BỘ GIÁO DỤC & ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM

KHOA CÔNG NGHỆ THÔNG TIN

--📖--

BÁO CÁO CUỐI KỲ

MÔN HỌC: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Khám Phá Sức Mạnh của LINQ trong C# và Biểu Thức Lambda trong Java

Giảng Viên: Nguyễn Minh Đạo

Sinh viên: Nguyễn Vũ Bảo - 23110079

TP.HCM , THÁNG 12 NĂM 2024

MỤC LỤC

A. Tổng quan về công nghệ

1.1. LINQ to SQL

LINQ to SQL (Language Integrated Query to SQL) là một công nghệ trong .NET Framework, cho phép ánh xạ các bảng trong cơ sở dữ liệu SQL Server thành các đối tượng trong C#. Công nghệ này giúp lập trình viên thực hiện truy vấn dữ liệu bằng cú pháp C#, thay vì viết mã SQL truyền thống.

Ưu điểm:

- Cú pháp trực quan, dễ đọc.
- Tích hợp chặt chẽ với Visual Studio, giúp lập trình viên dễ dàng quản lý mô hình cơ sở dữ liệu.
- Tự động hóa quá trình ánh xạ giữa bảng cơ sở dữ liệu và đối tượng.

1.2. Java Stream API

Java Stream API, được giới thiệu từ Java 8, là một công cụ mạnh mẽ cho việc xử lý tập dữ liệu (collections) theo cách lập trình hàm. Thay vì xử lý dữ liệu theo kiểu lặp thủ công (vòng lặp for hoặc while), Stream API cho phép mô tả các thao tác như lọc, sắp xếp, ánh xạ, và giảm thiểu dữ liệu một cách ngắn gọn.

Ưu điểm:

- Cung cấp các thao tác xử lý dữ liệu ngắn gọn và mạnh mẽ.
- Kết hợp tốt với Lambda Expression để tăng tính linh hoạt.

1.3. Lambda Expression

Lambda Expression là một biểu thức đại diện cho các phương thức vô danh (anonymous methods). Biểu thức này giúp mã nguồn ngắn gọn, dễ đọc, và hỗ trợ tốt trong các thao tác xử lý dữ liệu theo phong cách lập trình hàm (functional programming).

1.4. JDBC

JDBC (Java Database Connectivity) là một API trong Java, cung cấp một cách chuẩn để kết nối và thao tác với cơ sở dữ liệu. JDBC cho phép các ứng dụng Java thực hiện các tác vụ như gửi truy vấn SQL, cập nhật dữ liệu, và lấy kết quả từ cơ sở dữ liệu. Nó hỗ trợ nhiều hệ quản trị cơ sở dữ liệu khác nhau như MySQL, Oracle, PostgreSQL thông qua các driver tương ứng. Các thành phần chính của JDBC bao gồm DriverManager, Connection, Statement, và ResultSet. JDBC giúp đơn giản hóa quá trình làm việc với cơ sở dữ

liệu, đồng thời cung cấp khả năng mở rộng và hiệu suất cao cho các ứng dụng Java.

1.5.SQL Server

SQL Server là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) được phát triển bởi Microsoft. Đây là một công cụ mạnh mẽ, phù hợp với các ứng dụng quy mô từ nhỏ đến lớn.

Lý do sử dụng trong dự án:

- Cung cấp khả năng tương thích tốt với LINQ to SQL trong ứng dụng C#.
- Khả năng xử lý dữ liệu lớn, phù hợp với các bài toán về quản lý thông tin.

1.6.MySQL

MySQL là một hệ quản trị cơ sở dữ liệu mã nguồn mở, được phát triển bởi Oracle Corporation. MySQL nổi tiếng với tốc độ xử lý nhanh và tính dễ dàng trong triển khai.

Lý do sử dụng trong dự án:

- Tích hợp linh hoạt với Java thông qua thư viện JDBC.
- Là lựa chọn phổ biến trong các dự án học thuật và công nghiệp, giúp dễ dàng triển khai ứng dụng đa nền tảng.

B. MÃ NGUỒN CƠ SỞ DỮ LIỆU TRONG SQL Server và MySQL

File mã nguồn MySQL : databaseMysql.txt

File mã nguồn SQL SERVER: sqlServer.txt

C. Các thực thể và thuộc tính

Khách hàng (CUSTOMER)

- **Mô tả**: Lưu thông tin khách hàng và điểm tích lũy để cung cấp ưu đãi và khuyến mãi.
- **Thuộc tính**:

Tên thuộc tính	Kiểu dữ liệu	Giải thích
MAKH	INT	Khóa chính, định danh duy nhất cho khách hàng.
HoTen	VARCHAR(100)	Họ và tên của khách hàng.
SDT	VARCHAR(15)	Số điện thoại liên lạc của khách hàng.
DiemTichLuy	INT	Tổng điểm tích lũy của khách hàng.

Nhân viên (EMPLOYEE)

- **Mô tả**: Quản lý thông tin nhân viên, bao gồm các nhân viên phục vụ và nhân viên quản lý.
- **Thuộc tính**:

Tên thuộc tính	Kiểu dữ liệu	Giải thích
MANV	INT	Khóa chính, định danh duy nhất cho nhân viên.
HoTen	VARCHAR(100)	Họ và tên của nhân viên.
ChucVu	ENUM	Vai trò của nhân viên: Nhân viên bán hàng, Quản lý.
SDT	VARCHAR(15)	Số điện thoại liên lạc của nhân viên.
NgayBatDau	DATE	Ngày bắt đầu làm việc.

Sản phẩm (PRODUCT)

- **Mô tả**: Lưu thông tin các món ăn, bao gồm giá cả và tình trạng hiện tại.
- **Thuộc tính**:

Tên thuộc tính	Kiểu dữ liệu	Giải thích
MASP	INT	Khóa chính, định danh duy nhất cho sản phẩm.

Tên thuộc tính	Kiểu dữ liệu	Giải thích
TenSP	VARCHAR(100)	Tên món ăn hoặc sản phẩm.
DonGia	DECIMAL(10, 2)	Giá của sản phẩm.
TinhTrang	ENUM	Trạng thái sản phẩm: Có sẵn, Hết hàng.
HinhAnh	VARCHAR(255)	Đường dẫn đến ảnh minh họa sản phẩm.

Chi tiết hóa đơn (INVOICE_DETAIL)

- **Mô tả**: Lưu chi tiết các sản phẩm trong từng hóa đơn.
- **Thuộc tính**:

Tên thuộc tính	Kiểu dữ liệu	Giải thích
MAHD	INT	Khóa ngoại, tham chiếu đến hóa đơn.
MASP	INT	Khóa ngoại, tham chiếu đến sản phẩm.
SoLuong	INT	Số lượng sản phẩm được mua.
ThanhTien	DECIMAL(10, 2)	Thành tiền (SoLuong × DonGia).

Hóa đơn (INVOICE)

- **Mô tả**: Lưu thông tin chi tiết về các hóa đơn thanh toán.
- **Thuộc tính**:

Tên thuộc tính	Kiểu dữ liệu	Giải thích
MAHD	INT	Khóa chính, định danh duy nhất cho hóa đơn.
MAKH	INT	Khóa ngoại, tham chiếu đến khách hàng đã tạo hóa đơn.
MANV	INT	Khóa ngoại, tham chiếu đến nhân viên xử lý hóa đơn.
ThoiGianTT	DATETIME	Thời gian thanh toán hóa đơn.
TongTien	DECIMAL(10, 2)	Tổng số tiền trong hóa đơn.

Tên thuộc tính	Kiểu dữ liệu	Giải thích
HinhThucTT	ENUM	Hình thức thanh toán: Tiền mặt, Thẻ, Chuyển khoản.
SoTienNhan	DECIMAL(10, 2)	Số tiền khách hàng thanh toán.

Khuyến mãi (PROMOTION)

- **Mô tả**: Lưu thông tin về các chương trình khuyến mãi.
- **Thuộc tính**:

Tên thuộc tính	Kiểu dữ liệu	Giải thích
MAKM	INT	Khóa chính, định danh duy nhất cho khuyến mãi.
ThoiGianHL	DATETIME	Thời gian hiệu lực của khuyến mãi.
DieuKien	TEXT	Điều kiện áp dụng khuyến mãi.
MucGiamGia	DECIMAL(10, 2)	Mức giảm giá.

Khách hàng có khuyến mãi(Customer_Promotion)

Tên thuộc tính	Kiểu dữ liệu	Giải thích
MAKH	INT	Mã khách hàng
MAKM	INT	Mã khuyến mãi

PRIMARY KEY (MAKH, MAKM): Mỗi khách hàng có thể tham gia nhiều chương trình khuyến mãi

2. Tóm tắt mối liên kết

2.1. Khách hàng và hóa đơn (CUSTOMER ↔ INVOICE):

- Một khách hàng có thể tạo nhiều hóa đơn.
- Một hóa đơn chỉ thuộc về một khách hàng.

2.2. Nhân viên và hóa đơn (EMPLOYEE ↔ INVOICE):

- Một nhân viên có thể xử lý nhiều hóa đơn.
- Một hóa đơn được xử lý bởi một nhân viên.

2.3. Hóa đơn và chi tiết hóa đơn (INVOICE ↔ INVOICE_DETAIL):

- Một hóa đơn có thể chứa nhiều sản phẩm.

- Một sản phẩm có thể thuộc nhiều hóa đơn thông qua bảng trung gian.

2.4. Khuyến mãi và khách hàng (PROMOTION ↔ CUSTOMER):

- Một khuyến mãi có thể áp dụng cho nhiều khách hàng.
- Một khách hàng có thể được áp dụng nhiều chương trình khuyến mãi.

D. Package “Database” : Để kết nối database .

Lớp JDBCUtil giúp:

- Đơn giản hóa việc kết nối đến cơ sở dữ liệu MySQL.
- Hỗ trợ quản lý tài nguyên như đóng kết nối và rollback khi xảy ra lỗi.
- Cung cấp tiện ích để in thông tin cơ sở dữ liệu.

1) Khai báo thuộc tính :

```
// URL kết nối đến cơ sở dữ liệu MySQL
private static final String URL = "jdbc:mysql://localhost:3306/FastFoodManagement";
// Tên người dùng đăng nhập MySQL
private static final String USER = "root";
// Mật khẩu đăng nhập MySQL
private static final String PASSWORD = "";
```

URL : Chuỗi kết nối tới cơ sở dữ liệu MySQL .

URL = "jdbc:mysql://localhost:3306/FastFoodManagement";

+) Localhost: Máy chủ đang chậy MySQL (nội bộ máy tính)

+) 3306: Cổng mặc định của MySQL.

+) FastFoodManagement: Tên cơ sở dữ liệu cần kết nối.

USER và PASSWORD: Thông tin đăng nhập vào MySQL. Ở đây, user là root và password trống.

2)Phương thức getConnection

```
public static Connection getConnection() throws SQLException {
    try {
        // Tải lớp Driver JDBC của MySQL
        // Class.forName() dùng để nạp driver vào bộ nhớ để JDBC sử dụng
     // "com.mysql.cj.jdbc.Driver" là driver hiện đại cho MySQL (JDBC 8+)
        Class.forName("com.mysql.cj.jdbc.Driver");
    } catch (ClassNotFoundException e) {
 // Nếu không tìm thấy driver, ném ngoại lệ SQLException
 // Thông báo lỗi "Unable to load MySQL driver" kèm thông tin lỗi gốc (e)
        throw new SQLException("Unable to load MySQL driver", e);
    }
    // Tạo và trả về đối tượng Connection bằng DriverManager
    // Dùng URL, USER, PASSWORD để kết nối đến cơ sở dữ liệu
    return DriverManager.getConnection(URL, USER, PASSWORD);
}
```

Chức năng: Tạo một kết nối tới cơ sở dữ liệu MySQL

Cách hoạt động :

Class.forName("com.mysql.cj.jdbc.Driver"): Tải driver JDBC của MySQL (phiên bản mới).

- **Lưu ý**: Trong Java hiện đại (JDBC 4.0+), bước này có thể không cần vì driver tự động được tải khi có trong classpath.

DriverManager.getConnection(URL, USER, PASSWORD): Tạo kết nối dựa trên thông tin URL, user, và password.

Ngoại lệ: Nếu driver không được tải hoặc kết nối thất bại, sẽ ném ra SQLException.

3)Phương thức closeConnection

```
public static void closeConnection(Connection conn) {
   // Kiểm tra kết nối có khác null không
   if (conn != null) {
      try {
         // Đóng kết nối nếu nó đang mở
         conn.close();
      } catch (SQLException e) {
         // Bắt lỗi nếu xảy ra lỗi trong quá trình đóng kết nối
         // In chi tiết lỗi ra màn hình để tiện debug
         e.printStackTrace();
      }
  }}
```

Chức năng: Đóng kết nối khi không còn sử dụng để giải phóng tài nguyên.
Cách hoạt động:

- Kiểm tra conn khác null.
- Gọi conn.close() để đóng kết nối.
- Nếu có lỗi, in lỗi ra màn hình (debug).

4)Phương thức rollbackConnection

```
public static void rollbackConnection(Connection conn) {
   // Kiểm tra nếu kết nối không phải null
   if (conn != null) {
      try {
      // Thực hiện rollback (hoàn tác) các thay đổi chưa được commit
         conn.rollback();
      } catch (SQLException e) {
         // Bắt lỗi nếu xảy ra vấn đề trong quá trình rollback
         // In chi tiết lỗi ra màn hình để debug
```

```
            e.printStackTrace();
        }
    }
}
```

Chức năng: Hoàn tác (rollback) các thay đổi chưa được commit trên kết nối trong trường hợp xảy ra lỗi.

Cách hoạt động:

- Kiểm tra conn khác null.
- Gọi conn.rollback() để hoàn tác các thay đổi.
- Nếu có lỗi khi rollback, in lỗi ra màn hình.

5) **Phương thức** printInfo

```
public static void printInfo(Connection c) {
    try {
     // Kiểm tra nếu kết nối (Connection) không phải null
        if (c != null) {
         // Lấy metadata (siêu dữ liệu) của cơ sở dữ liệu từ kết nối
            DatabaseMetaData mtdt = c.getMetaData();


          // In tên hệ quản trị cơ sở dữ liệu (VD: MySQL, PostgreSQL)
            System.out.println(mtdt.getDatabaseProductName());


            // In phiên bản của hệ quản trị cơ sở dữ liệu (VD: 8.0.30)
            System.out.println(mtdt.getDatabaseProductVersion());
        }
    } catch (SQLException e) {
        // Bắt lỗi nếu có vấn đề trong quá trình truy cập metadata
        // In chi tiết lỗi ra màn hình để debug
        e.printStackTrace();
    }
}
```

Chức năng: In thông tin cơ sở dữ liệu (tên và phiên bản) ra màn hình.
Cách hoạt động:

- Kiểm tra c khác null.

- Sử dụng DatabaseMetaData (lấy từ c.getMetaData()) để lấy thông tin:

    +) getDatabaseProductName(): Lấy tên hệ quản trị cơ sở dữ liệu (VD: "MySQL").

    +) getDatabaseProductVersion(): Lấy phiên bản của hệ quản trị (VD: "8.0.30").

- In thông tin này ra màn hình.

E. Package DAO :quản lý việc truy xuất và thao tác với cơ sở dữ liệu

Class GenericDAO:

Dưới đây là một mô tả tổng quát về các phương thức còn lại trong GenericDAO:

- getAllCustomers :Lấy tất cả thông tin khách hàng từ bảng customers
- getAllEmployees: Lấy tất cả thông tin nhân viên từ bảng employee.
- getAllInvoices: Lấy tất cả thông tin hóa đơn từ bảng invoice.
- getAllProducts: Lấy tất cả thông tin sản phẩm từ bảng product.
- getAllInvoiceDetails: Lấy tất cả chi tiết hóa đơn từ bảng invoice_detail.
- getAllPromotions: Lấy tất cả các khuyến mãi từ bảng promotion.
- getAllCustomerPromotion: Lấy tất cả các mối quan hệ giữa khách hàng và khuyến mãi từ bảng customer_promotion.

Dùng để thao tác với cơ sở dữ liệu thông qua các phương thức để lấy tất cả dữ liệu từ các bảng trong cơ sở dữ liệu. Mỗi phương thức trả về một danh sách (List) các đối tượng tương ứng với từng bảng. Dưới đây là **phân tích chi tiết** từng phần mã:

1) Khai báo thuộc tính và Constructor

```
private Connection connection;

// Constructor của lớp GenericDAO
public GenericDAO(Connection connection) {
    // Gán đối tượng Connection được truyền vào cho biến instance connection
    this.connection = connection;
}
```

Giải thích :

private Connection connection;:

Ý nghĩa: Biến connection là một thuộc tính của lớp GenericDAO, dùng để lưu đối tượng kết nối cơ sở dữ liệu (Connection).

Mục đích: Cho phép các phương thức trong lớp GenericDAO sử dụng kết nối này để thực hiện truy vấn SQL.

public GenericDAO(Connection connection):

Chức năng:

- Đây là constructor (hàm khởi tạo) của lớp GenericDAO.
- Hàm này được gọi khi tạo một đối tượng GenericDAO.
- Nó nhận một đối tượng Connection từ bên ngoài và gán vào thuộc tính connection của lớp.

this.connection = connection;:

Ý nghĩa:

- this.connection đề cập đến biến thuộc tính của lớp GenericDAO.
- connection là tham số được truyền vào từ bên ngoài.
- Lệnh này gán giá trị tham số connection cho biến thuộc tính connection của lớp.

2) Phương thức getAllCustomers() :

```
public List<Customer> getAllCustomers() throws SQLException {
    // Câu lệnh SQL để lấy tất cả các bản ghi từ bảng "customer"
    String query = "SELECT * FROM customer";

    // Danh sách để lưu các đối tượng Customer được lấy từ cơ sở dữ liệu
    List<Customer> customers = new ArrayList<Customer>();

    // Khối try-with-resources để tự động đóng PreparedStatement và ResultSet sau khi dùng
    try (PreparedStatement pstmt = connection.prepareStatement(query);
         ResultSet rs = pstmt.executeQuery()) {

        // Lặp qua từng bản ghi trong ResultSet
        while (rs.next()) {
            // Tạo đối tượng Customer từ dữ liệu của bản ghi hiện tại
            customers.add(new Customer(
                rs.getInt("MAKH"),       // Mã khách hàng
                rs.getString("HoTen"),   // Họ tên khách hàng
                rs.getString("SDT"),     // Số điện thoại
                rs.getInt("DiemTichLuy") // Điểm tích lũy
            ));
        }
    } catch (SQLException e) {
        // Xử lý ngoại lệ khi truy vấn
        e.printStackTrace(); // In thông tin lỗi ra console
        throw e;             // Ném lại ngoại lệ để xử lý ở lớp gọi
    }

    // Trả về danh sách các khách hàng
    return customers;
```

```
}
```

+) Mục đích của phương thức

- Phương thức này lấy tất cả các khách hàng từ bảng customer trong cơ sở dữ liệu.
- Kết quả trả về là một danh sách đối tượng Customer (List<Customer>).

+) Giải thích các phần trong mã

String query = "SELECT * FROM customer";

Câu lệnh SQL để lấy tất cả các bản ghi trong bảng customer.

List<Customer> customers = new ArrayList<Customer>();

Khởi tạo một danh sách trống để lưu các đối tượng Customer được lấy từ cơ sở dữ liệu.

try-with-resources

Sử dụng try (PreparedStatement ...; ResultSet ...) đảm bảo các tài nguyên (PreparedStatement và ResultSet) sẽ được tự động đóng sau khi sử dụng, giúp tránh rò rỉ tài nguyên.

connection.prepareStatement(query)

Tạo một đối tượng PreparedStatement để thực thi câu lệnh SQL.

pstmt.executeQuery()

Thực thi câu lệnh SQL và trả về một ResultSet chứa dữ liệu từ bảng customer.

while (rs.next())

Lặp qua từng hàng (row) trong ResultSet.

new Customer(...)

Tạo một đối tượng Customer từ dữ liệu trong ResultSet.

rs.getInt("MAKH"): Lấy cột MAKH (Mã khách hàng) từ bản ghi hiện tại.

rs.getString("HoTen"): Lấy cột HoTen (Họ tên khách hàng).

rs.getString("SDT"): Lấy cột SDT (Số điện thoại).

rs.getInt("DiemTichLuy"): Lấy cột DiemTichLuy (Điểm tích lũy).

customers.add(...)

Thêm đối tượng Customer vừa tạo vào danh sách customers.

catch (SQLException e)

Bắt ngoại lệ nếu xảy ra lỗi khi truy vấn dữ liệu.

e.printStackTrace(): In chi tiết lỗi ra console (có thể thay bằng logging).

throw e;: Ném lại ngoại lệ để các lớp gọi phương thức này xử lý tiếp.

return customers;

Trả về danh sách các đối tượng Customer.

3) **Tương tự với phương thức getAllCustomers** các phương thức còn lại như getAllEmployees, getAllInvoices, getAllProducts, getAllInvoiceDetails, getAllPromotions, và getAllCustomerPromotion cũng thực hiện các bước tương tự để truy vấn và lấy dữ liệu từ cơ sở dữ liệu.

Cụ thể:

- **Câu lệnh SQL** trong mỗi phương thức sẽ thay đổi tùy thuộc vào bảng dữ liệu cần truy vấn (ví dụ: employee, invoice, product, promotion, v.v.).
- **Kết quả truy vấn** từ ResultSet được sử dụng để tạo ra các đối tượng tương ứng (ví dụ: Employee, Invoice, Product, v.v.) và lưu vào danh sách (List).
- **Các ngoại lệ** (SQLException) được bắt và xử lý để đảm bảo mã hoạt động đúng, và trong trường hợp có lỗi, nó sẽ được ném lại (throw) để có thể xử lý tại các lớp gọi phương thức.

- **Dùng** try-with-resources để tự động đóng tài nguyên PreparedStatement và ResultSet, giúp tránh rò rỉ tài nguyên sau khi sử dụng.

F. Kiểm tra và xử lí dữ liệu với Java Lambdas

```
try (Connection conn = JDBCUtil.getConnection()) {
GenericDAO genericDAO = new GenericDAO(conn);
// Lấy danh sách tất cả khách hàng
customers = genericDAO.getAllCustomers();
// Lấy danh sách tất cả nhân viên
employees = genericDAO.getAllEmployees();
// Lấy danh sách tất cả hóa đơn
invoices = genericDAO.getAllInvoices();

// Lấy danh sách tất cả sản phẩm
products = genericDAO.getAllProducts();

// Lấy danh sách tất cả chi tiết hóa đơn
invoiceDetails = genericDAO.getAllInvoiceDetails();

// Lấy danh sách tất cả khuyến mãi
promotions = genericDAO.getAllPromotions();

customerPromotions = genericDAO.getAllCustomerPromotion();
} catch (SQLException e) {
e.printStackTrace();
}
```

Giải thích :

- customers = genericDAO.getAllCustomers();: Phương thức này sẽ trả về một danh sách (có thể là List<Customer>) chứa tất cả các khách hàng từ cơ sở dữ liệu.
- employees = genericDAO.getAllEmployees();: Phương thức này sẽ trả về một danh sách tất cả các nhân viên.
- invoices = genericDAO.getAllInvoices();: Phương thức này lấy tất cả các hóa đơn.
- products = genericDAO.getAllProducts();: Phương thức này lấy tất cả các sản phẩm từ cơ sở dữ liệu.
- invoiceDetails = genericDAO.getAllInvoiceDetails();: Phương thức này lấy tất cả chi tiết hóa đơn (các sản phẩm được mua trong mỗi hóa đơn).
- promotions = genericDAO.getAllPromotions();: Phương thức này lấy tất cả các chương trình khuyến mãi.
- customerPromotions = genericDAO.getAllCustomerPromotion();: Phương thức này lấy danh sách tất cả các khách hàng tham gia khuyến mãi.

1. Liệt kê tất cả các khách hàng.

SQL:

```
SELECT *
FROM CUSTOMER;
```

C#:

```
// 1.Liệt kê tất cả các khách hàng.
1 reference
public static void cau1()
{
    var customers = from c in dc1dc.CUSTOMERs
                    select c;
    Console.WriteLine("1. Liệt kê tất cả các khách hàng:");
    Console.WriteLine("-----------------------------------------------------");
    Console.WriteLine(String.Format("{0,-10}{1,-30}{2,-15}{3,-10}", "ID", "Họ Và Tên", "Số Điện Thoại", "Điểm tích lũy"));
    Console.WriteLine("-----------------------------------------------------");

    foreach (var customer in customers)
    {
        // Định dạng từng dòng thông tin khách hàng
        Console.WriteLine(String.Format("{0,-10}{1,-30}{2,-15}{3,-10}", customer.MAKH, customer.HoTen, customer.SDT, customer.DiemTichLuy));
    }
}
```

Giải thích:

· var customers: Khai báo biến customers để lưu kết quả truy vấn.

· from c in dc1dc.CUSTOMERs: Sử dụng Linq để lấy tất cả các khách hàng (CUSTOMERs) từ dc1dc.

· select c: Chọn tất cả các khách hàng.

Java:

```
// 1.Liệt kê tất cả các khách hàng.
public static void cau1() {
    System.out.printf("%-10s %-30s %-15s %-10s %n", "ID", "Họ Và Tên", "Số Điện Thoại" ,"Điểm tích lũy");
    System.out.println("-----------------------------------------------------");
    // In thông tin khách hàng
    customers.forEach(customer -> System.out.printf("%-10s %-30s %-15s %-10s %n",
            customer.getId(),
            customer.getName(),
            customer.getPhone(),
            customer.getPoints()));
}
```

Giải thích :

```
System.out.printf("%-10s %-30s %-15s %-10s %n", "ID", "Họ Và Tên", "Số Điện Thoại" ,"Điểm tích lũy");
```

Dòng này dùng để in ra tiêu đề cho bảng với các cột là ID, Họ và Tên, Số Điện Thoại và Điểm tích lũy. Cụ thể:

% -10s: In ra chuỗi có chiều rộng 10 ký tự, căn trái.

% -30s: In ra chuỗi có chiều rộng 30 ký tự, căn trái.

% -15s: In ra chuỗi có chiều rộng 15 ký tự, căn trái.

```
customers.forEach(customer ->

    System.out.printf("%-10s %-30s %-15s %-10s %n",
```

```
        customer.getId(), customer.getName(),

        customer.getPhone(),

        customer.getPoints()));
```

Dòng này lặp qua tất cả các khách hàng trong danh sách customers và in ra thông tin của mỗi khách hàng, bao gồm ID, tên, số điện thoại và điểm tích lũy.
customer.getId(), customer.getName(), customer.getPhone(), và customer.getPoints() là các phương thức để lấy thông tin của từng khách hàng.

Kết quả truy vấn:

```
1.Liệt kê tất cả các khách hàng:
-----------------------------------------------------
ID          Họ Và Tên                       Số Điện Thoại   Điểm tích lũy
-----------------------------------------------------
1           Nguyễn Hoàng Long               0901234567      120
2           Trần Thị Minh Hằng              0912345678      95
3           Phạm Quốc Anh                   0923456789      150
4           Hoàng Mai Phương                0934567890      80
5           Ngô Huy Hoàng                   0945678901      40
6           Lê Ngọc Tú                      0956789012      10
7           Vũ Thị Thu Hương                0967890123      55
8           Nguyễn Thanh Phong              0978901234      75
9           Đặng Hải Nam                    0989012345      30
10          Bùi Thanh Hải                   0990123456      200
11          Nguyễn Văn Hùng                 0911122233      65
12          Trần Thị Ngọc Hà                0912233445      85
13          Phạm Hoàng Anh                  0913344556      110
14          Lê Minh Tuấn                    0914455667      20
15          Ngô Bảo Trâm                    0915566778      90
16          Hoàng Thanh Hương               0916677889      105
17          Nguyễn Thu Hằng                 0917788990      125
18          Trần Quốc Bảo                   0918899001      70
19          Phạm Thị Hoa                    0919900112      50
```

2. Liệt kê các sản phẩm có giá lớn hơn 30,000 VND.

```
SELECT *
FROM product
WHERE  product.DonGia > 30000;
```

C#:

```
//2.Liệt kê các sản phẩm có giá lớn hơn 30,000 VND.
1 reference
public static void cau2(double price)
{
    Console.WriteLine("\n2. Liệt kê các sản phẩm có giá lớn hơn 30,000 VND:");
    Console.WriteLine("-----------------------------------------------------");
    Console.WriteLine(String.Format("{0,-10}{1,-30}{2,-15}{3,-10}", "Mã SP","Tên Sản Phẩm","Giá","Tình Trạng "));
    Console.WriteLine("-----------------------------------------------------");
    decimal priceDecimal = (decimal)price;
    dc1dc.PRODUCTs.Where(e => e.DonGia > priceDecimal)
        .ToList()
        .ForEach(p => Console.WriteLine(String.Format("{0,-10}{1,-30}{2,-15}{3,-10}", p.MASP, p.TenSP, p.DonGia, p.TinhTrang)));
}
```

Giải thích :

```
dc1dc.PRODUCTs.Where(e => e.DonGia > priceDecimal)
                        .ToList()
```

- dc1dc.PRODUCTs: Truy cập vào bảng PRODUCTs trong cơ sở dữ liệu qua đối tượng dc1dc.
- Where(e => e.DonGia > priceDecimal): Sử dụng phương thức Where để lọc các sản phẩm có đơn giá (DonGia) lớn hơn giá trị priceDecimal.
- .ToList()Chuyển kết quả lọc từ phương thức Where thành một danh sách các sản phẩm.

Java:

```
/2.Liệt kê các sản phẩm có giá lớn hơn 30,000 VND.
   public static void cau2(double price) {
       System.out.printf("%-10s %-30s %-15s %-10s %n", "Mã SP", "Tên Sản Phẩm", "Giá" ,"Tình trạng");
       System.out.println("-----------------------------------------------------");
       products.stream()
           .filter(e -> e.getPrice() > price)
           .collect(Collectors.toList())
           .forEach(e -> System.out.printf("%-10s %-30s %-15s %-10s %n",
                   e.getId(),
                   e.getName(),
                   e.getPrice() ,
                   e.getStatus()));
   }
```

Giải thích :

products.stream():

Chuyển danh sách sản phẩm products thành một stream, cho phép xử lý các phần tử theo kiểu tuần tự.

.filter(e -> e.getPrice() > price):

- Dùng phương thức filter để lọc các sản phẩm có giá (getPrice()) lớn hơn giá trị price được truyền vào.
- Mỗi sản phẩm (e) sẽ được kiểm tra, nếu điều kiện thỏa mãn (giá sản phẩm lớn hơn price), sản phẩm đó sẽ được giữ lại trong stream.

.collect(Collectors.toList()):

- Dùng phương thức collect để thu thập các sản phẩm lọc được thành một danh sách (List).

Kết quả truy vấn:

```
2. Liệt kê các sản phẩm có giá lớn hơn 30,000 VND:
-----------------------------------------------------
Mã SP      Tên Sản Phẩm                  Giá            Tình trạng
-----------------------------------------------------
1          Burger Bò                     50000.0        Có sẵn
2          Burger Gà                     45000.0        Có sẵn
3          Burger Phô Mai                55000.0        Có sẵn
6          Gà Rán Phần Nhỏ               35000.0        Có sẵn
7          Gà Rán Phần Lớn               70000.0        Có sẵn
8          Pizza Phô Mai                 120000.0       Có sẵn
9          Pizza Hải Sản                 150000.0       Có sẵn
10         Pizza Thập Cẩm                140000.0       Có sẵn
11         Mì Ý Sốt Bò                   85000.0        Có sẵn
12         Mì Ý Sốt Gà                   80000.0        Có sẵn
19         Salad Trộn                    40000.0        Có sẵn
20         Taco Gà                       45000.0        Có sẵn
21         Taco Bò                       50000.0        Hết hàng
22         Taco Phô Mai                  55000.0        Hết hàng
```

3. Hiển thị thông tin các hóa đơn cùng với tên khách hàng và tên nhân viên xử lý.

```
SELECT i.MAHD , c.HoTen , e.HoTen , i.TongTien , i.ThoiGianTT
FROM invoice AS i
JOIN customer AS c ON c.MAKH = i.MAKH
JOIN employee AS e ON e.MANV = i.MANV ;
```

C#:

```
//3.Hiển thị thông tin các hóa đơn cùng với tên khách hàng và tên nhân viên xử lý.
1 reference
public static void cau3()
{
    var result = from i in dc1dc.INVOICEs
                 join c in dc1dc.CUSTOMERs on i.MAKH equals c.MAKH
                 join e in dc1dc.EMPLOYEEs on i.MAKH equals e.MANV
                 select new
                 {
                     MAHD = i.MAHD,
                     HoTenKhachHang = c.HoTen,
                     HoTenNhanVien = e.HoTen,
                     TongTien = i.TongTien,
                     ThoiGianTT = i.ThoiGianTT
                 };
    Console.WriteLine("\n3. Hiển thị thông tin các hóa đơn cùng với tên khách hàng và tên nhân viên xử lý:");
    Console.WriteLine("-----------------------------------------------------------");
    Console.WriteLine("{0,-15} {1,-30} {2,-30} {3,-15} {4,-30}",
                "Mã Hóa Đơn", "Tên Khách Hàng", "Tên Nhân Viên", "Tổng Tiền", "Thời gian");
    Console.WriteLine("-----------------------------------------------------------");
    foreach (var item in result)
    {
        Console.WriteLine("{0,-15} {1,-30} {2,-30} {3,-15:F2} {4,-30}",
        item.MAHD,
        item.HoTenKhachHang ?? "Unknown",
        item.HoTenNhanVien ?? "Unknown",
        item.TongTien,
        item.ThoiGianTT);

    }
}
```

Giải thích :

Biến result:

```
var result = from i in dc1dc.INVOICEs
```

- Khai báo biến result để lưu trữ kết quả của truy vấn LINQ.
- có nghĩa là chúng ta bắt đầu truy vấn từ bảng INVOICEs và đặt tên biến i cho từng bản ghi trong bảng này.

Thực hiện phép nối giữa các bảng:

```
join c in dc1dc.CUSTOMERs on i.MAKH equals c.MAKH:
```

Thực hiện phép nối (join) giữa bảng INVOICEs và bảng CUSTOMERs dựa trên cột MAKH (Mã Khách Hàng). Chỉ những bản ghi có MAKH khớp nhau giữa hai bảng sẽ được kết hợp lại.

```
join e in dc1dc.EMPLOYEEs on i.MAKH equals e.MANV:
```

Thực hiện phép nối giữa bảng INVOICEs và bảng EMPLOYEEs dựa trên cột MAKH của bảng INVOICEs và cột MANV (Mã Nhân Viên) của bảng EMPLOYEEs.

Chọn thông tin cần thiết:

select new { ... }:

Tạo ra một đối tượng ẩn danh mới chứa các cột được chọn từ kết quả nối.Bao gồm các thông tin:

MAHD = i.MAHD: Mã hóa đơn.
HoTenKhachHang = c.HoTen: Họ tên khách hàng.
HoTenNhanVien = e.HoTen: Họ tên nhân viên.
TongTien = i.TongTien: Tổng tiền của hóa đơn.
ThoiGianTT = i.ThoiGianTT: Thời gian thanh toán của hóa đơn

Java:

```
//  Hiển thị thông tin các hóa đơn cùng với tên khách hàng và tên nhân viên xử lý.
    public static void cau3() {
        System.out.printf("%-15s %-30s %-30s  %-15s %-30s %n ", |
                "Mã Hóa Đơn","Tên Khách Hàng","Tên Nhân Viên","Tổng Tiền","Thời gian");
        System.out.println("-------------------------------------------------------------");
        invoices.forEach(invoice -> {
            Optional<String> customerName = customers.stream()
                    .filter(customer ->
                        customer.getId() == invoice.getCustomerId())
                                                .findFirst()
                                                .map(Customer::getName);
            Optional<String> employeeName = employees.stream()
                    .filter(employee ->
                        employee.getId() == invoice.getEmployeeId())
                                                .findFirst()
                                                .map(Employee::getName);

            System.out.printf("%-15s %-30s %-30s %-15.2f %-30s %n",
                    invoice.getId(),
                    customerName.orElse("Unknown"),
                    employeeName.orElse("Unknown"),
                    invoice.getTotalAmount(),
                    invoice.getPaymentTime());
        });
    }
```

Giải thích:

Optional<String> customerName = customers.stream().filter(customer -> customer.getId() == invoice.getCustomerId()).findFirst().map(Customer::getName);:

Tìm khách hàng có ID trùng với customerId trong hóa đơn, nếu tìm thấy thì lấy tên khách hàng, nếu không thì trả về Optional.empty().

Optional<String> employeeName = employees.stream().filter(employee -> employee.getId() == invoice.getEmployeeId()).findFirst().map(Employee::getName);:

Tìm nhân viên có ID trùng với employeeId trong hóa đơn, nếu tìm thấy thì lấy tên nhân viên, nếu không thì trả về Optional.empty().

Kết quả truy vấn:

```
3. Hiển thị thông tin các hóa đơn cùng với tên khách hàng và tên nhân viên xử lý:
-----------------------------------------------------
Mã Hóa Đơn      Tên Khách Hàng              Tên Nhân Viên               Tổng Tiền       Thời gian
 -----------------------------------------------------
1               Nguyễn Hoàng Long           Nguyễn Văn An               150000.00       2023-01-01 10:00:00.0
2               Trần Thị Minh Hằng          Lê Thị Hương                200000.00       2023-01-02 11:30:00.0
3               Phạm Quốc Anh               Trần Quốc Bảo               120000.00       2023-01-03 12:45:00.0
4               Hoàng Mai Phương            Phạm Hoàng Long             180000.00       2023-01-04 09:15:00.0
5               Ngô Huy Hoàng               Nguyễn Minh Đức             250000.00       2023-01-05 14:50:00.0
6               Lê Ngọc Tú                  Lê Ngọc Hà                  300000.00       2023-01-06 16:20:00.0
7               Vũ Thị Thu Hương            Trần Thị Mai                175000.00       2023-01-07 12:30:00.0
8               Nguyễn Thanh Phong          Phạm Quốc Hùng              220000.00       2023-01-08 13:40:00.0
9               Đặng Hải Nam                Lê Thanh Tú                 280000.00       2023-01-09 17:10:00.0
10              Bùi Thanh Hải               Ngô Bảo Châu                150000.00       2023-01-10 19:00:00.0
11              Nguyễn Văn Hùng             Nguyễn Văn An               140000.00       2023-01-11 08:30:00.0
12              Trần Thị Ngọc Hà            Lê Thị Hương                240000.00       2023-01-12 10:45:00.0
13              Phạm Hoàng Anh              Trần Quốc Bảo               190000.00       2023-01-13 13:15:00.0
14              Lê Minh Tuấn                Phạm Hoàng Long             150000.00       2023-01-14 11:00:00.0
15              Ngô Bảo Trâm                Nguyễn Minh Đức             210000.00       2023-01-15 14:30:00.0
16              Hoàng Thanh Hương           Lê Ngọc Hà                  270000.00       2023-01-16 16:20:00.0
17              Nguyễn Thu Hằng             Trần Thị Mai                160000.00       2023-01-17 12:30:00.0
18              Trần Quốc Bảo               Phạm Quốc Hùng              210000.00       2023-01-18 13:40:00.0
19              Phạm Thị Hoa                Lê Thanh Tú                 230000.00       2023-01-19 17:10:00.0
20              Lê Văn Minh                 Ngô Bảo Châu                180000.00       2023-01-20 19:00:00.0
21              Ngô Hoàng Hải               Nguyễn Văn An               160000.00       2023-01-21 08:30:00.0
22              Vũ Thanh Tú                 Lê Thị Hương                210000.00       2023-01-22 10:45:00.0
23              Nguyễn Hoàng Duy            Trần Quốc Bảo               190000.00       2023-01-23 13:15:00.0
24              Trần Thanh Phương           Phạm Hoàng Long             150000.00       2023-01-24 11:00:00.0
```

4. Tính tổng số tiền mà khách hàng đã chi tiêu (Tổng tiền của tất cả các hóa đơn).

```
SELECT c.MAKH , SUM(i.TongTien)
FROM invoice AS i
JOIN customer AS c ON i.MAKH = c.MAKH
GROUP BY c.MAKH
```

C#:

```
//4.Tính tổng số tiền mà khách hàng đã chi tiêu(Tổng tiền của tất cả các hóa đơn).
1 reference
public static void cau4()
{
    var result = from i in dc1dc.INVOICEs
                 join c in dc1dc.CUSTOMERs on i.MAKH equals c.MAKH
                 group i by new { c.MAKH, c.HoTen } into grouped
                 select new
                 {
                     MAKH = grouped.Key.MAKH,
                     HoTen = grouped.Key.HoTen,
                     TotalSpent = grouped.Sum(i => i.TongTien)
                 };
    Console.WriteLine("\n4. Tính tổng số tiền mà khách hàng đã chi tiêu (Tổng tiền của tất cả các hóa đơn):");
    Console.WriteLine("-----------------------------------------------------");
    Console.WriteLine("{0,-20} {1,-30} {2,-20}", "Mã Khách Hàng", "Họ Và Tên", "Tổng Tiền");
    Console.WriteLine("-----------------------------------------------------");
    result.ToList().ForEach(customer =>
    {
        Console.WriteLine($"{customer.MAKH,-20} {customer.HoTen,-30} {customer.TotalSpent,-20}");
    });
}
```

Giải thích:

· var result: Khai báo biến result để lưu trữ kết quả của truy vấn LINQ.

· from id in dc1dc.INVOICE_DETAILs: Bắt đầu truy vấn từ bảng INVOICE_DETAILs và đặt tên biến id cho từng bản ghi trong bảng này.

· join p in dc1dc.PRODUCTs on id.MASP equals p.MASP: Thực hiện phép nối (join) giữa bảng INVOICE_DETAILs và bảng PRODUCTs dựa trên cột MASP (Mã Sản Phẩm). Chỉ những bản ghi có MASP khớp nhau giữa hai bảng sẽ được kết hợp lại.

· orderby id.MAHD: Sắp xếp kết quả theo mã hóa đơn (MAHD).

· select new { ... }: Tạo ra một đối tượng ẩn danh mới chứa các cột được chọn từ kết quả nối:

MAHD = id.MAHD: Mã hóa đơn.

TenSP = p.TenSP: Tên sản phẩm.

SoLuong = id.SoLuong: Số lượng sản phẩm.

Java:

```
//  4. Tính tổng số tiền mà khách hàng đã chi tiêu (Tổng tiền của tất cả các hóa đơn).
    public static void cau4() {
        System.out.printf("%-20s %-30s %-20s %n","Mã Khách Hàng","Họ Và Tên","Tổng Tiền");
        System.out.println("-------------------------------------------------------");
        Map<Integer, Double> totalAmountByCustomer = invoices.stream().collect(
                Collectors.groupingBy(Invoice::getCustomerId, Collectors.summingDouble(Invoice::getTotalAmount)));

        totalAmountByCustomer.forEach((customerID, totalAmount) -> {
            String customerName = customers.stream().filter(customer -> customer.getId() == customerID)
                    .map(Customer::getName).findFirst().orElse("Unknow");
            System.out.printf("%-20s %-30s %-20s %n", customerID ,customerName ,totalAmount);
        });

    }
```

Giải thích :

Map<Integer, Double> totalAmountByCustomer = invoices.stream().collect(...):

Dùng phương thức stream() để duyệt qua tất cả các hóa đơn trong danh sách invoices.

Collectors.groupingBy(Invoice::getCustomerId, Collectors.summingDouble(Invoice::getTotalAmount))

Dùng để nhóm các hóa đơn theo customerId (mã khách hàng), rồi tính tổng số tiền chi tiêu của mỗi khách hàng.

totalAmountByCustomer.forEach((customerID, totalAmount) -> {...}):

Duyệt qua tất cả các mục trong totalAmountByCustomer, mỗi mục gồm customerID (mã khách hàng) và totalAmount (tổng số tiền khách hàng đã chi tiêu).

Kết quả truy vấn:

```
4. Tính tổng số tiền mà khách hàng đã chi tiêu (Tổng tiền của tất cả các hóa đơn):
-----------------------------------------------------
Mã Khách Hàng        Họ Và Tên                      Tổng Tiền
-----------------------------------------------------
1                    Nguyễn Hoàng Long              330000.0
2                    Trần Thị Minh Hằng             360000.0
3                    Phạm Quốc Anh                  370000.0
4                    Hoàng Mai Phương               370000.0
5                    Ngô Huy Hoàng                  430000.0
6                    Lê Ngọc Tú                     510000.0
7                    Vũ Thị Thu Hương               445000.0
8                    Nguyễn Thanh Phong             380000.0
9                    Đặng Hải Nam                   490000.0
10                   Bùi Thanh Hải                  380000.0
11                   Nguyễn Văn Hùng                320000.0
12                   Trần Thị Ngọc Hà               240000.0
13                   Phạm Hoàng Anh                 190000.0
14                   Lê Minh Tuấn                   150000.0
15                   Ngô Bảo Trâm                   210000.0
16                   Hoàng Thanh Hương              270000.0
17                   Nguyễn Thu Hằng                160000.0
18                   Trần Quốc Bảo                  210000.0
19                   Phạm Thị Hoa                   230000.0
20                   Lê Văn Minh                    180000.0
21                   Ngô Hoàng Hải                  160000.0
```

5. Liệt kê tên sản phẩm và số lượng của từng sản phẩm trong mỗi hóa đơn.

```
SELECT ID.MAHD, P.TenSP, ID.SoLuong
FROM INVOICE_DETAIL ID
JOIN PRODUCT P ON ID.MASP = P.MASP
ORDER BY ID.MAHD;
```

C#:

```
//5.Liệt kê tên sản phẩm và số lượng của từng sản phẩm trong mỗi hóa đơn.
1 reference
public static void cau5()
{
    var result = from id in dc1dc.INVOICE_DETAILs
                 join p in dc1dc.PRODUCTs on id.MASP equals p.MASP
                 orderby id.MAHD
                 select new
                 {
                     MAHD = id.MAHD,
                     TenSP = p.TenSP,
                     SoLuong = id.SoLuong
                 };
    Console.WriteLine("\n5. Liệt kê tên sản phẩm và số lượng của từng sản phẩm trong các hóa đơn:");
    Console.WriteLine("-----------------------------------------------------");
    result.ToList().ForEach(product => {
        Console.WriteLine($" {product.MAHD,-10}{product.TenSP,-30}: {product.SoLuong}");
    });
}
```

Giải thích :

```
var result = from id in dc1dc.INVOICE_DETAILs
join p in dc1dc.PRODUCTs on id.MASP equals p.MASP
orderby id.MAHD
select new {
      MAHD = id.MAHD,
      TenSP = p.TenSP,
      SoLuong = id.SoLuong
};
```

- var result: Khai báo biến result để lưu trữ kết quả của truy vấn LINQ
- from id in dc1dc.INVOICE_DETAILs: Bắt đầu truy vấn từ bảng INVOICE_DETAILs và đặt tên biến id cho từng bản ghi trong bảng này.

- join p in dc1dc.PRODUCTs on id.MASP equals p.MASP: Thực hiện phép nối (join) giữa bảng INVOICE_DETAILs và bảng PRODUCTs dựa trên cột MASP (Mã Sản Phẩm). Chỉ những bản ghi có MASP khớp nhau giữa hai bảng sẽ được kết hợp lại.
- orderby id.MAHD: Sắp xếp kết quả theo mã hóa đơn (MAHD) theo thứ tự tăng dần.
- select new {...}: Tạo ra một đối tượng ẩn danh mới chứa các cột được chọn từ kết quả nối

  MAHD = id.MAHD: Mã hóa đơn từ bảng INVOICE_DETAILs.

  TenSP = p.TenSP: Tên sản phẩm từ bảng PRODUCTs.

  SoLuong = id.SoLuong: Số lượng sản phẩm từ bảng INVOICE_DETAILs.

Java:

```
// 5. Liệt kê tên sản phẩm và số lượng của từng sản phẩm trong các hóa đơn.
public static void cau5() {
    invoices.forEach(invoice -> {
        System.out.printf("Hóa đơn ID: %-10d%n", invoice.getId());
        Map<String, Integer> productQuantitiesPerInvoice = invoiceDetails.stream()
                .filter(detail -> detail.getInvoiceId() == invoice.getId())
                .collect(Collectors.groupingBy(
                        detail -> products.stream()
                                .filter(product -> product.getId() == detail.getProductId())
                                .map(Product::getName)
                                .findFirst()
                                .orElse("Unknown"),
                        Collectors.summingInt(InvoiceDetail::getQuantity)));

        productQuantitiesPerInvoice.forEach((productName, quantity) ->
            System.out.printf(" %-30s: %d%n", productName, quantity));

        System.out.println();
    });
}
```

Giải thích:

Map<String, Integer> productQuantitiesPerInvoice = invoiceDetails.stream() .filter(detail -> detail.getInvoiceId() == invoice.getId()) .collect(Collectors.groupingBy(detail -> products.stream() .filter(product -> product.getId() == detail.getProductId()) .map(Product::getName) .findFirst() .orElse("Unknown"), Collectors.summingInt(InvoiceDetail::getQuantity)));
· invoiceDetails.stream(): Tạo một dòng (stream) từ danh sách invoiceDetails, mỗi phần tử là chi tiết hóa đơn. · filter(detail -> detail.getInvoiceId() == invoice.getId()): Lọc các chi tiết hóa đơn có invoiceId trùng với ID của hóa đơn hiện tại (invoice.getId()). · collect(Collectors.groupingBy(...)): Nhóm các chi tiết hóa đơn theo tên

sản phẩm và tính tổng số lượng sản phẩm trong mỗi nhóm.

detail -> products.stream(): Dùng Lambda để tìm sản phẩm có productId tương ứng với mỗi chi tiết hóa đơn (detail).

filter(product -> product.getId() == detail.getProductId()): Lọc sản phẩm có productId trùng với chi tiết hóa đơn.

map(Product::getName): Lấy tên sản phẩm từ đối tượng Product.

findFirst().orElse("Unknown"): Nếu không tìm thấy sản phẩm nào, trả về "Unknown" (tên sản phẩm không xác định).

Collectors.summingInt(InvoiceDetail::getQuantity): Tính tổng số lượng sản phẩm trong mỗi nhóm.

Kết quả sẽ là một Map có khóa là tên sản phẩm và giá trị là tổng số lượng của sản phẩm đó trong hóa đơn.

```
productQuantitiesPerInvoice.forEach((productName, quantity) ->
System.out.printf(" %-30s: %d%n", productName, quantity) );
```

· forEach: Duyệt qua các mục trong productQuantitiesPerInvoice, mỗi mục có một productName và quantity.
· (productName, quantity) -> ...: Đây là Lambda xử lý mỗi phần tử trong productQuantitiesPerInvoice. Với mỗi sản phẩm, in ra tên sản phẩm và số lượng của nó.

Kết quả truy vấn:

```
5. Liệt kê tên sản phẩm và số lượng của từng sản phẩm trong các hóa đơn:
-------------------------------------------------------
Hóa đơn ID: 1
 Khoai Tây Chiên Lớn            : 3
 Burger Bò                      : 2
 Pizza Phô Mai                  : 2
 Gà Rán Phần Lớn                : 1
 Burger Phô Mai                 : 1

Hóa đơn ID: 2
 Pizza Hải Sản                  : 1
 Khoai Tây Chiên Nhỏ            : 2
 Gà Rán Phần Nhỏ                : 2
 Burger Gà                      : 3

Hóa đơn ID: 3
 Gà Xiên Nướng                  : 1
 Pizza Thập Cẩm                 : 2
 Nước Ngọt Pepsi                : 2
 Mì Ý Sốt Gà                    : 3

Hóa đơn ID: 4
 Taco Bò                        : 1
 Nước Chanh                     : 2
 Súp Hải Sản                    : 1
 Salad Trộn                     : 3

Hóa đơn ID: 5
 Bánh Sandwich Bò               : 1
 Súp Gà                         : 3
 Hotdog Xúc Xích                : 2
 Taco Phô Mai                   : 2

Hóa đơn ID: 6
 Khoai Lang Chiên               : 1
```

6. Hiển thị thông tin các nhân viên đã làm việc từ ngày 01/01/2022 trở đi.

```
SELECT *
FROM employee e
WHERE e.NgayBatDau > "2022-01-01" ;
```

C#:

```
//6.Hiển thị thông tin các nhân viên đã làm việc từ ngày 01/01/2022 trở đi.
1 reference
public static void cau6(DateTime date)
{
    var result = from e in dc1dc.EMPLOYEEs
                 where e.NgayBatDau > date
                 select new
                 {
                     MANV = e.MANV,
                     HOVATEN = e.HoTen,
                     ChucVu = e.ChucVu,
                     SDT = e.SDT,
                     MgayBatDau = e.NgayBatDau
                 };
    Console.WriteLine("\n6. Hiển thị thông tin các nhân viên đã làm việc từ ngày 01/01/2022 trở đi:");
    Console.WriteLine("-------------------------------------------------------");
    Console.WriteLine($"{"Mã Nhân Viên",-20}{"Họ Và Tên",-30}{"Chức Vụ",-25}{"Số Điện Thoại",-15}{"Ngày Bắt Đầu",-20}");
    result.ToList().ForEach(employee =>
    {
        Console.WriteLine($"{employee.MANV,-20}{employee.HOVATEN,-30}{employee.ChucVu,-25}{employee.SDT,-15}{employee.MgayBatDau , -20}");
    });
```

Giải thích :

```
var result = from e in dc1dc.EMPLOYEEs
```

from e in dc1dc.EMPLOYEEs: Bắt đầu truy vấn từ bảng `EMPLOYEEs` và đặt tên biến `e` cho từng bản ghi trong bảng này.

```
where e.NgayBatDau > date
```

Lọc ra các nhân viên có ngày bắt đầu làm việc (`NgayBatDau`) sau ngày được chỉ định (`date`).

```
select new
{
   MANV = e.MANV,
   HOVATEN = e.HoTen,
   ChucVu = e.ChucVu,
   SDT = e.SDT,
   MgayBatDau = e.NgayBatDau
};
```

select new {...}: Tạo ra một đối tượng ẩn danh mới chứa các cột được chọn từ kết quả truy vấn.

MANV = e.MANV: Mã nhân viên.
HOVATEN = e.HoTen: Họ và tên nhân viên.
ChucVu = e.ChucVu: Chức vụ của nhân viên.
SDT = e.SDT: Số điện thoại của nhân viên.
MgayBatDau = e.NgayBatDau: Ngày bắt đầu làm việc của nhân viên.

Java:

```
6. Hiển thị thông tin các nhân viên đã làm việc từ ngày 01/01/2022 trở đi.
public static void cau6(Date startDate) {
    System.out.printf("%-20s %-30s %-25s %-15s %-20s %n",
            "Mã Nhân Viên","Họ Và Tên","Chức Vụ","Số Điện Thoại","Ngày Bắt Đầu");
    System.out.println("-------------------------------------------------------------");
    employees.stream().filter(employee -> employee.getStartDate().after(startDate))
            .forEach(employee -> System.out.printf("%-20s %-30s %-25s %-15s %-20s %n",
                    employee.getId(),
                    employee.getName() ,
                    employee.getPosition() ,
                    employee.getPhone() ,
                    employee.getStartDate()));
}
```

Giải thích :

```
.filter(employee -> employee.getStartDate().after(startDate))
```

là một biểu thức lambda, trong đó employee là một đối tượng Employee trong danh sách. Phần employee.getStartDate() lấy ngày bắt đầu làm việc của nhân viên, và after(startDate) kiểm tra xem ngày bắt đầu của nhân viên đó có sau ngày startDate hay không.
=>Kết quả của .filter() là một stream chỉ chứa những nhân viên có ngày bắt đầu làm việc sau ngày startDate.

```
.forEach(.....):
```

duyệt qua từng nhân viên trong stream đã lọc và in thông tin của họ ra màn hình, bao gồm mã nhân viên, tên, chức vụ, số điện thoại và ngày bắt đầu làm việc, với định dạng đã được chỉ định trong printf().

Kết quả truy vấn:

```
6. Hiển thị thông tin các nhân viên đã làm việc từ ngày 01/01/2022 trở đi
-------------------------------------------------------
Mã Nhân Viên        Họ Và Tên                   Chức Vụ                 Số Điện Thoại   Ngày Bắt Đầu
-------------------------------------------------------
4                   Phạm Hoàng Long             Nhân viên bán hàng      0904445566      2022-03-01
5                   Nguyễn Minh Đức             Nhân viên bán hàng      0905556677      2022-08-15
7                   Trần Thị Mai                Nhân viên bán hàng      0907778899      2023-02-05
8                   Phạm Quốc Hùng              Nhân viên bán hàng      0908889900      2023-06-10
```

7. Tìm tất cả các hóa đơn có tổng tiền lớn hơn 100,000 VND.

```
SELECT *
FROM invoice
WHERE invoice.TongTien > 100000 ;
```

C#:

```
//7.Tìm tất cả các hóa đơn có tổng tiền lớn hơn 100,000 VND.
1 reference
public static void cau7(decimal amount)
{
    var result = from i in dc1dc.INVOICEs
                 where i.TongTien > amount
                 select i;
    Console.WriteLine("\n7. Tìm tất cả các hóa đơn có tổng tiền lớn hơn 100,000 VND:");
    Console.WriteLine("-------------------------------------------------------");
    Console.WriteLine($"{"MAHD",-5}{"MAKH",-5}{"MANV",-10}{"Thời Gian TT",-30}{"Tổng Tiền",-20}{"Hình Thức TT",-20}{"Số Tiền Nhận",-20}");
    Console.WriteLine("-------------------------------------------------------");
    result.ToList().ForEach(invoice =>
    {
        Console.WriteLine($"{invoice.MAHD,-5}{invoice.MANV,-5}{invoice.MAKH,-10}" +
            $"{invoice.ThoiGianTT,-30}{invoice.TongTien,-20}{invoice.HinhThucTT,-20}{invoice.SoTienNhan,-20}");
    });
}
```

Truy vấn dữ liệu từ bảng INVOICEs:

var result = from i in dc1dc.INVOICEs

Các bước thực hiện:

- var result: **Khai báo biến** result để lưu trữ kết quả của truy vấn.
- from i in dc1dc.INVOICEs: **Bắt đầu truy vấn** từ bảng INVOICEs và đặt tên biến i cho từng bản ghi trong bảng này.

Lọc các hóa đơn có tổng tiền lớn hơn giá trị cụ thể:

where i.TongTien > amount

Các bước thực hiện:

- where i.TongTien > amount: **Lọc ra các hóa đơn** có tổng tiền (TongTien) lớn hơn giá trị amount.

Chọn các hóa đơn thỏa mãn điều kiện:

select i;

Các bước thực hiện:

- select i: **Chọn toàn bộ thông tin** của các hóa đơn thỏa mãn điều kiện từ bảng INVOICEs.

Java:

```
//7. Tìm tất cả các hóa đơn có tổng tiền lớn hơn 100,000 VND.
   public static void cau7(double amount) {
       System.out.printf("%-5s %-5s %-10s %-30s %-20s %-20s %-20s %n",
               "MAHD","MAKH","MANV","Thời Gian","Tổng Tiền","Hình Thức TT","Số Tiền Nhận" );
       System.out.println("------------------------------------------------------------");
       invoices.stream().filter(invoice -> invoice.getTotalAmount() > amount)
       .forEach(invoice -> System.out.printf("%-5s %-5s %-10s %-30s %-20s %-20s %-20s %n",
                   invoice.getId(),
                   invoice.getCustomerId(),
                   invoice.getEmployeeId(),
                   invoice.getPaymentTime().toString(),
                   invoice.getTotalAmount(),
                   invoice.getPaymentMethod(),
                   invoice.getReceivedAmount()));
   }
```

Giải thích :

`invoices.stream()`

invoices là danh sách các đối tượng Invoice (hóa đơn).

.stream() chuyển đổi danh sách invoices thành một **stream** (dòng chảy). Điều này cho phép chúng ta sử dụng các phương thức như filter, forEach, map, ... để xử lý từng phần tử trong danh sách.

`.filter(invoice -> invoice.getTotalAmount() > amount)`

.filter() là phương thức của Stream API, dùng để lọc các phần tử trong stream theo điều kiện được cung cấp.

Trong biểu thức invoice -> invoice.getTotalAmount() > amount, invoice là mỗi phần tử trong stream, tức là mỗi hóa đơn.

- invoice.getTotalAmount() trả về tổng tiền của hóa đơn.
- amount là một giá trị được truyền vào phương thức, dùng để so sánh với tổng tiền của hóa đơn.

.filter() chỉ giữ lại những hóa đơn có tổng tiền lớn hơn giá trị amount. Các hóa đơn không thỏa mãn điều kiện sẽ bị loại bỏ.

`.forEach(invoice -> System.out.printf(...)`

forEach() là phương thức duyệt qua từng phần tử còn lại trong stream sau khi lọc, và thực hiện một hành động với mỗi phần tử.

Trong biểu thức invoice -> System.out.printf(...), chúng ta sử dụng System.out.printf() để in thông tin của mỗi hóa đơn theo định dạng nhất định.

Kết quả truy vấn:

```
7. Tìm tất cả các hóa đơn có tổng tiền lớn hơn 100,000 VND.
-----------------------------------------------------
MAHD  MAKH  MANV       Thời Gian                 Tổng Tiền           Hình Thức TT        Số Tiền Nhận
-----------------------------------------------------
1     1     1          2023-01-01 10:00:00.0     150000.0            Tiền mặt            150000.0
2     2     2          2023-01-02 11:30:00.0     200000.0            Thẻ                 200000.0
3     3     3          2023-01-03 12:45:00.0     120000.0            Chuyển khoản        120000.0
4     4     4          2023-01-04 09:15:00.0     180000.0            Tiền mặt            180000.0
5     5     5          2023-01-05 14:50:00.0     250000.0            Thẻ                 250000.0
6     6     6          2023-01-06 16:20:00.0     300000.0            Chuyển khoản        300000.0
7     7     7          2023-01-07 12:30:00.0     175000.0            Tiền mặt            175000.0
8     8     8          2023-01-08 13:40:00.0     220000.0            Thẻ                 220000.0
9     9     9          2023-01-09 17:10:00.0     280000.0            Chuyển khoản        280000.0
10    10    10         2023-01-10 19:00:00.0     150000.0            Tiền mặt            150000.0
11    11    1          2023-01-11 08:30:00.0     140000.0            Tiền mặt            140000.0
12    12    2          2023-01-12 10:45:00.0     240000.0            Thẻ                 240000.0
13    13    3          2023-01-13 13:15:00.0     190000.0            Chuyển khoản        190000.0
14    14    4          2023-01-14 11:00:00.0     150000.0            Tiền mặt            150000.0
15    15    5          2023-01-15 14:30:00.0     210000.0            Thẻ                 210000.0
16    16    6          2023-01-16 16:20:00.0     270000.0            Chuyển khoản        270000.0
17    17    7          2023-01-17 12:30:00.0     160000.0            Tiền mặt            160000.0
18    18    8          2023-01-18 13:40:00.0     210000.0            Thẻ                 210000.0
19    19    9          2023-01-19 17:10:00.0     230000.0            Chuyển khoản        230000.0
20    20    10         2023-01-20 19:00:00.0     180000.0            Tiền mặt            180000.0
21    21    1          2023-01-21 08:30:00.0     160000.0            Tiền mặt            160000.0
22    22    2          2023-01-22 10:45:00.0     210000.0            Thẻ                 210000.0
23    23    3          2023-01-23 13:15:00.0     190000.0            Chuyển khoản        190000.0
24    24    4          2023-01-24 11:00:00.0     150000.0            Tiền mặt            150000.0
25    25    5          2023-01-25 14:30:00.0     210000.0            Thẻ                 210000.0
26    26    6          2023-01-26 16:20:00.0     270000.0            Chuyển khoản        270000.0
```

8. Liệt kê các khuyến mãi đang còn hiệu lực (trước ngày 31/12/2024).

```
SELECT *
FROM promotion
WHERE promotion.ThoiGianHL > "2023-31-12"
```

C#:

```
//8.Liệt kê các khuyến mãi đang còn hiệu lực(trước ngày 31/12/2024).
1 reference
public static void cau8(DateTime date)
{
    var result = from p in dc1dc.PROMOTIONs
                 where p.ThoiGianHL < date
                 select p;
    Console.WriteLine("\n8. Liệt kê các khuyến mãi đang còn hiệu lực (trước ngày 31/12/2024):");
    Console.WriteLine("-----------------------------------------------------");
    Console.WriteLine($"{"MAKM",-10}{"Thời Gian HIệu Lực",-20}{"Điều Kiện",-50}{"Mức Giảm Giá",-30}");
    Console.WriteLine("-----------------------------------------------------");
    result.ToList().ForEach(promotion =>
    {
        Console.WriteLine($"{promotion.MAKM,-10}{promotion.ThoiGianHL,-20}{promotion.DieuKien,-50}{promotion.MucGiamGia,-30}");
    });
}
```

Giải thích :

```
var result = from p in dc1dc.PROMOTIONs
```

`var result`: Khai báo biến `result` để lưu trữ kết quả của truy vấn.
`from p in dc1dc.PROMOTIONs`: Bắt đầu truy vấn từ bảng `PROMOTIONs` và đặt tên biến `p` cho từng bản ghi trong bảng này.

```
where p.ThoiGianHL < date
```

Lọc ra các khuyến mãi có thời gian hiệu lực (`ThoiGianHL`) nhỏ hơn ngày được chỉ định (`date`).

```
select p;
```

Chọn toàn bộ thông tin của các khuyến mãi thỏa mãn điều kiện từ bảng `PROMOTIONs`.

Java:

```java
// 8. Liệt kê các khuyến mãi đang còn hiệu lực (trước ngày 31/12/2024).
public static void cau8(Date endDate) {
    System.out.printf("%-10s %-20s %-50s %-30s %n",
            "MaKM","Thời Gian Hiệu Lực","Điều Kiện","Mức Giảm");
    System.out.println("-----------------------------------------------------");
    promotions.stream().filter(promotion -> promotion.getEffectiveTime().before(endDate))
            .forEach(promotion-> System.out.printf("%-10s %-20s %-50s %-30s %n",
                    promotion.getId(),
                    promotion.getEffectiveTime(),
                    promotion.getCondition(),
                    promotion.getDiscount()));
}
```

Giải thích:

.filter(promotion -> promotion.getEffectiveTime().before(endDate))

Giải thích biểu thức

Trong biểu thức promotion -> promotion.getEffectiveTime().before(endDate), promotion là mỗi phần tử trong stream, tức là mỗi đối tượng Promotion.

Các thành phần của biểu thức

- promotion.getEffectiveTime() trả về thời gian có hiệu lực của chương trình khuyến mãi (kiểu Date).
- endDate là một giá trị ngày tháng được truyền vào từ bên ngoài phương thức, đại diện cho mốc thời gian mà chúng ta muốn so sánh.
- .before(endDate) kiểm tra xem thời gian hiệu lực của khuyến mãi có trước thời gian endDate hay không.

Chức năng của .filter()

.filter() chỉ giữ lại các khuyến mãi mà thời gian hiệu lực của chúng trước endDate. Các khuyến mãi không thỏa mãn điều kiện này sẽ bị loại bỏ.

Kết quả truy vấn:

```
8. Liệt kê các khuyến mãi đang còn hiệu lực (trước ngày 31/12/2024):
-----------------------------------------------------
MaKM       Thời Gian Hiệu Lực   Điều Kiện                                          Mức Giảm
-----------------------------------------------------
1          2023-01-01           Áp dụng cho đơn hàng trên 500,000 VND              50000.0
2          2023-02-01           Áp dụng cho đơn hàng trên 1,000,000 VND            100000.0
3          2023-03-01           Giảm giá 10% cho tất cả các sản phẩm               10.0
4          2023-04-01           Áp dụng cho khách hàng có điểm tích lũy trên 100   150000.0
5          2023-05-01           Khuyến mãi ngày lễ 30/4                            70000.0
```

9. Tìm thông tin các khách hàng và số điểm tích lũy của họ, sắp xếp theo điểm tích lũy giảm dần.

```
SELECT *
FROM customer c
ORDER BY c.DiemTichLuy DESC
```

C#:

```
////9.Tìm thông tin các khách hàng và số điểm tích lũy của họ, sắp xếp theo điểm tích lũy giảm dần.
1 reference
public static void cau9()
{
    var result = from c in dc1dc.CUSTOMERs
                 orderby c.DiemTichLuy descending
                 select c;
    Console.WriteLine("\n9. Tìm thông tin các khách hàng và số điểm tích lũy của họ, sắp xếp theo điểm tích lũy giảm dần:");
    Console.WriteLine("-----------------------------------------------------");
    Console.WriteLine("{0,-10} {1,-30} {2,-15} {3,-10}", "ID", "Họ Và Tên", "Số Điện Thoại", "Điểm tích lũy");
    Console.WriteLine("-----------------------------------------------------");
    result.ToList().ForEach(c =>
    {
        Console.WriteLine("{0,-10} {1,-30} {2,-15} {3,-10}",c.MAKH,c.HoTen,c.SDT,c.DiemTichLuy);
    });
}
```

Giải thích:

Giải thích từng phần:

- **`var result`:** Khai báo biến `result` để lưu trữ kết quả của truy vấn.
- **`from c in dc1dc.CUSTOMERs`:** Bắt đầu truy vấn từ bảng `CUSTOMERs` và đặt tên biến `c` cho từng bản ghi trong bảng này.
- **`orderby c.DiemTichLuy descending`:** Sắp xếp kết quả theo cột `DiemTichLuy` (điểm tích lũy) theo thứ tự giảm dần (`descending`).
- **`select c`:** Chọn toàn bộ thông tin của các khách hàng từ bảng `CUSTOMERs` sau khi đã sắp xếp.

Java:

```
9. Tìm thông tin các khách hàng và số điểm tích lũy của họ, sắp xếp theo điểm tích lũy giảm dần.
public static void cau9() {
    System.out.printf("%-10s %-30s %-15s %-10s %n", "ID", "Họ Và Tên", "Số Điện Thoại" ,"Điểm tích lũy");
    System.out.println("-----------------------------------------------------");
    customers.stream().sorted(Comparator.comparingInt(Customer::getPoints).reversed())
    .forEach(c -> System.out.printf("%-10s %-30s %-15s %-10s %n",
            c.getId(),
            c.getName(),
            c.getPhone(),
            c.getPoints()));
}
```

Giải thích:

```
.sorted(Comparator.comparingInt(Customer::getPoints).reversed())
```

sorted() là phương thức của Stream API dùng để sắp xếp các phần tử trong stream theo một tiêu chí nào đó.

Comparator.comparingInt(Customer::getPoints):

- Comparator.comparingInt() là một phương thức tạo đối tượng Comparator để so sánh các phần tử dựa trên giá trị của một trường số nguyên.
- Customer::getPoints là phương thức lấy giá trị của trường điểm tích lũy (points) từ mỗi đối tượng Customer. Phương thức này trả về một giá trị kiểu int.

.reversed():

- reversed() là một phương thức để đảo ngược thứ tự sắp xếp. Nếu không có reversed(), các khách hàng sẽ được sắp xếp theo thứ tự tăng dần của điểm tích lũy.
- Thêm reversed() sẽ khiến các khách hàng được sắp xếp theo thứ tự giảm dần của điểm tích lũy.

Kết quả truy vấn:

```
9. Tìm thông tin các khách hàng và số điểm tích lũy của họ, sắp xếp theo điểm tích lũy giảm dần:
-----------------------------------------------------
ID        Họ Và Tên                   Số Điện Thoại   Điểm tích lũy
-----------------------------------------------------
10        Bùi Thanh Hải               0990123456      200
21        Ngô Hoàng Hải               0921122334      180
27        Ngô Ngọc Mai                0927788990      170
30        Trần Đăng Quang             0930011223      155
59        Nguyễn Văn Đại              0959900112      155
70        Hoàng Thu Phương            0970011223      155
3         Phạm Quốc Anh               0923456789      150
46        Hoàng Thu Hà                0946677889      150
68        Lê Quốc Anh                 0968899001      145
80        Lê Huy Hoàng                0980011223      145
20        Lê Văn Minh                 0920011223      140
39        Ngô Bảo Minh                0939900112      140
53        Nguyễn Thị Thanh Hằng       0953344556      140
74        Lê Thanh Bình               0974455667      140
84        Trần Văn Thái               0984455667      140
33        Ngô Thanh Tùng              0933344556      135
47        Nguyễn Văn Khánh            0947788990      135
62        Lê Quốc Huy                 0962233445      135
22        Vũ Thanh Tú                 0922233445      130
73        Phạm Minh Khang             0973344556      130
83        Nguyễn Huy Vũ               0983344556      130
17        Nguyễn Thu Hằng             0917788990      125
28        Hoàng Minh Hải              0928899001      125
```

10. Tính tổng doanh thu từ các hóa đơn trong tháng 1/2023.

```
SELECT SUM(TongTien) AS TotalRevenue
FROM INVOICE
WHERE MONTH(ThoiGianTT) = 1 AND YEAR(ThoiGianTT) = 2023;
```

C#:

```
////10.Tính tổng doanh thu từ các hóa đơn trong tháng 1/2023.
1 reference
public static void cau10(int year , int month)
{
    var totalRevenue = (from i in dc1dc.INVOICEs
                        where i.ThoiGianTT.Month == month && i.ThoiGianTT.Year == year
                        select i.TongTien).Sum();

    Console.WriteLine("\n10. Tính tổng doanh thu từ các hóa đơn trong tháng 1/2023:");
    Console.WriteLine("-----------------------------------------------------");
    Console.WriteLine("Tổng doanh thu cho tháng {0} năm {1}: {2:F2}",month , year , totalRevenue);
}
```

Tổng doanh thu

- var totalRevenue: Khai báo biến totalRevenue để lưu trữ kết quả tổng doanh thu.

- from i in dc1dc.INVOICEs: Bắt đầu truy vấn từ bảng INVOICEs và đặt tên biến i cho từng bản ghi trong bảng này.

Điều kiện lọc

- where i.ThoiGianTT.Month == month && i.ThoiGianTT.Year == year: Lọc ra các hóa đơn có thời gian thanh toán (ThoiGianTT) trong tháng (Month) và năm (Year) cụ thể.

Chọn cột tổng tiền

- select i.TongTien: Chọn cột TongTien (Tổng tiền) của các hóa đơn thỏa mãn điều kiện.

- - .Sum(): Tính tổng giá trị của các cột TongTien đã chọn.

Java:

```
10.Tính tổng doanh thu từ các hóa đơn trong tháng 11/2024.
public static void cau10(int year, int month) {
    double revenue = invoices.stream().filter(invoice -> {
        Calendar cal = Calendar.getInstance();
        cal.setTime(invoice.getPaymentTime());
        return cal.get(Calendar.MONTH) == month && cal.get(Calendar.YEAR) == year;
    }).mapToDouble(Invoice::getTotalAmount).sum();
    System.out.printf("Tổng doanh thu cho tháng %d năm %d: %.2f%n", month+1, year,revenue);
}
```

Giải thích :

invoices.stream():

invoices là một danh sách các đối tượng Invoice (hóa đơn).

.stream() chuyển danh sách invoices thành một **stream**, cho phép chúng ta xử lý các phần tử (hóa đơn) trong danh sách bằng cách sử dụng các phương thức như filter, mapToDouble, sum, v.v.

```
.filter(invoice -> {
        Calendar cal = Calendar.getInstance();
        cal.setTime(invoice.getPaymentTime());
        return cal.get(Calendar.MONTH) == month &&
                cal.get(Calendar.YEAR) == year;
})
```

.filter() là phương thức dùng để lọc các phần tử trong stream dựa trên một điều kiện nhất định.

Ở đây, điều kiện lọc là mỗi invoice (hóa đơn) có ngày thanh toán (paymentTime) trong tháng và năm được chỉ định (month, year).

Trong phần thân của .filter(), một đối tượng Calendar được tạo ra và gán ngày thanh toán của hóa đơn vào:

```
Calendar cal = Calendar.getInstance();
cal.setTime(invoice.getPaymentTime());
```

Sau đó, phương thức get(Calendar.MONTH) và get(Calendar.YEAR) được sử dụng để lấy tháng và năm của ngày thanh toán từ Calendar. Điều kiện kiểm tra:

return cal.get(Calendar.MONTH) == month && cal.get(Calendar.YEAR) == year;

.mapToDouble(Invoice::getTotalAmount):

Sau khi lọc các hóa đơn thỏa mãn điều kiện (tháng và năm), .mapToDouble() được sử dụng để chuyển mỗi hóa đơn thành giá trị tổng tiền (totalAmount) dưới dạng double.

Invoice::getTotalAmount là phương thức lấy tổng tiền của mỗi hóa đơn.

.sum():

.sum() tính tổng giá trị của các phần tử trong stream. Ở đây, nó sẽ cộng tất cả các tổng tiền của các hóa đơn thỏa mãn điều kiện tháng và năm.

Kết quả của .sum() sẽ là tổng doanh thu của các hóa đơn trong tháng và năm được yêu cầu.

Kết quả truy vấn:

```
10. Tính tổng doanh thu từ các hóa đơn trong tháng 1/2023:
-----------------------------------------------------------
Tổng doanh thu cho tháng 1 năm 2023: 5975000.00
```

11. Tìm thông tin của khách hàng có số tiền thanh toán lớn nhất trong một hóa đơn.

```
SELECT C.MAKH, C.HoTen, I.MAHD, I.TongTien
FROM invoice i
JOIN customer c ON c.MAKH = i.MAKH
WHERE i.TongTien = (
        SELECT MAX(i.TongTien)
        FROM invoice i
);
```

C#:

```
//11.Tìm thông tin của khách hàng có số tiền thanh toán lớn nhất trong một hóa đơn.
1 reference
public static void cau11()
{
    var maxAmount = dc1dc.INVOICEs.Max(i => i.TongTien);

    var result = (from i in dc1dc.INVOICEs
                  join c in dc1dc.CUSTOMERs on i.MAKH equals c.MAKH
                  where i.TongTien == maxAmount
                  select new
                  {
                      c.MAKH,
                      c.HoTen,
                      i.MAHD,
                      i.TongTien
                  }).FirstOrDefault();
    Console.WriteLine("\n11. Tìm thông tin của khách hàng có số tiền thanh toán lớn nhất trong một hóa đơn:");
    Console.WriteLine("-----------------------------------------------------");
    if(result != null)
    {
        Console.WriteLine("Khách hàng: {0,-30} \nSố tiền: {1,-10:F2}", result.HoTen, result.TongTien);
    }
    else
    {
        Console.WriteLine("Không có khách hàng nào thỏa mãn điều kiện.");
    }
```

Giải thích :

Tìm tổng tiền lớn nhất trong bảng INVOICEs

```
var maxAmount = dc1dc.INVOICEs.Max(invoice => invoice.TongTien);
```

- var maxAmount: Biến lưu tổng tiền lớn nhất.
- dc1dc.INVOICEs.Max(invoice => invoice.TongTien): Lấy giá trị lớn nhất của cột TongTien trong bảng INVOICEs.

Truy vấn thông tin của hóa đơn có tổng tiền lớn nhất

```
var result = dc1dc.INVOICEs
   .Where(invoice => invoice.TongTien == maxAmount)
   .Join(dc1dc.CUSTOMERs,
        invoice => invoice.MAKH,
        customer => customer.MAKH,
        (invoice, customer) => new
        {
           customer.MAKH,
           customer.HoTen,
           invoice.MAHD,
           invoice.TongTien
        })
   .FirstOrDefault();
```

Where(invoice => invoice.TongTien == maxAmount):
Lọc các hóa đơn có TongTien bằng với maxAmount.

Join(dc1dc.CUSTOMERs, ...):
Thực hiện phép **nối bảng** giữa:

- INVOICEs (dùng invoice.MAKH)
- CUSTOMERs (dùng customer.MAKH)

Kết quả nối ((invoice, customer) => new {...}**)**:
Tạo một đối tượng ẩn danh mới chứa:

- customer.MAKH: Mã khách hàng.
- customer.HoTen: Họ tên khách hàng.
- invoice.MAHD: Mã hóa đơn.
- invoice.TongTien: Tổng tiền hóa đơn.

FirstOrDefault():
Lấy bản ghi đầu tiên nếu có, hoặc trả về null nếu không tìm thấy.

Java:

```
11.Tìm thông tin của khách hàng có số tiền thanh toán lớn nhất trong một hóa đơn.
public static void cau11() {
    invoices.stream().max(Comparator.comparingDouble(Invoice::getTotalAmount)).ifPresent(invoice -> {
        Optional<Customer> maxCustomer = customers.stream()
                .filter(customer -> customer.getId() == invoice.getCustomerId()).findFirst();
        System.out.printf("Khách hàng: %-30s %nSố tiền: %-10.2f%n",
                maxCustomer.map(Customer::getName).orElse("Unknown"),
                invoice.getTotalAmount());
    });
}
```

Giải thích :

```
.max(Comparator.comparingDouble(Invoice::getTotalAmount)):
```

.max() là một phương thức trong stream để tìm giá trị lớn nhất trong stream.
Comparator.comparingDouble(Invoice::getTotalAmount) tạo một Comparator để so sánh các đối tượng Invoice dựa trên giá trị trả về từ phương thức getTotalAmount() (tổng tiền của hóa đơn).
Phương thức max sẽ tìm hóa đơn có tổng tiền lớn nhất trong danh sách invoices.

```
.ifPresent(invoice -> { ... })
```

.ifPresent() là một phương thức trong Optional. Nếu kết quả của phương thức max() (hóa đơn có tổng tiền lớn nhất) không phải là null, thì sẽ thực thi phần thân của ifPresent().

invoice là đối tượng Invoice có tổng tiền lớn nhất.

```
Optional<Customer> maxCustomer = customers.stream()
        .filter(customer -> customer.getId() ==
        invoice.getCustomerId()).findFirst();
```

Sau khi tìm được hóa đơn có tổng tiền lớn nhất, chúng ta cần tìm thông tin khách hàng của hóa đơn này.

customers.stream() chuyển danh sách khách hàng thành một stream.
.filter(customer -> customer.getId() == invoice.getCustomerId()) lọc các khách hàng để tìm khách hàng có id trùng với customerId trong hóa đơn.
.findFirst() trả về khách hàng đầu tiên tìm được trong danh sách (danh sách khách hàng có thể có nhiều người, nhưng ta chỉ lấy khách hàng đầu tiên phù hợp).

Kết quả truy vấn:

```
11. Tìm thông tin của khách hàng có số tiền thanh toán lớn nhất trong một hóa đơn:
-----------------------------------------------------
Khách hàng: Lê Ngọc Tú
Số tiền: 300000.00
```

12. Liệt kê tất cả các khách hàng và sản phẩm của họ trong hóa đơn (bao gồm tên khách hàng và tên sản phẩm).

```
SELECT C.HoTen, P.TenSP, ID.SoLuong
FROM INVOICE_DETAIL ID
JOIN INVOICE I ON ID.MAHD = I.MAHD
JOIN CUSTOMER C ON I.MAKH = C.MAKH
JOIN PRODUCT P ON ID.MASP = P.MASP
ORDER BY C.MAKH, I.MAHD;
```

C#:

```
////12.Liệt kê tất cả các khách hàng và sản phẩm của họ trong hóa đơn(bao gồm tên khách hàng và tên sản phẩm).
1 reference
public static void cau12()
{
    var result = from id in dc1dc.INVOICE_DETAILs
                 join i in dc1dc.INVOICEs on id.MAHD equals i.MAHD
                 join c in dc1dc.CUSTOMERs on i.MAKH equals c.MAKH
                 join p in dc1dc.PRODUCTs on id.MASP equals p.MASP
                 orderby c.MAKH, i.MAHD
                 select new
                 {
                     c.HoTen,
                     p.TenSP,
                     id.SoLuong
                 };
    Console.WriteLine("\n12. Liệt kê tất cả các khách hàng và sản phẩm của họ trong hóa đơn (bao gồm tên khách hàng và tên sản phẩm):");
    Console.WriteLine("-----------------------------------------------------");
    Console.WriteLine("{0,-30} {1,-30} {2,-15}", "Họ Tên", "Tên Sản Phẩm", "Số Lượng");
    Console.WriteLine("-----------------------------------------------------");
    result.ToList().ForEach(x => Console.WriteLine("{0,-30} {1,-30} {2,-15}", x.HoTen, x.TenSP, x.SoLuong));

}
```

```
var result = from id in dc1dc.INVOICE_DETAILs
```

- `var result`: Khai báo biến `result` để lưu trữ kết quả của truy vấn.
- `from id in dc1dc.INVOICE_DETAILs`: Bắt đầu truy vấn từ bảng `INVOICE_DETAILs` và đặt tên biến `id` cho từng bản ghi trong bảng này.

```
join i in dc1dc.INVOICEs on id.MAHD equals i.MAHD
```

- Thực hiện phép nối giữa bảng `INVOICE_DETAILs` và bảng `INVOICEs` dựa trên cột `MAHD` (mã hóa đơn). Điều này có nghĩa là chỉ các bản ghi có mã hóa đơn khớp nhau giữa hai bảng sẽ được kết hợp.

```
join c in dc1dc.CUSTOMERs on i.MAKH equals c.MAKH
```

- Thực hiện phép nối giữa bảng `INVOICEs` và bảng `CUSTOMERs` dựa trên cột `MAKH` (mã khách hàng). Điều này giúp lấy thông tin khách hàng cho mỗi hóa đơn.

```
join p in dc1dc.PRODUCTs on id.MASP equals p.MASP
```

- Thực hiện phép nối giữa bảng `INVOICE_DETAILs` và bảng `PRODUCTs` dựa trên cột `MASP` (mã sản phẩm). Điều này giúp lấy thông tin sản phẩm cho mỗi chi tiết hóa đơn.

```
orderby c.MAKH, i.MAHD
```

- Sắp xếp kết quả theo mã khách hàng (`MAKH`) và sau đó theo mã hóa đơn (`MAHD`).

Java:

```java
12.Liệt kê tất cả các khách hàng và sản phẩm của họ trong hóa đơn (bao gồm tên khách hàng và tên sản phẩm).
public static void cau12() {
    System.out.printf("%-30s %-30s %-15s %n", "Họ Tên","Tên Sản Phẩm","Số Lượng");
    System.out.println("-------------------------------------------------------");
    invoices.forEach(invoice -> {
        Optional<Customer> customer = customers.stream()
                .filter(c -> c.getId() == invoice.getCustomerId())
                .findFirst();

        Map<String, Integer> productQuantities = invoiceDetails.stream()
                .filter(detail -> invoice.getId() == detail.getInvoiceId())
                .collect(Collectors.groupingBy(
                    detail -> products.stream()
                        .filter(product -> product.getId() == detail.getProductId())
                        .map(Product::getName)
                        .findFirst()
                        .orElse("Unknown"),
                    Collectors.summingInt(InvoiceDetail::getQuantity)));

        productQuantities.forEach((productName, quantity) -> {
            System.out.printf("%-30s %-30s %-15d %n",
                    customer.map(Customer::getName).orElse("Unknown"),
                    productName, quantity);
        });
    });
}
```

Giải thích:

invoices.forEach(invoice -> { ... })

.forEach() là một phương thức của stream trong Java, được dùng để lặp qua từng phần tử trong stream.
Ở đây, chúng ta đang duyệt qua tất cả các hóa đơn trong danh sách invoices và xử lý từng hóa đơn (đối tượng invoice).

Optional<Customer> customer = customers.stream()...:

customers.stream() tạo ra một stream từ danh sách khách hàng customers.
.filter(c -> c.getId() == invoice.getCustomerId()) lọc các khách hàng để tìm khách hàng có id trùng với customerId của hóa đơn hiện tại.
.findFirst() trả về khách hàng đầu tiên tìm thấy (nếu có), được đóng gói trong Optional<Customer>. Nếu không có khách hàng phù hợp, sẽ trả về Optional.empty().

Map<String, Integer> productQuantities = invoiceDetails.stream() ...:

.filter(detail -> invoice.getId() == detail.getInvoiceId()) lọc các chi tiết hóa đơn để chỉ lấy những chi tiết thuộc về hóa đơn hiện tại (invoice.getId()).

.collect(Collectors.groupingBy(...)) nhóm các chi tiết hóa đơn theo tên sản phẩm (dựa trên productId trong chi tiết hóa đơn). Kết quả là một Map<String, Integer>, trong đó key là tên sản phẩm (String) và value là tổng số lượng bán được của sản phẩm đó (Integer).
+) Trong phần groupingBy:

·detail -> products.stream()... tìm kiếm tên sản phẩm thông qua productId trong chi tiết hóa đơn, sử dụng stream từ danh sách products.

·.map(Product::getName) lấy tên sản phẩm từ đối tượng Product tương ứng với productId.

·.findFirst().orElse("Unknown") trả về tên sản phẩm đầu tiên tìm được hoặc "Unknown" nếu không tìm thấy sản phẩm tương ứng.

```
customer.map(Customer::getName).orElse("Unknown")
```

map(Customer::getName) lấy tên khách hàng từ đối tượng Customer nếu khách hàng tồn tại.

orElse("Unknown") dùng để thay thế giá trị "Unknown" nếu không tìm thấy khách hàng (trường hợp Optional<Customer> là empty).

Kết quả truy vấn:

```
12. Liệt kê tất cả các khách hàng và sản phẩm của họ trong hóa đơn (bao gồm tên khách hàng và tên sản phẩm):
-------------------------------------------------------
Họ Tên                          Tên Sản Phẩm                Số Lượng
-------------------------------------------------------
Nguyễn Hoàng Long               Khoai Tây Chiên Lớn         3
Nguyễn Hoàng Long               Burger Bò                   2
Nguyễn Hoàng Long               Pizza Phô Mai               2
Nguyễn Hoàng Long               Gà Rán Phần Lớn             1
Nguyễn Hoàng Long               Burger Phô Mai              1
Trần Thị Minh Hằng              Pizza Hải Sản               1
Trần Thị Minh Hằng              Khoai Tây Chiên Nhỏ         2
Trần Thị Minh Hằng              Gà Rán Phần Nhỏ             2
Trần Thị Minh Hằng              Burger Gà                   3
Phạm Quốc Anh                   Gà Xiên Nướng               1
Phạm Quốc Anh                   Pizza Thập Cẩm              2
Phạm Quốc Anh                   Nước Ngọt Pepsi             2
Phạm Quốc Anh                   Mì Ý Sốt Gà                 3
Hoàng Mai Phương                Taco Bò                     1
Hoàng Mai Phương                Nước Chanh                  2
Hoàng Mai Phương                Súp Hải Sản                 1
Hoàng Mai Phương                Salad Trộn                  3
Ngô Huy Hoàng                   Bánh Sandwich Bò            1
Ngô Huy Hoàng                   Súp Gà                      3
Ngô Huy Hoàng                   Hotdog Xúc Xích             2
Ngô Huy Hoàng                   Taco Phô Mai                2
Lê Ngọc Tú                      Khoai Lang Chiên            1
```

13. Tìm sản phẩm có số lượng bán chạy nhất (tính tổng số lượng trong các hóa đơn).

```
SELECT P.TenSP, SUM(ID.SoLuong) AS TotalQuantity
FROM INVOICE_DETAIL ID
JOIN PRODUCT P ON ID.MASP = P.MASP
GROUP BY P.TenSP
ORDER BY TotalQuantity DESC
LIMIT 1;
```

C#:

```
//13.Tìm sản phẩm có số lượng bán chạy nhất(tính tổng số lượng trong các hóa đơn).
1 reference
public static void cau13()
{
    var result = (from id in dc1dc.INVOICE_DETAILs
                  join p in dc1dc.PRODUCTs on id.MASP equals p.MASP
                  group id by p.TenSP into grouped
                  orderby grouped.Sum(x => x.SoLuong) descending
                  select new
                  {
                      TenSP = grouped.Key,
                      TotalQuantity = grouped.Sum(x => x.SoLuong)
                  }).FirstOrDefault();
    Console.WriteLine("\n13. Tìm sản phẩm có số lượng bán chạy nhất (tính tổng số lượng trong các hóa đơn):");
    Console.WriteLine("-----------------------------------------------------------");
    Console.WriteLine("{0} :{1}", result.TenSP, result.TotalQuantity);

}
```

Giải thích :

```
var result = (from id in dc1dc.INVOICE_DETAILs
              join p in dc1dc.PRODUCTs on id.MASP equals p.MASP
```

- `var result`: Khai báo biến `result` để lưu trữ kết quả của truy vấn.
- `from id in dc1dc.INVOICE_DETAILs`: Bắt đầu truy vấn từ bảng `INVOICE_DETAILs` và đặt tên biến `id` cho từng bản ghi trong bảng này.
- `join p in dc1dc.PRODUCTs on id.MASP equals p.MASP`: Thực hiện phép nối giữa bảng `INVOICE_DETAILs` và bảng `PRODUCTs` dựa trên cột `MASP` (mã sản phẩm). Điều này giúp kết hợp thông tin sản phẩm với chi tiết hóa đơn.

```
group id by p.TenSP into grouped
```

- `group id by p.TenSP into grouped`: Nhóm các bản ghi trong bảng `INVOICE_DETAILs` theo tên sản phẩm (`TenSP`). Kết quả của việc nhóm được lưu trữ trong biến `grouped`.

```
orderby grouped.Sum(x => x.SoLuong) descending
```

- `orderby grouped.Sum(x => x.SoLuong) descending`: Tính tổng số lượng (`SoLuong`) của mỗi nhóm và sắp xếp các nhóm theo tổng số lượng bán ra giảm dần.

```
select new
{
    TenSP = grouped.Key,
    TotalQuantity = grouped.Sum(x => x.SoLuong)
}).FirstOrDefault();
```

- `select new {...}`: Tạo ra một đối tượng ẩn danh mới chứa các cột được chọn từ kết quả nhóm:
  - `TenSP = grouped.Key`: Tên sản phẩm (là khóa của nhóm).
  - `TotalQuantity = grouped.Sum(x => x.SoLuong)`: Tổng số lượng sản phẩm bán ra của nhóm.

- `FirstOrDefault()`: Lấy sản phẩm có số lượng bán ra nhiều nhất (sản phẩm đầu tiên trong danh sách kết quả).

Java:

```java
13.Tìm sản phẩm có số lượng bán chạy nhất (tính tổng số lượng trong các hóa đơn).
public static void cau13() {
    Map<String, Integer> productQuantities = invoiceDetails.stream()
            .collect(Collectors.groupingBy(
                    detail -> products.stream().filter(product -> product.getId() == detail.getProductId())
                            .map(Product::getName).findFirst().orElse("Unknow"),
                    Collectors.summingInt(InvoiceDetail::getQuantity)));

    productQuantities.entrySet().stream().max(Map.Entry.comparingByValue()).ifPresent(topProduct -> System.out
            .println("Sản phẩm: " + topProduct.getKey() + ", Số lượng: " + topProduct.getValue()));
}
```

Giải thích :

detail -> products.stream().filter(product -> product.getId() == detail.getProductId()):

· Đối với mỗi InvoiceDetail, đoạn mã này tìm kiếm trong danh sách products (danh sách sản phẩm) để tìm sản phẩm có id khớp với productId trong chi tiết hóa đơn.

· products.stream() tạo một stream từ danh sách sản phẩm, và .filter(product -> product.getId() == detail.getProductId()) lọc các sản phẩm có id trùng với productId của chi tiết hóa đơn.

.map(Product::getName)

Sau khi tìm được sản phẩm phù hợp, .map(Product::getName) lấy tên sản phẩm (dự đoán rằng Product có phương thức getName() trả về tên sản phẩm).

.findFirst().orElse("Unknow")

.findFirst() trả về sản phẩm đầu tiên tìm được (nếu có). Nếu không tìm thấy sản phẩm (trong trường hợp danh sách products không chứa sản phẩm với id khớp), sử dụng .orElse("Unknow") để thay thế bằng tên "Unknow".

Collectors.summingInt(InvoiceDetail::getQuantity)

Sau khi nhóm theo tên sản phẩm, Collectors.summingInt(InvoiceDetail::getQuantity) sẽ tính tổng số lượng của mỗi sản phẩm trong nhóm. InvoiceDetail::getQuantity là phương thức lấy số lượng sản phẩm trong mỗi chi tiết hóa đơn.

productQuantities.entrySet().stream():

.entrySet() trả về một tập hợp các cặp key-value trong Map. Trong trường hợp này, key là tên sản phẩm, value là tổng số lượng sản phẩm bán được.

max(Map.Entry.comparingByValue()):

.max() tìm ra phần tử lớn nhất trong stream.

Map.Entry.comparingByValue() là một Comparator giúp so sánh giá trị (số lượng sản phẩm) trong mỗi cặp key-value của Map.
Kết quả là một Optional<Map.Entry<String, Integer>>, chứa sản phẩm có số lượng bán cao nhất. Nếu không có phần tử nào trong productQuantities, kết quả sẽ là Optional.empty().

Kết quả truy vấn:

```
13. Tìm sản phẩm có số lượng bán chạy nhất (tính tổng số lượng trong các hóa đơn):
---------------------------------------------------------
Sản phẩm: Gà Rán Phần Nhỏ, Số lượng: 36
```

14. Liệt kê các khuyến mãi đã áp dụng cho khách hàng và điều kiện khuyến mãi.

```
SELECT C.HoTen, P.MAKM, P.DieuKien
FROM CUSTOMER_PROMOTION CP
JOIN CUSTOMER C ON CP.MAKH = C.MAKH
JOIN PROMOTION P ON CP.MAKM = P.MAKM
ORDER BY C.HoTen;
```

C#:

```
////14.Liệt kê các khuyến mãi đã áp dụng cho khách hàng và điều kiện khuyến mãi.
1 reference
public static void cau14()
{
    var result = from cp in dc1dc.CUSTOMER_PROMOTIONs
                 join c in dc1dc.CUSTOMERs on cp.MAKH equals c.MAKH
                 join p in dc1dc.PROMOTIONs on cp.MAKM equals p.MAKM
                 orderby c.HoTen
                 select new
                 {
                     HoTen = c.HoTen,
                     MAKM = p.MAKM,
                     DieuKien = p.DieuKien
                 };
    Console.WriteLine("\n14. Liệt kê các khuyến mãi đã áp dụng cho khách hàng và điều kiện khuyến mãi:");
    Console.WriteLine("-----------------------------------------------------------");
    Console.WriteLine("{0,-30} {1,-10} {2,-50}", "Họ Và Tên", "Ma KM", "Điều kiện");
    Console.WriteLine("-----------------------------------------------------------");
    result.ToList().ForEach(x => Console.WriteLine("{0,-30} {1,-10} {2,-50}",x.HoTen,x.MAKM,x.DieuKien));

}
```

Giải thích

```
var result = from cp in dc1dc.CUSTOMER_PROMOTIONs
```

- `var result`: Khai báo biến `result` để lưu trữ kết quả của truy vấn.
- `from cp in dc1dc.CUSTOMER_PROMOTIONs`: Bắt đầu truy vấn từ bảng `CUSTOMER_PROMOTIONs` và đặt tên biến `cp` cho từng bản ghi trong bảng này.

```
join c in dc1dc.CUSTOMERs on cp.MAKH equals c.MAKH
```

- Thực hiện phép nối giữa bảng `CUSTOMER_PROMOTIONs` và bảng `CUSTOMERs` dựa trên cột `MAKH` (mã khách hàng). Điều này giúp lấy thông tin khách hàng cho mỗi khuyến mãi.

```
join p in dc1dc.PROMOTIONs on cp.MAKM equals p.MAKM
```

- Thực hiện phép nối giữa bảng `CUSTOMER_PROMOTIONs` và bảng `PROMOTIONs` dựa trên cột `MAKM` (mã khuyến mãi). Điều này giúp lấy thông tin khuyến mãi cho mỗi khách hàng.

```
orderby c.HoTen
```

- Sắp xếp kết quả theo họ tên khách hàng (`HoTen`) theo thứ tự bảng chữ cái.

```
select new
{
    HoTen = c.HoTen,
    MAKM = p.MAKM,
    DieuKien = p.DieuKien
};
```

- Tạo ra một đối tượng ẩn danh mới chứa các cột được chọn từ kết quả nối:
  - `HoTen = c.HoTen`: Họ tên khách hàng từ bảng `CUSTOMERs`.
  - `MAKM = p.MAKM`: Mã khuyến mãi từ bảng `PROMOTIONs`.
  - `DieuKien = p.DieuKien`: Điều kiện khuyến mãi từ bảng `PROMOTIONs`.

Java:

```java
// 14.Liệt kê các khuyến mãi đã áp dụng cho khách hàng và điều kiện khuyến mãi.
public static void cau14() {
    System.out.printf("%-30s %-10s %-50s %n", "Họ Và Tên","Ma KM","Điều kiện");
    System.out.println("-----------------------------------------------------");
    customerPromotions.stream().forEach(cp -> {
        Optional<Customer> customer = customers.stream()
                .filter(c -> c.getId() == cp.getCustomerId()).findFirst();
        Optional<Promotion> promotion = promotions.stream()
                .filter(p -> p.getId() == cp.getPromotionId())
                .findFirst();
        if (customer.isPresent() && promotion.isPresent()) {
            {
                System.out.printf("%-30s %-10s %-50s %n",
                        customer.get().getName(),
                        promotion.get().getId(),
                        promotion.get().getCondition());
            }
        }
    });
}
```

Giải thích:

customerPromotions.stream().forEach(cp -> { ... });

.forEach(cp -> {...}) lặp qua từng đối tượng CustomerPromotion và thực hiện các thao tác bên trong lambda expression (biểu thức lambda).

Optional<Customer> customer = customers.stream()

.filter(c -> c.getId() == cp.getCustomerId()).findFirst();

.filter(c -> c.getId() == cp.getCustomerId()) lọc các khách hàng có id trùng với customerId trong CustomerPromotion.
.findFirst() trả về phần tử đầu tiên tìm được khớp, trong trường hợp này là một đối tượng Optional<Customer>, có thể chứa khách hàng tìm được hoặc Optional.empty() nếu không tìm thấy.

Optional<Promotion> promotion = promotions.stream()
.filter(p -> p.getId() == cp.getPromotionId()).findFirst();

ương tự như bước trước, tìm kiếm khuyến mãi trong danh sách promotions bằng cách lọc theo promotionId và trả về một Optional<Promotion>.

if (customer.isPresent() && promotion.isPresent()) { ... }

· customer.isPresent() kiểm tra xem có khách hàng nào khớp với customerId không (tức là nếu Optional<Customer> không rỗng).
· promotion.isPresent() kiểm tra xem có khuyến mãi nào khớp với promotionId không.
Nếu cả hai đều tồn tại (khách hàng và khuyến mãi đều được tìm thấy), đoạn mã bên trong if sẽ được thực thi.

Kết quả truy vấn:

```
14. Liệt kê các khuyến mãi đã áp dụng cho khách hàng và điều kiện khuyến mãi:
-----------------------------------------------------
Họ Và Tên                       Ma KM       Điều kiện
-----------------------------------------------------
Nguyễn Hoàng Long               1           Áp dụng cho đơn hàng trên 500,000 VND
Trần Thị Minh Hằng              1           Áp dụng cho đơn hàng trên 500,000 VND
Phạm Quốc Anh                   2           Áp dụng cho đơn hàng trên 1,000,000 VND
Hoàng Mai Phương                3           Giảm giá 10% cho tất cả các sản phẩm
Ngô Huy Hoàng                   3           Giảm giá 10% cho tất cả các sản phẩm
Lê Ngọc Tú                      4           Áp dụng cho khách hàng có điểm tích lũy trên 100
Vũ Thị Thu Hương                5           Khuyến mãi ngày lễ 30/4
Nguyễn Thanh Phong              2           Áp dụng cho đơn hàng trên 1,000,000 VND
Đặng Hải Nam                    1           Áp dụng cho đơn hàng trên 500,000 VND
Bùi Thanh Hải                   4           Áp dụng cho khách hàng có điểm tích lũy trên 100
Nguyễn Văn Hùng                 3           Giảm giá 10% cho tất cả các sản phẩm
Trần Thị Ngọc Hà                5           Khuyến mãi ngày lễ 30/4
Phạm Hoàng Anh                  4           Áp dụng cho khách hàng có điểm tích lũy trên 100
Lê Minh Tuấn                    2           Áp dụng cho đơn hàng trên 1,000,000 VND
Ngô Bảo Trâm                    1           Áp dụng cho đơn hàng trên 500,000 VND
Hoàng Thanh Hương               5           Khuyến mãi ngày lễ 30/4
```

15. Tính tổng số tiền mà một khách hàng đã chi tiêu trong tất cả các hóa đơn (dựa trên mã khách hàng).

```
SELECT MAKH, SUM(TongTien) AS TotalSpent
FROM INVOICE
WHERE MAKH = 1
GROUP BY MAKH;
```

C#:

```
////15.Tính tổng số tiền mà một khách hàng đã chi tiêu trong tất cả các hóa đơn(dựa trên mã khách hàng).
1 reference
public static void cau15(int MAKH)
{
    var result = (from i in dc1dc.INVOICEs
                  where i.MAKH == MAKH
                  group i by i.MAKH into g
                  select new
                  {
                      MAKH = g.Key,
                      TONGTIEN = g.Sum(x => x.TongTien)
                  }).FirstOrDefault();

    if (result != null)
    {
        Console.WriteLine("\n15. Tính tổng số tiền mà một khách hàng đã chi tiêu trong tất cả các hóa đơn (dựa trên mã khách hàng):");
        Console.WriteLine("-----------------------------------------------------------");
        Console.WriteLine($"Khách hàng: {result.MAKH}");
        Console.WriteLine($"Tổng số tiền: {result.TONGTIEN:F2}");
    }
    else
    {
        Console.WriteLine("Không tìm thấy thông tin khách hàng.");
    }
```

Giải thích :

```
var result = (from i in dc1dc.INVOICEs
```

- `var result`: Khai báo biến `result` để lưu trữ kết quả của truy vấn.
- `from i in dc1dc.INVOICEs`: Bắt đầu truy vấn từ bảng `INVOICEs` và đặt tên biến `i` cho từng bản ghi trong bảng này.

```
where i.MAKH == MAKH\
```

- `where i.MAKH == MAKH`: Lọc ra các hóa đơn có mã khách hàng (`MAKH`) bằng với giá trị `MAKH` được cung cấp.

```
group i by i.MAKH into g
```

- `group i by i.MAKH into g`: Nhóm các hóa đơn theo mã khách hàng (`MAKH`). Kết quả nhóm được lưu trữ trong biến `g`.

```
select new
  {
     MAKH = g.Key,
     TONGTIEN = g.Sum(x => x.TongTien)
  }).FirstOrDefault();
```

- `select new {...}`: Tạo ra một đối tượng ẩn danh mới chứa các cột được chọn từ kết quả nhóm:
  - `MAKH = g.Key`: Mã khách hàng (là khóa của nhóm).
  - `TONGTIEN = g.Sum(x => x.TongTien)`: Tính tổng số tiền (`TongTien`) của các hóa đơn trong nhóm.
- `FirstOrDefault()`: Lấy bản ghi đầu tiên trong danh sách kết quả hoặc giá trị mặc định nếu danh sách trống.

Java:

```
15.Tính tổng số tiền mà một khách hàng đã chi tiêu trong tất cả các hóa đơn (dựa trên mã khách hàng).
public static void cau15(int customerId) {
    double totalSpentByCustomer = invoices.stream().filter(invoice -> invoice.getCustomerId() == customerId)
            .mapToDouble(Invoice::getTotalAmount).sum();
    System.out.println("Khách hàng " + customerId + " đã chi tiêu: " + totalSpentByCustomer);
}
```

Giải thích :

.filter(invoice -> invoice.getCustomerId() == customerId):

· .filter(...) là một phép toán trong Stream API, dùng để lọc các phần tử trong stream.
· Biểu thức lambda invoice -> invoice.getCustomerId() == customerId sẽ kiểm tra xem customerId của hóa đơn (thông qua invoice.getCustomerId()) có bằng customerId mà bạn đang tìm kiếm không. Nếu có, hóa đơn này sẽ được giữ lại trong stream; nếu không, nó sẽ bị loại bỏ.
· Kết quả của thao tác này là một stream chứa tất cả các hóa đơn của một khách hàng cụ thể.

.mapToDouble(Invoice::getTotalAmount):

· .mapToDouble(...) chuyển đổi mỗi đối tượng Invoice trong stream thành một giá trị kiểu double bằng cách sử dụng phương thức getTotalAmount() của đối tượng Invoice. Phương thức này trả về tổng số tiền của hóa đơn (loại double).
· Kết quả của .mapToDouble() là một stream các giá trị kiểu double tương ứng với tổng số tiền của từng hóa đơn.

.sum():

Sau khi stream đã được chuyển thành các giá trị double (số tiền của từng hóa đơn), .sum() sẽ tính tổng tất cả các giá trị này và trả về kết quả là tổng số tiền mà khách hàng đã chi tiêu.

Kết quả truy vấn:

```
15. Tính tổng số tiền mà một khách hàng đã chi tiêu trong tất cả các hóa đơn (dựa trên mã khách hàng):
---------------------------------------------------------
Khách hàng 1 đã chi tiêu: 330000.0
```

16. Liệt kê tất cả các hóa đơn có ít nhất một sản phẩm có trạng thái "Hết hàng".

```
SELECT DISTINCT I.MAHD, I.ThoiGianTT, I.MAKH
FROM INVOICE I
JOIN INVOICE_DETAIL ID ON I.MAHD = ID.MAHD
JOIN PRODUCT P ON ID.MASP = P.MASP
WHERE P.TinhTrang= 'Hết hàng';
```

C#:

```
////16.Liệt kê tất cả các hóa đơn có ít nhất một sản phẩm có trạng thái "Hết hàng".
1 reference
public static void cau16()
{
    var result = (from i in dc1dc.INVOICEs
                  join id in dc1dc.INVOICE_DETAILs on i.MAHD equals id.MAHD
                  join p in dc1dc.PRODUCTs on id.MASP equals p.MASP
                  where p.TinhTrang == "Hết hàng"
                  select new
                  {
                      MAHD = i.MAHD,
                      ThoiGianTT = i.ThoiGianTT,
                      MAKH = i.MAKH
                  }).Distinct().ToList();
    Console.WriteLine("\n16. Liệt kê tất cả các hóa đơn có ít nhất một sản phẩm có trạng thái \"Hết hàng\":");
    Console.WriteLine("-----------------------------------------------------");
    Console.WriteLine("{0,-7} {1,-30} {2,-7}", "Ma HD", "Thời Gian TT", "Ma KH");
    Console.WriteLine("-----------------------------------------------------");
    foreach (var detail in result)
    {
        Console.WriteLine("{0,-7} {1,-30} {2,-7}", detail.MAHD, detail.ThoiGianTT, detail.MAKH);
    }
}
```

Giải thích :

```
var result = (from i in dc1dc.INVOICEs
```

- `var result`: Khai báo biến `result` để lưu trữ kết quả của truy vấn.
- `from i in dc1dc.INVOICEs`: Bắt đầu truy vấn từ bảng `INVOICEs` và đặt tên biến `i` cho từng bản ghi trong bảng này.

```
join id in dc1dc.INVOICE_DETAILs on i.MAHD equals id.MAHD
```

- Thực hiện phép nối giữa bảng `INVOICEs` và bảng `INVOICE_DETAILs` dựa trên cột `MAHD` (mã hóa đơn). Điều này giúp kết hợp thông tin của hóa đơn với các chi tiết hóa đơn.

```
join p in dc1dc.PRODUCTs on id.MASP equals p.MASP
```

- Thực hiện phép nối giữa bảng `INVOICE_DETAILs` và bảng `PRODUCTs` dựa trên cột `MASP` (mã sản phẩm). Điều này giúp lấy thông tin sản phẩm cho mỗi chi tiết hóa đơn.

```
where p.TinhTrang == "Hết hàng"
```

- `where p.TinhTrang == "Hết hàng"`: Lọc ra các bản ghi chỉ chứa các sản phẩm có tình trạng là "Hết hàng".

```
select new
{
    MAHD = i.MAHD,
    ThoiGianTT = i.ThoiGianTT,
    MAKH = i.MAKH
}
```

- Tạo ra một đối tượng ẩn danh mới chứa các cột được chọn từ kết quả nối:
  - `MAHD = i.MAHD`: Mã hóa đơn từ bảng `INVOICEs`.

- `ThoiGianTT = i.ThoiGianTT`: Thời gian thanh toán từ bảng `INVOICEs`.
- `MAKH = i.MAKH`: Mã khách hàng từ bảng `INVOICEs`.

Distinct().ToList();

- **`Distinct()`: Loại bỏ các bản ghi trùng lặp từ kết quả.**
- **`ToList()`: Chuyển kết quả thành một danh sách.**

Java:

```
16.Liệt kê tất cả các hóa đơn có ít nhất một sản phẩm có trạng thái "Hết hàng".
public static void cau16() {
    System.out.printf("%-7s %-30s %-7s %n","Ma HD","Thời Gian TT","Ma KH");
    System.out.println("-----------------------------------------------------");
    invoices.stream().filter(invoice -> invoiceDetails.stream()
            .filter(detail -> detail.getInvoiceId() == invoice.getId())
            .anyMatch(detail -> products.stream()
                    .filter(product -> product.getId() == detail.getProductId())
                    .anyMatch(product -> product.getStatus().equalsIgnoreCase("Hết hàng"))))
            .forEach(detail->System.out.printf("%-7s %-30s %-7s %n",
                    detail.getId(),
                    detail.getPaymentTime(),
                    detail.getCustomerId()));

}
```

Giải thích :

invoiceDetails.stream().filter(detail -> detail.getInvoiceId() == invoice.getId()):

.filter(detail -> detail.getInvoiceId() == invoice.getId()) lọc các chi tiết hóa đơn sao cho chỉ giữ lại những chi tiết mà invoiceId của chúng khớp với id của hóa đơn hiện tại (tức là các chi tiết hóa đơn thuộc về hóa đơn này).

.anyMatch(detail -> products.stream()...):

.anyMatch(...) kiểm tra xem có bất kỳ chi tiết hóa đơn nào thỏa mãn điều kiện bên trong.
Điều kiện bên trong là lọc qua danh sách products (sản phẩm) để kiểm tra xem sản phẩm nào có productId khớp với sản phẩm trong detail của hóa đơn.
· .filter(product -> product.getId() == detail.getProductId()) lọc ra sản phẩm có id khớp với productId trong chi tiết hóa đơn.
· .anyMatch(product -> product.getStatus().equalsIgnoreCase("Hết hàng")) kiểm tra xem sản phẩm có trạng thái "Hết hàng" hay không. Nếu tìm thấy ít nhất một sản phẩm có trạng thái "Hết hàng", .anyMatch(...) sẽ trả về true.

Kết quả truy vấn:

```
16. Liệt kê tất cả các hóa đơn có ít nhất một sản phẩm có trạng thái "Hết hàng".:
-----------------------------------------------------
Ma HD   Thời Gian TT                    Ma KH
-----------------------------------------------------
3       2023-01-03 12:45:00.0           3
4       2023-01-04 09:15:00.0           4
5       2023-01-05 14:50:00.0           5
10      2023-01-10 19:00:00.0           10
11      2023-01-11 08:30:00.0           11
14      2023-01-14 11:00:00.0           14
15      2023-01-15 14:30:00.0           15
17      2023-01-17 12:30:00.0           17
20      2023-01-20 19:00:00.0           20
30      2023-01-30 19:00:00.0           30
32      2023-02-02 12:30:00.0           32
34      2023-02-04 15:20:00.0           34
35      2023-02-05 16:45:00.0           35
37      2023-02-07 19:30:00.0           37
40      2023-02-10 09:15:00.0           40
41      2023-02-11 10:20:00.0           41
44      2023-02-14 15:30:00.0           44
45      2023-02-15 16:40:00.0           45
```

17. Tính tổng tiền của từng hóa đơn và tính toán tiền giảm giá nếu có (giảm giá theo khuyến mãi).

```sql
SELECT I.MAHD,
    I.MAKH,
    I.TongTien AS TotalAmount,
    COALESCE(P.MucGiamGia, 0) AS Discount,
    I.TongTien - COALESCE(P.MucGiamGia, 0) AS FinalAmount
FROM INVOICE I
LEFT JOIN CUSTOMER_PROMOTION CP ON I.MAKH = CP.MAKH
LEFT JOIN PROMOTION P ON CP.MAKM = P.MAKM
GROUP BY I.MAHD, I.MAKH, I.TongTien;
```

C#:

```csharp
////17.Tính tổng tiền của từng hóa đơn và tính toán tiền giảm giá nếu có(giảm giá theo khuyến mãi).
1 reference
public static void cau17()
{
    var result = (from i in dc1dc.INVOICEs
                  join cp in dc1dc.CUSTOMER_PROMOTIONs on i.MAKH equals cp.MAKH into cpGroup
                  from cp in cpGroup.DefaultIfEmpty()
                  join p in dc1dc.PROMOTIONs on cp.MAKM equals p.MAKM into pGroup
                  from p in pGroup.DefaultIfEmpty()
                  group new { i, p } by new { i.MAHD, i.MAKH, i.TongTien } into grouped
                  select new
                  {
                      MAHD = grouped.Key.MAHD,
                      MAKH = grouped.Key.MAKH,
                      TotalAmount = grouped.Key.TongTien,
                      Discount = grouped.Max(x => x.p != null ? x.p.MucGiamGia : 0),
                      FinalAmount = grouped.Key.TongTien - (grouped.Max(x => x.p != null ? x.p.MucGiamGia : 0))
                  }).ToList();
    Console.WriteLine("\n17.Tính tổng tiền của từng hóa đơn và tính toán tiền giảm giá nếu có(giảm giá theo khuyến mãi)");
    Console.WriteLine("-----------------------------------------------------");
    Console.WriteLine("{0,-10} {1,-10} {2,-15} {3,-10} {4,-15}", "MAHD", "MAKH", "TotalAmount", "Discount", "FinalAmount");
    Console.WriteLine("-----------------------------------------------------------------------------");
    foreach (var item in result) { Console.WriteLine("{0,-10} {1,-10} {2,-15:F2} {3,-10:F2} {4,-15:F2}",
        item.MAHD, item.MAKH, item.TotalAmount, item.Discount, item.FinalAmount); }
}
```

Giải thích :

```
var result = (from i in dc1dc.INVOICEs
```

- `var result`: Khai báo biến `result` để lưu trữ kết quả của truy vấn.

- `from i in dc1dc.INVOICEs`: Bắt đầu truy vấn từ bảng `INVOICEs` và đặt tên biến `i` cho từng bản ghi trong bảng này.

```
  join cp in dc1dc.CUSTOMER_PROMOTIONs on i.MAKH equals cp.MAKH into cpGroup
  from cp in cpGroup.DefaultIfEmpty()
```

- `join cp in dc1dc.CUSTOMER_PROMOTIONs on i.MAKH equals cp.MAKH into cpGroup`: Thực hiện phép nối giữa bảng `INVOICEs` và bảng `CUSTOMER_PROMOTIONs` dựa trên cột `MAKH` (mã khách hàng) và lưu kết quả vào nhóm `cpGroup`.
- `from cp in cpGroup.DefaultIfEmpty()`: Lấy từng bản ghi từ `cpGroup`, nếu không có kết quả, sẽ sử dụng giá trị mặc định `null`.

```
  join p in dc1dc.PROMOTIONs on cp.MAKM equals p.MAKM into pGroup
  from p in pGroup.DefaultIfEmpty()
```

- `join p in dc1dc.PROMOTIONs on cp.MAKM equals p.MAKM into pGroup`: Thực hiện phép nối giữa bảng `CUSTOMER_PROMOTIONs` và bảng `PROMOTIONs` dựa trên cột `MAKM` (mã khuyến mãi) và lưu kết quả vào nhóm `pGroup`.
- `from p in pGroup.DefaultIfEmpty()`: Lấy từng bản ghi từ `pGroup`, nếu không có kết quả, sẽ sử dụng giá trị mặc định `null`.

```
  group new { i, p } by new { i.MAHD, i.MAKH, i.TongTien } into grouped
```

- `group new { i, p } by new { i.MAHD, i.MAKH, i.TongTien } into grouped`: Nhóm các bản ghi theo mã hóa đơn (`MAHD`), mã khách hàng (`MAKH`) và tổng tiền (`TongTien`). Kết quả nhóm được lưu trữ trong biến `grouped`.

```
 select new
  {
     MAHD = grouped.Key.MAHD,
     MAKH = grouped.Key.MAKH,
     TotalAmount = grouped.Key.TongTien,
     Discount = grouped.Max(x => x.p != null ? x.p.MucGiamGia : 0),
     FinalAmount = grouped.Key.TongTien - (grouped.Max(x => x.p != null ? x.p.MucGiamGia : 0))
  }).ToList();
```

- `select new {...}`: Tạo ra một đối tượng ẩn danh mới chứa các cột được chọn từ kết quả nhóm:

- `MAHD = grouped.Key.MAHD`: Mã hóa đơn từ khóa của nhóm.
- `MAKH = grouped.Key.MAKH`: Mã khách hàng từ khóa của nhóm.
- `TotalAmount = grouped.Key.TongTien`: Tổng số tiền từ khóa của nhóm.
- `Discount = grouped.Max(x => x.p != null ? x.p.MucGiamGia : 0)`: Tính mức giảm giá lớn nhất từ bảng `PROMOTIONs`, nếu có giá trị, ngược lại giá trị mặc định là 0.
- `FinalAmount = grouped.Key.TongTien - (grouped.Max(x => x.p != null ? x.p.MucGiamGia : 0))`: Tính số tiền cuối cùng sau khi trừ đi mức giảm giá lớn nhất.
- `ToList()`: Chuyển kết quả thành một danh sách.

Java:

```
17.Tính tổng tiền của từng hóa đơn và tính toán tiền giảm giá nếu có (giảm giá theo khuyến mãi).
public static void cau17() {
    System.out.printf("%-7s %-7s %-20s %-20s %-20s %n",
            "MaHD","MaKH","Tổng Tiền","Giảm Giá","Tổng tiền sau giảm");
    System.out.println("-----------------------------------------------------");
    Map<Integer, Double> promotionMap = promotions.stream()
            .collect(Collectors.toMap(Promotion::getId, Promotion::getDiscount));
    invoices.forEach(invoice -> {
        double totalAmount = invoice.getTotalAmount();
        double discount = customerPromotions.stream()
                .filter(cp -> cp.getCustomerId() == invoice.getCustomerId())
                .mapToDouble(cp -> promotionMap.getOrDefault(cp.getPromotionId(), 0.0)).sum();
        double finalAmount = totalAmount - (discount );
        System.out.printf("%-7s %-7s %-20s %-20s %-20s %n",
                invoice.getId(),
                invoice.getCustomerId(),
                totalAmount,discount,finalAmount);
    });
}
```

Giải thích:

.filter(cp -> cp.getCustomerId() == invoice.getCustomerId()):

Lọc ra các chương trình khuyến mãi dành cho khách hàng của hóa đơn hiện tại (so sánh getCustomerId() của customerPromotions với invoice.getCustomerId()).

.mapToDouble(cp ->

promotionMap.getOrDefault(cp.getPromotionId(), 0.0)): Chuyển mỗi chương trình khuyến mãi (customer promotion) thành giá trị giảm giá:

- promotionMap.getOrDefault(cp.getPromotionId(), 0.0) lấy giá trị giảm giá từ promotionMap dựa trên promotionId. Nếu không tìm thấy mã khuyến mãi trong promotionMap, mặc định là 0.0 (không có giảm giá).
- .sum(): Tính tổng giá trị giảm giá của tất cả các khuyến mãi cho khách hàng của hóa đơn hiện tại.

double finalAmount = totalAmount - discount;

tính toán số tiền cuối cùng phải trả cho hóa đơn, bằng tổng tiền trừ đi tổng số tiền giảm giá.

Kết quả truy vấn:

```
17. Tính tổng tiền của từng hóa đơn và tính toán tiền giảm giá nếu có (giảm giá theo khuyến mãi):
-----------------------------------------------------------
MaHD    MaKH    Tổng Tiền           Giảm Giá            Tổng tiền sau giảm
-----------------------------------------------------------
1       1       150000.0            50000.0             100000.0
2       2       200000.0            50000.0             150000.0
3       3       120000.0            100000.0            20000.0
4       4       180000.0            10.0                179990.0
5       5       250000.0            10.0                249990.0
6       6       300000.0            150000.0            150000.0
7       7       175000.0            70000.0             105000.0
8       8       220000.0            100000.0            120000.0
9       9       280000.0            50000.0             230000.0
10      10      150000.0            150000.0            0.0
11      11      140000.0            10.0                139990.0
12      12      240000.0            70000.0             170000.0
13      13      190000.0            150000.0            40000.0
14      14      150000.0            100000.0            50000.0
15      15      210000.0            50000.0             160000.0
16      16      270000.0            70000.0             200000.0
17      17      160000.0            10.0                159990.0
18      18      210000.0            150000.0            60000.0
19      19      230000.0            100000.0            130000.0
20      20      180000.0            50000.0             130000.0
21      21      160000.0            70000.0             90000.0
22      22      210000.0            10.0                209990.0
23      23      190000.0            150000.0            40000.0
24      24      150000.0            100000.0            50000.0
25      25      210000.0            50000.0             160000.0
26      26      270000.0            70000.0             200000.0
27      27      160000.0            10.0                159990.0
28      28      210000.0            150000.0            60000.0
29      29      230000.0            100000.0            130000.0
30      30      180000.0            50000.0             130000.0
31      31      140000.0            0.0                 140000.0
```

18. Tìm tất cả các nhân viên đã xử lý hóa đơn có tổng tiền vượt quá 200,000 VND.

```
SELECT DISTINCT E.HoTen
FROM INVOICE I
JOIN EMPLOYEE E ON I.MANV = E.MANV
WHERE I.TongTien > 200000;
```

C#:

```
//18.Tìm tất cả các nhân viên đã xử lý hóa đơn có tổng tiền vượt quá 200,000 VND.
1 reference
public static void cau18()
{
    var result = (from i in dc1dc.INVOICEs
                  join e in dc1dc.EMPLOYEEs on i.MANV equals e.MANV
                  where i.TongTien > 200000
                  select e.HoTen).Distinct().ToList();
    Console.WriteLine("\n18. Tìm tất cả các nhân viên đã xử lý hóa đơn có tổng tiền vượt quá 200,000 VND:");
    Console.WriteLine("-----------------------------------------------------------");

    foreach(var e in result)
    {
        Console.WriteLine(e);
    }
```

Giải thích :

```
var result = (from i in dc1dc.INVOICEs
```

- `var result`: Khai báo biến `result` để lưu trữ kết quả của truy vấn.
- `from i in dc1dc.INVOICEs`: Bắt đầu truy vấn từ bảng `INVOICEs` và đặt tên biến `i` cho từng bản ghi trong bảng này.

```
join e in dc1dc.EMPLOYEEs on i.MANV equals e.MANV
```

- Thực hiện phép nối giữa bảng `INVOICEs` và bảng `EMPLOYEEs` dựa trên cột `MANV` (mã nhân viên). Điều này giúp kết hợp thông tin của nhân viên với các hóa đơn mà họ đã xử lý.

```
where i.TongTien > 200000
```

- `where i.TongTien > 200000`: Lọc ra các hóa đơn có tổng tiền (`TongTien`) lớn hơn 200,000.

```
select e.HoTen
```

- `select e.HoTen`: Chọn họ tên nhân viên (`HoTen`) từ bảng `EMPLOYEEs`.

}).Distinct().ToList();
- `Distinct()`: Loại bỏ các tên nhân viên trùng lặp từ kết quả.
- `ToList()`: Chuyển kết quả thành một danh sách.

Java:

```java
18.Tìm tất cả các nhân viên đã xử lý hóa đơn có tổng tiền vượt quá 200,000 VND.
public static void cau18(double amount) {
    System.out.printf("%-20s %-30s %-25s %-15s %-20s %n",
            "Mã Nhân Viên","Họ Và Tên","Chức Vụ","Số Điện Thoại","Ngày Bắt Đầu");
    System.out.println("-------------------------------------------------------");

    employees.stream()
            .filter(employee -> invoices.stream().filter(
                    invoice -> invoice.getEmployeeId() == employee.getId()
                            && invoice.getTotalAmount() > amount)
                    .count() > 0)
            .forEach(e -> System.out.printf("%-20s %-30s %-25s %-15s %-20s %n",
                    e.getId(),
                    e.getName(),
                    e.getPosition(),
                    e.getPhone(),
                    e.getStartDate()));
}
```

Giải thích :

.filter(employee -> invoices.stream().filter(...)...):

Lọc danh sách nhân viên để chỉ giữ lại các nhân viên có ít nhất một hóa đơn có tổng tiền lớn hơn amount.

.filter(invoice -> invoice.getEmployeeId() == employee.getId() && invoice.getTotalAmount() > amount):

· Lọc ra các hóa đơn mà có EmployeeId (mã nhân viên) bằng với employee.getId(), tức là các hóa đơn thuộc về nhân viên hiện tại trong dòng employee.
· invoice.getTotalAmount() > amount: Lọc thêm để chỉ giữ lại những hóa đơn có tổng tiền lớn hơn amount được chỉ định.

.count() > 0:

· Sau khi lọc các hóa đơn của nhân viên hiện tại, phương thức count() sẽ trả về số lượng hóa đơn thỏa mãn điều kiện trên.
· count() > 0: Nếu số lượng hóa đơn thỏa mãn điều kiện lớn hơn 0 (nghĩa là nhân viên có ít nhất một hóa đơn thỏa mãn điều kiện), thì nhân viên này sẽ được giữ lại trong danh sách cuối cùng.

Kết quả truy vấn:

```
18. Tìm tất cả các nhân viên đã xử lý hóa đơn có tổng tiền vượt quá 200,000 VND:
-----------------------------------------------------
Mã Nhân Viên        Họ Và Tên                   Chức Vụ                 Số Điện Thoại   Ngày Bắt Đầu
-----------------------------------------------------
2                   Lê Thị Hương                Nhân viên bán hàng      0902223344      2021-05-20
3                   Trần Quốc Bảo               Quản lý                 0903334455      2020-10-01
5                   Nguyễn Minh Đức             Nhân viên bán hàng      0905556677      2022-08-15
6                   Lê Ngọc Hà                  Quản lý                 0906667788      2020-01-10
8                   Phạm Quốc Hùng              Nhân viên bán hàng      0908889900      2023-06-10
9                   Lê Thanh Tú                 Nhân viên bán hàng      0909990011      2021-09-25
```

19. Liệt kê tên sản phẩm và số lần sản phẩm đó xuất hiện trong hóa đơn.

```
SELECT P.TenSP, COUNT(ID.MASP) AS SoLanXuatHien
FROM INVOICE_DETAIL ID
JOIN PRODUCT P ON ID.MASP = P.MASP
GROUP BY P.TenSP
ORDER BY SoLanXuatHien DESC;
```

C#:

```
//19.Liệt kê tên sản phẩm và số lần sản phẩm đó xuất hiện trong hóa đơn.
1 reference
public static void cau19()
{
    var result = (from id in dc1dc.INVOICE_DETAILs
                  join p in dc1dc.PRODUCTs on id.MASP equals p.MASP
                  group id by p.TenSP into grouped
                  orderby grouped.Count() descending
                  select new
                  {
                      TenSP = grouped.Key,
                      SoLanXuatHien = grouped.Count()
                  }).ToList();
    Console.WriteLine("\n19. Liệt kê tên sản phẩm và số lần sản phẩm đó xuất hiện trong hóa đơn:");
    Console.WriteLine("-----------------------------------------------------");
    Console.WriteLine("{0,-35} {1,-15}", "Tên Sản Phẩm", "Số lượng");
    Console.WriteLine("-----------------------------------------------------");
    foreach( var e in result)
    {
        Console.WriteLine("{0,-35} {1,-15}", e.TenSP,e.SoLanXuatHien);
    }
```

Giải thích :

```
var result = (from id in dc1dc.INVOICE_DETAILs
              join p in dc1dc.PRODUCTs on id.MASP equals p.MASP
```

- `var result`: Khai báo biến `result` để lưu trữ kết quả của truy vấn.
- `from id in dc1dc.INVOICE_DETAILs`: Bắt đầu truy vấn từ bảng `INVOICE_DETAILs` và đặt tên biến `id` cho từng bản ghi trong bảng này.
- `join p in dc1dc.PRODUCTs on id.MASP equals p.MASP`: Thực hiện phép nối giữa bảng `INVOICE_DETAILs` và bảng `PRODUCTs` dựa trên cột `MASP` (mã sản phẩm). Điều này giúp kết hợp thông tin sản phẩm với chi tiết hóa đơn.

```
group id by p.TenSP into grouped
```

- `group id by p.TenSP into grouped`: Nhóm các bản ghi trong bảng `INVOICE_DETAILs` theo tên sản phẩm (`TenSP`). Kết quả của việc nhóm được lưu trữ trong biến `grouped`.

```
orderby grouped.Count() descending
```

- `orderby grouped.Count() descending`: Đếm số lần mỗi sản phẩm xuất hiện (`grouped.Count()`) và sắp xếp các nhóm theo số lần xuất hiện giảm dần.

```
select new
{
    TenSP = grouped.Key,
    SoLanXuatHien = grouped.Count()
}).ToList();
```

- `select new {...}`: Tạo ra một đối tượng ẩn danh mới chứa các cột được chọn từ kết quả nhóm:
  - `TenSP = grouped.Key`: Tên sản phẩm (là khóa của nhóm).
  - `SoLanXuatHien = grouped.Count()`: Số lần sản phẩm xuất hiện trong bảng `INVOICE_DETAILs`.

- `ToList()`: Chuyển kết quả truy vấn thành một danh sách.

Java:

```
19.Liệt kê tên sản phẩm và số lần sản phẩm đó xuất hiện trong hóa đơn.
public static void cau19() {
    System.out.printf("%-35s %-15s %n", "Tên Sản Phẩm","Số lượng");
    System.out.println("-----------------------------------------------------");
    Map<String, Long> productOccurrences = invoiceDetails.stream()
            .collect(Collectors.groupingBy(
                    detail -> products.stream()
                            .filter(product -> product.getId() == detail.getProductId())
                            .map(Product::getName)
                            .findFirst()
                            .orElse("Unknow"),
                    Collectors.counting()));

    productOccurrences.entrySet().stream()
        .sorted((e1, e2) -> e2.getValue().compareTo(e1.getValue()))
        .forEach(entry -> System.out.printf("%-35s %-15s %n",entry.getKey(),entry.getValue()));
}
```

Giải thích :

.collect(Collectors.groupingBy(...)):

· hương thức collect() được sử dụng để thu thập các phần tử trong stream vào một cấu trúc dữ liệu. Ở đây, chúng ta sử dụng Collectors.groupingBy(), một collector giúp nhóm các phần tử trong stream theo một khóa (key) xác định.

· Cú pháp: groupingBy nhận vào một hàm để xác định khóa (key) của nhóm và trả về một Map (ở đây là Map<String, Long>) với khóa là tên sản phẩm và giá trị là số lần xuất hiện của sản phẩm đó.

detail -> products.stream().filter(...):

· Mỗi phần tử trong invoiceDetails là một đối tượng InvoiceDetail, đại diện cho một chi tiết của hóa đơn (mỗi chi tiết bao gồm thông tin về sản phẩm, số lượng, giá cả…).

· Để xác định sản phẩm liên quan đến một InvoiceDetail, ta sử dụng products.stream() để chuyển danh sách các sản phẩm thành một stream. Sau đó, ta lọc các sản phẩm có product.getId() == detail.getProductId() để tìm sản phẩm có mã ID tương ứng với ID sản phẩm trong chi tiết hóa đơn

.map(Product::getName):

Sau khi lọc ra sản phẩm tương ứng, ta lấy tên của sản phẩm đó thông qua product.getName(). map() sẽ chuyển đổi đối tượng Product thành tên sản phẩm (kiểu String).

.findFirst():

Vì sử dụng filter(), có thể có nhiều sản phẩm thỏa mãn điều kiện product.getId() == detail.getProductId(). Tuy nhiên, findFirst() chỉ lấy phần tử đầu tiên (do mỗi chi tiết hóa đơn chỉ có một sản phẩm). Nếu không tìm thấy sản phẩm nào, findFirst() sẽ trả về một Optional.

.orElse("Unknow"):

Phương thức orElse("Unknow") được sử dụng để đảm bảo rằng nếu không tìm thấy sản phẩm (trong trường hợp không có sản phẩm nào với ID trùng khớp), giá trị mặc định "Unknow" sẽ được trả về thay vì null.

Collectors.counting():

· Đây là một collector đặc biệt dùng để đếm số phần tử trong mỗi nhóm. Sau khi nhóm các chi tiết hóa đơn theo tên sản phẩm, Collectors.counting() sẽ đếm số lượng chi tiết hóa đơn có tên sản phẩm đó.

· Ví dụ: Nếu sản phẩm "Laptop" xuất hiện trong 3 chi tiết hóa đơn, giá trị đếm được cho khóa "Laptop" sẽ là 3.

Kết quả truy vấn:

```
19. Liệt kê tên sản phẩm và số lần sản phẩm đó xuất hiện trong hóa đơn:
-------------------------------------------------------
Tên Sản Phẩm                        Số lượng
-------------------------------------------------------
Gà Rán Phần Nhỏ                     18
Burger Phô Mai                      16
Gà Rán Phần Lớn                     15
Khoai Tây Chiên Lớn                 15
Pizza Phô Mai                       15
Burger Gà                           14
Bánh Sandwich Gà                    11
Pizza Hải Sản                       11
Khoai Tây Chiên Nhỏ                 11
Burger Bò                           10
Pizza Thập Cẩm                      10
Nước Cam                            9
Hotdog Phô Mai                      9
Taco Gà                             9
Mì Ý Sốt Gà                         9
Salad Trộn                          9
Taco Phô Mai                        8
Gà Xiên Nướng                       7
Nước Ngọt Coca-Cola                 7
Nước Ngọt Pepsi                     6
Nước Ngọt 7-Up                      5
Hotdog Xúc Xích                     5
Bánh Sandwich Bò                    5
Nước Chanh                          4
Súp Gà                              4
Gà Xiên Phô Mai                     3
Súp Hải Sản                         3
Mì Ý Sốt Bò                         2
```

20. Liệt kê các khách hàng có điểm tích lũy lớn hơn 150 và có tham gia chương trình khuyến mãi.

```
SELECT DISTINCT C.HoTen, C.DiemTichLuy
FROM CUSTOMER C
JOIN CUSTOMER_PROMOTION CP ON C.MAKH = CP.MAKH
```

```
JOIN PROMOTION P ON CP.MAKM = P.MAKM
WHERE C.DiemTichLuy > 150;
```

C#:

```
//20. Liệt kê các khách hàng có điểm tích lũy lớn hơn 150 và có tham gia chương trình khuyến mãi.
1 reference
public static void cau20()
{
    var result = (from c in dc1dc.CUSTOMERs
                  join cp in dc1dc.CUSTOMER_PROMOTIONs on c.MAKH equals cp.MAKH
                  join p in dc1dc.PROMOTIONs on cp.MAKM equals p.MAKM
                  where c.DiemTichLuy > 150
                  select new
                  {
                      HoTen = c.HoTen,
                      DiemTichLuy = c.DiemTichLuy
                  }).Distinct().ToList();
    Console.WriteLine("\n20. Liệt kê các khách hàng có điểm tích lũy lớn hơn 150 và có tham gia chương trình khuyến mãi:");
    Console.WriteLine("-----------------------------------------------------------");
    foreach (var customer in result) {
        Console.WriteLine("Khách hàng: {0}, Điểm tích lũy: {1}", customer.HoTen, customer.DiemTichLuy);
    }
}
```

Giải thích :

```
var result = (from c in dc1dc.CUSTOMERs
```

- `var result`: Khai báo biến `result` để lưu trữ kết quả của truy vấn.
- `from c in dc1dc.CUSTOMERs`: Bắt đầu truy vấn từ bảng `CUSTOMERs` và đặt tên biến `c` cho từng bản ghi trong bảng này.

```
join cp in dc1dc.CUSTOMER_PROMOTIONs on c.MAKH equals cp.MAKH
```

- Thực hiện phép nối giữa bảng `CUSTOMERs` và bảng `CUSTOMER_PROMOTIONs` dựa trên cột `MAKH` (mã khách hàng). Điều này giúp kết hợp thông tin khách hàng với các khuyến mãi mà họ nhận được.

```
join p in dc1dc.PROMOTIONs on cp.MAKM equals p.MAKM
```

- Thực hiện phép nối giữa bảng `CUSTOMER_PROMOTIONs` và bảng `PROMOTIONs` dựa trên cột `MAKM` (mã khuyến mãi). Điều này giúp lấy thông tin khuyến mãi mà khách hàng đã nhận.

```
where c.DiemTichLuy > 150
```

- `where c.DiemTichLuy > 150`: Lọc ra các khách hàng có điểm tích lũy (`DiemTichLuy`) lớn hơn 150.

```
select new
{
    HoTen = c.HoTen,
    DiemTichLuy = c.DiemTichLuy
}
```

- Tạo ra một đối tượng ẩn danh mới chứa các cột được chọn từ kết quả nối:
  - `HoTen = c.HoTen`: Họ tên khách hàng từ bảng `CUSTOMERs`.

- `DiemTichLuy = c.DiemTichLuy`: Điểm tích lũy của khách hàng từ bảng `CUSTOMERs`.

```
}).Distinct().ToList();
```

- `Distinct()`: Loại bỏ các bản ghi trùng lặp từ kết quả.
- `ToList()`: Chuyển kết quả thành một danh sách.

Java:

```
20. Liệt kê các khách hàng có điểm tích lũy lớn hơn 150 và có tham gia chương trình khuyến mãi.
public static void cau20(int pointsThreshold) {
    customers.stream()
            .filter(customer -> customer.getPoints() > pointsThreshold
                    && customerPromotions.stream().anyMatch(cp -> cp.getCustomerId() == customer.getId()))
            .distinct()
            .forEach(customer -> System.out
                    .println("Khách hàng: " + customer.getName() + ", Điểm tích lũy: " + customer.getPoints()));
}
```

Giải thích :

.filter(customer -> customer.getPoints() > pointsThreshold && ...):

filter() là một phương thức của Stream dùng để lọc các phần tử trong stream theo một điều kiện.
Điều kiện lọc ở đây bao gồm hai phần:

- customer.getPoints() > pointsThreshold: Lọc khách hàng có số điểm tích lũy lớn hơn giá trị ngưỡng pointsThreshold.
- customerPromotions.stream().anyMatch(cp -> cp.getCustomerId() == customer.getId()): Lọc các khách hàng mà có ít nhất một bản ghi khuyến mãi liên quan đến họ trong danh sách customerPromotions. Phương thức anyMatch() trả về true nếu có ít nhất một phần tử trong customerPromotions thỏa mãn điều kiện cp.getCustomerId() == customer.getId(). Điều này có nghĩa là khách hàng phải có ít nhất một khuyến mãi liên kết với họ.

.distinct():

Phương thức distinct() loại bỏ các phần tử trùng lặp trong stream. Điều này đảm bảo rằng mỗi khách hàng chỉ xuất hiện một lần trong kết quả cuối cùng, ngay cả khi có nhiều bản ghi khuyến mãi liên quan đến khách hàng đó.

Kết quả truy vấn:

```
20. Liệt kê các khách hàng có điểm tích lũy lớn hơn 150 và có tham gia chương trình khuyến mãi:
--------------------------------------------------------
Khách hàng: Bùi Thanh Hải, Điểm tích lũy: 200
Khách hàng: Ngô Hoàng Hải, Điểm tích lũy: 180
Khách hàng: Ngô Ngọc Mai, Điểm tích lũy: 170
Khách hàng: Trần Đăng Quang, Điểm tích lũy: 155
```