import javax.swing.\*;

import java.sql.\*;

import java.util.Vector;

import javax.swing.table.DefaultTableModel;

import java.util.Locale;

class ProductDTO {

int prodID, quantity, userID;

double costPrice, sellPrice;

Double totalCost, totalRevenue;

String prodCode, prodName, date, suppCode, custCode, custName, brand;

public int getProdID() {

return prodID;

}

public void setProdID(int prodID) {

this.prodID = prodID;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

public int getUserID() {

return userID;

}

public void setUserID(int userID) {

this.userID = userID;

}

public double getCostPrice() {

return costPrice;

}

public void setCostPrice(double costPrice) {

this.costPrice = costPrice;

}

public double getSellPrice() {

return sellPrice;

}

public void setSellPrice(double sellPrice) {

this.sellPrice = sellPrice;

}

public Double getTotalCost() {

return totalCost;

}

public void setTotalCost(Double totalCost) {

this.totalCost = totalCost;

}

public Double getTotalRevenue() {

return totalRevenue;

}

public void setTotalRevenue(Double totalRevenue) {

this.totalRevenue = totalRevenue;

}

public String getProdCode() {

return prodCode;

}

public void setProdCode(String prodCode) {

this.prodCode = prodCode;

}

public String getProdName() {

return prodName;

}

public void setProdName(String prodName) {

this.prodName = prodName;

}

public String getDate() {

return date;

}

public void setDate(String date) {

this.date = date;

}

public String getSuppCode() {

return suppCode;

}

public void setSuppCode(String suppCode) {

this.suppCode = suppCode;

}

public String getCustCode() {

return custCode;

}

public void setCustCode(String custCode) {

this.custCode = custCode;

}

public String getCustName() {

return custName;

}

public void setCustName(String custName) {

this.custName = custName;

}

public String getBrand() {

return brand;

}

public void setBrand(String brand) {

this.brand = "Dummy Brand";

}

}

public class ProductDAO {

Connection conn = null;

PreparedStatement prepStatement = null;

PreparedStatement prepStatement2 = null;

Statement statement = null;

Statement statement2 = null;

ResultSet resultSet = null;

public ProductDAO() {

try {

conn = new ConnectionFactory().getConn();

statement = conn.createStatement();

statement2 = conn.createStatement();

} catch (Exception ex) {

}

}

public ResultSet getSuppInfo() {

try {

String query = "SELECT \* FROM suppliers";

resultSet = statement.executeQuery(query);

} catch (Exception e) {

}

return resultSet;

}

public ResultSet getCustInfo() {

try {

String query = "SELECT \* FROM customers";

resultSet = statement.executeQuery(query);

} catch (Exception e) {

}

return resultSet;

}

public ResultSet getProdStock() {

try {

String query = "SELECT \* FROM currentstock";

resultSet = statement.executeQuery(query);

} catch (Exception e) {

}

return resultSet;

}

public ResultSet getProdInfo() {

try {

String query = "SELECT \* FROM products";

resultSet = statement.executeQuery(query);

} catch (Exception e) {

}

return resultSet;

}

public Double getProdCost(String prodCode) {

Double costPrice = null;

try {

String query = "SELECT costprice FROM products WHERE productcode='" +prodCode+ "'";

resultSet = statement.executeQuery(query);

if (resultSet.next())

costPrice = resultSet.getDouble("costprice");

} catch (Exception e) {

e.printStackTrace();

}

return costPrice;

}

public Double getProdSell(String prodCode) {

Double sellPrice = null;

try {

String query = "SELECT sellprice FROM products WHERE productcode='" +prodCode+ "'";

resultSet = statement.executeQuery(query);

if (resultSet.next())

sellPrice = resultSet.getDouble("sellprice");

} catch (Exception e) {

e.printStackTrace();

}

return sellPrice;

}

String suppCode;

public String getSuppCode(String suppName) {

try {

String query = "SELECT suppliercode FROM suppliers WHERE fullname='" +suppName+ "'";

resultSet = statement.executeQuery(query);

while (resultSet.next()) {

suppCode = resultSet.getString("suppliercode");

}

} catch (SQLException e) {

}

return suppCode;

}

String prodCode;

public String getProdCode(String prodName) {

try {

String query = "SELECT productcode FROM products WHERE productname='" +prodName+ "'";

resultSet = statement.executeQuery(query);

while (resultSet.next()) {

suppCode = resultSet.getString("productcode");

}

} catch (SQLException e) {

}

return prodCode;

}

String custCode;

public String getCustCode(String custName) {

try {

String query = "SELECT customercode FROM suppliers WHERE fullname='" +custName+ "'";

resultSet = statement.executeQuery(query);

while (resultSet.next()) {

suppCode = resultSet.getString("customercode");

}

} catch (SQLException e) {

}

return custCode;

}

// Method to check for availability of stock in Inventory

boolean flag = false;

public boolean checkStock(String prodCode) {

try {

String query = "SELECT \* FROM currentstock WHERE productcode='" +prodCode+ "'";

resultSet = statement.executeQuery(query);

while (resultSet.next()) {

flag = true;

}

} catch (SQLException e) {

}

return flag;

}

// Methods to add a new product

public void addProductDAO(ProductDTO productDTO) {

try {

String query = "SELECT \* FROM products WHERE productname='"

+ productDTO.getProdName()

+ "' AND costprice='"

+ productDTO.getCostPrice()

+ "' AND sellprice='"

+ productDTO.getSellPrice()

+ "' AND brand='"

+ productDTO.getBrand()

+ "'";

resultSet = statement.executeQuery(query);

if (resultSet.next())

JOptionPane.showMessageDialog(null, "Product has already been added.");

else

addFunction(productDTO);

} catch (SQLException e) {

}

}

public void addFunction(ProductDTO productDTO) {

try {

String query = "INSERT INTO products VALUES(null,?,?,?,?,?)";

prepStatement = (PreparedStatement) conn.prepareStatement(query);

prepStatement.setString(1, productDTO.getProdCode());

prepStatement.setString(2, productDTO.getProdName());

prepStatement.setDouble(3, productDTO.getCostPrice());

prepStatement.setDouble(4, productDTO.getSellPrice());

prepStatement.setString(5, productDTO.getBrand());

String query2 = "INSERT INTO currentstock VALUES(?,?)";

prepStatement2 = conn.prepareStatement(query2);

prepStatement2.setString(1, productDTO.getProdCode());

prepStatement2.setInt(2, productDTO.getQuantity());

prepStatement.executeUpdate();

prepStatement2.executeUpdate();

JOptionPane.showMessageDialog(null, "Product added and ready for sale.");

} catch (SQLException throwables) {

}

}

// Method to add a new purchase transaction

public void addPurchaseDAO(ProductDTO productDTO) {

try {

String query = "INSERT INTO purchaseinfo VALUES(null,?,?,?,?,?)";

prepStatement = conn.prepareStatement(query);

prepStatement.setString(1, productDTO.getSuppCode());

prepStatement.setString(2, productDTO.getProdCode());

prepStatement.setString(3, productDTO.getDate());

prepStatement.setInt(4, productDTO.getQuantity());

prepStatement.setDouble(5, productDTO.getTotalCost());

prepStatement.executeUpdate();

JOptionPane.showMessageDialog(null, "Purchase log added.");

} catch (SQLException throwables) {

}

String prodCode = productDTO.getProdCode();

if(checkStock(prodCode)) {

try {

String query = "UPDATE currentstock SET quantity=quantity+? WHERE productcode=?";

prepStatement = conn.prepareStatement(query);

prepStatement.setInt(1, productDTO.getQuantity());

prepStatement.setString(2, prodCode);

prepStatement.executeUpdate();

} catch (SQLException throwables) {

}

}

else if (!checkStock(prodCode)) {

try {

String query = "INSERT INTO currentstock VALUES(?,?)";

prepStatement = (PreparedStatement) conn.prepareStatement(query);

prepStatement.setString(1, productDTO.getProdCode());

prepStatement.setInt(2, productDTO.getQuantity());

prepStatement.executeUpdate();

} catch (SQLException throwables) {

}

}

deleteStock();

}

// Method to update existing product details

public void editProdDAO(ProductDTO productDTO) {

try {

String query = "UPDATE products SET productname=?,costprice=?,sellprice=?,brand=? WHERE productcode=?";

prepStatement = (PreparedStatement) conn.prepareStatement(query);

prepStatement.setString(1, productDTO.getProdName());

prepStatement.setDouble(2, productDTO.getCostPrice());

prepStatement.setDouble(3, productDTO.getSellPrice());

prepStatement.setString(4, productDTO.getBrand());

prepStatement.setString(5, productDTO.getProdCode());

String query2 = "UPDATE currentstock SET quantity=? WHERE productcode=?";

prepStatement2 = conn.prepareStatement(query2);

prepStatement2.setInt(1, productDTO.getQuantity());

prepStatement2.setString(2, productDTO.getProdCode());

prepStatement.executeUpdate();

prepStatement2.executeUpdate();

JOptionPane.showMessageDialog(null, "Product details updated.");

} catch (SQLException throwables) {

}

}

// Methods to handle updating of stocks in Inventory upon any transaction made

public void editPurchaseStock(String code, int quantity) {

try {

String query = "SELECT \* FROM currentstock WHERE productcode='" +code+ "'";

resultSet = statement.executeQuery(query);

if(resultSet.next()) {

String query2 = "UPDATE currentstock SET quantity=quantity-? WHERE productcode=?";

prepStatement = conn.prepareStatement(query2);

prepStatement.setInt(1, quantity);

prepStatement.setString(2, code);

prepStatement.executeUpdate();

}

} catch (SQLException throwables) {

}

}

public void editSoldStock(String code, int quantity) {

try {

String query = "SELECT \* FROM currentstock WHERE productcode='" +code+ "'";

resultSet = statement.executeQuery(query);

if(resultSet.next()) {

String query2 = "UPDATE currentstock SET quantity=quantity+? WHERE productcode=?";

prepStatement = conn.prepareStatement(query2);

prepStatement.setInt(1, quantity);

prepStatement.setString(2, code);

prepStatement.executeUpdate();

}

} catch (SQLException throwables) {

}

}

public void deleteStock() {

try {

String query = "DELETE FROM currentstock WHERE productcode NOT IN(SELECT productcode FROM purchaseinfo)";

String query2 = "DELETE FROM salesinfo WHERE productcode NOT IN(SELECT productcode FROM products)";

statement.executeUpdate(query);

statement.executeUpdate(query2);

} catch (SQLException throwables) {

}

}

// Method to permanently delete a product from inventory

public void deleteProductDAO(String code) {

try {

String query = "DELETE FROM products WHERE productcode=?";

prepStatement = conn.prepareStatement(query);

prepStatement.setString(1, code);

String query2 = "DELETE FROM currentstock WHERE productcode=?";

prepStatement2 = conn.prepareStatement(query2);

prepStatement2.setString(1, code);

prepStatement.executeUpdate();

prepStatement2.executeUpdate();

JOptionPane.showMessageDialog(null, "Product has been removed.");

} catch (SQLException e){

}

deleteStock();

}

public void deletePurchaseDAO(int ID){

try {

String query = "DELETE FROM purchaseinfo WHERE purchaseID=?";

prepStatement = conn.prepareStatement(query);

prepStatement.setInt(1, ID);

prepStatement.executeUpdate();

JOptionPane.showMessageDialog(null, "Transaction has been removed.");

} catch (SQLException e){

}

deleteStock();

}

// Products data set retrieval for display

public ResultSet getQueryResult() {

try {

String query = "SELECT productcode,productname,costprice,sellprice,brand FROM products ORDER BY pid";

resultSet = statement.executeQuery(query);

} catch (SQLException throwables) {

throwables.printStackTrace();

}

return resultSet;

}

// Purchase table data set retrieval

public ResultSet getPurchaseInfo() {

try {

String query = "SELECT PurchaseID,purchaseinfo.ProductCode,ProductName,Quantity,Totalcost " +

"FROM purchaseinfo INNER JOIN products " +

"ON products.productcode=purchaseinfo.productcode ORDER BY purchaseid;";

resultSet = statement.executeQuery(query);

} catch (SQLException throwables) {

throwables.printStackTrace();

}

return resultSet;

}

// Stock table data set retrieval

public ResultSet getCurrentStockInfo() {

try {

String query = """

SELECT currentstock.ProductCode,products.ProductName,

currentstock.Quantity,products.CostPrice,products.SellPrice

FROM currentstock INNER JOIN products

ON currentstock.productcode=products.productcode;

""";

resultSet = statement.executeQuery(query);

} catch (SQLException throwables) {

throwables.printStackTrace();

}

return resultSet;

}

// Search method for products

public ResultSet getProductSearch(String text) {

try {

String query = "SELECT productcode,productname,costprice,sellprice,brand FROM products " +

"WHERE productcode LIKE '%"+text+"%' OR productname LIKE '%"+text+"%' OR brand LIKE '%"+text+"%'";

resultSet = statement.executeQuery(query);

} catch (SQLException e) {

}

return resultSet;

}

public ResultSet getProdFromCode(String text) {

try {

String query = "SELECT productcode,productname,costprice,sellprice,brand FROM products " +

"WHERE productcode='" +text+ "' LIMIT 1";

resultSet = statement.executeQuery(query);

} catch (SQLException e) {

e.printStackTrace();

}

return resultSet;

}

// Search method for purchase logs

public ResultSet getPurchaseSearch(String text) {

try {

String query = "SELECT PurchaseID,purchaseinfo.productcode,products.productname,quantity,totalcost " +

"FROM purchaseinfo INNER JOIN products ON purchaseinfo.productcode=products.productcode " +

"INNER JOIN suppliers ON purchaseinfo.suppliercode=suppliers.suppliercode" +

"WHERE PurchaseID LIKE '%"+text+"%' OR productcode LIKE '%"+text+"%' OR productname LIKE '%"+text+"%' " +

"OR suppliers.fullname LIKE '%"+text+"%' OR purchaseinfo.suppliercode LIKE '%"+text+"%' " +

"OR date LIKE '%"+text+"%' ORDER BY purchaseid";

resultSet = statement.executeQuery(query);

} catch (SQLException e) {

e.printStackTrace();

}

return resultSet;

}

public ResultSet getProdName(String code) {

try {

String query = "SELECT productname FROM products WHERE productcode='" +code+ "'";

resultSet = statement.executeQuery(query);

} catch (SQLException throwables) {

throwables.printStackTrace();

}

return resultSet;

}

public String getSuppName(int ID) {

String name = null;

try {

String query = "SELECT fullname FROM suppliers " +

"INNER JOIN purchaseinfo ON suppliers.suppliercode=purchaseinfo.suppliercode " +

"WHERE purchaseid='" +ID+ "'";

resultSet = statement.executeQuery(query);

if (resultSet.next())

name = resultSet.getString("fullname");

} catch (SQLException throwables) {

throwables.printStackTrace();

}

return name;

}

public String getCustName(int ID) {

String name = null;

try {

String query = "SELECT fullname FROM customers " +

"INNER JOIN salesinfo ON customers.customercode=salesinfo.customercode " +

"WHERE salesid='" +ID+ "'";

resultSet = statement.executeQuery(query);

if (resultSet.next())

name = resultSet.getString("fullname");

} catch (SQLException throwables) {

throwables.printStackTrace();

}

return name;

}

public String getPurchaseDate(int ID) {

String date = null;

try {

String query = "SELECT date FROM purchaseinfo WHERE purchaseid='" +ID+ "'";

resultSet = statement.executeQuery(query);

if (resultSet.next())

date = resultSet.getString("date");

} catch (SQLException throwables) {

throwables.printStackTrace();

}

return date;

}

public String getSaleDate(int ID) {

String date = null;

try {

String query = "SELECT date FROM salesinfo WHERE salesid='" +ID+ "'";

resultSet = statement.executeQuery(query);

if (resultSet.next())

date = resultSet.getString("date");

} catch (SQLException throwables) {

throwables.printStackTrace();

}

return date;

}

// Method to display product-related data set in tabular form

public DefaultTableModel buildTableModel(ResultSet resultSet) throws SQLException {

ResultSetMetaData metaData = resultSet.getMetaData();

Vector<String> columnNames = new Vector<String>();

int colCount = metaData.getColumnCount();

for (int col=1; col <= colCount; col++){

columnNames.add(metaData.getColumnName(col).toUpperCase(Locale.ROOT));

}

Vector<Vector<Object>> data = new Vector<Vector<Object>>();

while (resultSet.next()) {

Vector<Object> vector = new Vector<Object>();

for (int col=1; col<=colCount; col++) {

vector.add(resultSet.getObject(col));

}

data.add(vector);

}

return new DefaultTableModel(data, columnNames);

}

}