.

3- DATABASE ERD

- **Entities:** User, Transaction, StockTrade, StockAsset
- **Relationships:** User to Transaction (one-to-many), User to StockTrade (one-to-many), User to StockAsset (one-to-many)
- **Relationship degree:** Binary for all relationships
- **Relationship cardinality:** one-to-many for all relationships
- **Cardinality constraints:** A user can have multiple transactions, stock trades, and stock assets, but each transaction, stock trade, and stock asset is associated with only one user.
- **Entity attributes:**
  - User: User: user_id (PK), email, first_name, last_name, balance, date_of_birth, password
  - Transaction: transaction_id (PK), sender_id (FK), receiver_id (FK), amount, date, transaction_type
  - StockTrade: trade_id (PK), user_id (FK), stock_symbol, purchase_price, quantity, date
  - StockAsset: asset_id (PK), user_id (FK), stock_symbol, total_quantity
- **Primary keys:** user_id, transaction_id, trade_id, asset_id

3- DATABASE ERD



4- DATABASE MAPPING AND THE RELATIONAL MODEL

Steps to convert the updated ERD to the relational model:

1. Create a relation for each entity with its attributes.
2. Convert relationships and cardinality constraints into foreign keys.
3. Establish primary keys for each relation.

Relational Model Schema:

- User (user_id: PK, email, first_name, last_name, balance, password)
- Transaction (transaction_id: PK, sender_id: FK, receiver_id: FK, amount, date, transaction_type)
- StockTrade (trade_id: PK, user_id: FK, stock_symbol, purchase_price, quantity, date)
- StockAsset (asset_id: PK, user_id: FK, stock_symbol, total_quantity)

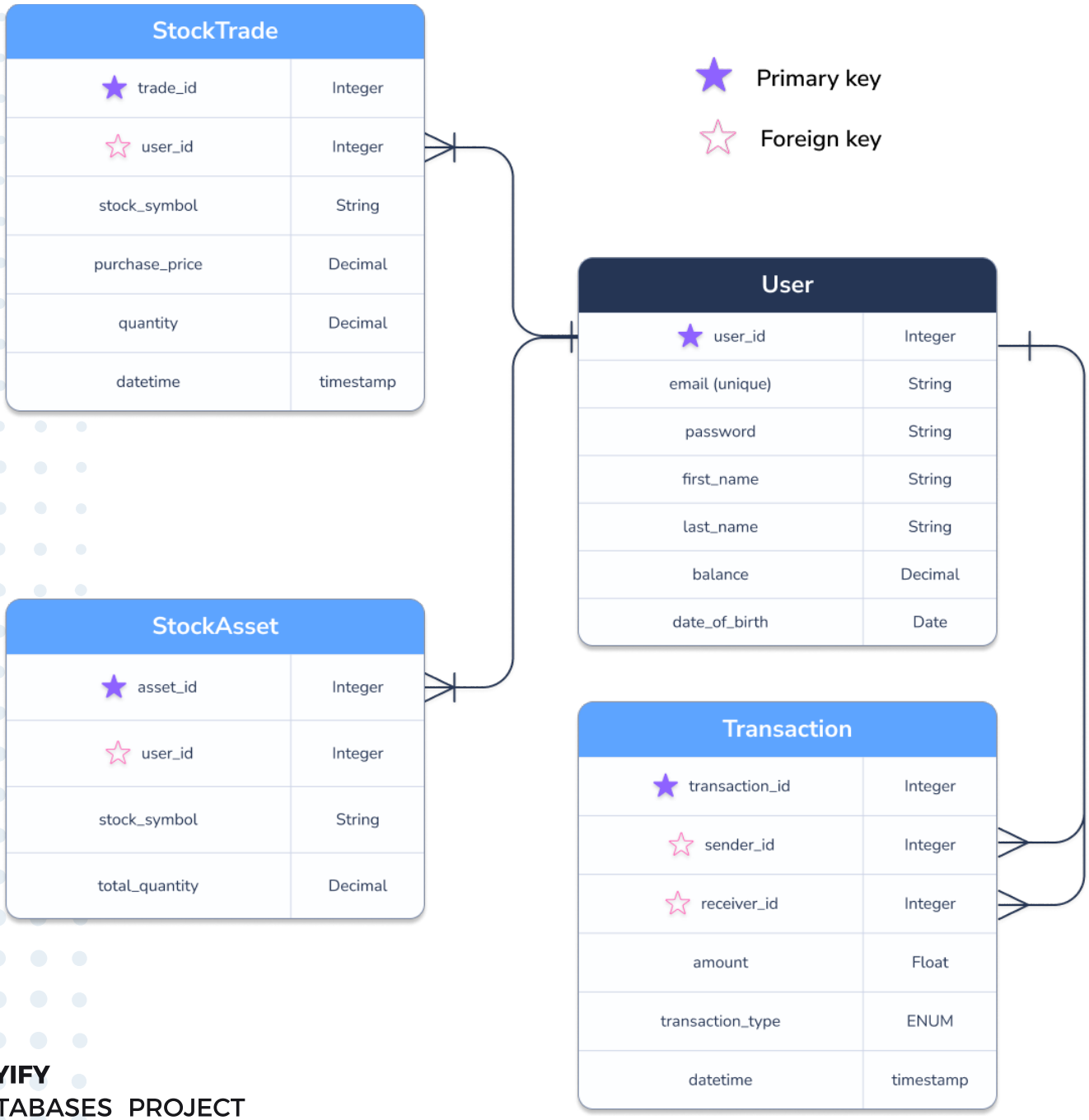
Relationships:

- User to Transaction: one-to-many through sender_id and receiver_id foreign keys in the Transaction relation.
- User to StockTrade: one-to-many through user_id foreign key in the StockTrade relation.
- User to StockAsset: one-to-many through user_id foreign key in the StockAsset relation.

Primary and Foreign Keys:

- **Primary keys:** User.user_id, Transaction.transaction_id, StockTrade.trade_id, StockAsset.asset_id
- **Foreign keys:** Transaction.sender_id and Transaction.receiver_id referencing User.user_id, StockTrade.user_id referencing User.user_id, StockAsset.user_id referencing User.user_id

4- DATABASE MAPPING AND THE RELATIONAL MODEL



5. SAMPLE OF SQL USED IN THE PROJECT

```sql
-- Create the 'User' table:
CREATE TABLE User (
  user_id INT AUTO_INCREMENT PRIMARY KEY,
  email VARCHAR(255) UNIQUE NOT NULL,
  first_name VARCHAR(50) NOT NULL,
  last_name VARCHAR(50) NOT NULL,
  bank_account VARCHAR(50) NOT NULL
);

-- ALTER TABLE:
ALTER TABLE User
ADD COLUMN phone_number VARCHAR(20);

-- INSERT INTO
INSERT INTO User (email, first_name, last_name, bank_account)
VALUES ('john.doe@example.com', 'John', 'Doe', '1234567890');

-- UPDATE
UPDATE User
SET phone_number = '555-123-4567'
WHERE user_id = 1;

-- DELETE
DELETE FROM Transaction
WHERE transaction_id = 100;

-- SELECT
SELECT *
FROM Transaction
WHERE sender_id = 1 OR receiver_id = 1;
```

5. SQL OF MAKING THE PHYSICAL MODEL AT MYSQL WORKBENCH

1 of 2

```sql
-- Create the Payify database
CREATE DATABASE payify;
USE payify;

-- Create the User table
CREATE TABLE user (
  user_id INT AUTO_INCREMENT PRIMARY KEY,
  email VARCHAR(255) UNIQUE NOT NULL,
  password VARCHAR(255) NOT NULL,
  first_name VARCHAR(50) NOT NULL,
  last_name VARCHAR(50) NOT NULL,
  date_of_birth DATE NOT NULL,
  balance DECIMAL(10, 2) NOT NULL
);

-- Create the Transaction table
CREATE TABLE Transaction (
  transaction_id INT AUTO_INCREMENT PRIMARY KEY,
  sender_id INT NOT NULL,
  receiver_id INT NOT NULL,
  amount DECIMAL(10, 2) NOT NULL,
  transaction_type ENUM('sent', 'received') NOT NULL,
  datetime TIMESTAMP NOT NULL,
  FOREIGN KEY (sender_id) REFERENCES User(user_id),
  FOREIGN KEY (receiver_id) REFERENCES User(user_id)
);
```

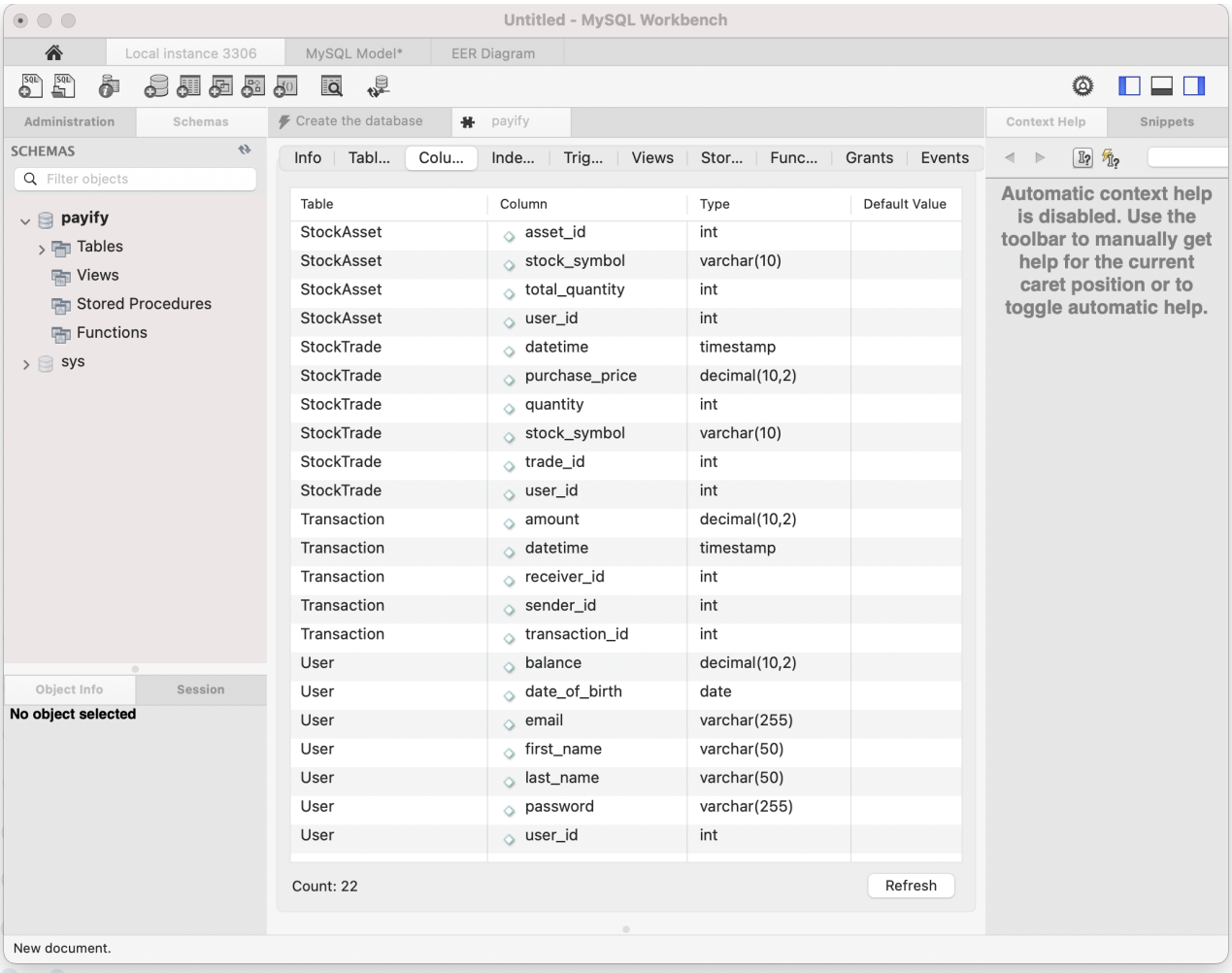
5. SQL OF MAKING THE PHYSICAL MODEL AT MYSQL WORKBENCH

2 of 2

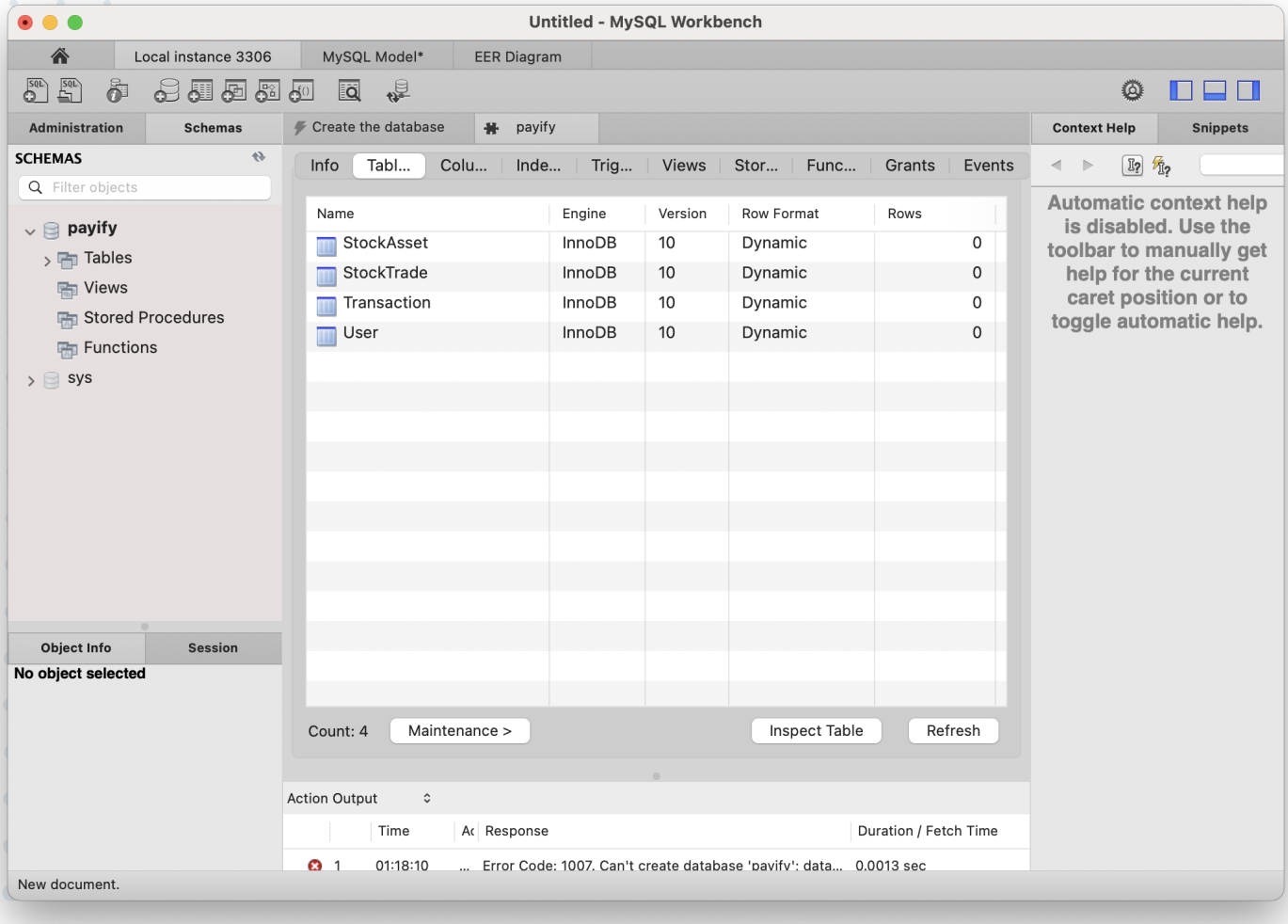
```sql
-- Create the StockTrade table
CREATE TABLE StockTrade (
  trade_id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  stock_symbol VARCHAR(10) NOT NULL,
  purchase_price DECIMAL(10, 2) NOT NULL,
  quantity INT NOT NULL,
  datetime TIMESTAMP NOT NULL,
  FOREIGN KEY (user_id) REFERENCES User(user_id)
);


-- Create the StockAsset table
CREATE TABLE StockAsset (
  asset_id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  stock_symbol VARCHAR(10) NOT NULL,
  total_quantity INT NOT NULL,
  FOREIGN KEY (user_id) REFERENCES User(user_id)
);
```

6. DATABASE PHYSICAL MODEL AT MYSQL WORKBENCH



6. DATABASE PHYSICAL MODEL AT MYSQL WORKBENCH



MANSOURA UNIVERSITY
FACULTY OF COMPUTER AND INFORMATION

Thanks