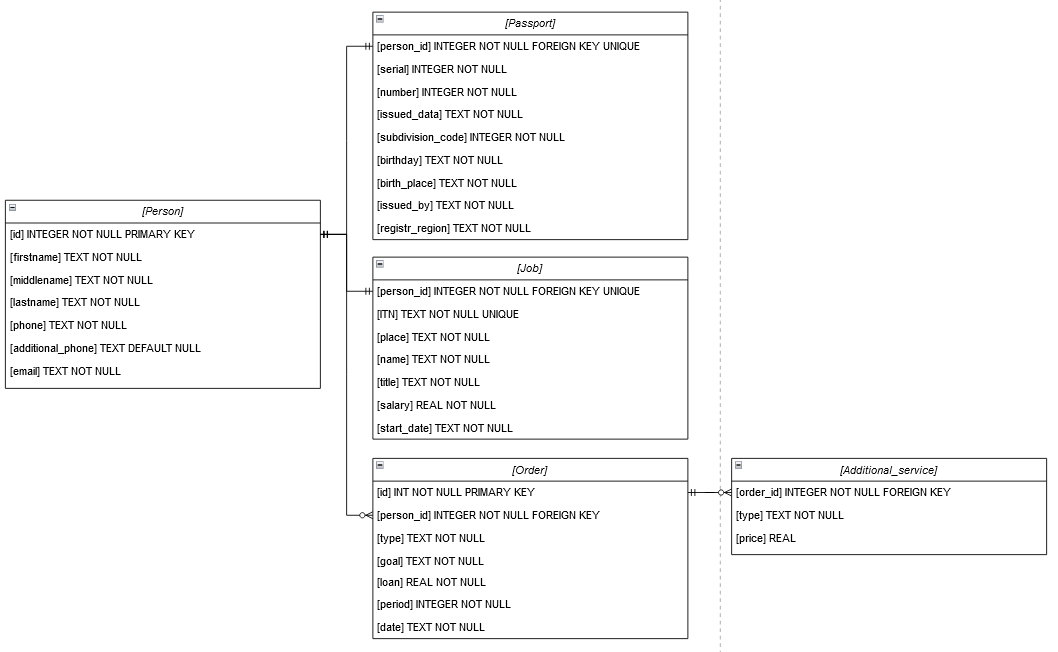
Задание 1

База данных: SQLite

ER диаграмма:



Сущность [Person] представляет из себя группу поле вводов из формы “Персональные данные”.

[id] – целочисленный тип. Цель: уникальная индефикация пользователей.

Для представления имени [firstname] , фамилии [lastname], отчества [middlename], телефона [phone], доп. телефона [additional\_phone], email [email] используется строковый формат.

Сущность [Passport] отражает группу вводов “Паспорт”.

[person\_id] – целочисленный тип. Уникальный индефикатор указывает, что человек может иметь только один паспорт в РФ в определённый период времени.

Следующие атрибуты в жизни представляются в виде цифр, поэтому используем целочисленный тип.

[serial] – серия паспорта.

[number] – номер паспорта.

[subdivision\_code] – код подразделения.

Атрибуты, представленные ниже, являются датами, но в SQLite отсутсвует тип данных, связанный со времением, поэтому используем тип TEXT и накладываем на него ограничения.

[issued\_data] – дата выдачи.

[birthday] – дата рождения.

Следующие атрибуты это поля для ввода текстовой информации.

[issued\_by] – кем выдан паспорт.

[birth\_place] – место рождения.

[registr\_region] – регион регистрации.

Сущность [Job] реализует поля вводов под названием “Сведения о месте работы”.

[person\_id] – челочисленный тип. Согласуется с значением [id] из сущности [Person].

[ITN] – ИНН. Представляются в реальном мире в целочисленом виде.

[salary] – размер зарплаты. Тип REAL, поскольку зарплата может не только быть в целочисленном виде.

Следующим атрибутам характерен тип TEXT.

[place] – регион работы.

[name] – название организации.

[title] – название должности.

[start\_date] – дата начала работы. На дату нужно накладывать ограничения.

Сущность [Order] отражает поля вводов “Параметры заявки”.

[id] – целочисленный тип. Один человек может иметь несколько заявок, поэтому отношение один ко многим.

[person\_id] – целочиленный тип. Связь с сущностью [Person].

В ниже представленных атрибутах используется тип TEXT.

[type] – вид продукта.

[goal] – цель кредита.

[date] – дата оформления зявки. Нужно для быстрой навигации по истории заявок.

Атрибуты в которых используется целочисленный тип.

[period] – срок кредита.

[loan] – сумма. Тип REAL.

Сущность [Additional\_service] отражает поля вводов “Параметры заявки”.

[order\_id] – целочисленный тип. Заявка может иметь несколько дополнительных услуг, поэтому отношение один ко многим.

[type] – вид услуги. Используемый тип – TEXT.

[price] – стоимость дополнительной услуги. Тип REAL.

SQL файл:

DROP TABLE IF EXISTS [Order];

DROP TABLE IF EXISTS [Additional\_service];

DROP TABLE IF EXISTS [Person];

DROP TABLE IF EXISTS [Passport];

DROP TABLE IF EXISTS [Job];

CREATE TABLE [Person] (

    [id] INTEGER NOT NULL PRIMARY KEY,

    [firstname] TEXT NOT NULL,

    [middlename] TEXT NOT NULL,

    [lastname] TEXT NOT NULL,

    [phone] TEXT NOT NULL,

    [additional\_phone] TEXT DEFAULT NULL,

    [email] TEXT NOT NULL

);

CREATE TABLE [Passport] (

    [person\_id] INTEGER NOT NULL UNIQUE,

    [serial] TEXT NOT NULL,

    [num] TEXT NOT NULL,

    [issued\_date] TEXT NOT NULL,

    [subdivision\_code] INT NOT NULL,

    [birthday] TEXT NOT NULL,

    [birth\_place] TEXT NOT NULL,

    [issued\_by] TEXT NOT NULL,

    [registr\_region] TEXT NOT NULL,

    FOREIGN KEY ([person\_id]) REFERENCES [Person]([id])

);

CREATE TABLE [Job](

    [person\_id] INTEGER NOT NULL UNIQUE,

    [ITN] TEXT NOT NULL UNIQUE,

    [place] TEXT NOT NULL,

    [name] TEXT NOT NULL,

    [title] TEXT NOT NULL,

    [salary] INTEGER NOT NULL,

    [state\_date] TEXT NOT NULL,

    FOREIGN KEY ([person\_id]) REFERENCES [Person]([id])

);

CREATE TABLE [Order] (

        [id] INTEGER NOT NULL PRIMARY KEY,

        [person\_id] INTEGER NOT NULL,

        [type] TEXT NOT NULL,

        [goal] TEXT NOT NULL,

        [loan] REAL,

        [period] INTEGER,

        [date] TEXT,

        FOREIGN KEY ([person\_id]) REFERENCES [Person]([id])

    );

CREATE TABLE [Additional\_service](

    [order\_id] INT NOT NULL, /\*\*/

    [type] TEXT NOT NULL,

    [price] REAL,

    FOREIGN KEY ([order\_id]) REFERENCES [Order]([id])

);

Задание 2

SELECT type, COUNT(\*) as order\_count

FROM [Order]

GROUP BY type

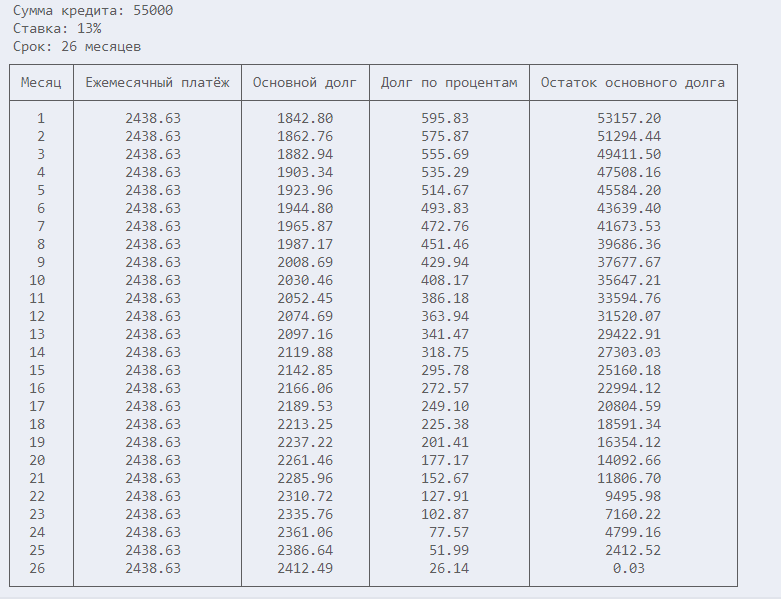
ORDER BY order\_count DESC LIMIT 1;

Задание 3

Язык программирования: JavaScript .

Скрипт разрабатывался на базе NodeJS.

Результат:



Файл:

//*\*\* Зависимости для создания таблицы \*/*

const { Console } = require('console');

const { Transform } = require('stream');

/\*\**Вывод данных в таблице\**\*/

function table(input) {

  const ts = new Transform({ transform(chunk, enc, cb) { cb(null, chunk) } })

  const logger = new Console({ stdout: ts });

  logger**.**table(input);

  const table = (ts**.**read() || '')**.**toString();

  let result = '';

  for (let row of table**.**split(/[\r\n]+/)) {

    let r = row**.**replace(/[^┬]\*┬/, '┌');

    r = r**.**replace(/^├─\*┼/, '├');

    r = r**.**replace(/│[^│]\*/, '');

    r = r**.**replace(/^└─\*┴/, '└');

    r = r**.**replace(/'/g, ' ');

    result += `${r}\n`;

  }

  console**.**log(result);

}

/\**Параметры: сумма кредита, ставка, срок*\*/

function\* getAnnuityPayment(creditAmount, rate, period) { // *Используем генератор*

    const coof = rate/(100\*12), // *Расчёт процентной ставки*

    leftSide = coof / (1 - Math**.**pow(1+coof, -period)), // *Выделение левой части уравнения ежемесячного платежа*

    monthlyPayment = (creditAmount \* leftSide)**.**toFixed(2); // *Расчёт ежемесячного платежа*

    for(let i = 0; i < period; i++) { // *Для каждого месяца расчитываем требуемые данные*

        //*\*\* Используем toFixed для позиционирования плавающей точки\*/*

        const debtIntrest = (creditAmount \* coof)**.**toFixed(2),

        principalDebt = (monthlyPayment - debtIntrest)**.**toFixed(2);

        creditAmount = (creditAmount - principalDebt)**.**toFixed(2);

        yield {'Месяц': i + 1,

               'Ежемесячный платёж': monthlyPayment,

               'Основной долг': principalDebt,

               'Долг по процентам':  debtIntrest,

                'Остаток основного долга': creditAmount > 0? creditAmount: 0};

    }

}

/\**Параметры: сумма кредита, ставка, срок*\*/

function showAnnuityPaymentTable(creditAmount, rate, period) {

    const annityPayments = []; // *Обёртка для информации, поступающей из генератора*

    //*\*\*Генерируем значения для таблицы и помещаем их обёртку\*/*

    for(let annityPayment of getAnnuityPayment(creditAmount, rate, period)) {

        annityPayments**.**push(annityPayment);

    }

    console**.**log(` Cумма кредита: ${creditAmount} \n Ставка: ${rate}% \n Срок: ${period} месяцев`);

    table(annityPayments); // *Помещаем данные в таблицу*

}

// *Распечатываем таблицу*

showAnnuityPaymentTable(55000, 13, 26);