Проект по „Бази от данни“

Тема: „Куриерска фирма“

Изготвил: Димитър Коджабашев, Ф. номер: 8MI0400006

I. Обхват на модела и дефиниране на задачата:

В дадена информационна система се съхраняват данни за куриерска фирма – информация за клиенти, работници, пратки, офиси и транспортните курсове, извършвани от фирмата.

За клиентите се пазят данните: ЕГН на служител (UCN – уникален идентификатор), адрес, тел. номер, име и фамилия.

За пратките се пази информация относно: номер - уникален за всяка пратка, адрес за доставка, приблизително време за доставка (в дни), статус на пратката (стойност измежду: „подготвяне“; „в процес на доставяне“; „доставена“) и цена за доставка на пратката.

Един клиент може да изпраща и получава множество пратки, като в системата се пазят датите на изпращане и получаване на съответните пратки.

Данни се пазят и за офисите на куриерската фирма: оценка на офис, уникален номер на офис, град, в който се намира, улица и брой работници.

Дадена пратка може да се намира в точно един офис в някакъв момент.

В офис работят куриери, за които се пази информация за заплата, номер на работник (уникален за всеки работник), оценка на обслужването, години стаж и име на работника.

В един офис могат да работят множество работници, докато работник е нает точно в един определен офис. Една пратка може да бъде доставена точно от един работник.

Един транспортен курс в системата се определя от: начална и крайна дата, статус на курса, уникален номер на курса, начална и крайна точка (градове) и транспортирано тегло.

Един служител може да извършва множество курсове, но един курс бива извършван от един служител. Дадена пратка се транспортирана по точно един определен (директен) курс.

II. Правила и проверки:

В сила са следните ограничения:

За таблица “Packages”:

- Датата на изпращане на пакет винаги е преди датата на получаването му
- Ориентировъчното време за доставка винаги е > 0 (в часове)
- Таксата за доставка винаги е > 0

За таблица “Course”:

- Датата на стартиране на курс винаги е преди датата на привършването му
- Номерът на курса винаги е положително число
- Теглото на курса (в кг.) винаги е число > 0

За таблица “DeliveryWorker”:

- Годините опит на работник са винаги >= 0
- Рейтингът на един работник е число между 0 и 5
- Заплатата на работник винаги е > 0
- Идентификационният номер на работник е низ с точно 6 символа

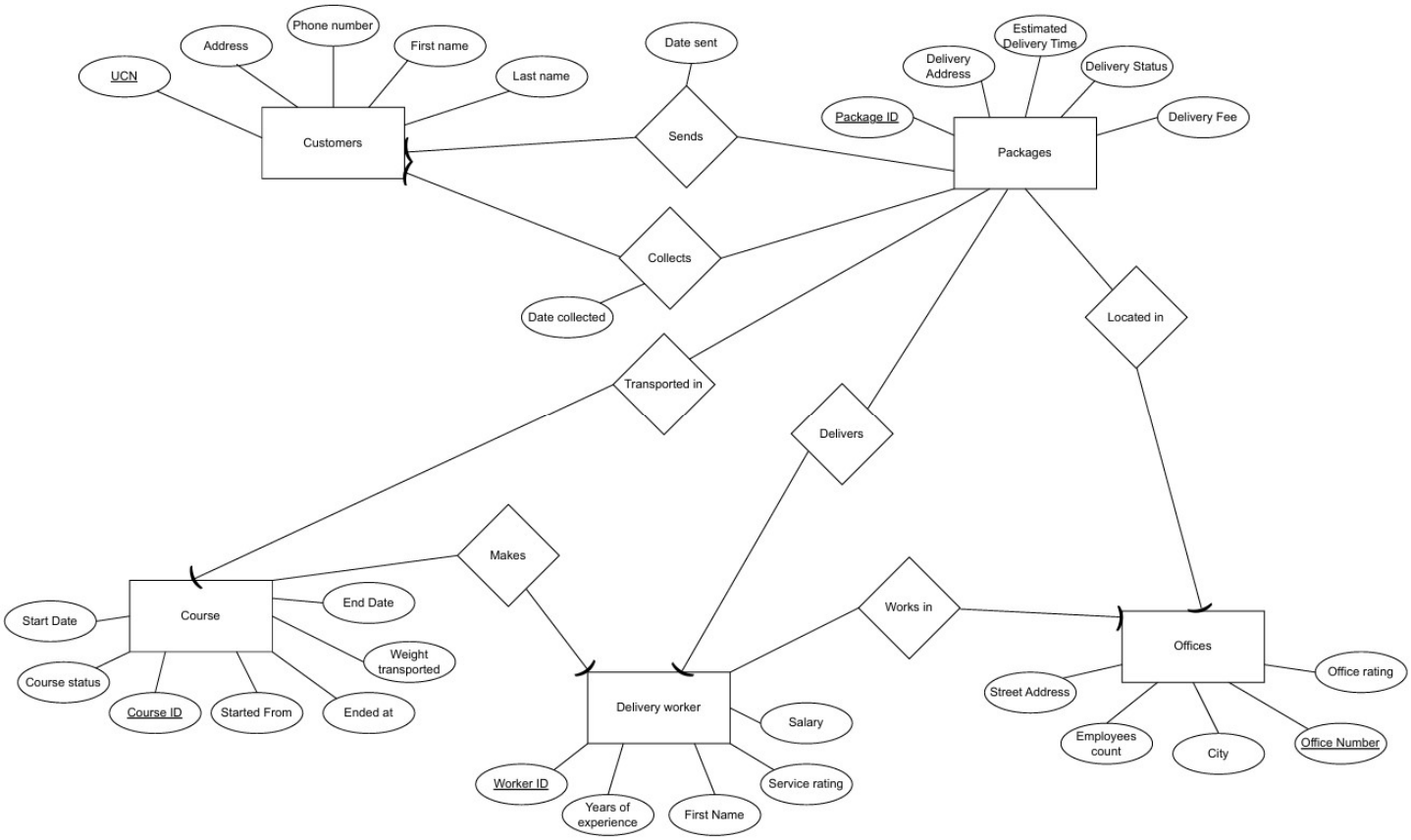
За таблица “Offices”:

- Рейтингът на офис е число между 0 и 5

За таблица “Customers”:

- ЕГН-то на клиент е низ с точно 10 символа

III. E/R Модел на данни:



IV. Релационен модел на данни:

След преобразуване на модела „Същност-връзки“ към релационен модел и оптимизиране на връзките много-един се получава следният релационен модел на базата от данни:

- Customers (UCN, Address, PhoneNumber, FirstName, LastName)

- Course (CourseStatus, CourseID, StartedFrom, EndedAt, WeightTransported, StartDate, EndDate, WorkerID)

- DeliveryWorker (WorkerID, YearsOfExperience, FirstName, ServiceRating, Salary, OfficeNumber)

- Offices (StreetAddress, EmployeesCount, City, OfficeNumber, OfficeRating)

- Packages (PackageID, DeliveryAddress, EstimatedDeliveryTime, DeliveryStatus, DeliveryFee, CourseID, WorkerID, OfficeNumber, DateCollected, RecieverUCN, DateSent, SenderUCN)

Първични ключове (Primary keys):

Customers: PK(UCN)

Course: PK(CourseID)

DeliveryWorker: PK (WorkerID)

Offices: PK (OfficeNumber)

Packages: PK (PackageID)

Външни ключове (Foreign Keys):

Packages: FK(CourseID) -> Course(CourseID)

Packages: FK(WorkerID) -> DeliveryWorker(WorkerID)

Packages: FK(OfficeNumber) -> Offices(OfficeNumber)

Packages: FK(RecieverUCN) -> Customers(UCN)

Packages: FK(SenderUCN) -> Customers(UCN)

DeliveryWorker: FK(OfficeNumber) -> Offices(OfficeNumber)

Course: FK(WorkerID) -> DeliveryWorker(WorkerID)

Check:

Packages: (DateSent < DateCollected)

Packages: (EstimatedDeliveryTime > 0)

Packages: (DeliveryFee > 0)

Course: (StartDate < EndDate)

Course: (CourseID > 0)

Course: (CourseWeight > 0)

DeliveryWorker: (YearsOfExperience >= 0)

DeliveryWorker: (ServiceRating >= 0 && ServiceRating <= 5)

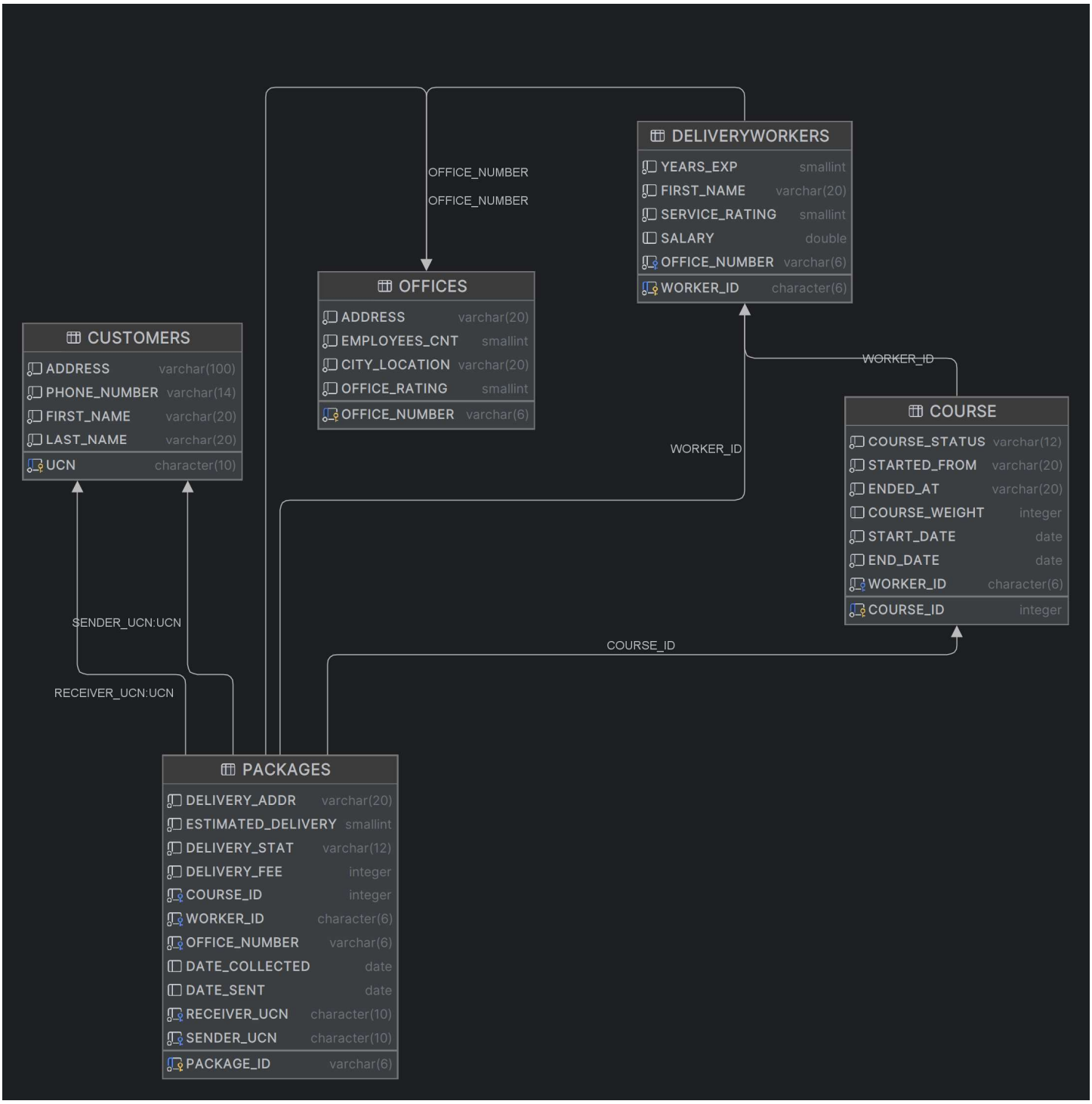
DeliveryWorker: (Salary > 0)

DeliveryWorker: (LENGTH(WorkerID) = 6)

Offices: (OfficeRating>= 0 && OfficeRating <= 5)

Customers: (LENGTH(UCN) = 10)

V. Схема на базата от данни:



VI. Функции, тригери и изгледи:

Функция 1:

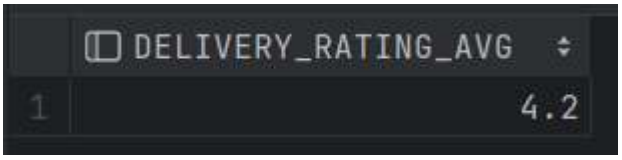
При извикване функцията връща средноаритметично на рейтинга на всички доставчици във фирмата:

```
CREATE OR REPLACE FUNCTION GetAverageServiceRating()
RETURNS DOUBLE
DETERMINISTIC
BEGIN
    DECLARE avgRating DOUBLE;

    SELECT SUM(SERVICE_RATING) / (COUNT(*) * 1.0)
    INTO avgRating
    FROM DeliveryWorkers;

    IF avgRating IS NULL THEN
        SET avgRating = 0;
    END IF;

    RETURN avgRating;
END;
```

Извикване:

```
SELECT FN24_8MI0400006.GETAVERAGESERVICERATING() as DELIVERY_RATING_AVG
from SYSIBM.SYSDUMMY1;
```

Резултат:



Функция 2:

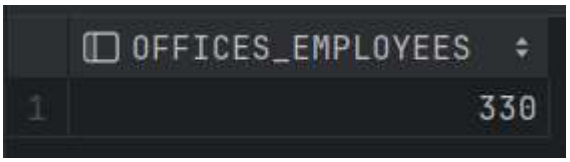
При извикване функцията връща броя на всички служители във фирмата:

```
CREATE OR REPLACE FUNCTION GetTotalEmployeesCount()
RETURNS INTEGER
BEGIN
    DECLARE totalEmployeesCount INTEGER DEFAULT 0;

    SELECT SUM(OFFICES.EMPLOYEES_CNT) INTO totalEmployeesCount
    FROM OFFICES;

    RETURN totalEmployeesCount;
END;
```

Извикване:

```
SELECT FN24_8MI0400006.GETTOTALEMPLOYEESCOUNT() as OFFICES_EMPLOYEES
from SYSIBM.SYSDUMMY1;
```

Резултат:



Функция 3:

При извикване с дадено валидно ЕГН на клиент - функцията връща сумата която този клиент трябва да заплати за всички пратки:

```
CREATE OR REPLACE FUNCTION GetTotalDeliveryFees(UCN CHAR(10))
RETURNS INTEGER
DETERMINISTIC
BEGIN
    DECLARE totalFees INTEGER;

    SELECT SUM(DELIVERY_FEE)
    INTO totalFees
    FROM Packages
    WHERE RECEIVER_UCN = UCN;

    IF totalFees IS NULL THEN
        SET totalFees = 0;
    END IF;

    RETURN totalFees;
END;
```

Извикване:

```
SELECT FN24_8MI0400006.GETTOTALDELIVERYFEES('3344556677') as CUSTOMER_FEES

from SYSIBM.SYSDUMMY1;
```

Резултат:

Функция 4:

При извикване - функцията връща имена, адрес и тел.номер на клиентите:

```
CREATE OR REPLACE FUNCTION FN24_8MI0400006.GetCustomerNamesAndAddresses()
RETURNS TABLE (
    CUSTOMER_NAME VARCHAR(50),
    CUSTOMER_LAST_NAME VARCHAR(50),
    CUSTOMER_ADDRESS VARCHAR(100),
    CUSTOMER_NUMBER CHAR(12)
)
RETURN
    SELECT FIRST_NAME AS CUSTOMER_NAME,
           LAST_NAME AS CUSTOMER_LAST_NAME,
           ADDRESS AS CUSTOMER_ADDRESS,
           PHONE_NUMBER AS CUSTOMER_NUMBER
    FROM Customers;
```

Извикване:

```
SELECT *
FROM TABLE(FN24_8MI0400006.GETCUSTOMERNAMESANDADDRESSES()) T;
```

Резултат:

	CUSTOMER_NAME	CUSTOMER_LAST_NAME	CUSTOMER_ADDRESS	CUSTOMER_NUMBER
1	John	Doe	123 Main St	123-456-7890
2	Jane	Smith	456 Elm St	098-765-4321
3	Mike	Johnson	789 Oak St	112-233-4455
4	Sara	Williams	101 Maple St	223-344-5566
5	Paul	Brown	202 Birch St	334-455-6677
6	Emily	Davis	303 Pine St	445-566-7788
7	David	Miller	404 Cedar St	556-677-8899
8	Emma	Wilson	505 Spruce St	667-788-9900
9	James	Taylor	606 Fir St	778-899-0011
10	Olivia	Anderson	707 Chestnut St	889-900-1122

Тригери:

Тригер 1: Този тригер задава стойност по подразбиране, ако не е зададена такава при добавянето на нова пратка в таблицата Packages.

```
CREATE OR REPLACE TRIGGER SetDefaultDeliveryStatus
BEFORE INSERT ON Packages
REFERENCING NEW AS NEW
FOR EACH ROW
BEGIN ATOMIC
    IF NEW.DELIVERY_STAT IS NULL OR NEW.DELIVERY_STAT = '' THEN
        SET NEW.DELIVERY_STAT = 'In Transit';
    END IF;
END;
```

Пример:

```
INSERT INTO Packages (PACKAGE_ID, DELIVERY_ADDR, ESTIMATED_DELIVERY, DELIVERY_FEE,
COURSE_ID, WORKER_ID, OFFICE_NUMBER, DATE_COLLECTED, DATE_SENT, RECEIVER_UCN, SENDER_UCN)
VALUES ('123456', '123 Main St', 3, 10, 1, 'DW001', 'OFF001','',2024-01-02, 2024-01-01,
'9876543210', '1234567890');
```

Тригер 2: Този тригер увеличава броя на служителите в даден офис при добавянето на нов служител - куриер в таблицата Offices.

```
CREATE TRIGGER IncreaseEmployeeCount
AFTER INSERT ON DeliveryWorkers
REFERENCING NEW AS NEW
FOR EACH ROW
BEGIN ATOMIC
    UPDATE Offices
    SET EMPLOYEES_CNT = EMPLOYEES_CNT + 1
    WHERE OFFICE_NUMBER = NEW.OFFICE_NUMBER;
END;
```

Пример:

```
INSERT INTO DeliveryWorkers (WORKER_ID, YEARS_EXP, FIRST_NAME, SERVICE_RATING, SALARY,
OFFICE_NUMBER) VALUES ('DW002', 2, 'John', 4, 52000, 'OFF001');
```

Тригер 3: Този тригер намалява броя на служителите в даден офис при отстраняването на служител от таблицата Offices.

```
CREATE TRIGGER DecreaseEmployeeCount
AFTER DELETE ON DeliveryWorkers
REFERENCING OLD AS OLD
FOR EACH ROW
BEGIN ATOMIC
    UPDATE Offices
    SET EMPLOYEES_CNT = EMPLOYEES_CNT - 1
    WHERE OFFICE_NUMBER = OLD.OFFICE_NUMBER;
END;
```

Пример:

```
DELETE FROM DeliveryWorkers WHERE WORKER_ID = 'DW002';
```

Изгледи:

Изглед 1:

Този изглед комбинира данни от таблиците „Packages“ и „Customers“, за да покаже подробности за пакета заедно с имената на изпращача и получателя.

```
CREATE VIEW PackageDetails AS
SELECT P.PACKAGE_ID,
       P.DELIVERY_ADDR,
       P.ESTIMATED_DELIVERY,
       P.DELIVERY_STAT,
       P.DELIVERY_FEE,
       P.COURSE_ID,
       P.WORKER_ID,
       P.OFFICE_NUMBER,
       P.DATE_COLLECTED,
       P.DATE_SENT,
       P.RECEIVER_UCN,
       P.SENDER_UCN,
       C_SENDER.FIRST_NAME AS SENDER_FIRST_NAME,
       C_SENDER.LAST_NAME AS SENDER_LAST_NAME,
       C_RECEIVER.FIRST_NAME AS RECEIVER_FIRST_NAME,
       C_RECEIVER.LAST_NAME AS RECEIVER_LAST_NAME
FROM Packages P
JOIN Customers C_SENDER ON P.SENDER_UCN = C_SENDER.UCN
JOIN Customers C_RECEIVER ON P.RECEIVER_UCN = C_RECEIVER.UCN;
```

Изглед 2:

Този изглед комбинира данни от таблиците „DeliveryWorkers“ и „Offices“, за да покаже подробности за работниците заедно с информация за техните офиси.

```
CREATE VIEW WorkerOfficeDetails AS
SELECT DW.WORKER_ID,
       DW.YEARS_EXP,
       DW.FIRST_NAME,
       DW.SERVICE_RATING,
       DW.SALARY,
       DW.OFFICE_NUMBER,
       O.ADDRESS AS OFFICE_ADDRESS,
       O.EMPLOYEES_CNT AS OFFICE_EMPLOYEES_COUNT,
       O.CITY_LOCATION AS OFFICE_CITY,
       O.OFFICE_RATING
FROM DeliveryWorkers DW
JOIN Offices O ON DW.OFFICE_NUMBER = O.OFFICE_NUMBER;
```

VII. Приложение за достъп до базата (Java):

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;


public class DB2Test {

    private Connection connection;
    private Statement statement;
    private ResultSet resultSet;


    public void openConnection(){

// Step 1: Load IBM DB2 JDBC driver

        try {

            DriverManager.registerDriver(new com.ibm.db2.jcc.DB2Driver());

        }

        catch(Exception cnfex) {

            System.out.println("Problem in loading or registering IBM DB2 JDBC driver");

            cnfex.printStackTrace();
        }

// Step 2: Opening database connection

        try {

            connection =
```

```
DriverManager.getConnection("jdbc:db2://62.44.108.24:50000/SAMPLE", "db2admin",
"db2admin");

            statement = connection.createStatement();

        }

        catch(SQLException s){

            s.printStackTrace();

        }
    }

    public void closeConnection(){

        try {

            if(null != connection) {

                // cleanup resources, once after processing

                resultSet.close();
                statement.close();

                // and then finally close connection

                connection.close();

            }
        }

        catch (SQLException s) {

            s.printStackTrace();

        }
    }

    public void select(String stmnt, int column) {
        // Example querry : select title, year from db2movie.movie - column = 2
        try{
            resultSet = statement.executeQuery(stmnt);
            String result = "";

            while(resultSet.next()) {
                for (int i = 1; i <= column; i++) {

                    result += resultSet.getString(i);

                    if (i == column) result += " \n";
                    else             result += ", ";
                }
            }

            System.out.println("Executing query: " + stmnt + "\n");
            System.out.println("Result output \n");
            System.out.println("---------------------------------- \n");
            System.out.println(result);

        }
        catch (SQLException s)
        {
```

```
            s.printStackTrace();
        }

    }

    public void insert(String stmnt) {

        try{
            statement.executeUpdate(stmnt);
            System.out.println("Successfully inserted!");
        }

        catch (SQLException s){
            System.out.println("NOT inserted!");
            s.printStackTrace();
        }

    }

    public void delete(String stmnt) {

        try{

            statement.executeUpdate(stmnt);

        }

        catch (SQLException s){

            s.printStackTrace();

        }

        System.out.println("Successfully deleted!");

    }

    public static int getColumns(String statement) {

        int cols = 1;

        String word = "";

        for (int i = 0; i < statement.length(); i++) {
            word += statement.charAt(i);

            if(statement.charAt(i) == ' ') {
                word = word.trim();
                if (word.equals("FROM")) {
                    return cols;
                } else {
                    word = "";
                }

            }

            if(statement.charAt(i) == ',') {
                cols++;
            }
        }

        return cols;
    }
```

```
    public static void selectI(String statement, DB2Test db2Obj) {

        int cols = getColumns(statement);

        db2Obj.select(statement, cols);
    }

    public static void main(String[] args) {

        DB2Test db2Obj = new DB2Test();
        String stmnt = "";

        db2Obj.openConnection();

        //query 1:
        stmnt = "SELECT ADDRESS, EMPLOYEES_CNT FROM " +
                "FN24_8MI0400006.OFFICES " +
                "WHERE ADDRESS like \'789%\'";
        selectI(stmnt, db2Obj);

        //query 2:
        stmnt = "SELECT WORKER_ID, FIRST_NAME, SALARY, OFFICE_NUMBER " +
                "FROM FN24_8MI0400006.DELIVERYWORKERS " +
                "WHERE WORKER_ID IN (\'DW001\', \'DW002\', \'DW003\')";
        selectI(stmnt, db2Obj);

        //query 3:
        stmnt = "SELECT SENDER_UCN, SUM(DELIVERY_FEE) " +
                "FROM FN24_8MI0400006.PACKAGES " +
                "GROUP BY SENDER_UCN " +
                "ORDER BY SUM(DELIVERY_FEE) DESC";
        selectI(stmnt, db2Obj);

        //query 4:
        stmnt = "SELECT ADDRESS, CITY_LOCATION " +
                "FROM FN24_8MI0400006.OFFICES " +
                "WHERE OFFICE_RATING >= 4 " +
                "ORDER BY ADDRESS DESC";
        selectI(stmnt, db2Obj);

        db2Obj.closeConnection();

    }

}
```