**Министерство транспорта Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования «Российский университет транспорта»**

**(РУТ(МИИТ))**

**Институт управления и цифровых технологий**

**Кафедра «Цифровые технологии управления транспортными процессами»**

**(ЦТУТП)**

**Отчёт по летней практике**

**Предметная область «Корпорация»**

Выполнил: Яковлев Я. С.

Группа УВП-112

                       Проверили: ст. преп. Заманов Е. А.,

ст. преп. Разживайкин И. С.,

                                                                                           ст. преп., Заманова Е.А.

Оглавление

[Введение 4](#_Toc172105595)

[Основная часть 4](#_Toc172105596)

[Теоретическая часть 4](#_Toc172105597)

[Реализация 5](#_Toc172105598)

[Метод добавляющий сотрудника 6](#_Toc172105599)

[Удаление сотрудника 7](#_Toc172105600)

[Получить всех сотрудников чья зарплата превышает 100\_000 7](#_Toc172105601)

[Получить всех сотрудников из отдела производства и разработки с должностью Senior Backend разработчик 7](#_Toc172105602)

[Изменить статус на "На этапе тестирования" проекту n 8](#_Toc172105603)

[Изменить id проекта N над которым работают сотрудники с должностью M на проект K. 8](#_Toc172105604)

[Увеличить зарплату на 30% сотрудникам, которые работают в компании более 3 лет. 8](#_Toc172105605)

[Получить id отдела, в котором работает наибольшее количество сотрудников. 9](#_Toc172105606)

[Удалить всех сотрудников с рейтингом ниже N 9](#_Toc172105607)

[Получить среднюю зарплату все сотрудников чей возраст не менее 30 лет. 9](#_Toc172105608)

[Выводы 9](#_Toc172105609)

[Список литературы 9](#_Toc172105610)

[Приложения 10](#_Toc172105611)

# 

# Введение

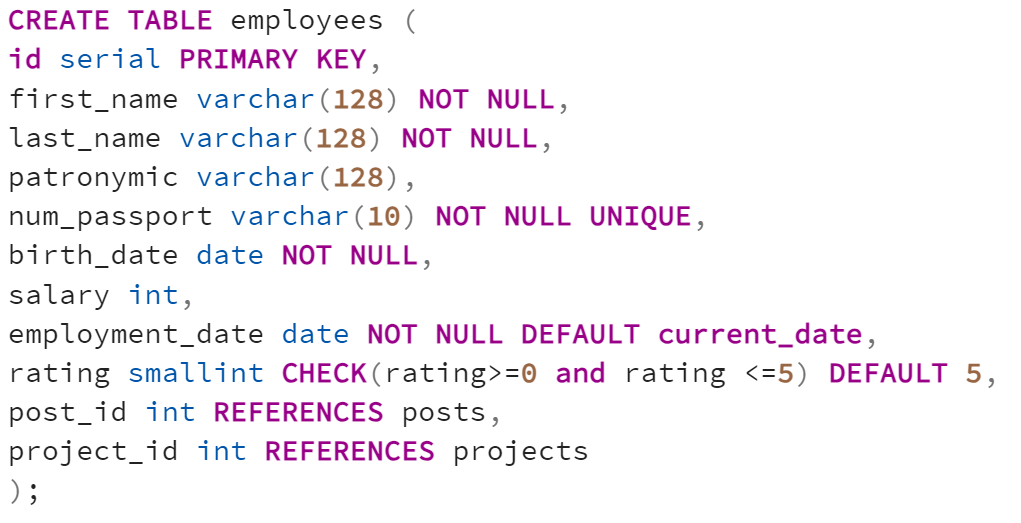
Предметная область данной работы – система из сотрудников, их проектов и должностей, а также отделов, которые образуют корпорацию. Цель работы – создать приложение, которое может взаимодействуя с реляционной базой данных сможет удалять, изменять, добавлять новые, а также на основании имеющейся информации делать статистические расчёты.

# Основная часть

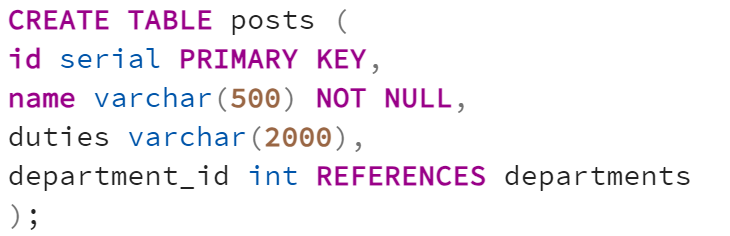
## Теоретическая часть

В базе данных будет 4 таблицы: сотрудники, должности, отделы, проекты и статусы. Таблицы связаны между собой следующим образом: каждый сотрудник связан с определенной должностью и проектом, должности и проекты в свою очередь связаны с определенным отделом. По мимо это присутствует вспомогательная таблица со статусами. Она нужна для обозначения текущего статуса проекта.

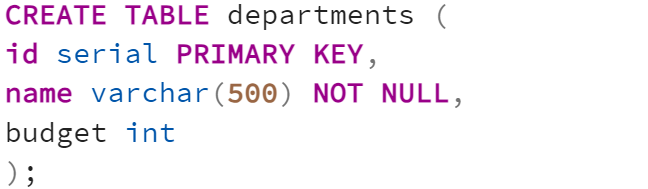
В таблице с сотрудниками представлены следующие поля: целочисленный идентификатор, имя длиною не более 128 символов, фамилия длиною не более 128 символов, отчество длиною не более 128 символов, серия и номер паспорта, дата рождения, целочисленная зарплата, дата приема на работу, рейтинг в диапазоне от 0 до 5, целочисленный внешний ключ, связывающий сотрудника с его должностью, целочисленный внешний ключ, связывающий сотрудника с данными о проекте, в котором он работает.



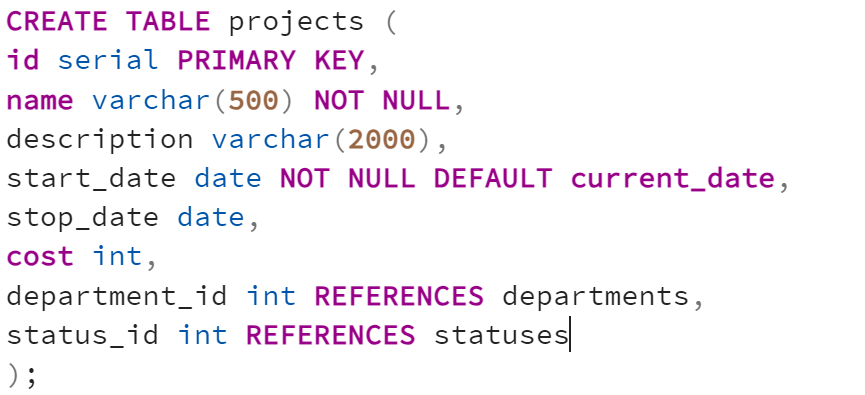
В таблице с должностями содержатся следующие поля: числовой идентификатор, название должности длиною не более 500 символов, обязанности присущие должности – текст не более 2000 символов и целочисленный внешний ключ, связывающий должность с отделом.



В таблице с отделами хранится целочисленный идентификатор, название отдела длиною не более 500 символов и целочисленный бюджет отдела.



В таблице с проектами хранится целочисленный идентификатор, название проекта длиною не более 500 символов, описание проекта не более 2000 символов, дата начала проекта, дата окончания проекта, стоимость проекта и два внешних ключа, один из них связывает проект с определенным отделом, другой связывает проект со вспомогательной таблицей со статусами проекта.



В вспомогательной таблице статусы хранится целочисленный идентификатор и само название статуса длиною не более 200 символов.

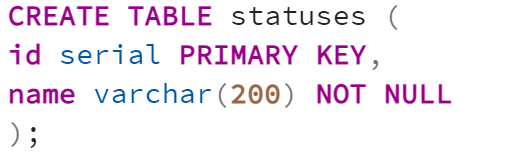
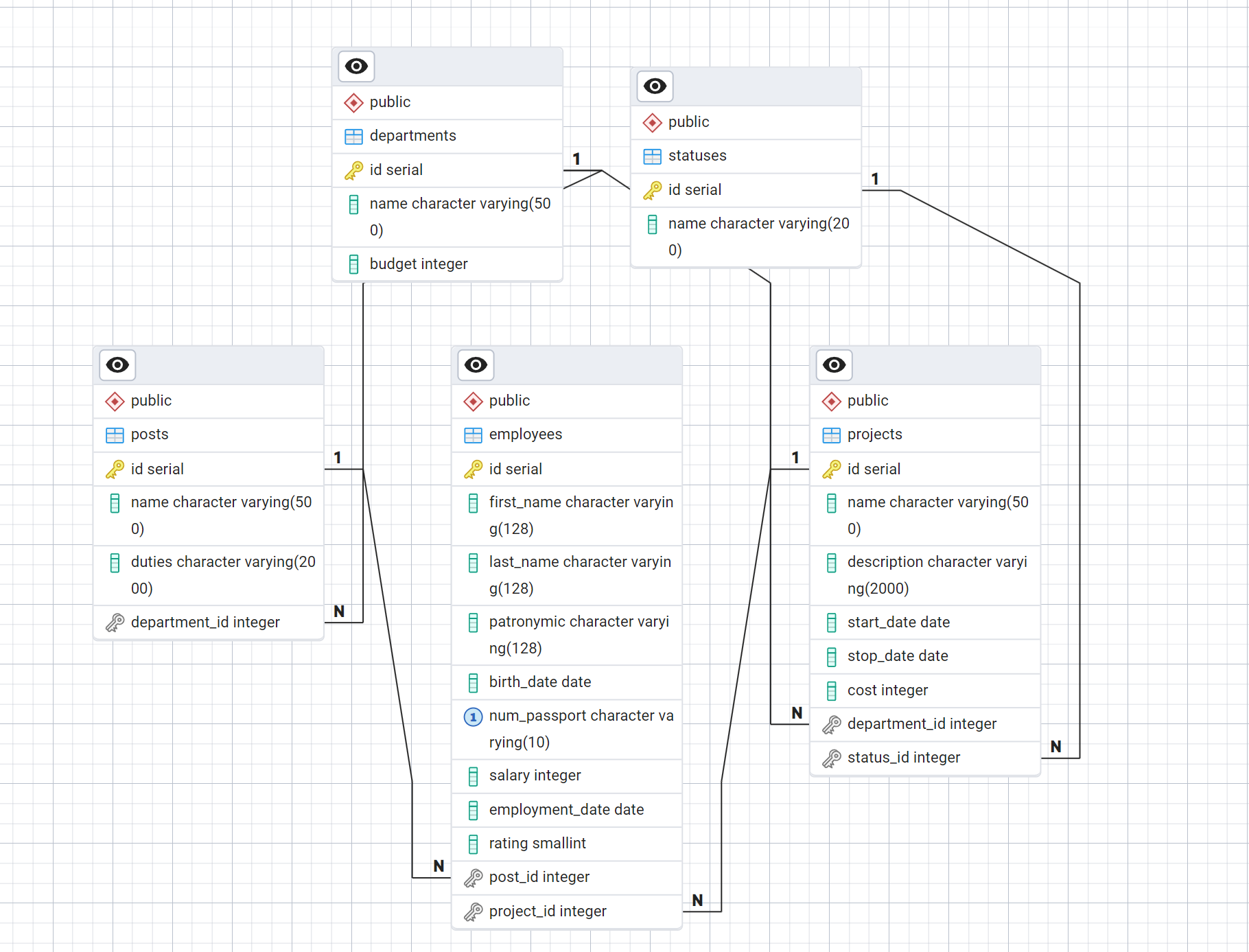


Диаграмма предметной области выглядит следующим образом:



## Реализация

**Целью является реализация запросов:**

1. Добавить сотрудника
2. Удалить сотрудника
3. Получить всех сотрудников чья зарплата превышает 100000
4. Получить всех сотрудников из отдела производства и разработки с должностью Senior Backend разработчик.
5. Изменить статус на "На этапе тестирования" проекту n.
6. Изменить id проекта N над которым работают сотрудники с должностью M на проект K.
7. Увеличить зарплату на 30% сотрудникам, которые работают в компании более 3 лет.
8. Получить id отдела, в котором работает наибольшее количество сотрудников.
9. Удалить всех сотрудников с рейтингом ниже N
10. Получить среднюю зарплату все сотрудников чей возраст не менее 30 лет.

### Метод добавляющий сотрудника

private static void addEmployee(Connection connection, String first\_name, String last\_name,  
 String patronymic, Date birth\_date, String num\_passport, int salary,  
 Date employment\_date, int rating, int post\_id, int project\_id) throws SQLException {  
 PreparedStatement statement = connection.prepareStatement(  
 "INSERT INTO employees(first\_name, last\_name, birth\_date, patronymic, " +  
 "num\_passport, salary, employment\_date, rating, post\_id, project\_id) " +  
 "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?) returning id;", Statement.*RETURN\_GENERATED\_KEYS*);  
 statement.setString(1, first\_name);  
 statement.setString(2, last\_name);  
 statement.setString(3, patronymic);  
 statement.setDate(4, birth\_date);  
 statement.setString(5, num\_passport);  
 statement.setInt(6, salary);  
 statement.setDate(7, employment\_date);  
 statement.setInt(8, rating);  
 statement.setInt(9, post\_id);  
 statement.setInt(10, project\_id);  
}

И его сокращенный вариант, где оставшиеся аргументы подставит сама база данных:

private static void addEmployee(Connection connection, String first\_name, String last\_name,  
 String patronymic, Date birth\_date, String num\_passport, int salary,  
 int post\_id, int project\_id) throws SQLException {  
  
 PreparedStatement statement = connection.prepareStatement(  
 "INSERT INTO employees(first\_name, last\_name, patronymic, birth\_date, " +  
 "num\_passport, salary, post\_id, project\_id) " +  
 "VALUES (?, ?, ?, ?, ?, ?, ?, ?) returning id;", Statement.*RETURN\_GENERATED\_KEYS*);  
 statement.setString(1, first\_name);  
 statement.setString(2, last\_name);  
 statement.setString(3, patronymic);  
 statement.setDate(4, birth\_date);  
 statement.setString(5, num\_passport);  
 statement.setInt(6, salary);  
 statement.setInt(7, post\_id);  
 statement.setInt(8, project\_id);  
  
 int count = statement.executeUpdate();  
 System.*out*.println("INSERT " + count + " strings");  
}

### Удаление сотрудника

private static void removeEmployee(Connection connection, int id) throws SQLException {  
 PreparedStatement statement = connection.prepareStatement("DELETE FROM employees WHERE id = ?");  
 statement.setInt(1, id);  
 int count = statement.executeUpdate();  
 System.*out*.println("DELETE " + count + " strings");  
}

### Получить всех сотрудников чья зарплата превышает 100\_000

private static void getEmployeesWhereSalaryMore100\_000(Connection connection) throws SQLException {  
 Statement statement = connection.createStatement();  
 ResultSet rs = statement.executeQuery("SELECT \* FROM employees WHERE salary > 100\_000;");  
  
 while (rs.next()) { // пока есть данные  
 int param0 = rs.getInt(1);  
 String param1 = rs.getString(2);  
 String param2 = rs.getString(3);  
 String param3 = rs.getString(4);  
 Date param4 = rs.getDate(5);  
 String param5 = rs.getString(6);  
 int param6 = rs.getInt(7);  
 Date param7 = rs.getDate(8);  
 int param8 = rs.getInt(9);  
 int param9 = rs.getInt(10);  
 int param10 = rs.getInt(11);  
 System.*out*.println(param0 + " | " + param1 + " | " + param2 + " | " +  
 param3 + " | " + param4 + " | " + param5 + " | " +  
 param6 + " | " + param7 + " | " + param8 + " | " +  
 param9 + " | " + param10);  
 }  
}

### Получить всех сотрудников из отдела производства и разработки с должностью Senior Backend разработчик

private static void getEmployeesWhereDepartmentIsN\_PostIsM(Connection connection) throws SQLException {  
 /\* Получим всех сотрудников с должностью Senior backend разработчик(id = 1) которые работают  
 в отделе производства и разработки(id = 1) \*/  
 Statement statement = connection.createStatement();  
 ResultSet rs = statement.executeQuery("SELECT employees.id, first\_name, last\_name, patronymic, " +  
 "(SELECT name FROM posts WHERE posts.id = post\_id), " +  
 "departments.name AS department FROM employees, departments WHERE \n" +  
 "post\_id = 1 and departments.id = " +  
 "(SELECT department\_id FROM posts WHERE posts.id = post\_id LIMIT 1);");  
  
 int param0 = -1;  
 String param1, param2, param3 = null, param4 = null, param5 = null;  
  
 while (rs.next()) { // пока есть данные  
 param0 = rs.getInt(1);  
 param1 = rs.getString(2);  
 param2 = rs.getString(3);  
 param3 = rs.getString(4);  
 param4 = rs.getString(5);  
 param5 = rs.getString(6);  
 System.*out*.println(param0 + " | " + param1 + " | " + param2 + " | " +  
 param3 + " | " + param4 + " | " + param5);  
 }  
}

### Изменить статус на "На этапе тестирования" проекту n

private static void correctStatusProjectNToStatusM(Connection connection, int id, int status\_id) throws SQLException {  
 // Изменим статус проекту под id = 1 на статус "На этапе тестирования"(id=2)  
 PreparedStatement statement =connection.prepareStatement("UPDATE projects SET status\_id = ? " +  
 "WHERE id = ?;");  
 statement.setInt(1, status\_id);  
 statement.setInt(2, id);  
  
 int count = statement.executeUpdate();  
 System.*out*.println("UPDATE " + count + " strings");  
}

### Изменить id проекта N над которым работают сотрудники с должностью M на проект K.

private static void correctEmployees\_WithProjectN\_WherePostIsM\_ToProjectK(Connection connection, int project\_id, int post\_id, int new\_project\_id) throws SQLException {  
 PreparedStatement statement =connection.prepareStatement("UPDATE employees SET project\_id = ? " +  
 "WHERE post\_id = ? AND project\_id = ?;");  
 statement.setInt(1, new\_project\_id);  
 statement.setInt(2, post\_id);  
 statement.setInt(3, project\_id);  
  
 int count = statement.executeUpdate();  
 System.*out*.println("UPDATE " + count + " strings");  
}

### Увеличить зарплату на 30% сотрудникам, которые работают в компании более 3 лет.

private static void correctSalaryWhereEmployment\_dateMore3Years(Connection connection) throws SQLException {  
 PreparedStatement statement = connection.prepareStatement("UPDATE employees SET salary = salary \* 1.3 " +  
 "WHERE current\_date - employment\_date >= 365 \* 3;");  
  
 int count = statement.executeUpdate();  
 System.*out*.println("UPDATE " + count + " strings");  
}

### Получить id отдела, в котором работает наибольшее количество сотрудников.

private static void getDepartmentWhereMaxEmployees(Connection connection) throws SQLException {  
 Statement statement = connection.createStatement();  
 ResultSet rs = statement.executeQuery("SELECT id, name FROM " +  
 "(SELECT d.id, d.name, COUNT(e.id) " +  
 "FROM employees e JOIN posts p ON e.post\_id = p.id " +  
 "JOIN departments d ON p.department\_id = d.id GROUP BY d.id) " +  
 "ORDER BY count DESC LIMIT 1;");  
  
 while (rs.next()) { // пока есть данные  
 int param0 = rs.getInt(1);  
 String param1 = rs.getString(2);  
 System.*out*.println(param0 + " | " + param1);  
 }  
  
}

### Удалить всех сотрудников с рейтингом ниже N

private static void removeEmployeesWhereRatingUnderN(Connection connection, int rating) throws SQLException {  
 PreparedStatement statement = connection.prepareStatement("DELETE from employees WHERE rating < ?");  
 statement.setInt(1, rating);  
 int count = statement.executeUpdate();  
 System.*out*.println("DELETE " + count + " strings");  
}

### Получить среднюю зарплату все сотрудников чей возраст не менее 30 лет.

private static void getAvgSalaryWhereAgeMore30(Connection connection) throws SQLException {  
 Statement statement = connection.createStatement();  
 ResultSet rs = statement.executeQuery("SELECT AVG(salary) FROM employees " +  
 "WHERE current\_date - birth\_date > 30 \* 365;");  
 if (rs.next()) System.*out*.println(rs.getInt(1));  
}

## Выводы

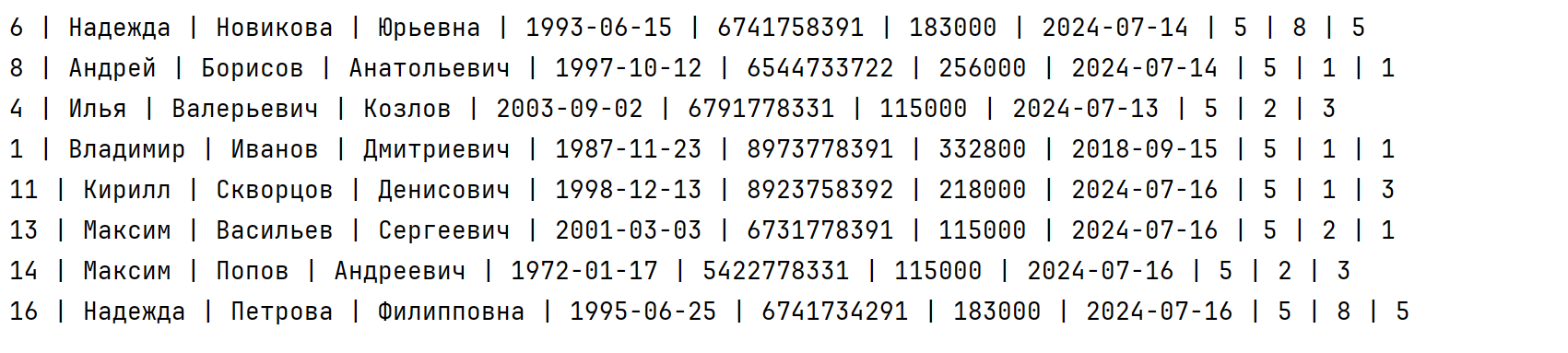
Создана программа позволяющая с помощью подключения к базе данных удалять, изменять и добавлять новые данные и выводить статистическую информацию. В это версии программы представлено 10 запросов к базе данных, но при необходимости в программу можно добавить любой новый метод или изменить уже имеющийся.

## Список литературы

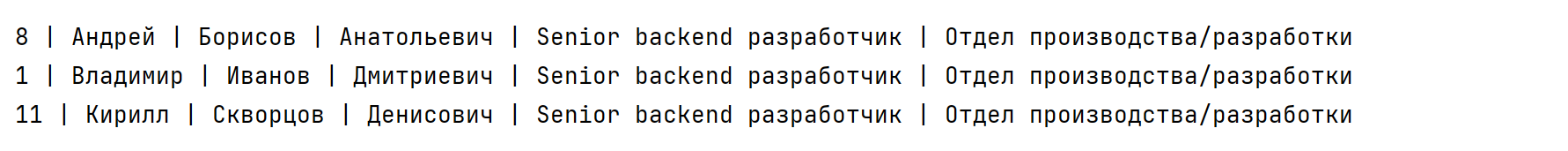
1. Учебные материалы, предоставленные преподавателями

## Приложения

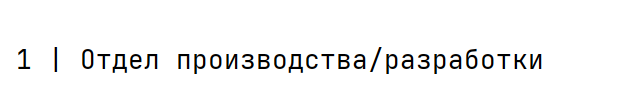
Получить всех сотрудников чья зарплата превышает 100000:



Получить всех сотрудников из отдела производства и разработки с должностью Senior Backend разработчик:



Получить отдел с наибольшим количеством сотрудников:



Получить среднюю зарплату все сотрудников чей возраст не менее 30 лет.

