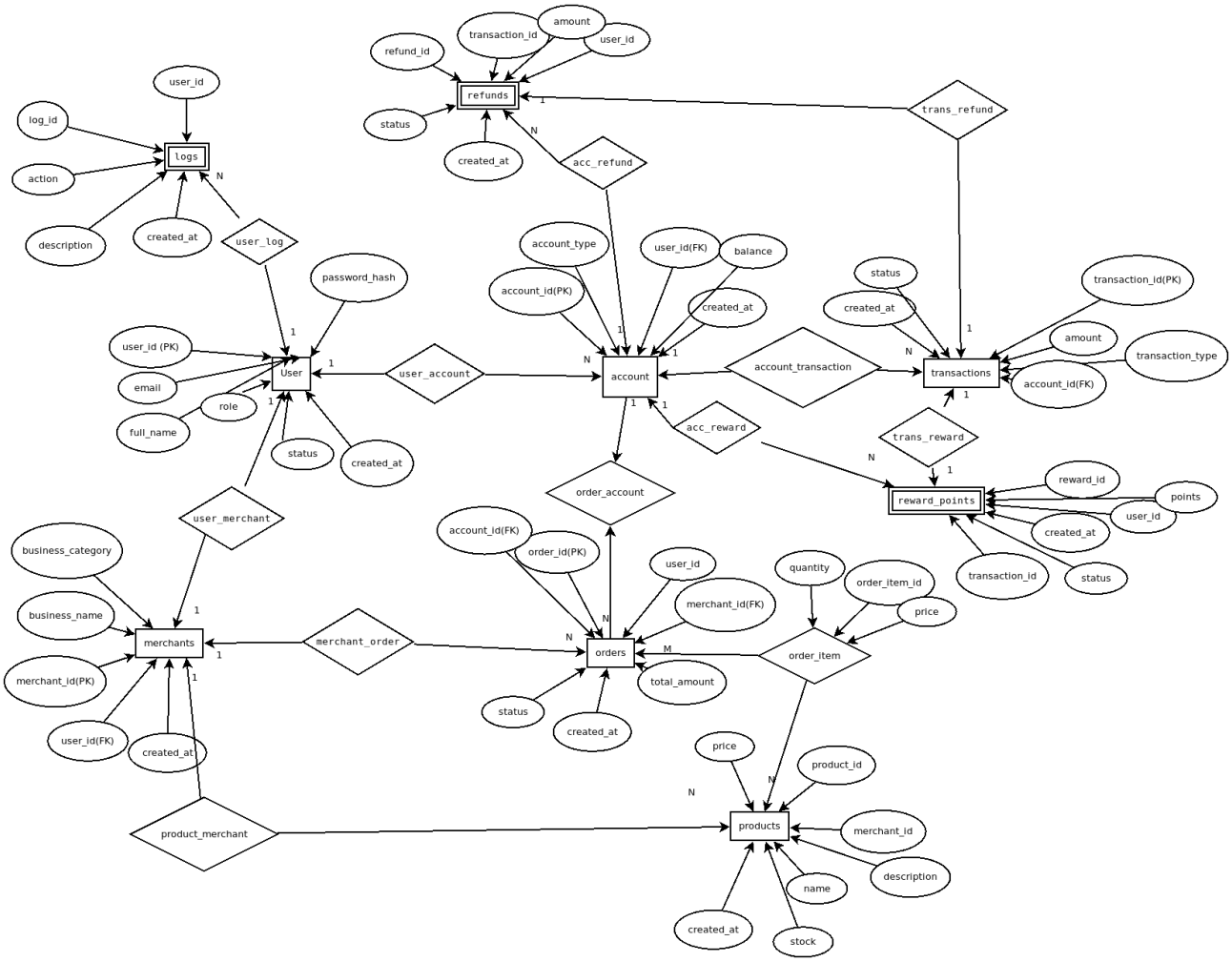
Midterm Project Report: Simulated E-Wallet and Rewards System for E-Commerce

1. ER Model

The **Entity-Relationship (ER) Model** for our e-wallet system consists of the following major entities:

- **Users** (Customers, Merchants, Admins, Support Staff)
- **Accounts** (Each user can have multiple accounts)
- **Transactions** (Top-ups, Purchases, Withdrawals, Refunds, Reward Redemptions)
- **Merchants** (Sellers registered on the platform)
- **Products** (Items listed for sale by merchants)
- **Orders** (Customer purchases)
- **Order_Items** (Many-to-Many relationship between Orders and Products)
- **Reward Points** (Earned and redeemed by accounts)
- **Refunds** (Processed for canceled transactions)
- **Logs** (Records user activities)



2. All tables and relational model

```
CREATE TABLE Users (
    user_id SERIAL PRIMARY KEY,
    full_name VARCHAR(255) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    role ENUM('customer', 'admin', 'merchant', 'support') NOT NULL,
    status ENUM('active', 'blocked') DEFAULT 'active',
    created_at TIMESTAMP DEFAULT now()
);

CREATE TABLE Accounts (
    account_id SERIAL PRIMARY KEY,
    user_id INT NOT NULL,
    account_type ENUM('wallet', 'merchant') NOT NULL,
    balance DECIMAL(10,2) DEFAULT 0.00,
    created_at TIMESTAMP DEFAULT now(),
    FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE
);

CREATE TABLE Transactions (
    transaction_id SERIAL PRIMARY KEY,
    account_id INT NOT NULL,
    transaction_type ENUM('top-up', 'purchase', 'withdrawal', 'refund',
'reward_redemption') NOT NULL,
    amount DECIMAL(10,2) NOT NULL,
    status ENUM('pending', 'completed', 'failed', 'reversed') DEFAULT 'pending',
    created_at TIMESTAMP DEFAULT now(),
    FOREIGN KEY (account_id) REFERENCES Accounts(account_id) ON DELETE CASCADE
);

CREATE TABLE Merchants (
    merchant_id SERIAL PRIMARY KEY,
    user_id INT UNIQUE NOT NULL,
    business_name VARCHAR(255) NOT NULL,
    business_category VARCHAR(255),
    created_at TIMESTAMP DEFAULT now(),
    FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE
);

CREATE TABLE Products (
    product_id SERIAL PRIMARY KEY,
    merchant_id INT NOT NULL,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    price DECIMAL(10,2) NOT NULL,
    stock INT NOT NULL,
    created_at TIMESTAMP DEFAULT now(),
    FOREIGN KEY (merchant_id) REFERENCES Merchants(merchant_id) ON DELETE
CASCADE
);

CREATE TABLE Orders (
    order_id SERIAL PRIMARY KEY,
    account_id INT NOT NULL,
    merchant_id INT NOT NULL,
    total_amount DECIMAL(10,2) NOT NULL,
    status ENUM('pending', 'completed', 'canceled', 'refunded') DEFAULT
'pending',
    created_at TIMESTAMP DEFAULT now(),
    FOREIGN KEY (account_id) REFERENCES Accounts(account_id) ON DELETE CASCADE,
```

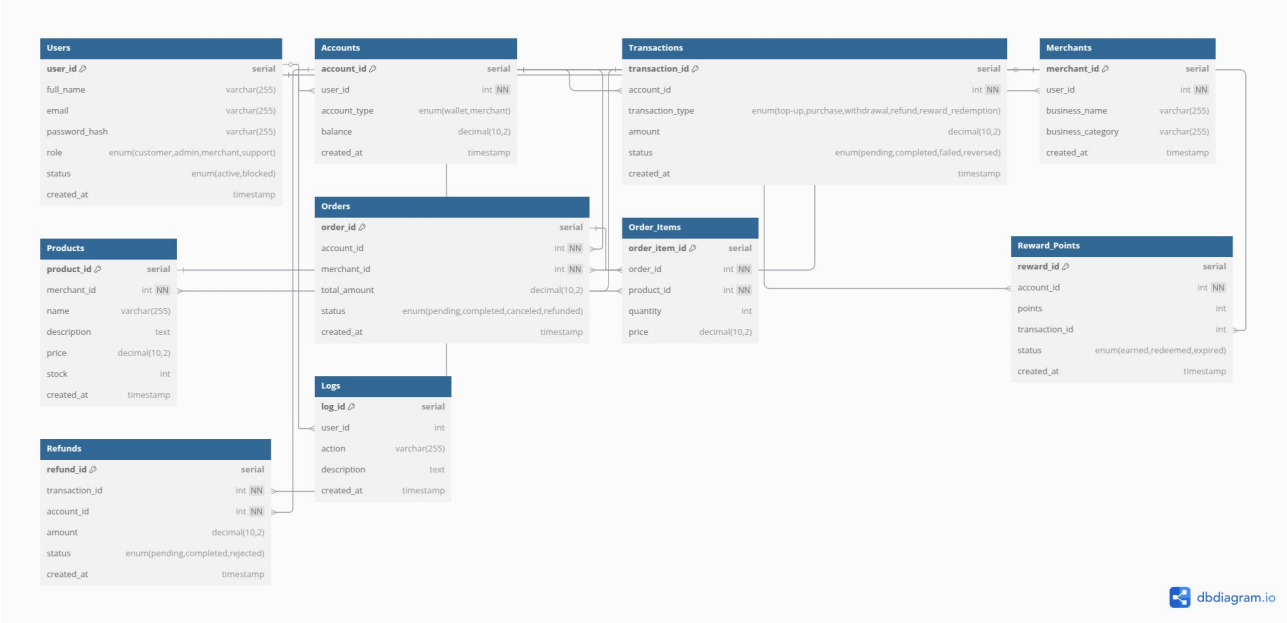
```
    FOREIGN KEY (merchant_id) REFERENCES Merchants(merchant_id) ON DELETE
CASCADE
);

CREATE TABLE Order_Items (
    order_item_id SERIAL PRIMARY KEY,
    order_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (order_id) REFERENCES Orders(order_id) ON DELETE CASCADE,
    FOREIGN KEY (product_id) REFERENCES Products(product_id) ON DELETE CASCADE
);

CREATE TABLE Reward_Points (
    reward_id SERIAL PRIMARY KEY,
    account_id INT NOT NULL,
    points INT NOT NULL,
    transaction_id INT,
    status ENUM('earned', 'redeemed', 'expired') DEFAULT 'earned',
    created_at TIMESTAMP DEFAULT now(),
    FOREIGN KEY (account_id) REFERENCES Accounts(account_id) ON DELETE CASCADE,
    FOREIGN KEY (transaction_id) REFERENCES Transactions(transaction_id) ON
DELETE SET NULL
);

CREATE TABLE Refunds (
    refund_id SERIAL PRIMARY KEY,
    transaction_id INT NOT NULL,
    account_id INT NOT NULL,
    amount DECIMAL(10,2) NOT NULL,
    status ENUM('pending', 'completed', 'rejected') DEFAULT 'pending',
    created_at TIMESTAMP DEFAULT now(),
    FOREIGN KEY (transaction_id) REFERENCES Transactions(transaction_id) ON
DELETE CASCADE,
    FOREIGN KEY (account_id) REFERENCES Accounts(account_id) ON DELETE CASCADE
);

CREATE TABLE Logs (
    log_id SERIAL PRIMARY KEY,
    user_id INT,
    action VARCHAR(255) NOT NULL,
    description TEXT,
    created_at TIMESTAMP DEFAULT now(),
    FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE SET NULL
);
```



3. Sample Data

Users Table

```
INSERT INTO Users (full_name, email, password_hash, role, status) VALUES
('Alice Johnson', 'alice@example.com', 'hashed_password1', 'customer',
'active'),
('Bob Merchant', 'bob@store.com', 'hashed_password2', 'merchant', 'active');
```

Accounts Table

```
INSERT INTO Accounts (user_id, account_type, balance) VALUES
(1, 'wallet', 500.00),
(2, 'merchant', 0.00);
```

Transactions Table

```
INSERT INTO Transactions (account_id, transaction_type, amount, status) VALUES
(1, 'top-up', 500.00, 'completed'),
(1, 'purchase', 200.00, 'completed');
```

4. Functions Implementation

Function 1: Reward Points Calculation for Purchases

```
CREATE OR REPLACE FUNCTION grant_reward_points()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.transaction_type = 'purchase' THEN
        INSERT INTO Reward_Points (account_id, points, transaction_id, status)
        VALUES (NEW.account_id, FLOOR(NEW.amount / 100) * 5, NEW.transaction_id,
'earned');
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER trigger_grant_rewards
AFTER INSERT ON Transactions
FOR EACH ROW
EXECUTE FUNCTION grant_reward_points();
```

Function 2: Cashback Redemption

```
CREATE OR REPLACE FUNCTION redeem_rewards(accountId INT, pointsToRedeem INT)
RETURNS VOID AS $$
DECLARE cashbackAmount DECIMAL(10,2);
BEGIN
    cashbackAmount := pointsToRedeem / 10;
    UPDATE Accounts
    SET balance = balance + cashbackAmount
    WHERE account_id = accountId;
END;
$$ LANGUAGE plpgsql;
```

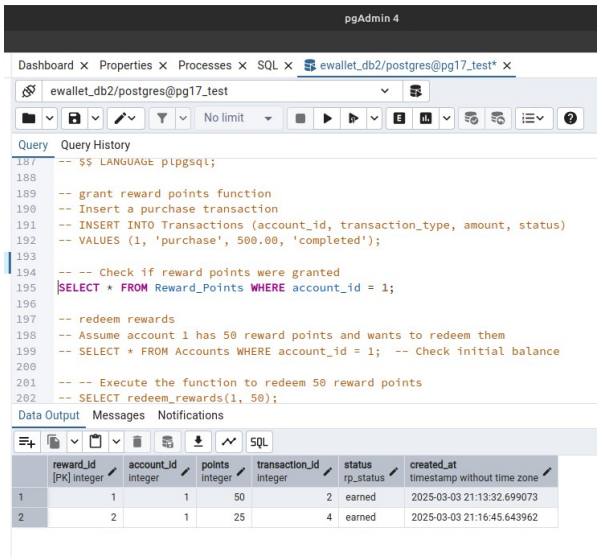
Function 3: Merchant Withdrawals

```
CREATE OR REPLACE FUNCTION process_withdrawal(merchantAccountId INT, amount
DECIMAL(10,2))
RETURNS VOID AS $$
BEGIN
    UPDATE Accounts
    SET balance = balance - amount
    WHERE account_id = merchantAccountId;
END;
$$ LANGUAGE plpgsql;
```

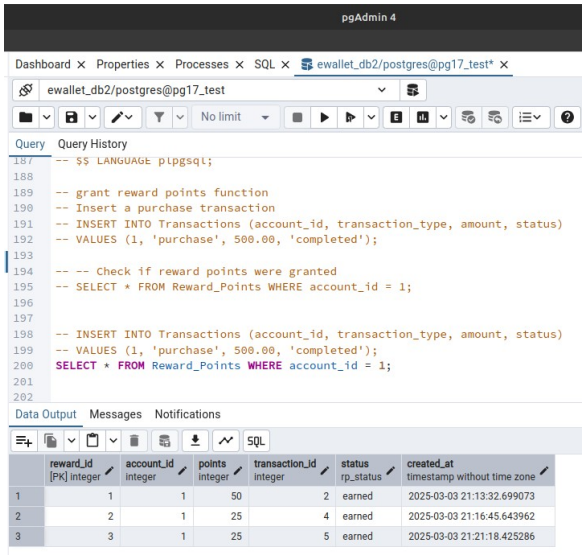
5. Screenshots of Function Execution

(Insert screenshots of reward points being credited, cashback redemption, and merchant withdrawal)

reward points before transaction

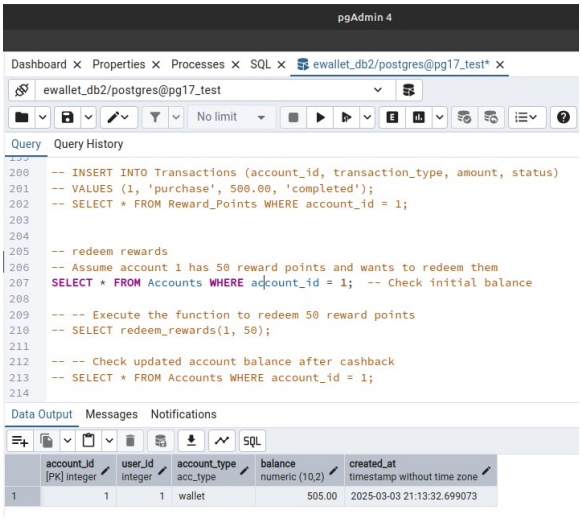


reward points after transaction

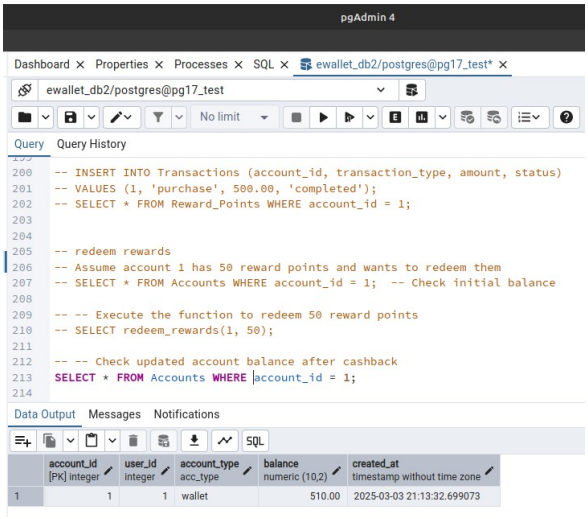


1. The first function is a trigger which executes automatically when a transaction is made. It grants reward points to user in the transaction. From the above output, it is clear that after a transaction, a reward is granted.

accounts before redeeming rewards

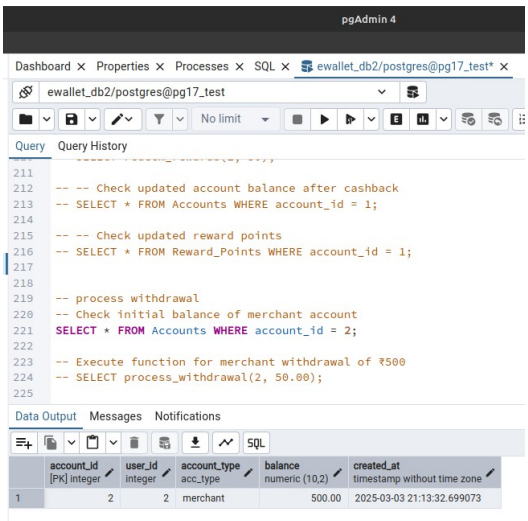


accounts after redeeming rewards

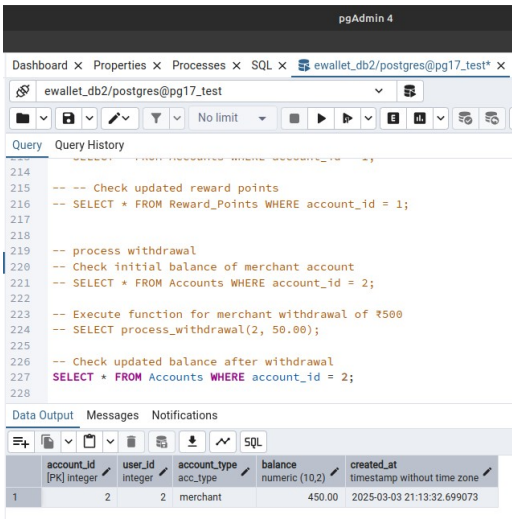


2. The second function is redeem rewards function. Here it makes a user to redeem the corresponding reward points obtained for his transactions. From the outputs, it is clear that a reward of 5 units is granted to user with id = 1.

accounts before withdrawal



accounts after withdrawal



3. The third function is process withdrawal function. This is the core part of a withdrawal. i.e, it decreases the balance of the withdrawing user's account by the requested amount. From the outputs, it is clear that 50 units is deducted from the account with id = 2.

6. Conclusion

This midterm report presents the **ER model, relational model, database schema, constraints, sample data, and stored functions** required for the simulated e-wallet and rewards system. The **reward points automation, cashback redemption, and merchant withdrawal functions** ensure smooth financial operations within the platform. The screenshots provided validate the proper execution of the implemented functions.

Future Improvements:

- Implementing stored procedures for batch processing.
- Improving performance with indexing strategies.
- Adding many more functions.
- Ensuring logging is done properly and implement a more general rewards scheme.

Project team:
Bhogaraju Shanmukha Sri Krishna (112201013)
Gajula Sri Siva Sai Shashank (112201014)
Ponnavolu Chakradhar Reddy (112201022)