Discount Rules Engine

Problem Statement

A rule engine is needed to evaluate order transactions for discounts based on various conditions such as product expiry, product type, special events, and quantity purchased in a large retail store.

Qualifying Rules and Calculation Rules

1. **Less than 30 days remaining for product expiry**:
    - Discounts based on remaining days:
        - 29 days: 1% discount
        - 28 days: 2% discount
        - ...and so on.
2. **Cheese and Wine products on sale**:
    - Cheese: 10% discount
    - Wine: 5% discount
3. **Products sold on March 23rd**:
    - Special discount: 50%
4. **More than 5 units of the same product**:
    - 6-9 units: 5% discount
    - 10-14 units: 7% discount
    - More than 15 units: 10% discount

Technical Approach

1. **Functional Programming**: Core logic implemented using pure functions with an emphasis on immutability and predictability.
2. **Data Processing**:

- Read order transactions from a CSV file.
- Apply qualifying rules to determine discounts.
- Calculate final prices after applying discounts.

3. **Database Interaction**:
   - Insert processed data into a database table.
   - Utilize JDBC for database connectivity.
4. **Logging**:
   - Log engine events in a file with timestamps and log levels.
5. **Documentation**:
   - Ensure code is well-commented and adheres to functional programming principles.
   - Emphasize readability, clarity, and self-explanation of the codebase.

Implemented Discount Rules

1. **Day Remaining Qualifying Rule**:
   - Function: `qualify_expiryDays`
   - Description: Checks if the remaining days for a product to expire is less than 30.
2. **On-Sale Products Qualifying Rule**:
   - Function: `qualify_category`
   - Description: Identifies whether a product is eligible for discount based on being a wine or cheese product.
3. **Special Discount for Products Sold on 23rd of March**:
   - Function: `qualify_23March`
   - Description: Applies a special discount if a product is sold on the 23rd of March.
4. **Quantity of Products Sold**:
   - Function: `qualify_quantity`

- Description: Determines if the quantity sold exceeds 5 units for the same product.

Database Interaction

- **Database Connection**:
  - Utilizes Oracle JDBC driver for database connectivity.
  - Inserts processed data into the `orders` table, including details such as order date, expiry date, product category, quantity, unit price, channel, payment method, discount, and total price.

Logging Mechanisms

- **Logging Engine Rule Interactions**:
  - Function: `log_event`
  - Description: Writes information about engine rule interactions into the `logs.log` file.
  - Structure: Logs timestamp, log level, and message for each rule interaction event.

Usage

1. **Data Input**:
   - Ensure that Scala and necessary dependencies are installed.
   - Place the order data CSV file (`TRX1000.csv`) in the `src/main/resources` directory.
   - Update database connection details in the `DiscountsEngine` object.
   - Run the `DiscountsEngine` object to execute the application.

- Check the logs for information about the execution process.

2. **Configuration**:
   - Update the database connection details (URL, username, password) in the application.
   - Create table as follows:

```
CREATE TABLE orders (
    order_date DATE,
    expiry_date DATE,
    days_to_expiry NUMBER,
    product_category VARCHAR2(100),
    product_name VARCHAR2(100),
    quantity NUMBER,
    unit_price NUMBER,
    channel VARCHAR2(50),
    payment_method VARCHAR2(50),
    discount NUMBER,
    total_due NUMBER
);
```

3. **Execution**:
   - Run the application to process order transactions, calculate discounts, and insert data into the database.
4. **Verification**:
   - Check the `orders` table in the database for inserted records with calculated discounts and total prices.
   - Review the `logs.log` file for logged engine rule interactions and any error messages.

```
Timestamp: 2024-05-08T18:30:54.089Z     LogLevel: Info/Debug    Message: Program Started
Timestamp: 2024-05-08T18:30:54.203Z     LogLevel: Info/Debug    Message: Started processing an order
Timestamp: 2024-05-08T18:30:54.206Z     LogLevel: Info  Message: Successful qualification
Timestamp: 2024-05-08T18:30:54.208Z     LogLevel: Info  Message: calculated discount: 0.05
Timestamp: 2024-05-08T18:30:54.209Z     LogLevel: Info  Message: Successful qualification
Timestamp: 2024-05-08T18:30:54.209Z     LogLevel: Info  Message: calculated discount: 0.05
Timestamp: 2024-05-08T18:30:54.209Z     LogLevel: Info  Message: Failed qualification
Timestamp: 2024-05-08T18:30:54.211Z     LogLevel: Info  Message: Failed qualification
Timestamp: 2024-05-08T18:30:54.211Z     LogLevel: Info  Message: Failed qualification
Timestamp: 2024-05-08T18:30:54.211Z     LogLevel: Info  Message: Successful qualification
Timestamp: 2024-05-08T18:30:54.211Z     LogLevel: Info  Message: calculated discount: 0.05
Timestamp: 2024-05-08T18:30:54.241Z     LogLevel: Info/Debug    Message: Ended processing an order
Timestamp: 2024-05-08T18:30:54.241Z     LogLevel: Debug Message: Time taken to process order: 38ms
Timestamp: 2024-05-08T18:30:54.243Z     LogLevel: Info/Debug    Message: Started processing an order
```

....

```
Timestamp: 2024-05-08T18:30:56.802Z     LogLevel: Debug Message: Successfully Opened database connection
Timestamp: 2024-05-08T18:30:56.997Z     LogLevel: Info  Message: Successfully inserted into database
Timestamp: 2024-05-08T18:30:56.997Z     LogLevel: Debug Message: Closed database connection
Timestamp: 2024-05-08T18:30:56.997Z     LogLevel: Info/Debug    Message: Program Finished
Timestamp: 2024-05-08T18:30:56.997Z     LogLevel: Debug Message: Program took 2908ms to run
```

Dependencies

- Java JDBC for database connection.

Future Improvements

- Enhance error handling and logging.
- Implement unit tests for critical components.
- Allow customization of discount rules and database connection details through configuration files.
- Optimize database interaction for better performance.
