**Hệ thống thông tin quản lý của hàng Circle K**

Dưới đây là hệ thống thông tin chi tiết quản lý các hoạt động và thông tin của cửa hàng Circle K. Hệ thống này bao gồm các bảng lưu trữ thông tin về nhà cung cấp, sản phẩm, nhóm hàng, nhiên viên, khách hàng, hóa đơn, khuyến mãi, phản hồi và doanh thu.

**Xác định các thự thể và thuộc tính**

**1. Nhà cung cấp sản phẩm (SUPPLIER)**

* + MaNCC (INT, PK): Mã nhà cung cấp (duy nhất).
  + TenNCC (VARCHAR(100)): Tên nhà cung cấp.
  + DiaChi (VARCHAR(255)): Địa chỉ của nhà cung cấp.
  + SDT (VARCHAR(15)): Số điện thoại liên hệ.
  + Email (VARCHAR(100)): Email liên hệ.

**2. Sản phẩm (PRODUCT)**

* + MaSP (INT, PK): Mã sản phẩm (duy nhất).
  + TenSP (VARCHAR(100)): Tên sản phẩm.
  + DonGia (DECIMAL(10, 2)): Đơn giá sản phẩm.
  + HanSuDung (DATE): Hạn sử dụng của sản phẩm.
  + NoiSanXuat (VARCHAR(100)): Nơi sản xuất.
  + MaNhom (INT, FK): Mã nhóm hàng, tham chiếu đến bảng CATEGORY.

**3. Nhóm hàng (CATAGORY)**

* + MaNhom (INT, PK): Mã nhóm hàng (duy nhất).
  + TenNhom (VARCHAR(100)): Tên nhóm hàng.

**4. Nhân viên (EMPLOYEE)**

* + MaNV (INT, PK): Mã nhân viên (duy nhất).
  + HoTen (VARCHAR(100)): Họ và tên nhân viên.
  + GioiTinh (ENUM): Giới tính (Nam, Nữ).
  + NgaySinh (DATE): Ngày sinh của nhân viên.
  + DiaChi (VARCHAR(255)): Địa chỉ của nhân viên.
  + SDT (VARCHAR(15)): Số điện thoại liên hệ.
  + MaCV (INT, FK): Mã chức vụ, tham chiếu đến bảng JOB\_TITLE.

**5. Chức vụ (JOB\_TITLE)**

* + MaCV (INT, PK): Mã chức vụ (duy nhất).
  + TenCV (VARCHAR(100)): Tên chức vụ.

**6. Quyền đăng nhập (USER\_ACCOUNT)**

* + MaNV (INT, PK, FK): Mã nhân viên, tham chiếu đến bảng EMPLOYEE.
  + TenDangNhap (VARCHAR(50)): Tên đăng nhập.
  + MatKhau (VARCHAR(50)): Mật khẩu.

**7. Khách hàng (CUSTOMER)**

* + MaKH (INT, PK): Mã khách hàng (duy nhất).
  + HoTen (VARCHAR(100)): Họ và tên khách hàng.
  + SDT (VARCHAR(15)): Số điện thoại.
  + SoLanMua (INT): Số lần mua hàng.

**8. Phản hồi khách hàng (FEEDBACK)**

* + MaPH (INT, PK): Mã phản hồi (duy nhất).
  + MaKH (INT, FK): Mã khách hàng, tham chiếu đến bảng CUSTOMER.
  + NgayPhanHoi (DATE): Ngày khách hàng gửi phản hồi.
  + NoiDung (VARCHAR(255)): Nội dung phản hồi.
  + SoDiemDanhGia (INT): Số điểm đánh giá từ khách hàng (1-5).

**9. Hóa đơn (INVOICE)**

* + MaHD (INT, PK): Mã hóa đơn (duy nhất).
  + MaKH (INT, FK): Mã khách hàng, tham chiếu đến bảng CUSTOMER.
  + NgayLap (DATE): Ngày lập hóa đơn.
  + TongSoTien (DECIMAL(10, 2)): Tổng số tiền trong hóa đơn.

**10. Khuyến mãi (PROMOTION)**

* + MaKM (INT, PK): Mã khuyến mãi (duy nhất).
  + NoiDung (VARCHAR(255)): Nội dung khuyến mãi.
  + ThoiGianBatDau (DATE): Thời gian bắt đầu khuyến mãi.
  + ThoiGianKetThuc (DATE): Thời gian kết thúc khuyến mãi.
  + DoiTuongApDung (VARCHAR(100)): Đối tượng áp dụng khuyến mãi.
  + MucGiamGia (DECIMAL(5, 2)): Mức giảm giá (%).

**Mối liên kết giữa các thực thể**

1. SUPPLIER và PRODUCT: quan hệ 1-n vì một nhà cung cấp có thể cung cấp nhiều sản phẩm và một sản phẩn chỉ có một nhà cung cấp trong cửa hàng Circle K.
2. CATAGORY và PRODUCT: quan hệ 1-n vì một nhóm hàng có thể chứa nhiều sản phẩm và một sản phẩm chỉ thuộc một nhóm hàng.
3. EMPLOYEE và JOB\_TITLE: quan hệ n-1 vì một nhần viên chỉ thuộc một chức vụ trong của hàng và một chức vụ trong cửa hàng có thể có nhiều nhân viên.
4. EMPLOYEE và USER\_ACCOUNT: quan hệ 1-1 vì mỗi nhân viên có một tài khoản đăng nhập duy nhất.
5. CUSTOMER và INVOICE: quan hệ 1-n vì một khách hàng có thể có nhiều hóa đơn và một hóa đơn thì chỉ thuộc về một khách hàng.
6. CUSTOMER và FEEDBACK: quan hệ 1-n vì một khách hàng có thể gửi nhiều phản hồi và một phản hồi chỉ do một khách hàng phản hồi.
7. PROMOTION và PRODUCT: quan hệ 1-n vì mội khuyến mãi có thể áp dụng cho nhiều sản phẩm và một sản phẩm thì chỉ có thể có một mã khuyến mãi.

**Chuẩn hóa và quan hệ ứng viên**

Hệ thống thông tin này tuân theo chuẩn 3NF:

1. Mỗi thực thể đều có **khóa chính** duy nhất
2. Các thuộc tính không khó đều phụ thuộc và khóa chính, không phụ thuộc bắc cầu.
3. Không có thuộc tính đa trị hoặc nhóm thuộc tính lặp.

Để tạo một hệ thống thông tin quản lý của hàng Circle K ta cần kết với cơ sở dữ liệu MySQL bằng **JDBC**. Nhưng trước tiên ra cần tạo ra cơ sở dữ liệu bên MySQL, các bảng tương ứng và sau đó viết mã Java để thực hiện các truy vấn **JDBC**. Dưới đây là phần tạo cơ sở dữ liệu.

**1. Thiết lập cơ sở dữ liệu MySQL**

Đầu tiên cần tạo cơ sở dữ liệu và các bảng trong MySQL.

**Tạo cơ sở dữ liệu và các bảng trong MySQL.**

|  |
| --- |
| -- Tạo cơ sở dữ liệu  CREATE DATABASE StoreCircleK;  USE StoreCircleK;  -- Tạo bảng Supplier  CREATE TABLE SUPPLIER (  MaNCC INT AUTO\_INCREMENT PRIMARY KEY,  TenNCC VARCHAR(100) NOT NULL,  DiaChi VARCHAR(255),  SDT VARCHAR(15),  Email VARCHAR(100)  );  -- Tạo bảng Category  CREATE TABLE CATEGORY (  MaNhom INT AUTO\_INCREMENT PRIMARY KEY,  TenNhom VARCHAR(100) NOT NULL  );  -- Tạo bảng Product  CREATE TABLE PRODUCT (  MaSP INT AUTO\_INCREMENT PRIMARY KEY,  TenSP VARCHAR(100) NOT NULL,  DonGia DECIMAL(10, 2) NOT NULL,  HanSuDung DATE,  NoiSanXuat VARCHAR(100),  MaNhom INT,  MaNCC INT,  FOREIGN KEY (MaNhom) REFERENCES CATEGORY(MaNhom),  FOREIGN KEY (MaNCC) REFERENCES SUPPLIER(MaNCC)  );  -- Tạo bảng Job\_Title  CREATE TABLE JOB\_TITLE (  MaCV INT AUTO\_INCREMENT PRIMARY KEY,  TenCV VARCHAR(100) NOT NULL  );  -- Tạo bảng Employee  CREATE TABLE EMPLOYEE (  MaNV INT AUTO\_INCREMENT PRIMARY KEY,  HoTen VARCHAR(100) NOT NULL,  GioiTinh ENUM('Nam', 'Nu') NOT NULL,  NgaySinh DATE,  DiaChi VARCHAR(255),  SDT VARCHAR(15),  MaCV INT,  FOREIGN KEY (MaCV) REFERENCES JOB\_TITLE(MaCV)  );  -- Tạo bảng Account  CREATE TABLE USER\_ACCOUNT (  MaNV INT PRIMARY KEY,  TenDangNhap VARCHAR(50) NOT NULL,  MatKhau VARCHAR(50) NOT NULL,  FOREIGN KEY (MaNV) REFERENCES EMPLOYEE(MaNV)  );  -- Tạo bảng Customer  CREATE TABLE CUSTOMER (  MaKH INT AUTO\_INCREMENT PRIMARY KEY,  HoTen VARCHAR(100) NOT NULL,  SDT VARCHAR(15) NOT NULL,  SoLanMua INT DEFAULT 0  );  -- Tạo bảng FeedBack  CREATE TABLE FEEDBACK (  MaPH INT AUTO\_INCREMENT PRIMARY KEY,  MaKH INT,  NgayPhanHoi DATE NOT NULL,  NoiDung VARCHAR(255),  SoDiemDanhGia INT CHECK (SoDiemDanhGia BETWEEN 1 AND 5),  FOREIGN KEY (MaKH) REFERENCES CUSTOMER(MaKH)  );  -- Tạo bảng Invoice  CREATE TABLE INVOICE (  MaHD INT AUTO\_INCREMENT PRIMARY KEY,  MaKH INT,  NgayLap DATE NOT NULL,  TongSoTien DECIMAL(10, 2) NOT NULL,  FOREIGN KEY (MaKH) REFERENCES CUSTOMER(MaKH)  );  -- Tạo bảng Promotion  CREATE TABLE PROMOTION (  MaKM INT AUTO\_INCREMENT PRIMARY KEY,  NoiDung VARCHAR(255) NOT NULL,  ThoiGianBatDau DATE,  ThoiGianKetThuc DATE,  DoiTuongApDung VARCHAR(100),  MucGiamGia DECIMAL(5, 2) NOT NULL  ); |

**Tóm tắt**

Hệ thống quản lý này bao gồm việc quản lý thông tin về:

* **Nhà cung cấp, sản phẩm, nhóm hàng** giúp quản lý nguồn cung và phân loại sản phẩm.
* **Nhân viên, chức vụ, quyền đăng nhập** giúp quản lý quyền hạn và thông tin nhân viên.
* **Khách hàng, phản hồi, hóa đơn** giúp theo dõi thông tin khách hàng và lịch sử mua săm.
* **Khuyến mãi** giúp quản lý các chương trình khuyến mãi áp dụng cho sản phẩm.

Hệ thống thông tin quản lý của hàng Cirlce K này đảm bảo quản lý chi tiết và dễ dàng mở rộng khi có thêm yêu cầu mới. Dưới đây là một số lệnh để thêm dữ liệu ban đầu vào các dòng trong hệ thống thông tin quản lý Circle K. Những lệnh này sẽ giúp khởi tạo hệ thống với dữ liệu mẫu để dễ dàng kiểm tra và chạy thử.

|  |
| --- |
| -- Thêm dữ liệu vào bảng Supplier  INSERT INTO SUPPLIER (TenNCC, DiaChi, SDT, Email) VALUES  ('Vinamilk', '10 Tan Trao, Phuong Tan Pha, Quan 7, Tp.HCM', '02854155555', 'vinamilk@gmail.com'),  ('Coca-Cola', 'Xa lo Ha Noi, Phuong Linh Trung, Tp.Thu Đuc', '1900555584', 'cocacola@gmail.com'),  ('PepsCo', '88 Đuong Le Van Viet, Phuong Hiep Phu, Quan 9, Tp.HCM', '02862890000', 'pepsico@gmail.com'),  ('Acecook Viet Nam', 'Lo II-3, Đuong số II, Khu Cong Nghiep Tan Binh, Quan Tan Phu, Tp.HCM', NULL, 'acecook@gmail.com');  -- Thêm dữ liệu vào bảng Category  INSERT INTO CATEGORY (TenNhom)  VALUES  ('Đo uong'),  ('Đo an');  -- Thêm dữ liệu vào bảng Product  INSERT INTO PRODUCT (TenSP, DonGia, HanSuDung, NoiSanXuat, MaNhom, MaNCC)  VALUES  ('Sua tuoi 100% Vinamilk', 30000, '2025-04-12', 'Trang trai bo sua tai Viet Nam', 1, 1),  ('Yomilk', 20000, '2025-03-13', 'Trang trai bo sua tai Viet Nam', 1, 1),  ('Fanst', 12000, '2024-12-04', 'Nha may Coca - Cola tai Viet Nam', 1, 2),  ('7UP', 10000, '2025-01-05', 'Nha may của Suntory PepsiCo Viet Nam', 1, 3),  ('Pho Nhip Song', 9000, '2025-01-22', 'Nha may cua Acecook Viet Nam', 2, 4),  ('My Hao Hao', 4000, '2025-03-19', 'Nha may cua Acecook Viet Nam', 2, 4);  -- Thêm dữ liệu vào bảng Job\_Title  INSERT INTO JOB\_TITLE (TenCV)  VALUES  ('Nhan vien ban hang'),  ('Quan ly cua hang'),  ('Nhan Vien Don Dep');  -- Thêm dữ liệu vào bảng Employee  INSERT INTO EMPLOYEE (HoTen, GioiTinh, NgaySinh, DiaChi, SDT, MaCV)  VALUES  ('Nguyen Vu Triet', 'Nam', '2005-05-02', 'Bu To, Phuoc tan, Phu Rieng, Binh Phuoc', '0367182578', 1),  ('Bui Thi Thu Thao', 'Nu', '2005-07-11', 'Thon 1, Đuc Lieu, Bu Đang, Binh Phuoc', '0934567890', 2),  ('Truong Thanh Long', 'Nam', '2005-03-28', 'Thon 1, Đuc Lieu, Bu Đang, Binh Phuoc', '0922334455', 1),  ('Nguyen Thi Phuong Anh', 'Nu', '2005-05-19', 'Thôn 2, Nghia Trung, Bu Đang, Binh Phuoc', '0933334651', 3);  -- Thêm dữ liệu vào bảng User\_Account  INSERT INTO USER\_ACCOUNT (MaNV, TenDangNhap, MatKhau)  VALUES  (1, 'nguyenvutriet', 'pw01'),  (2, 'thuthao', 'pw02'),  (3, 'thanhlong', 'pw03'),  (4, 'phuonganh', 'pw04');  -- Thêm dữ liệu vào bảng Customer  INSERT INTO CUSTOMER (HoTen, SDT, SoLanMua)  VALUES  ('Mai Thi Hue', '0909998887', 10),  ('Phan Tran Huyen Trang', '0912349876', 2),  ('Pham Ngoc Nam', '0923456789', 5),  ('Le Minh Quan', '0998772116', 1);  -- Thêm dữ liệu vào bảng Feedback  INSERT INTO FEEDBACK (MaKH, NgayPhanHoi, NoiDung, SoDiemDanhGia)  VALUES  (1, '2024-12-01', 'San pham rat ngon, se tiep tuc ung ho!', 5),  (3, '2024-12-05', 'Yomilk rat ngon, sẽ ung ho!', 4),  (4, '2024-12-25', 'Cua hang trang tri đep!', 4);  -- Thêm dữ liệu vào bảng Invoice  INSERT INTO INVOICE (MaKH, NgayLap, TongSoTien)  VALUES  (1, '2024-11-19', 234000),  (2, '2024-11-20', 25000),  (3, '2024-11-25', 36000),  (4, '2024-11-28', 51000),  (2,'2024-11-21', 20000),  (4,'2024-11-26', 13000);  -- Thêm dữ liệu vào bảng Promotion  INSERT INTO PROMOTION (NoiDung, ThoiGianBatDau, ThoiGianKetThuc, DoiTuongApDung, MucGiamGia)  VALUES  ('Giam gia 10% cho cac looi đo uong', '2024-11-01', '2024-12-01', 'Đo uong', 10.00),  ('Mua 5 goi my hao hao duoc tang ve xem concert Adele o Munich', '2024-11-18', '2024-11-19', NULL, 0.00),  ('Giam 5% cho cac loai đo an', '2024-11-18', '2024-11-25', 'Đo an', 5.00); |

**2. Cấu hình kết nối JDBC**

Để kết nối được Java với MySQL ta cần có thư viện MySQL JDBC (MySQL Connector). Ta cần tải thư viện xuống sau đó vào Project của mình tạo một package copy thư viện sau đó dán vào package này. Tiếp theo nhấn chuột phải vào thư viện đó chọn Build Path -> Add to Build Path.

**3. Chương trình Java kết nối JDBC**

|  |
| --- |
| package database;  import java.sql.Connection;  import java.sql.DatabaseMetaData;  import java.sql.DriverManager;  import java.sql.SQLException;  public class JDBCUtil {  public static Connection getConnection() {  Connection c = null;  try {  // Thông tin kết nối MySQL  String url = "jdbc:mysql://localhost:3306/storecirclek";  String username = "root";  String password = "123456";    // Tạo kết nối  c = DriverManager.getConnection(url, username, password);  } catch (SQLException e) {  e.printStackTrace();  }  return c;  }  public static void closeConnection(Connection c) {  try {  if (c != null && !c.isClosed()) {  c.close();  }  } catch (SQLException e) {  e.printStackTrace();  }  }  public static void printInfo(Connection c) {  try {  if(c != null) {  DatabaseMetaData mtdt = c.getMetaData();  System.out.println(mtdt.getDatabaseProductName());  System.out.println(mtdt.getDatabaseProductVersion());  }  } catch (SQLException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  }  } |

**4. Giải thích mã nguồn**

* **Kết nối CSDL:** Được khởi tạo trong class JDBCUtil trong đó hàm getConnection thực hiện việc kết nối tới CSDL trên MySQL. Các url, username, password được cập nhật theo đúng thông tin tài khoản trên MySQL.
* Từng phương thức: closeConnection thực hiện việc đóng kết nối với CSDL. getConnection thực hiện việc tạo kết nối tới CSDL. printInfo thực hiện hiển thị thông tin cơ bản về cơ sở dữ liệu.

Dưới đây là **20 câu truy vấn SQL** cho hệ thống thông tin cửa hàng Circle K, được sắp xếp từ cơ bản đến nâng cao. Mỗi câu truy vấn tập trung vào các khịa cạnh khác nhau, giúp ta hiểu rõ hơn về truy vấn SQL phổ biến.

**1. Thông tin nhân viên của Circle K**

|  |
| --- |
| SELECT HoTen, GioiTinh, NgaySinh, DiaChi FROM EMPLOYEEs; |

**2. Tìm nhân viên bán hàng**

|  |
| --- |
| SELECT HoTen FROM EMPLOYEEs WHERE MaCV = 1; |

**3. Tên và số điện thoại của khách hàng có số lượt mua trên 5**

|  |
| --- |
| SELECT HoTen, SDT FROM CUSTOMERs WHERE SoLanMua > 5; |

**4. Lời phản hồi của khách hàng có mã khách hàng là 3 và lượt đánh giá từ 4 sao**

|  |
| --- |
| SELECT NgayPhanHoi, NoiDung FROM FEEDBACKs WHERE MaKH = 3 AND SoDiemDanhGia >= 4; |

**5. Tên của khách hàng đã đánh giá 5 sao**

|  |
| --- |
| SELECT c.HoTen, c.SDT  FROM CUSTOMERs c  JOIN FEEDBACKs f ON c.MaKH = f.MaKH  WHERE f.SoDiemDanhGia = 5; |

**6. Hóa đơn kèm theo tên của người mua**

|  |
| --- |
| SELECT c.HoTen, i.TongSoTien  FROM INVOICEs i  JOIN CUSTOMERs c ON i.MaKH = c.MaKH; |

**7. Những sản phẩm của Vinamilk có đơn giá trên 25000**

|  |
| --- |
| SELECT p.TenSP, p.DonGia  FROM PRODUCTs p  JOIN SUPPLIERs s ON p.MaNCC = s.MaNCC  WHERE s.TenNCC = 'Vinamilk' AND p.DonGia > 25000; |

**8. Đếm số lượng nhân viên trong Circle K**

|  |
| --- |
| SELECT COUNT(\*) FROM EMPLOYEEs; |

**9. Tổng số sản phẩm của mỗi nhà cung cấp**

|  |
| --- |
| SELECT s.TenNCC, COUNT(p.MaSP) AS TongSanPham  FROM SUPPLIERs s  JOIN PRODUCTs p ON s.MaNCC = p.MaNCC  GROUP BY s.TenNCC; |

**10. Tổng số tiền trung bình của các khách hàng**

|  |
| --- |
| SELECT AVG(TongSoTien)  FROM INVOICEs; |

**11. Phiếu khuyến mãi được áp dụng trong thời gian hiện tại**

|  |
| --- |
| SELECT NoiDung, MucGiamGia, ThoiGianBatDau, ThoiGianKetThuc  FROM PROMOTIONs  WHERE ThoiGianBatDau <= NOW() AND ThoiGianKetThuc >= NOW(); |

**12. Sản phẩm hết hạn trong 30 ngày tính từ ngày hiện tại**

|  |
| --- |
| SELECT TenSP, HanSuDung, NoiSanXuat  FROM PRODUCTs  WHERE HanSuDung <= CURDATE() + INTERVAL 30 DAY; |

**13. Mỗi tên công việc lấy ra tên nhân viên trong đó**

|  |
| --- |
| SELECT j.TenCV, e.HoTen  FROM JOB\_TITLEs j  JOIN EMPLOYEEs e ON e.MaCV = j.MaCV; |

**14. Tính tổng đơn giá của các sản phẩm của nhà cung cấp**

|  |
| --- |
| SELECT s.TenNCC, SUM(p.DonGia) AS TongDonGia  FROM SUPPLIERs s  JOIN PRODUCTs p ON s.MaNCC = p.MaNCC  GROUP BY s.TenNCC; |

**15. Lấy ra sản phẩm thuộc nhóm đồ uống và sắp xếp theo đơn giá giảm dần**

|  |
| --- |
| SELECT p.TenSP, p.DonGia  FROM PRODUCTs p  JOIN CATEGORies c ON p.MaNhom = c.MaNhom  WHERE c.TenNhom = 'Đo uong'  ORDER BY p.DonGia DESC; |

**16. Lấy ra sản phẩm thuộc nhóm đồ ăn và sắp xếp theo đơn giá tăng dần**

|  |
| --- |
| SELECT p.TenSP, p.DonGia  FROM PRODUCTs p  JOIN CATEGORies c ON p.MaNhom = c.MaNhom  WHERE c.TenNhom = 'Đo an'  ORDER BY p.DonGia ASC; |

**17. Thông tin khách hàng đã đánh giá và sắp xếp theo số lần mua**

|  |
| --- |
| SELECT c.HoTen, c.SoLanMua, f.NoiDung  FROM CUSTOMERs c  JOIN FEEDBACKs f ON c.MaKH = f.MaKH  WHERE f.SoDiemDanhGia IS NOT NULL  ORDER BY c.SoLanMua; |

**18. Thông tin khách hàng không đánh giá của hàng Circle K**

|  |
| --- |
| SELECT c.HoTen, c.SDT  FROM CUSTOMERs c  LEFT JOIN FEEDBACKs f ON c.MaKH = f.MaKH  WHERE f.SoDiemDanhGia IS NULL; |

**19. Thông tin khách hàng mua đơn dưới 100k kèm theo lời phản hồi**

|  |
| --- |
| SELECT c.HoTen, f.NoiDung, SUM(i.TongSoTien) AS TongTien  FROM INVOICEs i  JOIN CUSTOMERs c ON i.MaKH = c.MaKH  LEFT JOIN FEEDBACKs f ON c.MaKH = f.MaKH  GROUP BY c.HoTen, f.NoiDung  HAVING SUM(i.TongSoTien) < 100000; |

**20. Tổng số tiền của các hóa đơn theo khách hàng và sắp xếp theo thứ tự giảm dần**

|  |
| --- |
| SELECT c.HoTen, c.SDT, SUM(i.TongSoTien) AS TongTien  FROM INVOICEs i  JOIN CUSTOMERs c ON i.MaKH = c.MaKH  GROUP BY c.HoTen, c.SDT  ORDER BY TongTien DESC; |

**Giải thích một số câu lệnh:**

* **Sử dụng Join để liên kết các bảng**: Các câu truy vấn như 5, 6, 7, 9, 13, 14, 15, 16, 17, 18, 19 và 20 yêu cầu hiểu về Join để nối dữ liệu từ nhiều bảng.
* **Tính toán trên dữ liệu**: Các câu lệnh như 8, 14, 19, và 20 yêu cầu sử dụng các hàm tổng hợp (Count, Sum).
* **Lọc dữ liệu với điều kiện**: Sử dụng các điều kiện where kết hợp với các hàm AND, NOW, CURDATE() + INTERVAL 30 DAY và IS NOT NULL để lọc dữ liệu theo yêu cầu của đề bài.

Những truy vấn này giúp ta làm quen với các dạng câu hỏi SQL phổ biến, từ các truy vấn đơn giản đến các truy vấn phức tạp.

Dưới đây là các truy vấn SQL trên được chuyển thành các biểu thức đại số quan hệ, giúp hiểu rõ cách viết và biểu diễn bằng đại số quan hệ.

**1. Thông tin nhân viên của Circle K.**

πHoTen, GioiTinh, NgaySinh, DiaChi​(EMPLOYEEs)

**2. Tìm nhân viên bán hàng.**

πHoTen​(σMaCV=1​(EMPLOYEEs))

**3. Tên và số điện thoại của khách hàng có số lượt mua trên 5.**

πHoTen, SDT​(σSoLanMua>5​(CUSTOMERs))

**4. Lời phản hồi của khách hàng có mã khách hàng là 3 và lượt đánh giá từ 4 sao.**

πNgayPhanHoi, NoiDung​(σMaKH=3∧SoDiemDanhGia≥4​(FEEDBACKs))

**5. Tên của khách hàng đã đánh giá 5 sao.**

πHoTen, SDT​(σSoDiemDanhGia=5​(CUSTOMERs⋈CUSTOMERs.MaKH=FEEDBACKs.MaKH​FEEDBACKs))

**6. Hóa đơn kèm theo tên của người mua.**

πHoTen, TongSoTien​(INVOICEs⋈INVOICEs.MaKH=CUSTOMERs.MaKH​CUSTOMERs)

**7. Những sản phẩm của Vinamilk có đơn giá trên 25000.**

πTenSP, DonGia​(σTenNCC=′Vinamilk′∧DonGia>25000​(PRODUCTs⋈PRODUCTs.MaNCC=SUPPLIERs.MaNCC​SUPPLIERs))

**8. Đếm số lượng nhân viên trong Circle K.**

*Đại số quan hệ không hỗ trợ việc đếm nhân viên trong Circle K.*

**9. Tổng số sản phẩm của mỗi nhà cung cấp.**

*Đại số quan hệ không hỗ trợ việc đếm số sản phẩm của nhà cung cấp.*

**10. Tổng số tiền trung bình của các khách hàng.**

*Đại số quan hệ không hỗ trợ tính tổng và đếm.*

**11. Phiếu khuyến mãi được áp dụng trong thời gian hiện tại.**

πNoiDung, MucGiamGia, ThoiGianBatDau, ThoiGianKetThuc​(σThoiGianBatDau≤NOW∧ThoiGianKetThuc≥NOW​(PROMOTIONs))

**12. Sản phẩm hết hạn trong 30 ngày tính từ ngày hiện tại.**

πTenSP, HanSuDung, NoiSanXuat​(σHanSuDung≤CURDATE+30​(PRODUCTs))

**13. Mỗi tên công việc lấy ra tên nhân viên trong đó.**

πTenCV,HoTen​(σe.MaCV=j.MaCV​(JOB\_TITLEs×EMPLOYEEs))

**14. Tính tổng đơn giá của các sản phẩm của nhà cung cấp.**

*Đại số quan hệ không hỗ trợ tính tổng nên không thể tính tổng đơn giá được.*

**15. Lấy ra sản phẩm thuộc nhóm đồ uống và sắp xếp theo đơn giá giảm dần.**

*Đại số quan hệ không hỗ trợ việc sắp xếp giảm dần.*

**16. Lấy ra sản phẩm thuộc nhóm đồ ăn và sắp xếp theo đơn giá tăng dần.**

*Đại số quan hệ trong hỗ trợ việc sắp xếp tăng dần hay giảm dần.*

**17. Thông tin khách hàng đã đánh giá và sắp xếp theo số lần mua.**

*Đại số quan hệ không hỗ trợ việc sắp xếp.*

**18. Thông tin khách hàng không đánh giá của hàng Circle K.**

πHoTen, SDT​(σSoDiemDanhGia=NULL​(CUSTOMERs LEFT OUTER JOIN FEEDBACKs))

**19. Thông tin khách hàng mua đơn dưới 100k kèm theo lời phản hồi.**

*Đại số quan hệ không hỗ trợ phép tính tổng.*

**20. Tổng số tiền của các hóa đơn theo khách hàng và sắp xếp theo thứ tự giảm dần.**

*Đại số quan hệ không hỗ trợ phép tính tổng.*

**Chú ý**: Một số truy vấn nâng cao, như tính toán trung bình, tổng số, đếm số lượng hoặc các truy vấn phụ thuộc vào thời gian, không thể biểu diễn bằng đại số quan hệ truyền thống. Đại số quan hệ chủ yếu hỗ trợ các phép toán cơ bản như *SELECT* (σ), *PROJECT* (π), *JOIN* (⨝), *DIFFERENCE* (-), và *UNION* (∪), thiếu các phép toán tổng hợp mà SQL cung cấp.

Để sử **dụng các câu truy vấn với Java lambdas** trong chương trình Java sử dụng JDBC, ta cần tải dữ liệu từ cơ sở dữ liệu vào các đối tượng Java (category, customer, employee, feedback, invoice, job\_title, product, supplier, promotion, user\_account), sau đó thực hiện việc truy vấn bằng cách sử dụng Java Lambdas và Stream API. Dưới đây là cách tích hợp các câu truy vấn này vào chương trình Java sử dụng JDBC.

**1. Tạo lớp đối tượng để lưu trữ dữ liệu từ MySQL**

**Class category**

|  |
| --- |
| **package** database;  **public** **class** category {  **private** **int** MaNhom;  **private** String TenNhom;  **public** category(**int** maNhom, String tenNhom) {  MaNhom = maNhom;  TenNhom = tenNhom;  }  **public** **int** getMaNhom() {  **return** MaNhom;  }  **public** **void** setMaNhom(**int** maNhom) {  MaNhom = maNhom;  }  **public** String getTenNhom() {  **return** TenNhom;  }  **public** **void** setTenNhom(String tenNhom) {  TenNhom = tenNhom;  }  @Override  **public** String toString() {  **return** "category [MaNhom=" + MaNhom + ", TenNhom=" + TenNhom + "]";  }  } |

**Class customer**

|  |
| --- |
| **package** database;  **public** **class** customer {  **private** **int** MaKH;  **private** String HoTen;  **private** String SDT;  **private** **int** SoLanMua;  **public** customer(**int** maKH, String hoTen, String sDT, **int** soLanMua) {  MaKH = maKH;  HoTen = hoTen;  SDT = sDT;  SoLanMua = soLanMua;  }  **public** **int** getMaKH() {  **return** MaKH;  }  **public** **void** setMaKH(**int** maKH) {  MaKH = maKH;  }  **public** String getHoTen() {  **return** HoTen;  }  **public** **void** setHoTen(String hoTen) {  HoTen = hoTen;  }  **public** String getSDT() {  **return** SDT;  }  **public** **void** setSDT(String sDT) {  SDT = sDT;  }  **public** **int** getSoLanMua() {  **return** SoLanMua;  }  **public** **void** setSoLanMua(**int** soLanMua) {  SoLanMua = soLanMua;  }  @Override  **public** String toString() {  **return** "customer [MaKH=" + MaKH + ", HoTen=" + HoTen + ", SDT=" + SDT + ", SoLanMua=" + SoLanMua + "]";  }  } |

**Class employee**

|  |
| --- |
| **package** database;  **import** java.sql.Date;  **public** **class** employee {  **private** **int** MaNV;  **private** String HoTen;  **private** String GioiTinh;  **private** Date NgaySinh;  **private** String DiaChi;  **private** String SDT;  **private** **int** MaCV;  **public** employee(**int** maNV, String hoTen, String gioiTinh, Date ngaySinh, String diaChi, String sDT, **int** maCV) {  MaNV = maNV;  HoTen = hoTen;  GioiTinh = gioiTinh;  NgaySinh = ngaySinh;  DiaChi = diaChi;  SDT = sDT;  MaCV = maCV;  }  **public** **int** getMaNV() {  **return** MaNV;  }  **public** **void** setMaNV(**int** maNV) {  MaNV = maNV;  }  **public** String getHoTen() {  **return** HoTen;  }  **public** **void** setHoTen(String hoTen) {  HoTen = hoTen;  }  **public** String getGioiTinh() {  **return** GioiTinh;  }  **public** **void** setGioiTinh(String gioiTinh) {  GioiTinh = gioiTinh;  }  **public** Date getNgaySinh() {  **return** NgaySinh;  }  **public** **void** setNgaySinh(Date ngaySinh) {  NgaySinh = ngaySinh;  }  **public** String getDiaChi() {  **return** DiaChi;  }  **public** **void** setDiaChi(String diaChi) {  DiaChi = diaChi;  }  **public** String getSDT() {  **return** SDT;  }  **public** **void** setSDT(String sDT) {  SDT = sDT;  }  **public** **int** getMaCV() {  **return** MaCV;  }  **public** **void** setMaCV(**int** maCV) {  MaCV = maCV;  }  @Override  **public** String toString() {  **return** "employee [MaNV=" + MaNV + ", HoTen=" + HoTen + ", GioiTinh=" + GioiTinh + ", NgaySinh=" + NgaySinh  + ", DiaChi=" + DiaChi + ", SDT=" + SDT + ", MaCV=" + MaCV + "]";  }  } |

**Class feedback**

|  |
| --- |
| **package** database;  **import** java.sql.Date;  **public** **class** feedback {  **private** **int** MaPH;  **private** **int** MaKH;  **private** Date NgayPhanHoi;  **private** String NoiDung;  **private** Integer SoDiemDanhGia;    **public** **int** getMaPH() {  **return** MaPH;  }  **public** **void** setMaPH(**int** maPH) {  MaPH = maPH;  }  **public** **int** getMaKH() {  **return** MaKH;  }  **public** **void** setMaKH(**int** maKH) {  MaKH = maKH;  }  **public** Date getNgayPhanHoi() {  **return** NgayPhanHoi;  }  **public** **void** setNgayPhanHoi(Date ngayPhanHoi) {  NgayPhanHoi = ngayPhanHoi;  }  **public** String getNoiDung() {  **return** NoiDung;  }  **public** **void** setNoiDung(String noiDung) {  NoiDung = noiDung;  }  **public** Integer getSoDiemDanhGia() {  **return** SoDiemDanhGia;  }  **public** **void** setSoDiemDanhGia(Integer soDiemDanhGia) {  SoDiemDanhGia = soDiemDanhGia;  }  **public** feedback(**int** maPH, **int** maKH, Date ngayPhanHoi, String noiDung, Integer soDiemDanhGia) {  **super**();  MaPH = maPH;  MaKH = maKH;  NgayPhanHoi = ngayPhanHoi;  NoiDung = noiDung;  SoDiemDanhGia = soDiemDanhGia;  }  @Override  **public** String toString() {  **return** "feedback [MaPH=" + MaPH + ", MaKH=" + MaKH + ", NgayPhanHoi=" + NgayPhanHoi + ", NoiDung=" + NoiDung  + ", SoDiemDanhGia=" + SoDiemDanhGia + "]";  }  } |

**Class invoice**

|  |
| --- |
| **package** database;  **import** java.sql.Date;  **public** **class** invoice {  **private** **int** MaHD;  **private** **int** MaKH;  **private** Date NgayLap;  **private** **double** TongSoTien;  **public** invoice(**int** maHD, **int** maKH, Date ngayLap, **double** tongSoTien) {  MaHD = maHD;  MaKH = maKH;  NgayLap = ngayLap;  TongSoTien = tongSoTien;  }  **public** **int** getMaHD() {  **return** MaHD;  }  **public** **void** setMaHD(**int** maHD) {  MaHD = maHD;  }  **public** **int** getMaKH() {  **return** MaKH;  }  **public** **void** setMaKH(**int** maKH) {  MaKH = maKH;  }  **public** Date getNgayLap() {  **return** NgayLap;  }  **public** **void** setNgayLap(Date ngayLap) {  NgayLap = ngayLap;  }  **public** **double** getTongSoTien() {  **return** TongSoTien;  }  **public** **void** setTongSoTien(**double** tongSoTien) {  TongSoTien = tongSoTien;  }  @Override  **public** String toString() {  **return** "invoice [MaHD=" + MaHD + ", MaKH=" + MaKH + ", NgayLap=" + NgayLap + ", TongSoTien=" + TongSoTien + "]";  }  } |

**Class job\_title**

|  |
| --- |
| **package** database;  **public** **class** job\_title {  **private** **int** MaCV;  **private** String TenCV;  **public** job\_title(**int** maCV, String tenCV) {  MaCV = maCV;  TenCV = tenCV;  }  **public** **int** getMaCV() {  **return** MaCV;  }  **public** **void** setMaCV(**int** maCV) {  MaCV = maCV;  }  **public** String getTenCV() {  **return** TenCV;  }  **public** **void** setTenCV(String tenCV) {  TenCV = tenCV;  }  @Override  **public** String toString() {  **return** "job\_title [MaCV=" + MaCV + ", TenCV=" + TenCV + "]";  }  } |

Class product

|  |
| --- |
| **package** database;  **import** java.sql.Date;  **public** **class** product {  **private** **int** MaSP;  **private** String TenSP;  **private** **double** DonGia;  **private** Date HanSuDung;  **private** String NoiSanXuat;  **private** **int** MaNhom;  **private** **int** MaNCC;  **public** product(**int** maSP, String tenSP, **double** donGia, Date hanSuDung, String noiSanXuat, **int** maNhom, **int** maNCC) {  MaSP = maSP;  TenSP = tenSP;  DonGia = donGia;  HanSuDung = hanSuDung;  NoiSanXuat = noiSanXuat;  MaNhom = maNhom;  MaNCC = maNCC;  }  **public** **int** getMaSP() {  **return** MaSP;  }  **public** **void** setMaSP(**int** maSP) {  MaSP = maSP;  }  **public** String getTenSP() {  **return** TenSP;  }  **public** **void** setTenSP(String tenSP) {  TenSP = tenSP;  }  **public** **double** getDonGia() {  **return** DonGia;  }  **public** **void** setDonGia(**double** donGia) {  DonGia = donGia;  }  **public** Date getHanSuDung() {  **return** HanSuDung;  }  **public** **void** setHanSuDung(Date hanSuDung) {  HanSuDung = hanSuDung;  }  **public** String getNoiSanXuat() {  **return** NoiSanXuat;  }  **public** **void** setNoiSanXuat(String noiSanXuat) {  NoiSanXuat = noiSanXuat;  }  **public** **int** getMaNhom() {  **return** MaNhom;  }  **public** **void** setMaNhom(**int** maNhom) {  MaNhom = maNhom;  }  **public** **int** getMaNCC() {  **return** MaNCC;  }  **public** **void** setMaNCC(**int** maNCC) {  MaNCC = maNCC;  }  @Override  **public** String toString() {  **return** "product [MaSP=" + MaSP + ", TenSP=" + TenSP + ", DonGia=" + DonGia + ", HanSuDung=" + HanSuDung  + ", NoiSanXuat=" + NoiSanXuat + ", MaNhom=" + MaNhom + ", MaNCC=" + MaNCC + "]";  }  } |

**Class promotion**

|  |
| --- |
| **package** database;  **import** java.sql.Date;  **public** **class** promotion {  **private** **int** MaKM;  **private** String NoiDung;  **private** Date ThoiGianBatDau;  **private** Date ThoiGianKetThuc;  **private** String DoiTuongApDung;  **private** **double** MucGiamGia;  **public** promotion(**int** maKM, String noiDung, Date thoiGianBatDau, Date thoiGianKetThuc, String doiTuongApDung,  **double** mucGiamGia) {  MaKM = maKM;  NoiDung = noiDung;  ThoiGianBatDau = thoiGianBatDau;  ThoiGianKetThuc = thoiGianKetThuc;  DoiTuongApDung = doiTuongApDung;  MucGiamGia = mucGiamGia;  }  **public** **int** getMaKM() {  **return** MaKM;  }  **public** **void** setMaKM(**int** maKM) {  MaKM = maKM;  }  **public** String getNoiDung() {  **return** NoiDung;  }  **public** **void** setNoiDung(String noiDung) {  NoiDung = noiDung;  }  **public** Date getThoiGianBatDau() {  **return** ThoiGianBatDau;  }  **public** **void** setThoiGianBatDau(Date thoiGianBatDau) {  ThoiGianBatDau = thoiGianBatDau;  }  **public** Date getThoiGianKetThuc() {  **return** ThoiGianKetThuc;  }  **public** **void** setThoiGianKetThuc(Date thoiGianKetThuc) {  ThoiGianKetThuc = thoiGianKetThuc;  }  **public** String getDoiTuongApDung() {  **return** DoiTuongApDung;  }  **public** **void** setDoiTuongApDung(String doiTuongApDung) {  DoiTuongApDung = doiTuongApDung;  }  **public** **double** getMucGiamGia() {  **return** MucGiamGia;  }  **public** **void** setMucGiamGia(**double** mucGiamGia) {  MucGiamGia = mucGiamGia;  }  @Override  **public** String toString() {  **return** "promotion [MaKM=" + MaKM + ", NoiDung=" + NoiDung + ", ThoiGianBatDau=" + ThoiGianBatDau  + ", ThoiGianKetThuc=" + ThoiGianKetThuc + ", DoiTuongApDung=" + DoiTuongApDung + ", MucGiamGia="  + MucGiamGia + "]";  }  } |

**Class suplier**

|  |
| --- |
| **package** database;  **public** **class** supplier {  **private** **int** MaNCC;  **private** String TenNCC;  **private** String DiaChi;  **private** String SDT;  **private** String Email;  **public** supplier(**int** maNCC, String tenNCC, String diaChi, String sDT, String email) {  MaNCC = maNCC;  TenNCC = tenNCC;  DiaChi = diaChi;  SDT = sDT;  Email = email;  }  **public** **int** getMaNCC() {  **return** MaNCC;  }  **public** **void** setMaNCC(**int** maNCC) {  MaNCC = maNCC;  }  **public** String getTenNCC() {  **return** TenNCC;  }  **public** **void** setTenNCC(String tenNCC) {  TenNCC = tenNCC;  }  **public** String getDiaChi() {  **return** DiaChi;  }  **public** **void** setDiaChi(String diaChi) {  DiaChi = diaChi;  }  **public** String getSDT() {  **return** SDT;  }  **public** **void** setSDT(String sDT) {  SDT = sDT;  }  **public** String getEmail() {  **return** Email;  }  **public** **void** setEmail(String email) {  Email = email;  }  @Override  **public** String toString() {  **return** "supplier [MaNCC=" + MaNCC + ", TenNCC=" + TenNCC + ", DiaChi=" + DiaChi + ", SDT=" + SDT + ", Email="  + Email + "]";  }  } |

**Class user\_account**

|  |
| --- |
| **package** database;  **public** **class** user\_account {  **private** String TenDangNhap;  **private** String MatKhau;  **public** user\_account(String tenDangNhap, String matKhau) {  TenDangNhap = tenDangNhap;  MatKhau = matKhau;  }  **public** String getTenDangNhap() {  **return** TenDangNhap;  }  **public** **void** setTenDangNhap(String tenDangNhap) {  TenDangNhap = tenDangNhap;  }  **public** String getMatKhau() {  **return** MatKhau;  }  **public** **void** setMatKhau(String matKhau) {  MatKhau = matKhau;  }  @Override  **public** String toString() {  **return** "user\_account [TenDangNhap=" + TenDangNhap + ", MatKhau=" + MatKhau + "]";  }  } |

**2. Tạo những danh sách để lấy dữ liệu từ MySQL và thực thi 20 câu lệnh Java Lambdas**

Đầu tiên, ta cần tạo lớp ManagementStoreCircleK để lấy dữ liệu từ MySQL và lưu trong danh sách (Lists). Ta sẽ sử dụng các danh sách này với Java Lambdas.

|  |
| --- |
| package mypack;  import java.sql.Connection;  import java.sql.Date;  import java.sql.ResultSet;  import java.sql.Statement;  import java.util.ArrayList;  import java.util.Comparator;  import java.util.List;  import java.util.Map;  import java.util.OptionalDouble;  import java.util.stream.Collectors;  import database.JDBCUtil;  import database.category;  import database.customer;  import database.employee;  import database.feedback;  import database.invoice;  import database.job\_title;  import database.product;  import database.promotion;  import database.supplier;  import database.user\_account;  public class ManagementStoreCircleK {  public static void main(String[] args) {  try {  //Bước 1: tạo kết nối  Connection connection = new JDBCUtil().getConnection();  System.out.println("Connection database Completed");      //Bước 2 tạo đối tượng statement  Statement st = connection.createStatement();    //Bước 3: Thực thi câu lệnh sql  //1. Bảng category  List<category> Category = new ArrayList<category>();  String table\_category = "SELECT \* FROM Category";  ResultSet rs = st.executeQuery(table\_category);  while(rs.next()) {  int maNhom = rs.getInt("MaNhom");  String tenNhom = rs.getString("TenNhom");  Category.add(new category(maNhom, tenNhom));  }  //2. Bảng Customer  List<customer> Customer = new ArrayList<customer>();  String table\_customer = "SELECT \* FROM Customer";  ResultSet rs\_customer = st.executeQuery(table\_customer);  while(rs\_customer.next()) {  int makh = rs\_customer.getInt("MaKH");  String hoten = rs\_customer.getString("HoTen");  String sdt = rs\_customer.getString("SDT");  int solanmua = rs\_customer.getInt("SoLanMua");  Customer.add(new customer(makh, hoten, sdt, solanmua));  }  //3. Bảng Employee  List<employee> Employee = new ArrayList<employee>();  String table\_employee = "SELECT \* FROM Employee";  ResultSet rs\_employee = st.executeQuery(table\_employee);  while(rs\_employee.next()) {  int manv = rs\_employee.getInt("MaNV");  String hoten = rs\_employee.getString("HoTen");  String gioitinh = rs\_employee.getString("GioiTinh");  Date ngaysinh = rs\_employee.getDate("NgaySinh");  String diachi = rs\_employee.getString("DiaChi");  int macv = rs\_employee.getInt("MaCV");  Employee.add(new employee(manv, hoten, gioitinh, ngaysinh, diachi, diachi, macv));  }  //4. Bảng feedback  List<feedback> Feedback = new ArrayList<feedback>();  String table\_feedback = "SELECT \* FROM Feedback";  ResultSet re\_feedback = st.executeQuery(table\_feedback);  while(re\_feedback.next()) {  int maph = re\_feedback.getInt("MaPH");  int makh = re\_feedback.getInt("MaKH");  Date ngayphaihoi = re\_feedback.getDate("NgayPhanHoi");  String noidung = re\_feedback.getString("NoiDung");  int sodiemdanhgia = re\_feedback.getInt("SoDiemDanhGia");  Feedback.add(new feedback(maph, makh, ngayphaihoi, noidung, sodiemdanhgia));  }  //5. Bảng Invoice  List<invoice> Invoice = new ArrayList<invoice>();  String table\_invoice = "SELECT \* FROM Invoice";  ResultSet rs\_invoice = st.executeQuery(table\_invoice);  while(rs\_invoice.next()) {  int mahd = rs\_invoice.getInt("MaHD");  int makh = rs\_invoice.getInt("MaKH");  Date ngaylap = rs\_invoice.getDate("NgayLap");  double tongsotien = rs\_invoice.getDouble("TongSoTien");  Invoice.add(new invoice(mahd, makh, ngaylap, tongsotien));  }  //6. Bảng Job\_Title  List<job\_title> Job\_Title = new ArrayList<job\_title>();  String table\_jobtitle = "SELECT \* FROM Job\_Title";  ResultSet rs\_jobtitle = st.executeQuery(table\_jobtitle);  while(rs\_jobtitle.next()) {  int macv = rs\_jobtitle.getInt("MaCV");  String tencv = rs\_jobtitle.getString("TenCV");  Job\_Title.add(new job\_title(macv, tencv));  }  //7. Bảng Product  List<product> Product = new ArrayList<product>();  String table\_product = "SELECT \* FROM Product";  ResultSet rs\_product = st.executeQuery(table\_product);  while(rs\_product.next()) {  int masp = rs\_product.getInt("MaSP");  String tensp = rs\_product.getString("TenSP");  double dongia = rs\_product.getDouble("DonGia");  Date hansudung = rs\_product.getDate("HanSuDung");  String noisanxuat = rs\_product.getString("NoiSanXuat");  int manhom = rs\_product.getInt("MaNhom");  int mancc = rs\_product.getInt("MaNCC");  Product.add(new product(masp, tensp, dongia, hansudung, noisanxuat, manhom, mancc));  }  //8. Bảng Promotion  List<promotion> Promotion = new ArrayList<promotion>();  String table\_promotion = "SELECT \* FROM Promotion";  ResultSet rs\_promotion = st.executeQuery(table\_promotion);  while(rs\_promotion.next()) {  int makm = rs\_promotion.getInt("MaKM");  String noidung = rs\_promotion.getString("NoiDung");  Date thoigianbatdau = rs\_promotion.getDate("ThoiGianBatDau");  Date thoigianketthuc = rs\_promotion.getDate("ThoiGianKetThuc");  String doituongapdung = rs\_promotion.getString("DoiTuongApDung");  double mucgiamgia = rs\_promotion.getDouble("MucGiamGia");  Promotion.add(new promotion(makm, noidung, thoigianbatdau, thoigianketthuc, doituongapdung, mucgiamgia));  }  //9. Supplier  List<supplier> Supplier = new ArrayList<supplier>();  String table\_supplier = "SELECT \* FROM Supplier";  ResultSet rs\_supplier = st.executeQuery(table\_supplier);  while(rs\_supplier.next()) {  int mancc = rs\_supplier.getInt("MaNCC");  String tenncc = rs\_supplier.getString("TenNCC");  String diacchi = rs\_supplier.getString("DiaChi");  String sdt = rs\_supplier.getString("SDT");  String email = rs\_supplier.getString("Email");  Supplier.add(new supplier(mancc, tenncc, diacchi, sdt, email));  }  //10. User\_Account  List<user\_account> User\_Account = new ArrayList<user\_account>();  String table\_useraccount = "SELECT \* FROM User\_Account";  ResultSet rs\_useraccount = st.executeQuery(table\_useraccount);  while(rs\_useraccount.next()) {  String tendangnhap = rs\_useraccount.getString("TenDangNhap");  String matkhau = rs\_useraccount.getString("MatKhau");  User\_Account.add(new user\_account(tendangnhap, matkhau));  }    //Bước 4: Xử lý các truy vấn.  //20 câu lệnh java lambada  //1. Lấy ra thông tin nhân viên của Circle K (tên, giới tính, ngày sáng năm sinh, địa chỉ ở)  System.out.println("Cau 1: Thong tin nhan vien.");  Employee.forEach(e -> System.out.println("Ho ten: " + e.getHoTen() + "- Gioi tinh: " + e.getGioiTinh() + "- Ngay sinh: " + e.getNgaySinh() + "- Dia chi: " + e.getDiaChi()));  System.out.println("---------------------------------------------------");  //2. Tìm nhân viên bán hàng  System.out.println("Cau 2: Nhan vien ban hang.");  //nhân viên bán hàng có MaCV = 1;  int nhanvienbanhang = 1;  Employee.stream()  .filter(e -> e.getMaCV() == nhanvienbanhang)  .forEach(s -> System.out.println(s.getHoTen()));  System.out.println("---------------------------------------------------");  //3. lấy ra tên và số điện thoại của khách hàng có số lượt mua trên 5  System.out.println("Cau 3: Tim khach hang co so lan mua tren 5.");  Customer.stream()  .filter(c -> c.getSoLanMua() > 5)  .forEach(c -> System.out.println(c.getHoTen() + " - " + c.getSDT()));  System.out.println("---------------------------------------------------");  //4. lấy ra lời phản hồi của khách hàng có mã khách hàng là 3 và lượt đánh giá từ 4 sao  System.out.println("Cau 4: MaKH = 3 va danh gia tu 4 sao .");  Feedback.stream()  .filter(f -> f.getMaKH() == 3 && f.getSoDiemDanhGia() >= 4)  .forEach(f -> System.out.println(f.getNgayPhanHoi() + " - " + f.getNoiDung()));  System.out.println("---------------------------------------------------");  //5. Lấy tên của khách hàng đã đánh giá 5 sao  System.out.println("Cau 5: Ten khach hang danh gia 5 sao:");  Feedback.stream()  .filter(f -> f.getSoDiemDanhGia() == 5)  .flatMap(f -> Customer.stream()  .filter(c -> c.getMaKH() == f.getMaKH())  .map(c -> new String[]{c.getHoTen(), c.getSDT()}))  .distinct()  .forEach(c -> System.out.println(c[0] + " - " + c[1]));  System.out.println("---------------------------------------------------");  //6. In ra những hóa đơn kèm theo tên của người mua  System.out.println("Cau 6: Ten va so tien:");  Invoice.stream()  .map(i -> new Object[]{i.getMaKH(), i.getTongSoTien()})  .forEach(i -> {  Customer.stream()  .filter(c -> c.getMaKH() == (int) i[0])  .map(cs -> cs.getHoTen())  .forEach(h ->{  System.out.println(h + " - " + i[1]);  });  });  System.out.println("---------------------------------------------------");  //7.tìm những sản phẩm của Vinamilk có đơn giá trên 25000  System.out.println("Cau 7: San pham cua Vinamilk co don gia tren 25000");  Supplier.stream()  .filter(s -> s.getTenNCC().equals("Vinamilk"))  .map(s -> s.getMaNCC())  .forEach(ma -> {  Product.stream()  .filter(p -> p.getMaNCC() == ma && p.getDonGia() > 25000)  .forEach(p -> {  System.out.println(p.getTenSP() + " - " + p.getDonGia());  });  });  System.out.println("---------------------------------------------------");  // 8. Đếm số lượng nhân viên trong Circle K  System.out.println("Cau 8: So luong nhan vien trong Circle K");  long soluongnhanvien = Employee.stream().count();  System.out.println(soluongnhanvien);  System.out.println("---------------------------------------------------");  //9. Tính tổng sản phẩm của mỗi nhà cung cấp  System.out.println("Cau 9: Tong san pham cua moi nha cung cap");  Map<Integer, Long> tongsanpham = Product.stream()  .collect(Collectors.groupingBy(p -> Integer.valueOf(p.getMaNCC()), Collectors.counting()));  tongsanpham.forEach((maNCC, tong) -> {  Supplier.stream()  .filter(s -> s.getMaNCC() == maNCC)  .forEach(s -> System.out.println(s.getTenNCC() + " - " + tong));  });  System.out.println("---------------------------------------------------");  //10. Tính tổng số tiền trung bình của các khách hàng  System.out.println("Cau 10: Tinh tong so tien trung binh cua cac khach hang: ");  double tongInvoice = Invoice.stream()  .mapToDouble(i -> i.getTongSoTien()).sum();  long soLuongKhachHang = Customer.stream().count();  System.out.println(tongInvoice / soLuongKhachHang);  System.out.println("---------------------------------------------------");  //11. Lấy ra những phiếu khuyến mãi được áp dụng trong thời gian hiện tại  System.out.println("Cau 11: khuyen mai dang dien ra");  java.util.Date currentDate = new java.util.Date();  Promotion.stream()  .filter(pm -> !pm.getThoiGianBatDau().after(currentDate) && !pm.getThoiGianKetThuc().before(currentDate))  .forEach(pm -> System.out.println(pm.getMaKM() + " - "+ pm.getNoiDung() + " - " + pm.getMucGiamGia()));  System.out.println("---------------------------------------------------");  //12. Tìm sản phẩm hết hạn trong 30 ngày tính từ ngày thời điểm hiện tại  System.out.println("Cau 12: San pham het han trong 30 ngay");  java.util.Date today = new java.util.Date();  Product.stream()  .filter(p -> p.getHanSuDung().before(new java.sql.Date(today.getTime() + (30L \* 24 \* 60 \* 60 \* 1000))))  .forEach(p -> System.out.println(p.getTenSP() + " - " + p.getHanSuDung() + " - " + p.getNoiSanXuat()));  System.out.println("---------------------------------------------------");  //13. Với mỗi Tên Công việc lấy ra tên của từng nhân viên trong đó  System.out.println("Cau 13: Lay ra nhan vien cua tung cong viec");  Job\_Title.stream()  .forEach(job -> {  System.out.println(job.getTenCV() + ": ");  Employee.stream()  .filter(e -> e.getMaCV() == job.getMaCV())  .forEach(e -> System.out.println(e.getHoTen()));  });  System.out.println("---------------------------------------------------");  //14. Tính tổng đơn giá của các sản phẩm của mỗi nhà cung cấp  System.out.println("Cau 14: Don gia cua nha cung cap.");  Map<Integer, Double> tongdongia = Product.stream()  .collect(Collectors.groupingBy(  p -> Integer.valueOf(p.getMaNCC()),  Collectors.summingDouble(p -> p.getDonGia())  ));  Supplier.forEach(s -> {  Double tonggia = tongdongia.get(s.getMaNCC());  if (tonggia != null) {  System.out.println(s.getTenNCC() + " - " + tonggia);  }  });  System.out.println("---------------------------------------------------");  //15. Lấy ra sản phẩm thuộc nhóm "Đồ uống" và sắp xếp theo đơn giá giảm dần  System.out.println("Câu 15: San pham đo uong.");  Product.stream()  .filter(p -> Category.stream()  .anyMatch(ca -> ca.getMaNhom() == p.getMaNhom() && ca.getTenNhom().equals("Đo uong")))  .sorted((p1, p2) -> Double.compare(p2.getDonGia(), p1.getDonGia()))  .forEach(p -> System.out.println(p.getTenSP() + " - " + p.getDonGia()));  System.out.println("---------------------------------------------------");  //16. lấy ra sản phẩm thuộc nhóm đồ ăn và sắp xếp theo đơn giá tăng dần  System.out.println("Câu 16: San pham do an.");  Product.stream()  .filter(p -> Category.stream()  .anyMatch(ca -> ca.getMaNhom() == p.getMaNhom() && ca.getTenNhom().equals("Đo an")))  .sorted((p1, p2) -> Double.compare(p1.getDonGia(), p2.getDonGia()))  .forEach(p -> System.out.println(p.getTenSP() + " - " + p.getDonGia()));  System.out.println("---------------------------------------------------");  //17. Lấy ra thông tin khách hàng đã đánh giá và sắp xếp tăng dần theo số lần mua  System.out.println("Cau 17: Thong tin khach hang danh gia va sap xep theo so lan mua");  Feedback.stream()  .filter(f -> f.getSoDiemDanhGia() != null)  .flatMap(f -> Customer.stream()  .filter(c -> c.getMaKH() == f.getMaKH())  .map(c -> new Object[]{c.getHoTen(), c.getSoLanMua(), f.getNoiDung() })  )  .sorted((p1, p2) -> {  Integer soLanMua1 = (Integer) p1[1];  Integer soLanMua2 = (Integer) p2[1];  return Integer.compare(soLanMua1, soLanMua2);  })  .forEach(tt -> System.out.println(tt[0] + " - " + tt[1] + " - " + tt[2]));  System.out.println("---------------------------------------------------");  //18. lấy ra thông tin khách hàng không đánh giá của hàng Circle K  System.out.println("Cau 18: Thong tin khach hang chua danh gia Circle K.");  Customer.stream()  .filter(c -> Feedback.stream()  .noneMatch(f -> f.getMaKH() == c.getMaKH() && f.getSoDiemDanhGia() != null))  .collect(Collectors.toList())  .forEach(c ->  System.out.println(c.getHoTen() + " - " + c.getSDT()));  System.out.println("---------------------------------------------------");  // 19. Lấy ra thông tin khách hàng mua đơn dưới 100.000 kèm theo lời phản hồi  System.out.println("Cau 19: Thong tin khach hang mua duoi 100000 va kem theo loi phan hoi");  Invoice.stream()  .filter(i -> i.getTongSoTien() < 100000)  .collect(Collectors.groupingBy(i -> i.getMaKH(), Collectors.summingDouble(i -> i.getTongSoTien())))  .entrySet().stream()  .flatMap(en -> Customer.stream()  .filter(c -> c.getMaKH() == en.getKey())  .flatMap(c -> Feedback.stream()  .filter(f -> f.getMaKH() == c.getMaKH())  .map(f -> new Object[]{c.getHoTen(), f.getNoiDung(), en.getValue()})))  .forEach(r -> System.out.println(r[0] + " - " + r[1] + " - " + r[2]));  System.out.println("---------------------------------------------------");  // 20. Tính tổng số tiền của các hóa đơn theo khách hàng và sắp xếp theo thứ tự giảm dần của tổng tiền  System.out.println("Cau 20: Tong tien hoa don theo khach hang duoc sap xep giam dan");  Map<Integer, Double> tongTienTheoKhachHang = Invoice.stream()  .collect(Collectors.groupingBy(  i -> i.getMaKH(),  Collectors.summingDouble(i -> i.getTongSoTien())  ));  tongTienTheoKhachHang.entrySet().stream()  .sorted(Map.Entry.<Integer, Double>comparingByValue().reversed())  .forEach(en -> {  int maKH = en.getKey();  double tongTien = en.getValue();  Customer.stream()  .filter(c -> c.getMaKH() == maKH)  .forEach(c -> {  System.out.println(c.getHoTen() + " - " + c.getSDT() + " - " + tongTien);  });  });  System.out.println("-----------------------------------------------");  //Bước 5: Đóng kết nối  JDBCUtil.closeConnection(connection);    } catch (Exception e) {  e.printStackTrace();  }      }  } |

**3. Giải thích chương trình**

1. **Tạo kết nối với CSDL**: để kết nối với CSDL ta thực hiện gọi lệnh Connection connection = **new** JDBCUtil().*getConnection*() để lấy ra kết nối đã được tạo ở lớp JDBCUtil.
2. **Tạo đối tượng Statement**: để thực thi các câu lệnh SQL, Statement là một interface của JDBC(Java Database Conectivity).
3. **Tải dữ liệu từ cơ sở dữ liệu**: Bằng việc tạo ra các List, các List này chỉ chứa dữ liệu thuộc kiểu đối tượng tương ứng với các class ta đã tạo bên trên (Category, Customer, Employee, Feedback, Incoice, Job\_Title, Product, Promotion, Supplier, User\_Account).
4. **Lấy ra thông tin nhân viên của Circle K (tên, giới tính, ngày sáng năm sinh, địa chỉ ở)**

|  |
| --- |
| System.***out***.println("Cau 1: Thong tin nhan vien.");  Employee.forEach(e -> System.***out***.println("Ho ten: " + e.getHoTen() + "- Gioi tinh: " + e.getGioiTinh() + "- Ngay sinh: " + e.getNgaySinh() + "- Dia chi: " + e.getDiaChi()));  System.***out***.println("---------------------------------------------------"); |

Câu lệnh Java Lambdas trên thực hiện lấy thông tin nhân viên Circle K. Để lấy thông tin ta cần truy cập vào bảng **Employee** lấy ra **HoTen, GioiTinh, NgaySinh, DiaChi** của nhân viên.

1. **Tìm nhân viên bán hàng**

|  |
| --- |
| System.***out***.println("Cau 2: Nhan vien ban hang.");  //nhân viên bán hàng có MaCV = 1;  **int** nhanvienbanhang = 1;  Employee.stream()  .filter(e -> e.getMaCV() == nhanvienbanhang)  .forEach(s -> System.***out***.println(s.getHoTen()));  System.***out***.println("---------------------------------------------------"); |

Câu lệnh Java Lambdas trên thực hiện việc tìm nhân viên bán hàng, nhân viên bán hàng thì có MaCV = 1. Ta cần tạo một biến **nhanvienbanhang** và có giá trị 1. Thực hiện việc so sánh từ bảng **Employee** lấy ra **MaCV** và so sánh với **nhanvienbanhang** bằng hàm **filter**. Nếu là nhân viên bán hàng thì thực hiện **foreach** để in ra **HoTen** của nhân viên bán hàng.

1. **lấy ra tên và số điện thoại của khách hàng có số lượt mua trên 5**

|  |
| --- |
| System.***out***.println("Cau 3: Tim khach hang co so lan mua tren 5.");  Customer.stream()  .filter(c -> c.getSoLanMua() > 5)  .forEach(c -> System.***out***.println(c.getHoTen() + " - " + c.getSDT()));  System.***out***.println("---------------------------------------------------"); |

Câu lệnh trên thực hiện việc tìm khách hàng có số lượt mua trên 5. Để thực hiện được ta cần truy cập vào bảng **Customer** và **SoLanMua** > 5 bằng lệnh **filter**. Sau đó ta in ra tên khách hàng và kèm theo số điện thoại.

1. **lấy ra lời phản hồi của khách hàng có mã khách hàng là 3 và lượt đánh giá từ 4 sao**

|  |
| --- |
| System.***out***.println("Cau 4: MaKH = 3 va danh gia tu 4 sao .");  Feedback.stream()  .filter(f -> f.getMaKH() == 3 && f.getSoDiemDanhGia() >= 4) .forEach(f -> System.***out***.println(f.getNgayPhanHoi() + " - " + f.getNoiDung()));  System.***out***.println("---------------------------------------------------"); |

Câu lệnh trên thực hiện việc lấy ra lời phản hồi của khách hàng có mã 3 và được đánh giá 4 sao cho Circle K. Để thực hiện được ra cần truy cập vào bảng **Feedback** lọc **MaKH** = 3 kèm theo điều kiện **SoDiemDanhGia** >= 4. Nếu tìm thấy sẽ trả về ngày phản hồi và nội dung phải hồi của khách hàng đó.

1. **Lấy tên của khách hàng đã đánh giá 5 sao**

|  |
| --- |
| System.***out***.println("Cau 5: Ten khach hang danh gia 5 sao:");  Feedback.stream()  .filter(f -> f.getSoDiemDanhGia() == 5)  .flatMap(f -> Customer.stream()  .filter(c -> c.getMaKH() == f.getMaKH())  .map(c -> **new** String[]{c.getHoTen(), c.getSDT()}))  .distinct()  .forEach(c -> System.***out***.println(c[0] + " - " + c[1]));  System.***out***.println("---------------------------------------------------"); |

Câu lệnh trên thực hiện việc tìm khách hàng đã đánh giá 5 sao cho khách hàng. Để thực hiện được ra cần truy cập vào bảng **Feedback** sau đó dùng lệnh **filter** để lấy điều kiện **SoDiemDanhGia** == 5. Sau đó ta thực hiện truy cập đến bảng **Customer** dùng **filter** để lọc lấy ra tên của khách hàng. Đồng thời gán tên và số điện thoại vào **map**. Sau đó chạy vòng **foreach** để thực hiện việc in thông tin từ **map** ra màn hình.

1. **In ra những hóa đơn kèm theo tên của người mua**

|  |
| --- |
| System.***out***.println("Cau 6: Ten va so tien:");  Invoice.stream()  .map(i -> **new** Object[]{i.getMaKH(), i.getTongSoTien()})  .forEach(i -> {  Customer.stream()  .filter(c -> c.getMaKH() == (**int**) i[0])  .map(cs -> cs.getHoTen())  .forEach(h ->{  System.***out***.println(h + " - " + i[1]);  });  });  System.***out***.println("---------------------------------------------------"); |

Câu lệnh trên thực hiện việc lấy ra hóa đơn của khách hàng đã mua hàng ở Circle K. Để thực hiện được ta cần truy cập vào bảng **Invoice** sau đó gán vào **map** những **MaKH** và **TongSoTien**. Sau đó ta dùng **foreach** để với mỗi **MaKH** ta truy cập vào bảng **Customer** để lấy ra tên của khách hàng.

1. **tìm những sản phẩm của Vinamilk có đơn giá trên 25000**

|  |
| --- |
| System.***out***.println("Cau 7: San pham cua Vinamilk co don gia tren 25000");  Supplier.stream()  .filter(s -> s.getTenNCC().equals("Vinamilk"))  .map(s -> s.getMaNCC())  .forEach(ma -> {  Product.stream()  .filter(p -> p.getMaNCC() == ma && p.getDonGia() > 25000)  .forEach(p -> {  System.***out***.println(p.getTenSP() + " - " + p.getDonGia());  });  });  System.***out***.println("---------------------------------------------------"); |

Câu lệnh trên thực hiện việc tìm kiếm những sản phẩm của Vinamilk mà những sản phẩm đó phải có giá trên 25000. Để thực hiện ta tần truy cập vào bảng **Supplier**. Sau đó ta dùng **filter** để tìm kiếm tên nhà cung cấp là Vinamilk. Nếu tìm được ta sẽ chạy vòng **foreach** và truy cập vào bảng **Product** để tìm kiếm những sản phẩm nào có **MaNCC** là của Vinamilk đồng thời sản phẩm đó phải có giá trên 25000. Cuối cùng là thực hiện in thông tin ra màn hình theo yêu cầu.

1. **Đếm số lượng nhân viên trong Circle K**

|  |
| --- |
| System.***out***.println("Cau 8: So luong nhan vien trong Circle K");  **long** soluongnhanvien = Employee.stream().count();  System.***out***.println(soluongnhanvien);  System.***out***.println("---------------------------------------------------"); |

Câu lệnh trên thực hiện việc đếm số lượng nhân viên có trong cửa hàng Circle K. Để thực hiện ra cần truy cập vào bảng **Employee** và dùng hàm **count** đế đếm.

1. **Tính tổng sản phẩm của mỗi nhà cung cấp**

|  |
| --- |
| System.***out***.println("Cau 9: Tong san pham cua moi nha cung cap");  Map<Integer, Long> tongsanpham = Product.stream()  .collect(Collectors.*groupingBy*(p -> Integer.*valueOf*(p.getMaNCC()), Collectors.*counting*()));  tongsanpham.forEach((maNCC, tong) -> {  Supplier.stream()  .filter(s -> s.getMaNCC() == maNCC)  .forEach(s -> System.***out***.println(s.getTenNCC() + " - " + tong));  }); |

Câu lệnh trên thực hiện việc đếm số lượng sản phẩm của mỗi nhà cung cấp trong cửa hàng Circle K. Để thực hiện ta cần truy cập vào bảng **Product** dùng **groupby** để gom nhóm theo **MaNCC** đồng thời thực hiện việc đếm bằng hàm **couting()**. Sau đó ta sẽ dùng vòng **foreach** để in ra thông tin theo yêu cầu.

1. **Tính tổng số tiền trung bình của các khách hàng**

|  |
| --- |
| System.***out***.println("Cau 10: Tinh tong so tien trung binh cua cac khach hang: ");  **double** tongInvoice = Invoice.stream()  .mapToDouble(i -> i.getTongSoTien()).sum();  **long** soLuongKhachHang = Customer.stream().count();  System.***out***.println(tongInvoice / soLuongKhachHang);  System.***out***.println("---------------------------------------------------"); |

Câu lệnh trên thực hiện việc tỉnh tổng số tiền bình quân theo số lượng khách hàng. Để thực hiện ta cần truy cập vào bảng **Invoice** để tính tổng số tiền. Và truy cập vào bảng **Customer** để đến số lượng khách hàng.

1. **Lấy ra những phiếu khuyến mãi được áp dụng trong thời gian hiện tại**

|  |
| --- |
| System.***out***.println("Cau 11: khuyen mai dang dien ra");  java.util.Date currentDate = **new** java.util.Date();  Promotion.stream()  .filter(pm -> !pm.getThoiGianBatDau().after(currentDate) && !pm.getThoiGianKetThuc().before(currentDate))  .forEach(pm -> System.***out***.println(pm.getMaKM() + " - "+ pm.getNoiDung() + " - " + pm.getMucGiamGia()));  System.***out***.println("---------------------------------------------------"); |

Câu lệnh trên thực hiện việc in ra phiếu khuyến mãi mà được áp dụng trong thời điển hiện tại. Để thực hiện ta cần lấy ra thời gian hiện tại bằng cách tạo biến **currentDate**, biến này có kiểu dữ liệu **Date** là kiễu dữ liệu có sẵn trong java. Sau đó truy cập vào bảng **Promotion** kiểm tra xem thời gian hiện tại có nằm trong khoảng thời gian bắt đầu và kết thúc của phiếu khuyến mãi hay không nếu có thì in phiếu đó ra.

1. **Tìm sản phẩm hết hạn trong 30 ngày tính từ ngày thời điểm hiện tại**

|  |
| --- |
| System.***out***.println("Cau 12: San pham het han trong 30 ngay");  java.util.Date today = **new** java.util.Date();  Product.stream()  .filter(p -> p.getHanSuDung().before(**new** java.sql.Date(today.getTime() + (30L \* 24 \* 60 \* 60 \* 1000))))  .forEach(p -> System.***out***.println(p.getTenSP() + " - " + p.getHanSuDung() + " - " + p.getNoiSanXuat()));  System.***out***.println("---------------------------------------------------"); |

Câu lệnh trên thực hiện tìm ra sản phẩm hết hạn trong 30 ngày tính từ thời điểm hiện tại. Đế thực hiện ta cần khai báo ngày hiện tại là **today**. Sau đó truy cập vào **Product** kiểm tra xem hạn sử dụng có nằm trước ngày hiện tại cộng thêm 30 ngày hay không. Nếu có thì in thông tin sản phẩm đó ra màn hình.

1. **Với mỗi Tên Công việc lấy ra tên của từng nhân viên trong đó**

|  |
| --- |
| System.***out***.println("Cau 13: Lay ra nhan vien cua tung cong viec");  Job\_Title.stream()  .forEach(job -> {  System.***out***.println(job.getTenCV() + ": ");  Employee.stream()  .filter(e -> e.getMaCV() == job.getMaCV())  .forEach(e -> System.***out***.println(e.getHoTen()));  });  System.***out***.println("---------------------------------------------------"); |

Câu lệnh trên thực hiện việc ứng với mỗi công việc trong quán lấy ra tên của từng nhân viên giữ các chức vụ đó. Để thực hiện ta cần truy cập vào bảng **Jpb\_Title** sau đó in ra tên từng công việc. Tiếp theo sẽ chạy vòng **foreach** để tìm kiếm nhân viên phụ trách công việc đó nằm trong bảng **Employee**.

1. **Tính tổng đơn giá của các sản phẩm của mỗi nhà cung cấp**

|  |
| --- |
| System.***out***.println("Cau 14: Don gia cua nha cung cap.");  Map<Integer, Double> tongdongia = Product.stream()  .collect(Collectors.*groupingBy*(  p -> Integer.*valueOf*(p.getMaNCC()),  Collectors.*summingDouble*(p -> p.getDonGia())  ));  Supplier.forEach(s -> {  Double tonggia = tongdongia.get(s.getMaNCC());  **if** (tonggia != **null**) {  System.***out***.println(s.getTenNCC() + " - " + tonggia);  }  });  System.***out***.println("---------------------------------------------------"); |

Câu lệnh trên thực hiện việc tính tổng giá của các sản phẩn theo mỗi nhà cung cấp. Để thực hiện ta cần tạo một **Map** để lưu giá trị key là **MaNCC** còn giá trị value là tổng giá thành theo các sản phẩm của **MaNCC**. Tiếp đến ta truy cập vào bảng **Supplier** in ra thông tin giá thành của các sản phẩm theo mỗi nhà cung cấp.

1. **Lấy ra sản phẩm thuộc nhóm "Đồ uống" và sắp xếp theo đơn giá giảm dần**

|  |
| --- |
| System.***out***.println("Câu 15: San pham đo uong.");  Product.stream()  .filter(p -> Category.stream()  .anyMatch(ca -> ca.getMaNhom() == p.getMaNhom() && ca.getTenNhom().equals("Đo uong")))  .sorted((p1, p2) -> Double.*compare*(p2.getDonGia(), p1.getDonGia()))  .forEach(p -> System.***out***.println(p.getTenSP() + " - " + p.getDonGia()));  System.***out***.println("---------------------------------------------------"); |

Câu lệnh trên thực hiện việc lấy ra những sản phẩm nào thuộc loại đồ uống và phải sắp xếp theo giá của sản phẩm giảm dần. Để thực hiện ta cần truy cập vào bảng **Product** sau đó truy cập đến bảng **Category** rồi kết nối theo **MaNhom** của 2 bảng. Sau đó thì ra lọc ra nhóm đồ uống và thực hiện sắp xếp giảm dần theo đơn giá của sản phẩm.

1. **lấy ra sản phẩm thuộc nhóm đồ ăn và sắp xếp theo đơn giá tăng dần**

|  |
| --- |
| System.***out***.println("Câu 16: San pham do an.");  Product.stream()  .filter(p -> Category.stream()  .anyMatch(ca -> ca.getMaNhom() == p.getMaNhom() && ca.getTenNhom().equals("Đo an")))  .sorted((p1, p2) -> Double.*compare*(p1.getDonGia(), p2.getDonGia()))  .forEach(p -> System.***out***.println(p.getTenSP() + " - " + p.getDonGia()));  System.***out***.println("---------------------------------------------------"); |

Câu lệnh trên trái ngược với câu 16, thay vì chọn ra đồ uống thì ở đây chọn ra đồ ăn và sắp xếp chúng theo giá thành của sản phẩm tăng dần.

1. **Lấy ra thông tin khách hàng đã đánh giá và sắp xếp tăng dần theo số lần mua**

|  |
| --- |
| System.***out***.println("Cau 17: Thong tin khach hang danh gia va sap xep theo so lan mua");  Feedback.stream()  .filter(f -> f.getSoDiemDanhGia() != **null**)  .flatMap(f -> Customer.stream()  .filter(c -> c.getMaKH() == f.getMaKH())  .map(c -> **new** Object[]{c.getHoTen(), c.getSoLanMua(), f.getNoiDung() })  )  .sorted((p1, p2) -> {  Integer soLanMua1 = (Integer) p1[1];  Integer soLanMua2 = (Integer) p2[1];  **return** Integer.*compare*(soLanMua1, soLanMua2);  })  .forEach(tt -> System.***out***.println(tt[0] + " - " + tt[1] + " - " + tt[2]));  System.***out***.println("---------------------------------------------------"); |

Câu lệnh trên thực hiện việc lấy thông tin của khách hàng và sắp xếp theo số lần mà khách hàng đã mua tại Circle K. Để thực hiện ta cần truy cập đến bảng **Feedback** và lấy ra số điểm đánh giá khác null. Sau đó truy cập thêm bảng **Customer** để lấy thông tin của khách hàng mà ta cần tìm. Sau khi lấy được ta sẽ sắp xếp theo số lần mua tăng dần của khách hàng.

1. **lấy ra thông tin khách hàng không đánh giá của hàng Circle K**

|  |
| --- |
| System.***out***.println("Cau 18: Thong tin khach hang chua danh gia Circle K.");  Customer.stream()  .filter(c -> Feedback.stream()  .noneMatch(f -> f.getMaKH() == c.getMaKH() && f.getSoDiemDanhGia() != **null**))  .collect(Collectors.*toList*())  .forEach(c ->  System.***out***.println(c.getHoTen() + " - " + c.getSDT()));  System.***out***.println("---------------------------------------------------"); |

Câu lệnh trên cách thức vận hàng giống câu 17 nhưng thay vì tìm những người đánh giá thì ở đây yêu cầu tìm người chưa đánh giá. Để thực hiện ta truy cập vào bảng **Customer** và **Feedback** để thực hiện việc nối bảng lại với nhau nhằm mục đích lấy ra thông tin khách hàng.

1. **Lấy ra thông tin khách hàng mua đơn dưới 100.000 kèm theo lời phản hồi**

|  |
| --- |
| System.***out***.println("Cau 19: Thong tin khach hang mua duoi 100000 va kem theo loi phan hoi");  Invoice.stream()  .filter(i -> i.getTongSoTien() < 100000)  .collect(Collectors.*groupingBy*(i -> i.getMaKH(), Collectors.*summingDouble*(i -> i.getTongSoTien())))  .entrySet().stream()  .flatMap(en -> Customer.stream()  .filter(c -> c.getMaKH() == en.getKey())  .flatMap(c -> Feedback.stream()  .filter(f -> f.getMaKH() == c.getMaKH())  .map(f -> **new** Object[]{c.getHoTen(), f.getNoiDung(), en.getValue()})))  .forEach(r -> System.***out***.println(r[0] + " - " + r[1] + " - " + r[2]));  System.***out***.println("---------------------------------------------------"); |

Câu lệnh trên thực hiện việc tìm khách hàng đã mua với những hóa đơn có tổng số tiền dưới 100000 VNĐ. Để thực hiện ta cần truy cập vào bảng **Invoicce** để lấy ra tổng số tiền kiểm của mỗi khách hàng có dưới 100000 hay không. Sau đó gán thông tin vào **map**. Cuối cùng là in thông tin khách hàng từ **map**.

1. **Tính tổng số tiền của các hóa đơn theo khách hàng và sắp xếp theo thứ tự giảm dần của tổng tiền**

|  |
| --- |
| System.***out***.println("Cau 20: Tong tien hoa don theo khach hang duoc sap xep giam dan");  Map<Integer, Double> tongTienTheoKhachHang = Invoice.stream()  .collect(Collectors.*groupingBy*(  i -> i.getMaKH(),  Collectors.*summingDouble*(i -> i.getTongSoTien())  ));  tongTienTheoKhachHang.entrySet().stream()  .sorted(Map.Entry.<Integer, Double>*comparingByValue*().reversed())  .forEach(en -> {  **int** maKH = en.getKey();  **double** tongTien = en.getValue();  Customer.stream()  .filter(c -> c.getMaKH() == maKH)  .forEach(c -> {  System.***out***.println(c.getHoTen() + " - " + c.getSDT() + " - " + tongTien);  });  });  System.***out***.println("-----------------------------------------------"); |

Câu lệnh trên thực hiện việc tính toán tổng số tiền của hóa đơn theo khách hàng và sắp xếp theo thứ tự giảm dần của tổng số tiền. Để thực hiện ta cần tạo ra **map** và truy cập đến bảng **Invoice**, nơi để lưu trữ key là **MaKH** và value là **TongSoTien**. Sau đó ta sắp xếp theo tổng số tiền. Cuối cùng trước khi in ta truy vấn đến bảng **Customer** để lấy thông tin của khách hàng bao gồm họ và tên, số điện thoại, tổng số tiền.

**28. Ngắt kết nối với CSDL**: sau khi thực hiện hết truy vấn ta cần ngắt kết nối tới CSDL bằng việc trong lớp JDBCUtil gọi phương thức *closeConnection*() để ngắt kết nối.