Scala Discounts Rule Engine

By: Israa Ali Ahmed

Information Technology Institute

Problem Statement

A huge retail store wants a rule engine that qualifies orders' transactions to discounts based on a set of qualifying rules. And automatically calculates the proper discount based on some calculation rules as follows:

QUALIFYING RULES	CALCULATION RULE
Less than 30 days remaining for product expiry	• 29 days: 1% discount • 28 days: 2% discount ...and so on.
Cheese and Wine products	• Cheese: 10% discount • Wine: 5% discount
Products sold on March 23rd:	• Special discount: 50%
More than 5 units of the same product	• 6-9 units: 5% discount • 10-14 units: 7% discount • More than 15 units: 10% discount
App Sales	• quantity: 1, 2, 3, 4, 5 -> discount 5% • quantity 6, 7, 8, 9, 10 -> discount 10% • quantity 11, 12, 13, 14, 15 -> discount 15% ...and so on.
Visa Card Usage	• 5% discount

Programming Approach

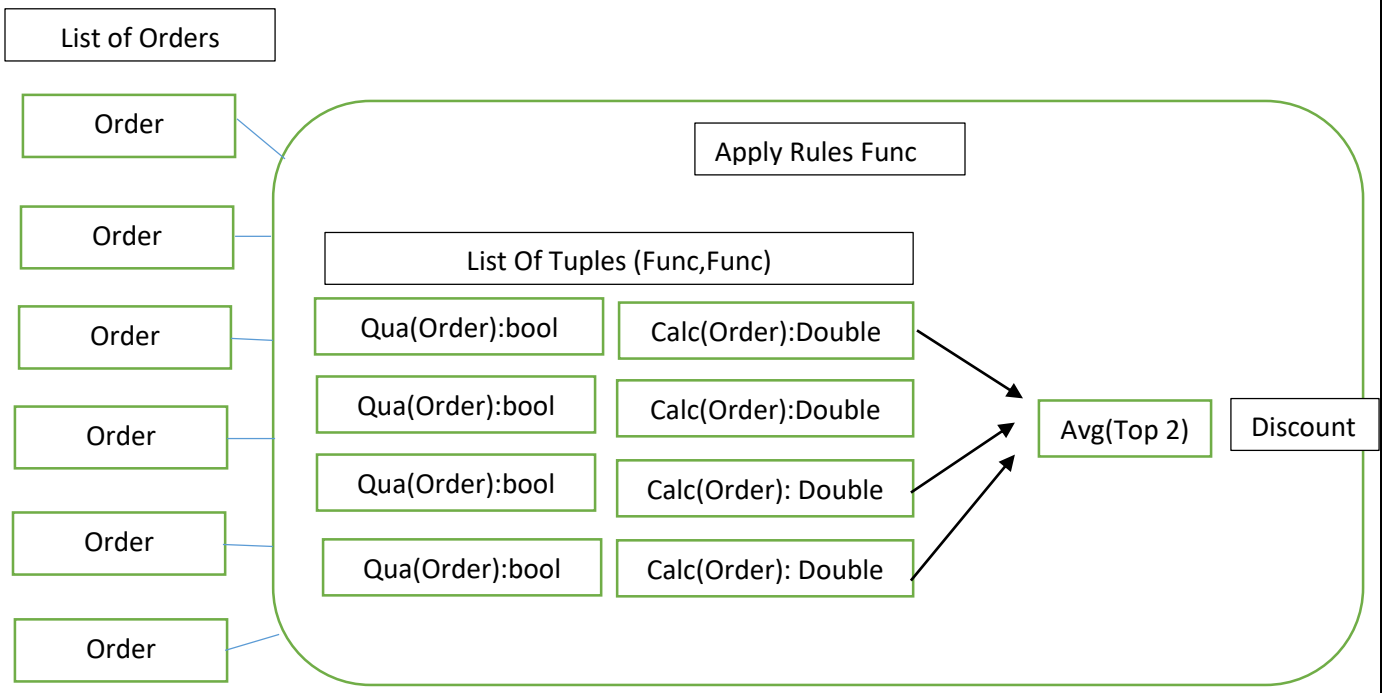
The core logic is written in a pure functional manner.

Functional programming offers several advantages over imperative programming paradigms:

- Immutability
- Pure Functions
- Higher-order Functions
- Function Composition

Information Technology Institute

Problem Solving Approach



Coding Steps

1) Read lines from the orders file and drop the header

This is the data format

```
timestamp,product_name,expiry_date,quantity,unit_price,channel,payment_method
2023-04-18T18:18:40Z,Wine - White Pinot Grigio,2023-06-10,6,122.47,Store,Visa
2023-04-09T20:17:37Z,Pepper - Red Thai,2023-05-22,5,23.18,Store,Cash
2023-05-09T13:28:27Z,Wine - Chardonnay Mondavi,2023-08-04,7,189.74,Store,Cash
2023-04-09T02:56:43Z,External Supplier,2023-05-23,14,9.92,App,Visa
2023-03-27T17:55:57Z,Beans - Black Bean Preserved,2023-05-28,13,145.85,Store,Cash
2023-04-27T04:41:13Z,Beef - Inside Round,2023-07-18,9,247.62,Store,Visa
```

342

2) Creating two Functions for each rule : qua_ruleName and cal_ruleName

1) qua_ruleName : to check if the order meets the specified rule
2) cal_ruleName: to calculate the discount based on the associated rule

```scala
//Qualifications and Calculations Functions that represent the rules
def quaExpireDate(order: String): Boolean = {...}
def calExpireDate(order: String): Double = {...}
// Define a function to check if an order contains Cheese or Wine
def quaCheeseAndWine(order: String): Boolean = {...}
// Define a function to calculate the discount based on Cheese or Wine
def calCheeseAndWine(order: String): Double = {...}
// Define a function to check if an order's transaction date is on March 23rd
def quaMarch(order: String): Boolean = order.substring(6, 10) == "03-23"
// Define a function to calculate the discount for orders on March 23rd
def calMarch(order: String): Double = 50
// Define a function to check if an order's quantity is greater than 5
def quaQty(order: String): Boolean = order.split(",")(3).toInt > 5
// Define a function to calculate the discount based on quantity
def calQty(order: String): Double = {...}
// Define a function to check if an order was made through the App channel
def quaChannel(order: String): Boolean = order.split(",")(5) == "App"
// Define a function to calculate the discount based on the channel
def calChannel(order: String): Double = math.ceil(order.split(",")(3).toInt.toDouble / 5) * 5
// Define a function to check if an order's payment method is Visa
def quaPayMethod(order: String): Boolean = order.split(",")(6) == "Visa"
// Define a function to calculate the discount for Visa payments
def calPayMethod(order: String): Double = 5
```

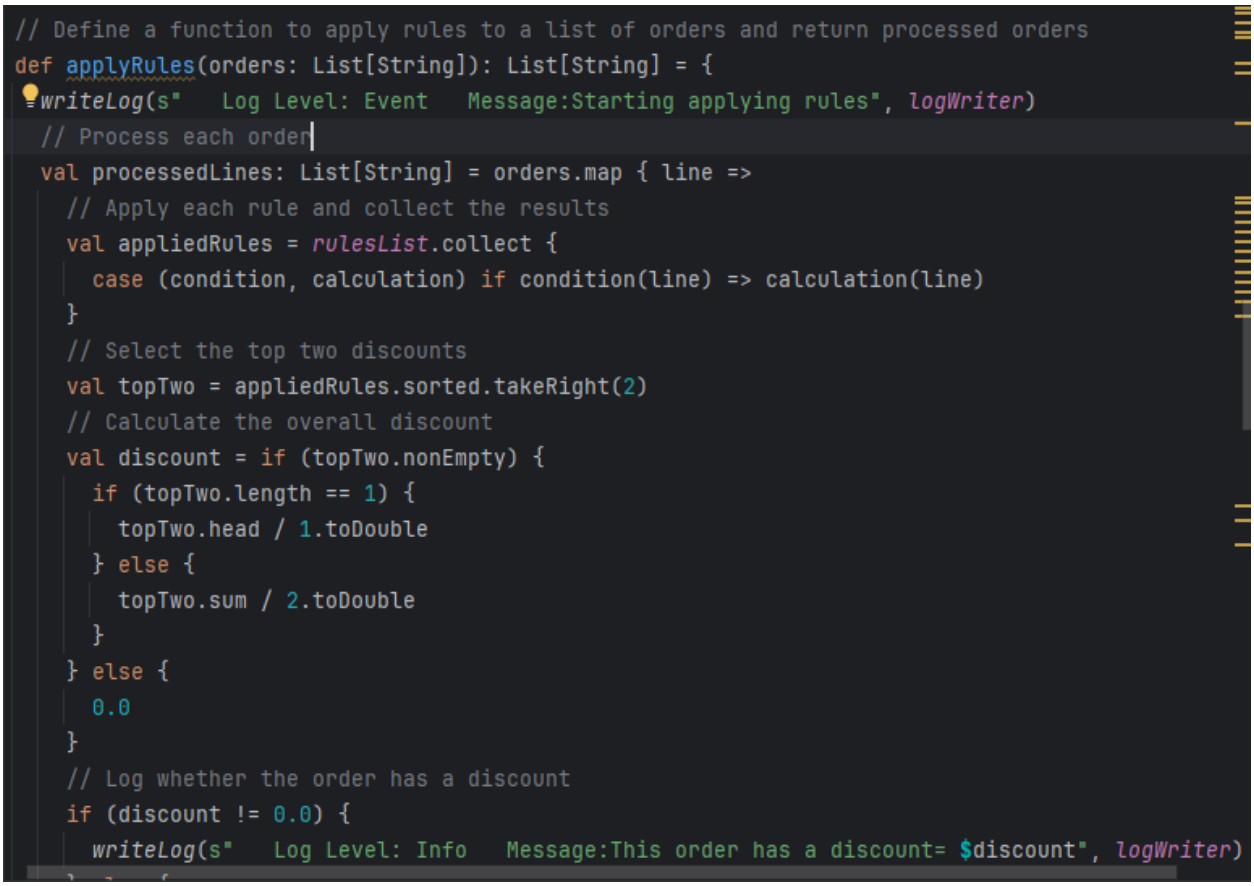
3) Define a list of rules as a tuple of condition and calculation functions

```scala
val rulesList: List[(String => Boolean, String => Double)] = List(
  (quaExpireDate, calExpireDate),
  (quaCheeseAndWine, calCheeseAndWine),
  (quaMarch, calMarch),
  (quaQty, calQty),
  (quaChannel, calChannel),
  (quaPayMethod, calPayMethod)
)
```

4) Define a function to apply the rules to a list of orders and return processed orders:

- **applyRules** function takes the lines as a list of string
- iterates over each line
- applies the rules list for it
- returns the corresponding discount if a rule applied
- returns the discounts as a list of double

- Then it will take the top 2 discounts and calculates the avg of the two and returns the discount
- Calculates the final price
- Returns a list of processed lines



```
// Define a function to apply rules to a list of orders and return processed orders
def applyRules(orders: List[String]): List[String] = {
  writeLog(s"   Log Level: Event   Message:Starting applying rules", logWriter)
  // Process each order
  val processedLines: List[String] = orders.map { line =>
    // Apply each rule and collect the results
    val appliedRules = rulesList.collect {
      case (condition, calculation) if condition(line) => calculation(line)
    }
    // Select the top two discounts
    val topTwo = appliedRules.sorted.takeRight(2)
    // Calculate the overall discount
    val discount = if (topTwo.nonEmpty) {
      if (topTwo.length == 1) {
        topTwo.head / 1.toDouble
      } else {
        topTwo.sum / 2.toDouble
      }
    } else {
      0.0
    }
    // Log whether the order has a discount
    if (discount != 0.0) {
      writeLog(s"   Log Level: Info   Message:This order has a discount= $discount", logWriter)
```

5) **Processed Orders: call the apply rules function and return the result**
6) **Define writeToDatabase function that will take the processed orders and insert them in a table at oracle data base**
   - Load the Oracle JDBC driver
   - Establish a connection to the database
   - Prepare the SQL insert statement
   - Create a prepared statement for batch insertion
   - Iterate over the processed orders and add them to the batch
   - Execute the batch insertion
   - Close the prepared statement and the database connection

```
try {
  // Iterate over the processed orders and add them to the batch
  data.foreach { order =>
    val orderData = order.split(",")
    val orderDate = orderData(0).substring(0,10)
    val expiryDate = orderData(2)
    val productName = orderData(1)
    val quantity = orderData(3).toInt
    val unitPrice = orderData(4).toDouble
    val channel = orderData(5)
    val paymentMethod = orderData(6)
    val discount = orderData(7).toDouble
    val finalPrice = orderData(8).toDouble
    preparedStatement.setString(1, orderDate)
    preparedStatement.setString(2, expiryDate)
    preparedStatement.setString(3, productName)
    preparedStatement.setInt(4, quantity)
    preparedStatement.setDouble(5, unitPrice)
    preparedStatement.setString(6, channel)
    preparedStatement.setString(7, paymentMethod)
    preparedStatement.setDouble(8, discount)
    preparedStatement.setDouble(9, finalPrice)
    preparedStatement.addBatch()
  }
```

Oracle table of the processed orders ordered by final price descinding

ORDER_DATE	EXPIRY_DATE	PRODUCT_NAME	QUANTITY	UNIT_PRICE	CHANNEL	PAYMENT_METHOD	DISCOUNT	FINAL_PRICE
2023-04-27	2023-06-21	Container - Hngd Cll Blk 7x7x3	17	247.07	Store	Visa	7.5	3885.18
2023-04-13	2023-06-23	Broom - Angled	17	243.87	Store	Cash	10	3731.21
2023-03-24	2023-06-04	Langers - Ruby Red Grapfruit	17	238.93	Store	Cash	10	3655.63
2023-04-05	2023-06-27	Table Cloth 72x144 White	17	229.19	Store	Visa	7.5	3604.01
2023-05-07	2023-06-01	Radish - Pickled	16	243.4	Store	Visa	7.5	3602.32
2023-03-18	2023-05-06	Wonton Wrappers	17	245.21	App	Visa	15	3543.28
2023-04-27	2023-07-06	Rum - White Gg White	17	230.88	Store	Cash	10	3532.46
2023-05-08	2023-07-16	Wooden Mop Handle	17	221.12	Store	Visa	7.5	3477.11
2023-03-16	2023-03-31	Lemons	17	247.42	App	Visa	17.5	3470.07
2023-03-22	2023-04-10	Canadian Emmenthal	16	240.98	Store	Cash	10.5	3450.83
2023-03-27	2023-05-14	Wine - Baron De Rothschild	16	230.14	Store	Cash	7.5	3406.07
2023-04-02	2023-06-05	Arizona - Green Tea	16	229.55	Store	Visa	7.5	3397.34
2023-04-04	2023-06-24	Sole - Iqf	17	235.06	App	Visa	15	3396.62
2023-05-12	2023-07-11	Red Currants	17	208.22	Store	Visa	7.5	3274.26

7) **Define a writeToCSV Function that will write the processed orders to a csv file**
8) **Define a function to write log messages during the running of the rule engine and write to a csv file**

TimeStamp: 2024-05-13T14:25:53.252	Log Level: Event	Message:Starting app
TimeStamp: 2024-05-13T14:25:53.253	Log Level: Event	Message:Opening orders.csv
TimeStamp: 2024-05-13T14:25:53.357	Log Level: Event	Message:Starting applying rules
TimeStamp: 2024-05-13T14:25:53.427	Log Level: Info	Message:This order has a discount= 5.0
TimeStamp: 2024-05-13T14:25:53.427	Log Level: Info	Message:This order has no discounts
TimeStamp: 2024-05-13T14:25:53.428	Log Level: Info	Message:This order has a discount= 5.0
TimeStamp: 2024-05-13T14:25:53.428	Log Level: Info	Message:This order has a discount= 11.0
TimeStamp: 2024-05-13T14:25:53.429	Log Level: Info	Message:This order has a discount= 7.0
TimeStamp: 2024-05-13T14:25:53.429	Log Level: Info	Message:This order has a discount= 5.0
TimeStamp: 2024-05-13T14:25:53.430	Log Level: Info	Message:This order has a discount= 4.5
TimeStamp: 2024-05-13T14:25:53.431	Log Level: Info	Message:This order has a discount= 7.5
TimeStamp: 2024-05-13T14:25:53.431	Log Level: Info	Message:This order has a discount= 16.5
TimeStamp: 2024-05-13T14:25:53.432	Log Level: Info	Message:This order has a discount= 7.5
TimeStamp: 2024-05-13T14:25:53.432	Log Level: Info	Message:This order has no discounts
TimeStamp: 2024-05-13T14:25:53.432	Log Level: Info	Message:This order has a discount= 5.0
TimeStamp: 2024-05-13T14:25:53.433	Log Level: Info	Message:This order has a discount= 8.0
TimeStamp: 2024-05-13T14:25:53.433	Log Level: Info	Message:This order has a discount= 10.0
TimeStamp: 2024-05-13T14:25:53.433	Log Level: Info	Message:This order has a discount= 7.0
TimeStamp: 2024-05-13T14:25:53.434	Log Level: Info	Message:This order has no discounts
TimeStamp: 2024-05-13T14:25:53.435	Log Level: Info	Message:This order has a discount= 7.0
TimeStamp: 2024-05-13T14:25:53.435	Log Level: Info	Message:This order has a discount= 7.5
TimeStamp: 2024-05-13T14:25:53.436	Log Level: Info	Message:This order has a discount= 5.0
TimeStamp: 2024-05-13T14:25:53.436	Log Level: Info	Message:This order has a discount= 15.0