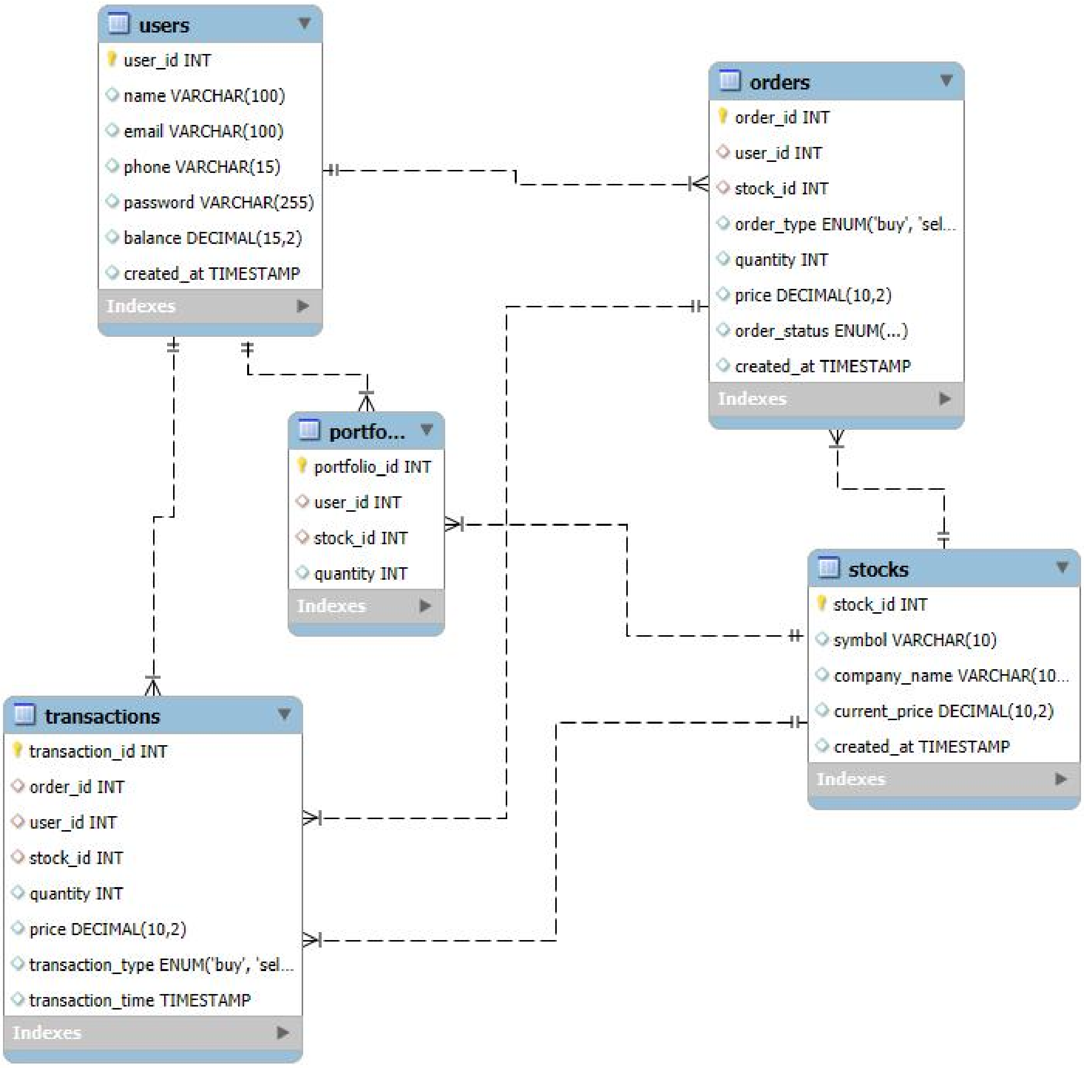
AngelOne Stock Trading Platform Database

The AngelOne Stock Trading Platform Database is designed to support the operations of a stock trading platform, offering users the ability to buy and sell stocks, manage their portfolios, and track stock prices and transaction histories. This MySQL-based project organizes and stores the essential data required for a trading platform, ensuring smooth and efficient management of user accounts, stock transactions, portfolios.

ER DIAGRAM:



Database Structure Overview:

1. Users Table (Accounts)
Stores user details such as personal information, email, and available balance for trading.
Columns : user_id, name, email, phone, password, balance, created_at.

2. Stocks Table
Contains stock data, including stock symbols, company names, market sectors, and current prices.
Columns : stock_id, symbol, current_price, created_at.

3. Orders Table
Tracks buy and sell orders placed by users, including order type, quantity, and status.
Columns: order_id, user_id, order_type, quantity, price, order-status, created_at.

4. Transactions Table
Records completed transactions (buy/sell) for stocks, linking users and stocks with transaction details.
Columns: transaction_id, order_id, user_id, stock_id, quantity, price, transaction_type, transaction_time.

5. Portfolio Table
Manages users' stock holdings, tracking the quantity of stocks each user owns.
Columns: user_id, stock_id, quantity.

Summary of Relationships:

Users → Orders: One user can place multiple orders (one-to-many).
Stocks → Orders: A stock can have multiple orders placed for it (one-to-many).
Orders → Transactions: One order can result in one or more transactions (one-to-many).
Users → Portfolio: A user can have a portfolio containing multiple stocks (one-to-many).
Stocks → Portfolio: A stock can be held by multiple users (one-to-many).

Technologies Used:

- ✓ MySQL: The relational database management system used for storing and querying data.
- ✓ SQL Queries: For retrieving stock prices, transaction history, and portfolio details.
- ✓ Foreign Keys: Used to maintain data integrity across users, stocks, transactions, and portfolios.
- ✓ Stored Procedures: Used for complex queries such as calculating portfolio values, fetching transaction history, and retrieving the most traded stocks.

PROGRAM CODES:

```
-- TO CREATE ANGELONE STOCK TRADING PLATFORM DATABASE
CREATE DATABASE AngeloneDB;

-- TO ACTIVE ANGELONE APP DATABASE
USE AngeloneDB;

-- TO CREATE TABLE (ANGELONE USERS)
CREATE TABLE Users (
   user_id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100),
   email VARCHAR(100) UNIQUE,
```

```
    phone VARCHAR(15),
    password VARCHAR(255),
    balance DECIMAL(15, 2) DEFAULT 0.00,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);


-- TO CREATE TABLE (ANGELONE STOCKS)
CREATE TABLE Stocks (
    stock_id INT AUTO_INCREMENT PRIMARY KEY,
    symbol VARCHAR(10) UNIQUE,
    company_name VARCHAR(100),
    current_price DECIMAL(10, 2),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);



-- TO CREATE TABLE ORDERS (Buy/Sell Orders)
CREATE TABLE Orders (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    stock_id INT,
    order_type ENUM('buy', 'sell'),
    quantity INT,
    price DECIMAL(10, 2),
    order_status ENUM('pending', 'completed', 'cancelled') DEFAULT 'pending',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (stock_id) REFERENCES Stocks(stock_id)
);


-- TO CREATE TABLE TRANSACTIONS (Completed Buy/Sell Transactions)
CREATE TABLE Transactions (
    transaction_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    user_id INT,
    stock_id INT,
    quantity INT,
    price DECIMAL(10, 2),
    transaction_type ENUM('buy', 'sell'),
    transaction_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (stock_id) REFERENCES Stocks(stock_id)
);


-- TO CREATE TABLE PORTFOLIO (Tracks user’s stock holdings)
CREATE TABLE Portfolio (
    portfolio_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    stock_id INT,
    quantity INT,
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (stock_id) REFERENCES Stocks(stock_id)
```

```sql
);




------------------------------------------------------------------------------------------------------------------

-- TO INSERT VALUES INTO TABLE (USERS)
INSERT INTO Users (name, email, phone, password, balance)
VALUES ('Arun kumar', 'Arunkumar@gmail.com.com', '9876543210', 'password123', 10000.00),
              ('Bala Krishna', 'BalaKrishna@gmail.com', '9876543211', 'password456', 5000.00),
     ('Chandru sankar', 'Chandrusankar@gmail.com', '9847365789', 'password176', 15000.00),
     ('Ajith kumar', 'Ajithkumar@gmail.com', '9882663571', 'password256', 7000.00),
     ('Prakash kumar', 'Prakashkumar@gmail.com', '9837445678', 'password006', 9000.00),
     ('Sathyamoorthy', 'Sathyamoorthy@gmail.com', '988230294', 'password316', 12000.00),
     ('Kannan Baskar', 'kannanBaskar@gmail.com', '9444814453', 'password161', 18000.00),
     ('Praveen kumar', 'Praveen kumar@gmail.com', '9008234465', 'password666', 5000.00),
     ('Jaya narayanan', 'Jayanarayanan@gmail.com', '9008743567', 'password116', 11000.00),
     ('Hari Krishna', 'HariKrishna@gmail.com', '9962836284', 'password446', 15000.00);


 -- TO INSERT VALUES INTO TABLE (STOCKS)
 INSERT INTO Stocks (symbol, company_name, current_price)
 VALUES ('TCS', 'Tata Consultancy Services Ltd.', '3910.15'),
                        ('HDFC', 'HDFC Bank Ltd.', '1697.70'),
                        ('AIRTEL', 'Bharti Airtel Ltd.', '1774.54'),
                        ('ICIC', 'ICICI Bank Ltd.', '1234.90'),
                        ('INFS', 'Infosys Ltd.', '1894.14'),
                        ('ITC', 'ITC Ltd.', '460.64'),
                        ('HCL', 'HCL Technologies Ltd.', '1703.54'),
                        ('MARSU', 'Maruti Suzuki India Ltd.', '12671.30'),
                        ('ZOM', 'Zomato Ltd.', '224.87'),
                        ('TAMOTOS', 'Tata Motors Ltd.', '694.54');




-- TO INSERT VALUES INTO TABLE (ORDERS)
INSERT INTO Orders (user_id, stock_id, order_type, quantity, price, order_status)
 VALUES (1, 101, 'buy', 50, 7500.50, 'pending'),
                        (2, 102, 'buy', 10, 2800.75, 'pending'),
      (3, 103, 'buy', 30, 5000.75, 'pending'),
      (4, 104, 'buy', 40, 6800.75, 'pending'),
      (5, 105, 'buy', 40, 6800.75, 'pending'),
      (6, 106, 'buy', 30, 5000.75, 'pending'),
      (7, 107, 'buy', 20, 3800.75, 'pending'),
      (8, 108, 'buy', 50, 7500.75, 'pending'),
      (9, 109, 'buy', 30, 5000.75, 'pending'),
      (10,110,'buy',10, 2800.75, 'pending');


-- TO INSERT VALUES INTO TABLE (TRANSACTIONS)
INSERT INTO Transactions (transaction_id,order_id,user_id, stock_id, quantity, price,
transaction_type)
VALUES  (1001,1,1, 101, 50, 7500.50, 'buy'),
                        (1002,2,2, 102, 10, 2800.75, 'buy'),
```

```
                            (1003,3,3, 103, 30, 5000.75, 'buy'),
                            (1004,4,4, 101, 40, 6800.75, 'sell'),
                            (1005,5,5, 104, 40, 6800.75, 'buy'),
                            (1006,6,6, 102, 30, 5000.75, 'sell'),
                            (1007,7,7, 105, 20, 3800.75, 'buy'),
                            (1008,8,8, 106, 50, 7500.75, 'buy'),
                            (1009,9,9, 101, 30, 5000.75, 'sell'),
                            (1010,10,10, 105, 10, 2800.75, 'sell');



-- TO INSERT VALUES INTO TABLE (PORTFOLIO)
INSERT INTO Portfolio (user_id, stock_id, quantity)
VALUES (1, 101, 50),
                (2, 102, 10),
                (3, 103, 30),
                (4, 104, 40),
                (5, 105, 40),
                (6, 106, 30),
                (7, 107, 20),
                (8, 108, 50),
                (9, 109, 30),
                (10, 110, 10);




------------------------------------------------------------------------------------------------------------------------

/*
ANGELONE DATABASE STORED PROCEDURES
*/

-- QUESTION 1. To Fetch the Top 5 Stocks Based on Current Price ?
DELIMITER $$

CREATE PROCEDURE GetTop5Stocks()
BEGIN
SELECT stock_id, symbol, company_name, current_price
FROM Stocks
ORDER BY current_price DESC
LIMIT 5;
END $$

DELIMITER ;



-- To call this stored procedure and retrieve the top 5 stocks:
CALL GetTop5Stocks();
```

-- QUESTION 2.To Get Users Who Bought the Most Popular Stock ?

```
DELIMITER $$

CREATE PROCEDURE GetUsersWhoBoughtMostStock()
BEGIN
SELECT DISTINCT user_id
FROM Transactions
WHERE stock_id = (SELECT stock_id
FROM Transactions
WHERE transaction_type = 'buy'
GROUP BY stock_id
ORDER BY SUM(quantity) DESC
LIMIT 1);
END $$

DELIMITER ;


-- To call this stored procedure and To retrieve Users Who Bought the Most Popular Stock ?
CALL GetUsersWhoBoughtMostStock();
```

-- Question 3.To Get User with Highest Portfolio Value ?

```
DELIMITER $$

CREATE PROCEDURE GetTopUserPortfolio()
BEGIN
SELECT user_id, balance
FROM users
WHERE balance = (SELECT MAX(balance) FROM Users);
END $$

DELIMITER ;


-- To call this stored procedure and To retrieve User with Highest Portfolio Value ?
CALL GetTopUserPortfolio();
```

-- Question 4. To Find Stocks Above Average Price ?

```
DELIMITER $$

CREATE PROCEDURE GetStocksAboveAveragePrice()
BEGIN
   SELECT stock_id, symbol, company_name, current_price
   FROM Stocks
   WHERE current_price > (SELECT AVG(current_price) FROM Stocks);
END $$

DELIMITER ;


-- To call this stored procedure and To retrieve Stocks Above Average Price  ?
CALL GetStocksAboveAveragePrice();
```

-- QUESTION 5.Top 3 Low Price Stocks Based on Current Price ?

```
DELIMITER $$

CREATE PROCEDURE GetTop3LowPriceStocks()
BEGIN
SELECT stock_id, symbol, company_name, current_price
FROM Stocks
ORDER BY current_price
LIMIT 3;
END $$

DELIMITER ;

-- To call this stored procedure and retrieve the top 3 low price stocks:
Call GetTop3LowPriceStocks();
```

-- Question 6. to Fetch a User's Transaction History within a specified date range ?

```
 DELIMITER $$

CREATE PROCEDURE GetUserTransactionHistory(
IN p_user_id INT,
IN p_start_date DATE,
IN p_end_date DATE
)
BEGIN
SELECT
transaction_id,
stock_id,
transaction_type,
quantity,
price,
transaction_time
FROM Transactions
WHERE user_id = p_user_id
AND DATE(transaction_time) BETWEEN p_start_date AND p_end_date
ORDER BY transaction_time DESC;
END $$

DELIMITER ;


-- To call this stored procedure and To retrieve a User's Transaction History within a specified date
range ?
CALL GetUserTransactionHistory(1, '2024-02-10' , '2024-02-14');
```

Project done by : Janakiraman B
