Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Военный факультет

Кафедра электронных вычислительных машин

Дисциплина: Базы данных

ОТЧЕТ
к лабораторной работе №6
на тему
СОЗДАНИЕ ПРИЛОЖЕНИЯ ДЛЯ БАЗЫ ДАННЫХ.
РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.

Курсант М.И. Семёнов

Преподаватель Д.В. Куприянова

МИНСК 2024

1. ЦЕЛЬ РАБОТЫ

В данной лабораторной работе целью является изучить порядок создания приложения для работы с данным, хранящиеся в базе данных. Также следует изучить предоставляемые библиотеки и интерфейсы для непосредственного подключения к СУБД PostgreSQL.

Изучить порядок написания руководства пользователя с описанием всех пунктов работы с приложением, от развертывания до функционала.

Во время написания приложения отобразить моменты, изменение которых позволит запускать приложение на других устройствах.

2. ИСХОДНЫЕ ДАННЫЕ К РАБОТЕ

Программное обеспечение для выполнения данной лабораторной работы используется Intellij idea и СУБД PostgreSQL. Задачей данной лабораторной работы является написание приложения и документации пользователя для непосредственной работы с базой данных.

В ходе данной лабораторной работы следует выполнить:

- написать приложение (по языку программирования и интерфейсу ограничений нет);
- написать руководство пользователя для непосредственного взаимодействия и работы с данными базы данных. Все исходные данные для написания и SQL-запросы брать с предыдущих лабораторных работ.

Оформить отчет с требованиями дипломного проектирования.

3. ВЫПОЛНЕНИЕ РАБОТЫ

Для реализации и написания данной лабораторной работы были выбраны следующие характеристики технологий:

1. средство разработки: Intellij IDEA;
2. язык программирования: Java;
3. стандарт: Java 8 и выше;
4. СУБД: PostgreSQL;
5. интерфейс API: jdbc (официальный интерфейс, предоставленный от PostgreSQL);

В подразделах ниже описано полное руководство пользователя:

- технические требования (минимальные системные требования для запуска данной программы);
- этапы разворачивания программы (описание того, как запустить, как развернуть);
- описание функционала программы (описание возможностей работы, таких как просмотр, удаление, редактирование данных, так же использование запросов с лабораторных работ №4 и №5).

3.1 Системные требования к проекту и устройству

Системные требования данного проекта к устройству на прямую зависят от системных требований ПО, на котором и при котором ведется разработка. В данном случае напрямую зависит от поддержки IntelliJ IDEA и PostgreSQL.

Минимальные требования к устройству:

- операционная система: Windows 8, 10, 11 (архитектура процессора только x64);
- ОЗУ: 2 ГБ;
- дисковое пространство: 3.5 ГБ и еще 1 ГБ для кэшей;
- разрешение монитора: 1024x768;
- версия Java: 8.

Рекомендуемые требования к устройству:

- ОЗУ: 8 ГБ;
- дисковое пространство: SSD-накопитель с не менее 5 ГБ свободного места;
- разрешение монитора: 1920x1080;
- версия Java: 8 и более.

3.2 Развертывание проекта на устройстве

В данном подразделе описывается полная пошаговая инструкция о развертывании проекта на другом устройстве.

Так как данный проект готовый, то в данном руководстве опишем его запуск с внедрением зависимостей. Для начала открываем Intellij IDEA. Для этого переходим в корень проекта файловой системы, на рисунке 3.1 изображен корень проекта.

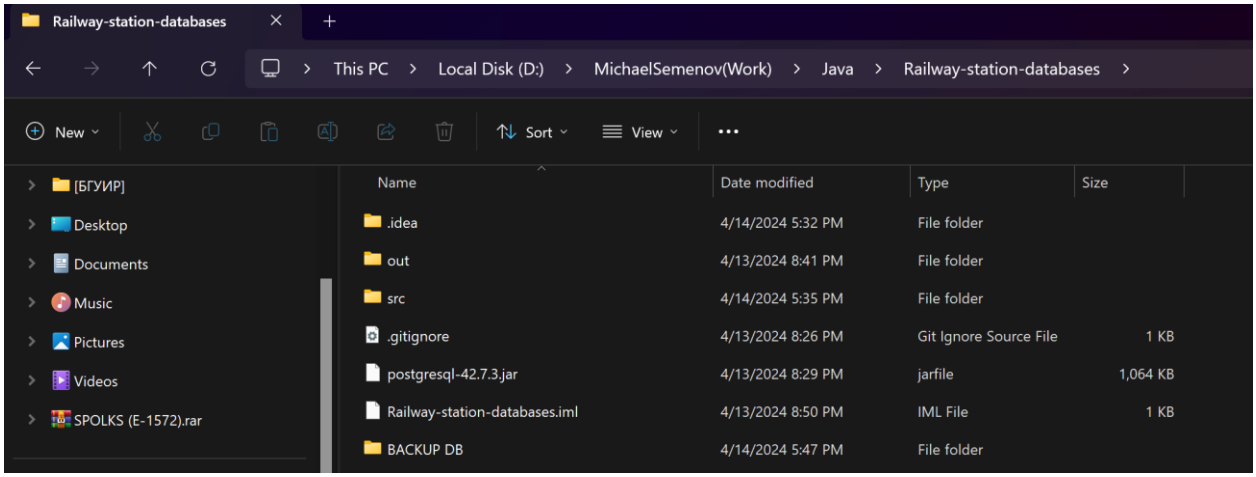

Рисунок 3.1 – Корень проекта

Далее открываем командную строку, по пути добираемся до корня проекта и вводим следующую команду, как показано на рисунке 3.2. В данном случае точка передается как параметр открытия проекта относительно перейденного пути.

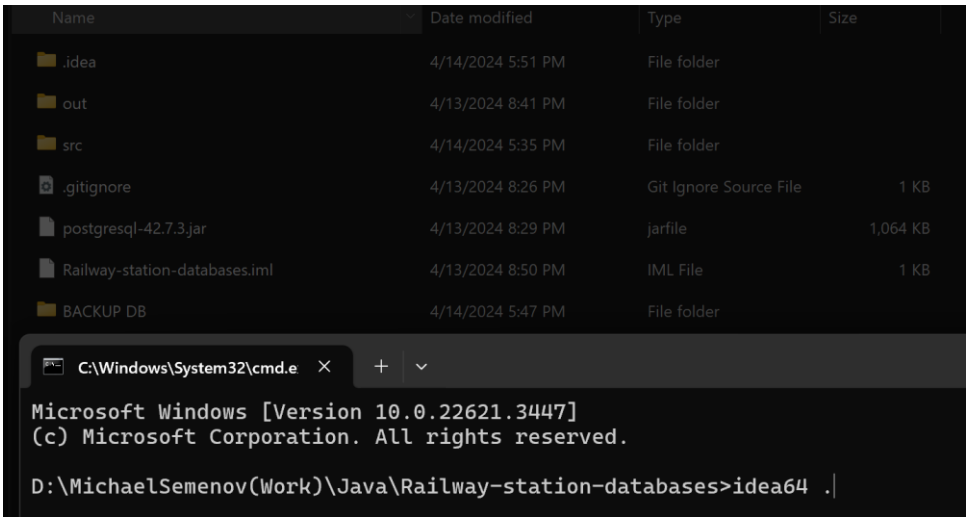

Рисунок 3.2 – Открытие проекта через IntelliJ IDEA

После открытия данного проекта через idea, как показано на рисунке 3.3, следует добавить зависимости – драйвер (модульный компонент) для взаимодействия программы с PostgreSQL. В исходных файл данный файл подписан как postgresql-42.7.3.jar. Далее, как показано на рисунке 3.4 открываем настройки структуры проекта через путь, а на рисунке 3.5 видим меню конфигурации зависимостей.

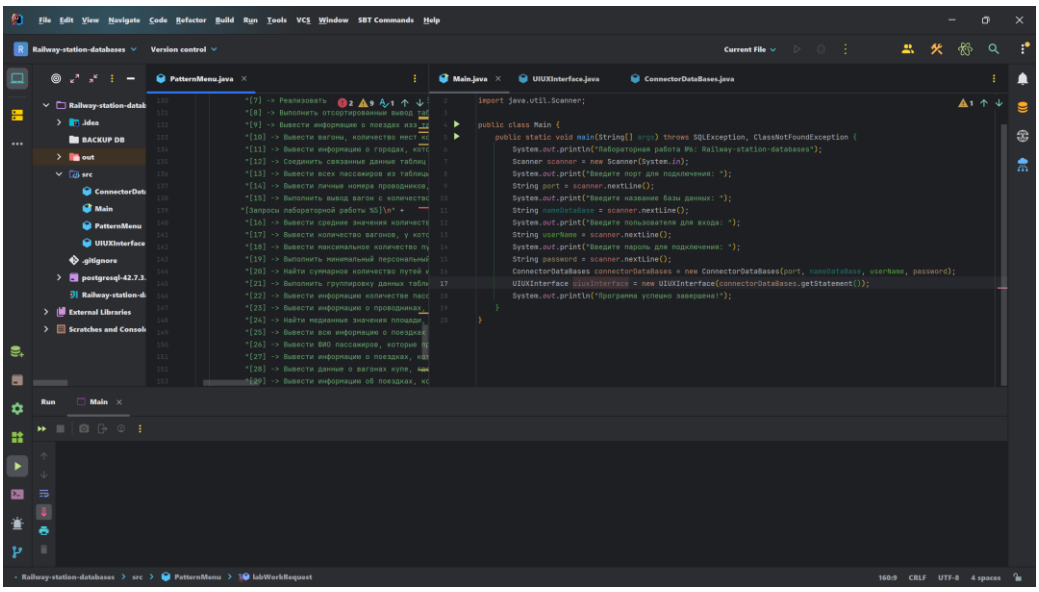

Рисунок 3.3 – Открытый проект через IntelliJ IDEA

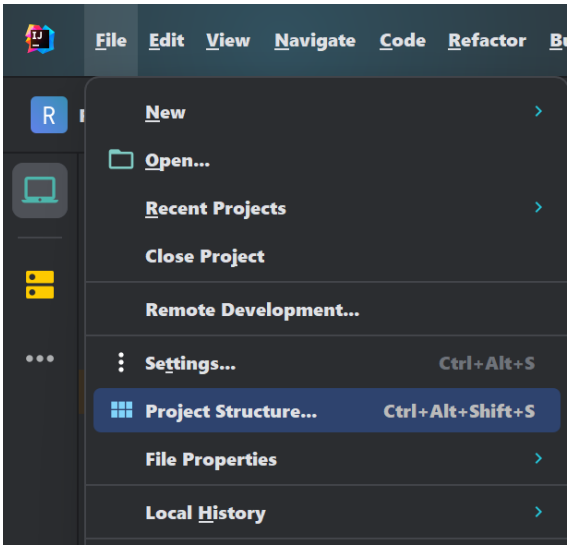

Рисунок 3.4 – Открытие конфигурации структуры проекта

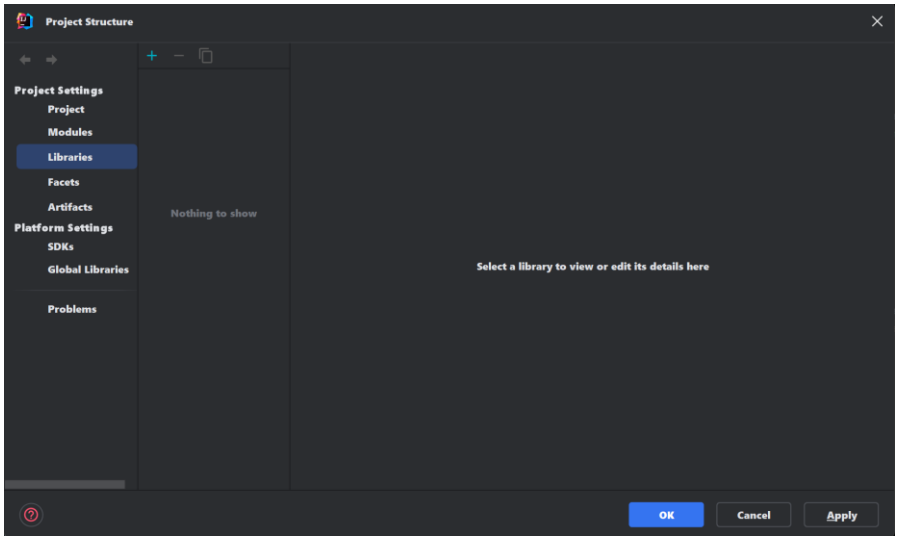

Рисунок 3.5 – Меню настройки структуры проекта

Далее для добавления нажимаем на значок «+» (New project library) и выбираем включение Java библиотеки как показано на рисунке 3.6.

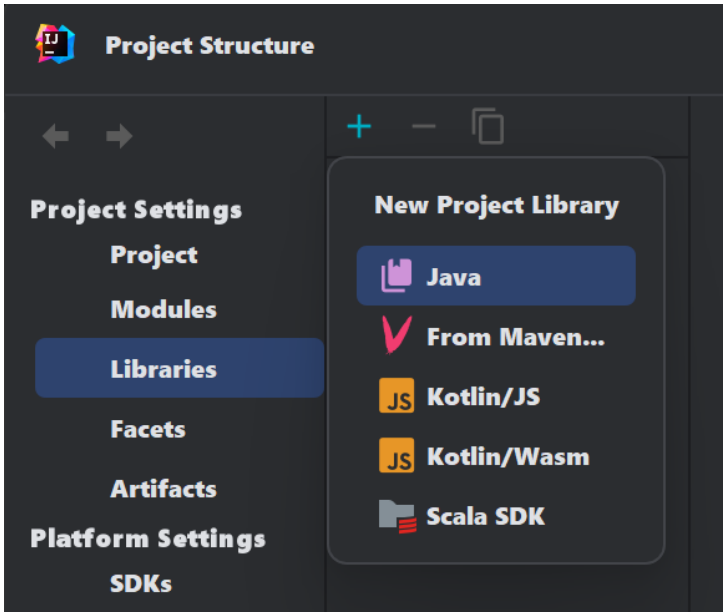

Рисунок 3.6 – Меню добавления Java библиотеки

После открытия открываем папку проекта и находим jar-файл под названием postgresql-42.7.3.jar и нажимаем «ОК», как показано на рисунке 3.7.

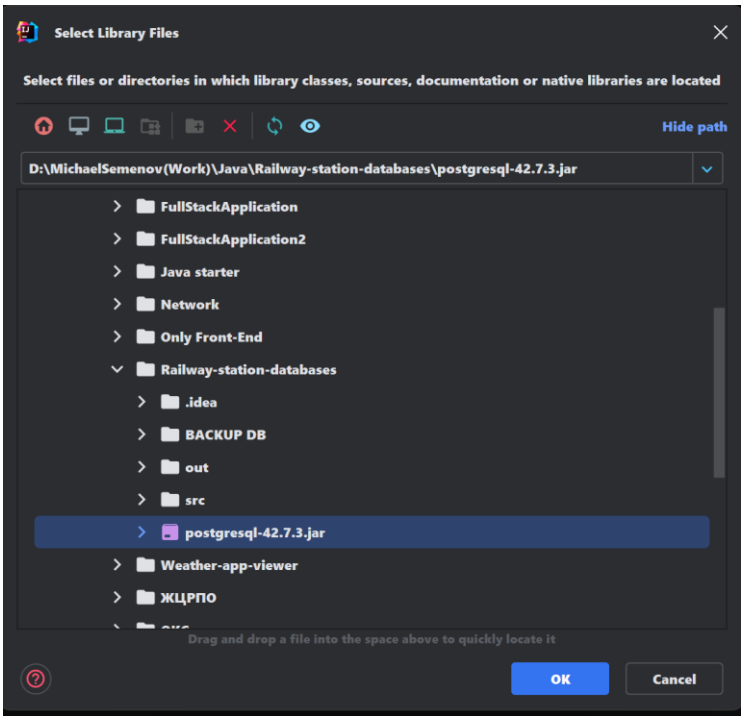

Рисунок 3.7 – Процесс добавления библиотеки (драйвера)

После нажатия кнопки «ОК» открывается окно подтверждения добавления внешней библиотеки, также нажимаем «ОК». На рисунке 3.8 изображено данное окно.

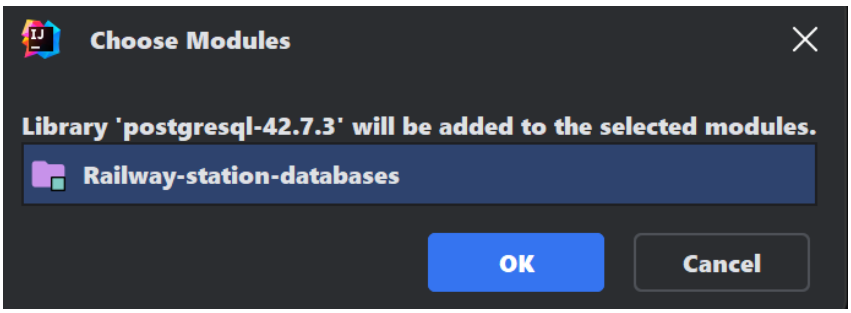

Рисунок 3.8 – Подтверждение добавления

После этого, как показано на рисунке 3.9 нажимаем кнопки «apply» и «ОК».

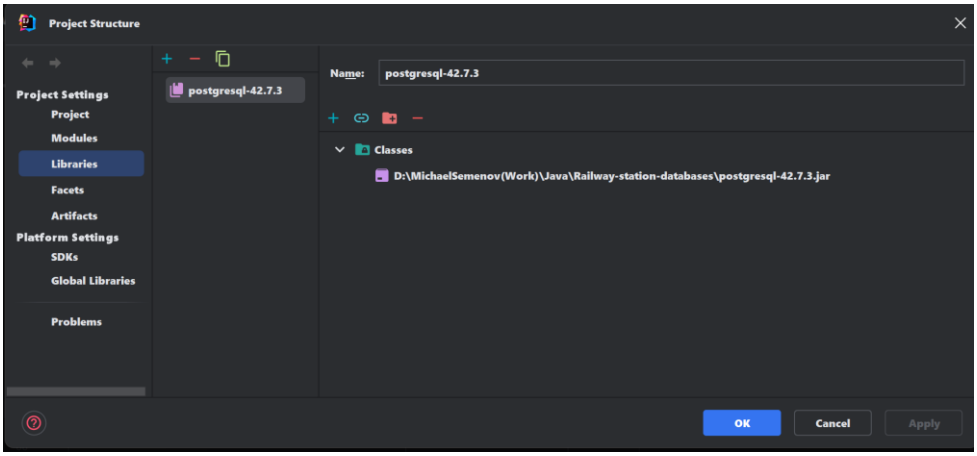

Рисунок 3.9 – Завершение добавления зависимостей

На данном этапе настройка IntelliJ IDEA закончена, следует теперь настроить базу данных для непосредственной работы.

В корне проекта, как показано на рисунке 3.1 показана папка «BACKUP DB», в которой хранится дамп базы данных «Railway-station-databases», который показан на рисунке 3.10.

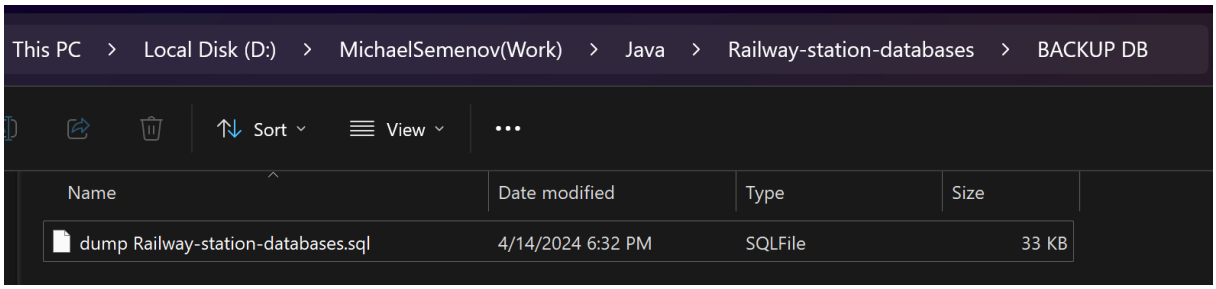

Рисунок 3.11 – Дамп исходной базы данных «Railway-station-databases»

Далее на рисунке 3.12 изображено открытие СУБД PostgreSQL. Так как данные манипуляции проходят на исходном ноутбуке, то восстановление исходников будет происходить на другом сервере «PostgreSQL 15», как показано на рисунке 3.13.

Рисунок 3.12 – Запуск программы pg admin 4

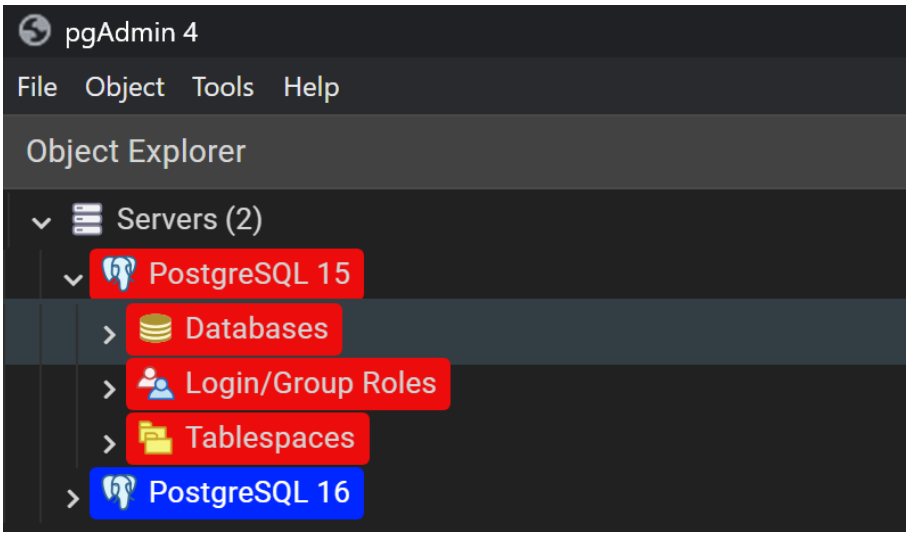

Рисунок 3.13 – Исходный сервер для восстановления базы данных

Далее создаем базу данных в данном сервере, как показано на рисунках 3.14 и 3.15. После ввода название «Railway-station-databases» и нажимаем «ОК».

Рисунок 3.14 – Путь создания базы данных

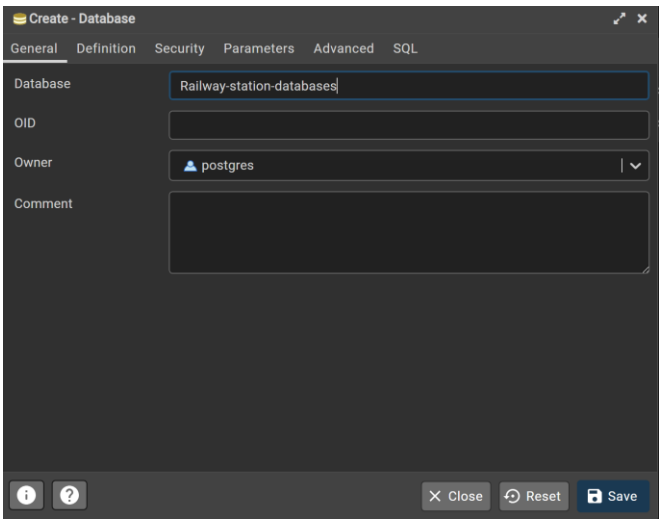

Рисунок 3.15 – Создание базы данных для восстановления

Далее на рисунке 3.16 изображен путь для восстановления базы данных. А на рисунке 3.17 вводим путь восстановления к папке «BACKUP DB» и к файлу «dump Railway-station-databases.sql».

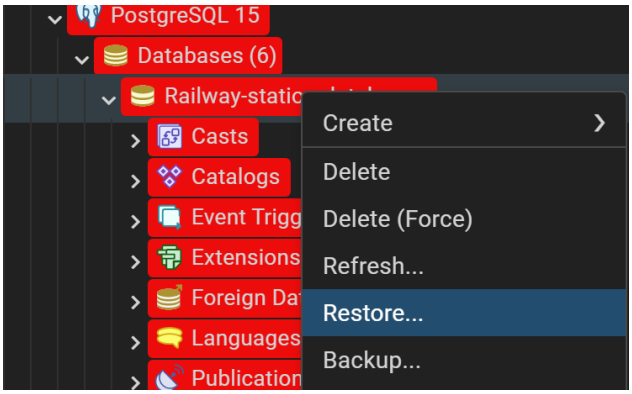

Рисунок 3.16 – Исходный путь восстановления

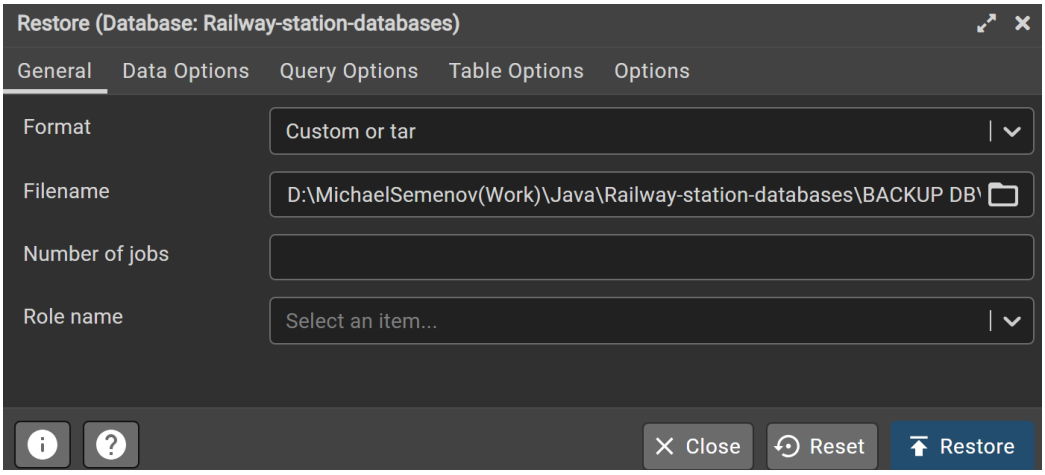

Рисунок 3.17 – Восстановление файла дампа базы данных

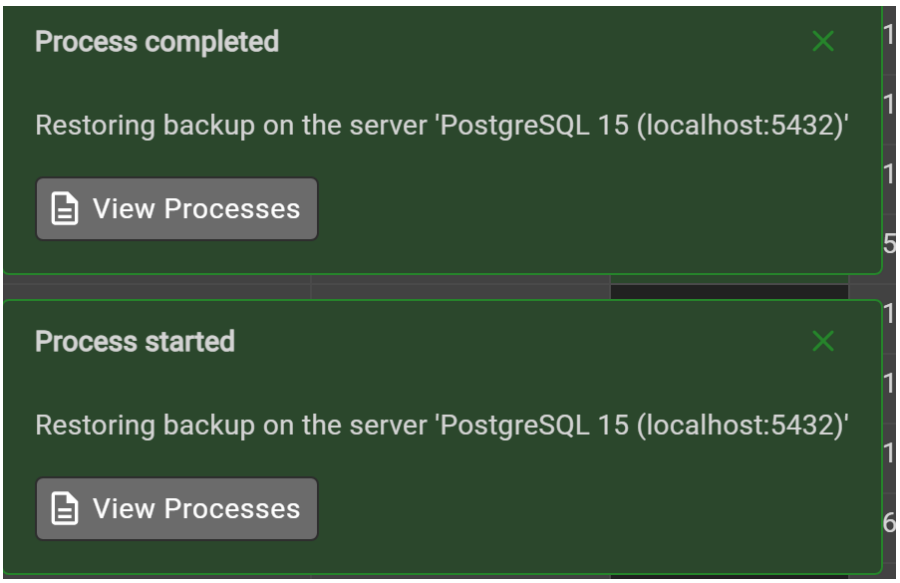

Рисунок 3.18 – Результаты процесса восстановления

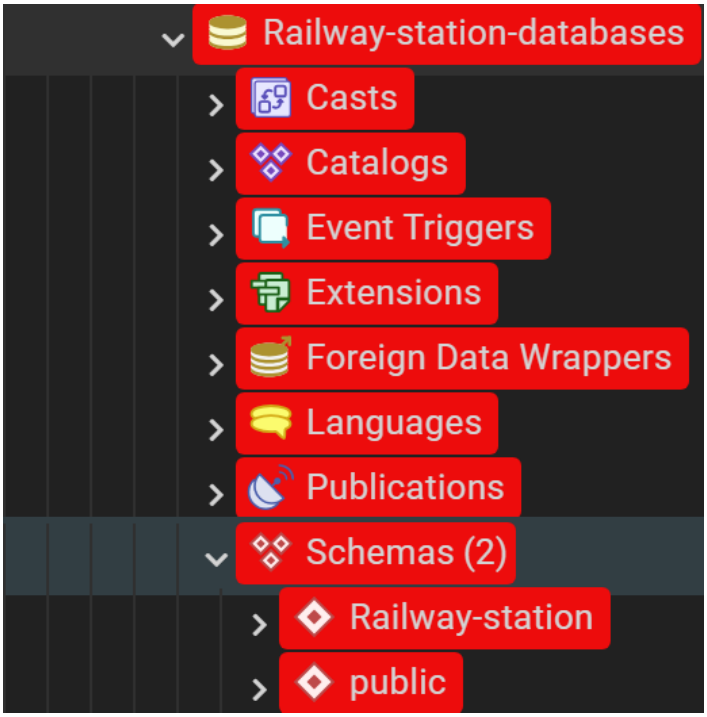

Рисунок 3.19 – Восстановленная схема данных

После данной настройки следует проверить фактическое подключение и доступ к базе данных через интерфейс Intellij idea. На рисунке 3.20 изображены настройки для подключения. Следует вводить фактические данные, где размещена база данных и каким способом на устройстве.

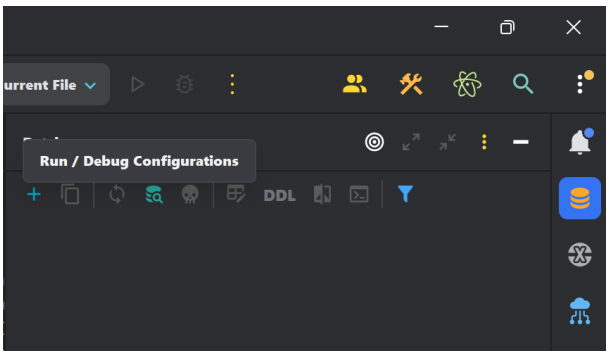

Рисунок 3.20 – Окно просмотра активных баз данных

Далее нажимаем на «+» и выбирает сервер, на которым размещена база данных, на рисунке 3.21 показан данный выбор.

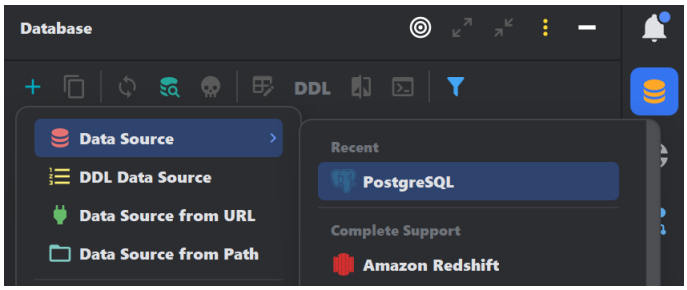

Рисунок 3.21 – Выбор на каком сервере размещена база данных

Как показано на рисунке 3.22 вводим конфигурационные данные для подключения к базе данных и нажимаем «Test connection». В нижнем левом углу видим успешно соединение.

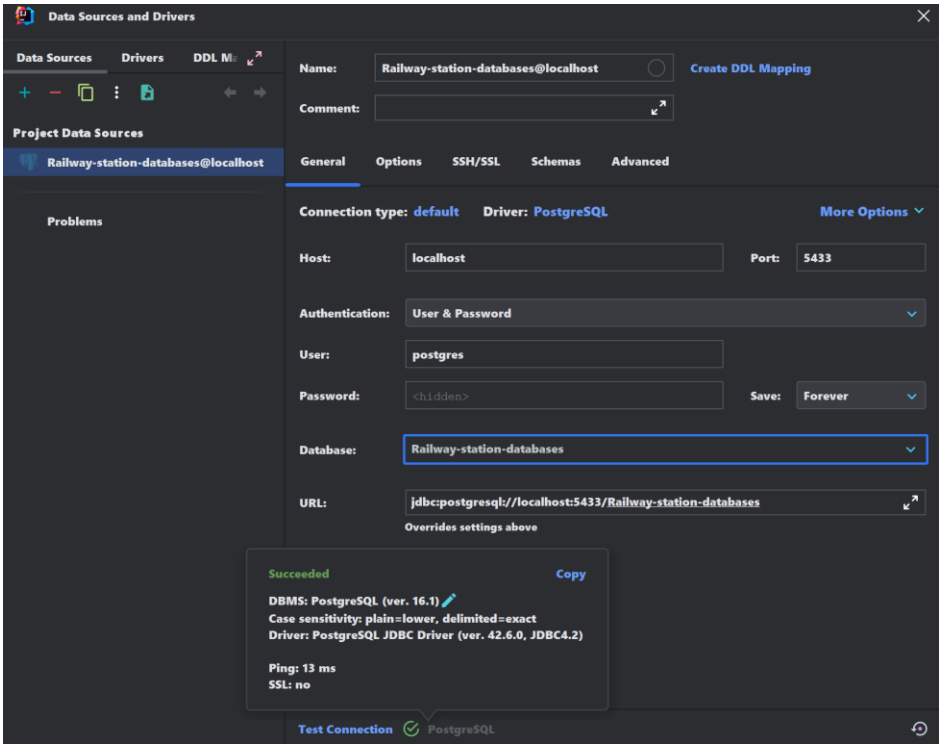

Рисунок 3.22 – Успешная проверка подключения

После соединения и установки всех настроек программа готова к использованию!

3.3 Руководство пользователя по эксплуатацию ПО

Далее рассмотрим функционал нашего консольного приложения после правильной его настройки, на рисунке 3.23 нажимам кнопку «Run» и смотрим на консольное окно снизу.

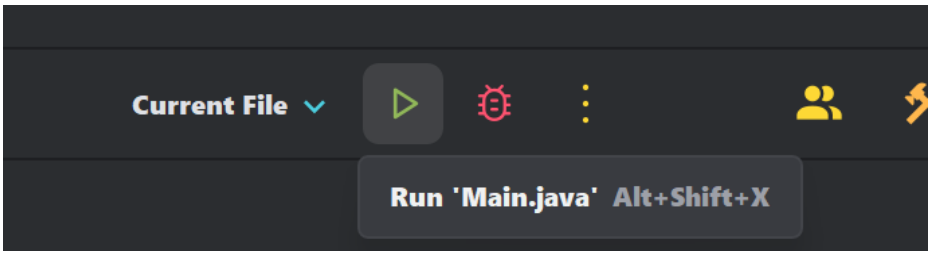

Рисунок 3.23 – Запуск приложения

Далее программа требует ввести конфигурационные данные для подключения. После ввода нажимаем «Entry» и видим надпись об успешном подключении к базе данных, как показано на рисунке 3.24.

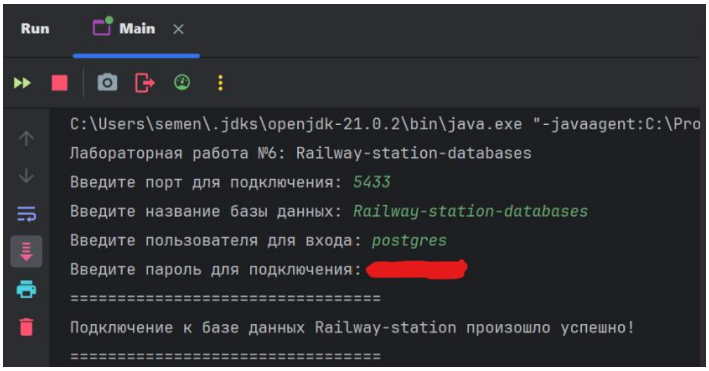

Рисунок 3.24 – Успешно подключение к базе данных

После на рисунке 3.25 изображено начальное меню, для взаимодействия пользователя с базой данной.

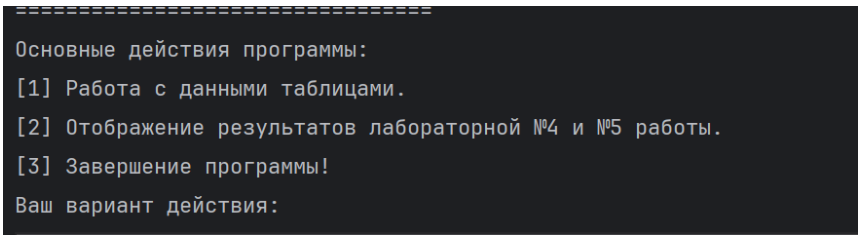

Рисунок 3.25 – Начальной меню программы

Далее для проверки нажмем «1» и «Entry». После этого на рисунке 3.26 программа предлагает выбрать таблицу для взаимодействия.

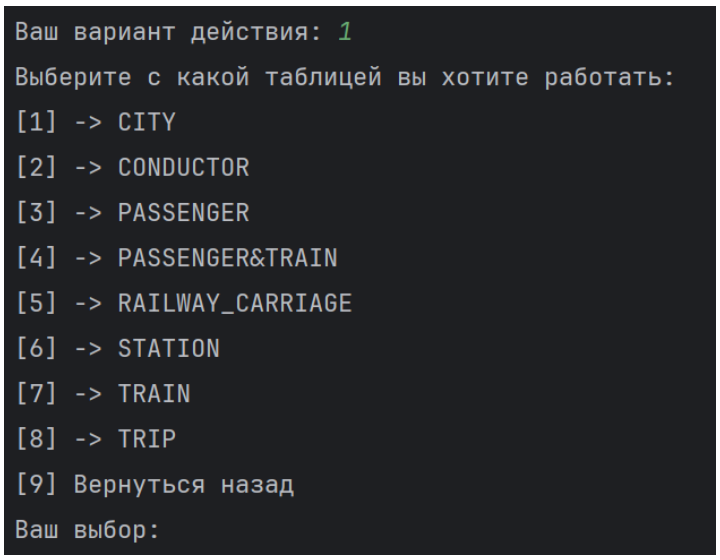

Рисунок 3.26 – Выбор таблицы для взаимодействия

Для примера выберем таблицу «CITY» и нажимаем «Entry». Так как работа и функционал таблиц идентичен, то проведем все манипуляции с одной таблицей. На рисунке 3.27 изображен меню работы с таблицами.

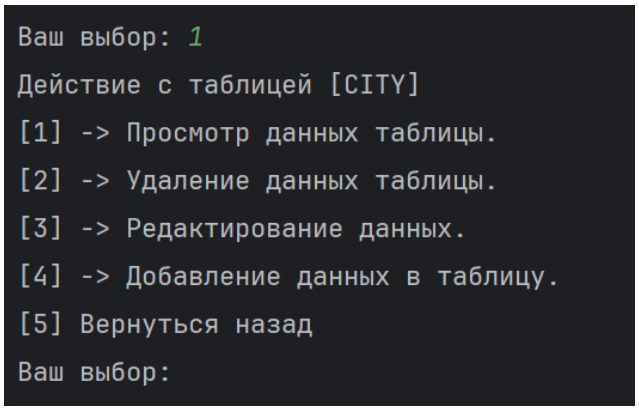

Рисунок 3.27 – Меню для работы с таблицей «CITY»

Для начала проверим вывод всех данных в таблице через меню кнопки «1». На рисунке 3.28 изображен результат работы вывода всех данных. Для проверки удаления нажимается кнопка «2» и вводится параметр, по какому происходит удаление. Удаление производим по первичному ключу, что делает удаление уникальным и однострочным. На рисунке 3.29 показан ввода данных для удаления, а на рисунке 3.30 показана результирующая таблица.

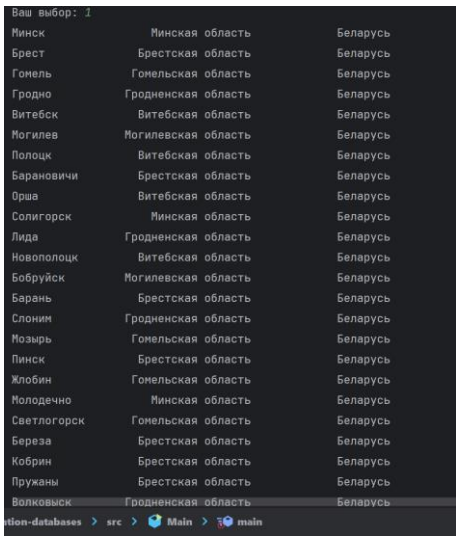

Рисунок 3.28 – Вывод всех данных таблицы

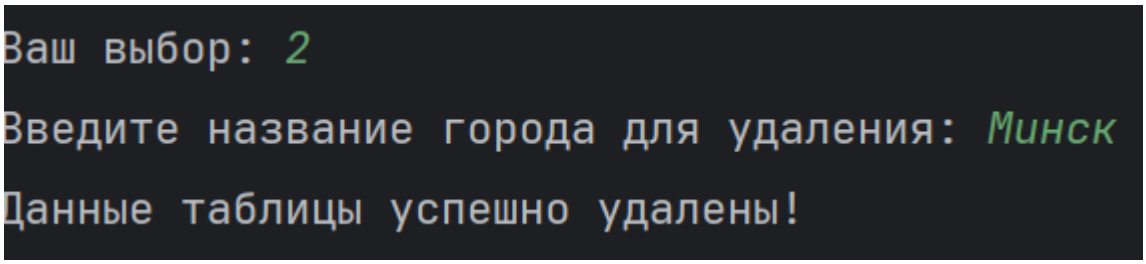

Рисунок 3.29 – Удаление данных из таблицы

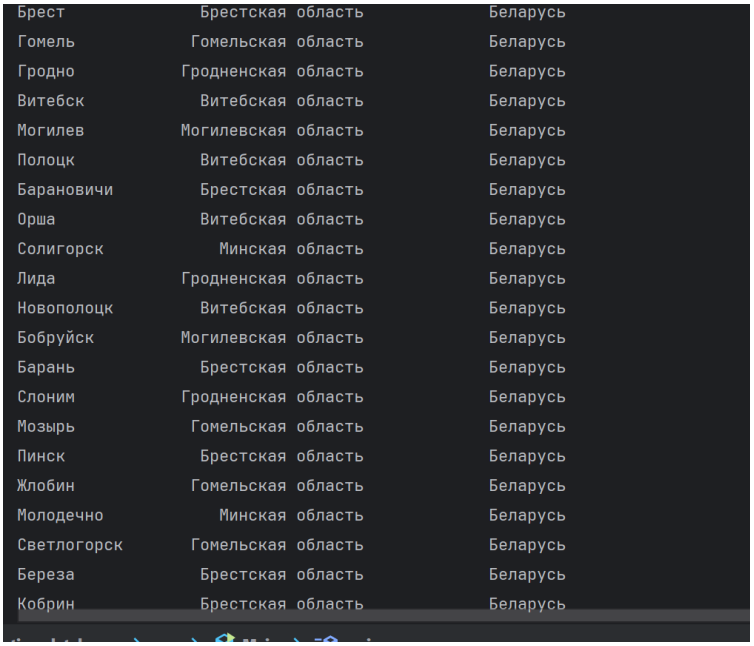

Рисунок 3.30 – Результат удаления

Далее проверим добавление данных в таблицу, для этого в контекстном меню нажимаем кнопку «4» и вводим данные, как показано на рисунке 3.31, нажимаем «Entry», после чего на рисунке 3.32 показана результирующая таблица с добавленной строкой данных.

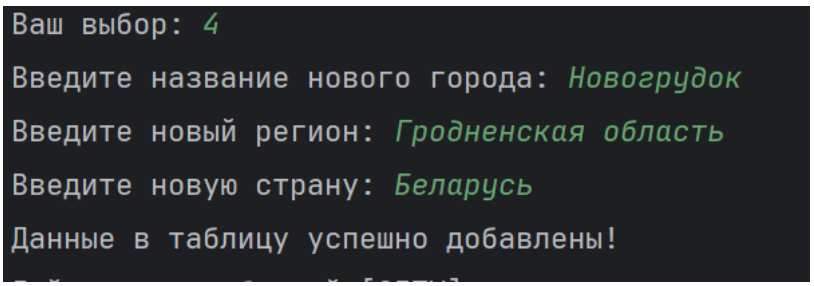

Рисунок 3.31 – Добавление нового города

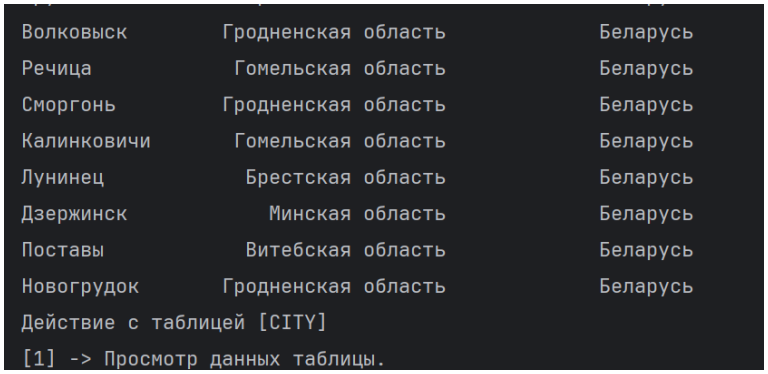

Рисунок 3.32 – Результат добавления в таблицу

Затем проверим выполнение команды изменения строчки данных в таблице, нажимаем кнопку «1». Далее вводим изменения, как показано на рисунке 3.33 и выводим результат изменения во всей таблице, которая изображена на рисунке 3.33.

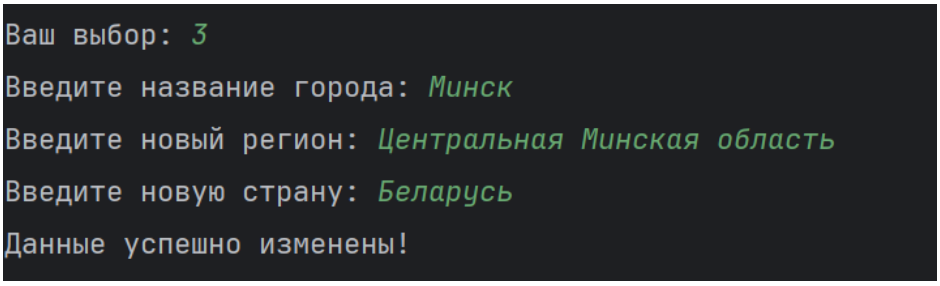

Рисунок 3.33 –Редактирование данных

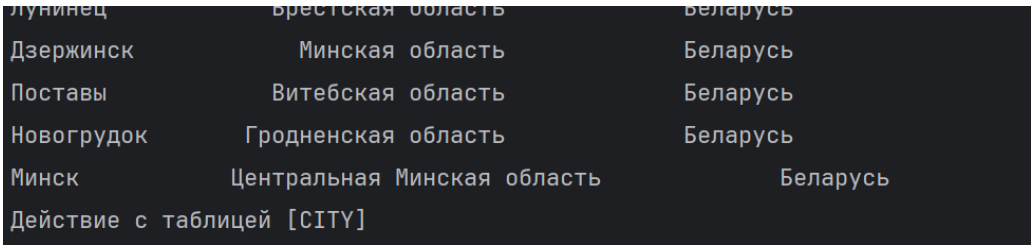

Рисунок 3.34 – Проверка изменения

Также в данной программе есть возможность вывода значений лабораторной работы №4 и №5. Для этого выходим назад в главное меню и нажимаем «2». На рисунке 3.35 изображен вывод меню для выбора вывода информации.

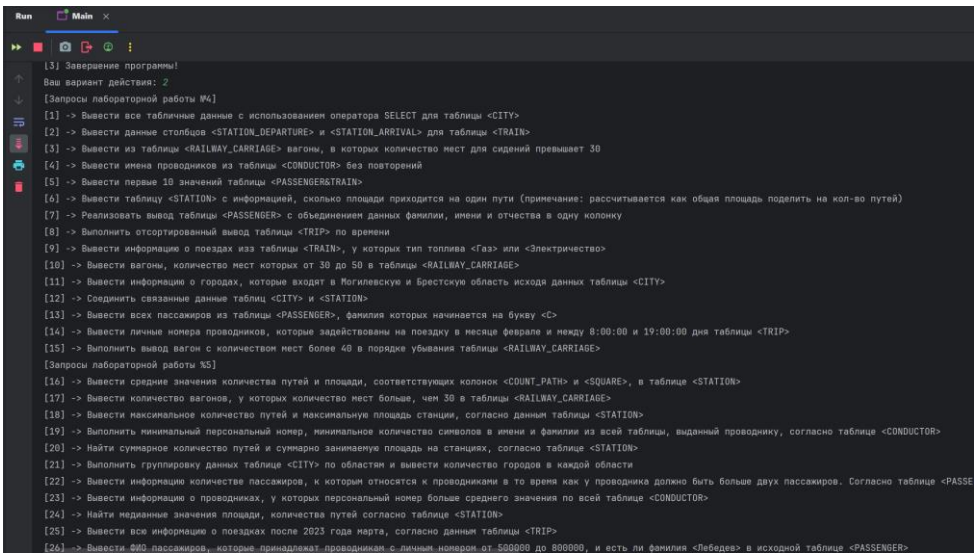

Рисунок 3.35 – Вывод данных запросов из лабораторных работ №4 и №5

Для примера посмотрим вывод первого запроса, для этого нажмем «1» и «Entry». На рисунке 3.36 вывод данных отображен по условию задания.

```
Ваш вариант действия: 1
Брест               Брестская область           Беларусь
Гомель              Гомельская область          Беларусь
Гродно              Гродненская область         Беларусь
Витебск             Витебская область           Беларусь
Могилев             Могилевская область         Беларусь
Полоцк              Витебская область           Беларусь
Барановичи          Брестская область           Беларусь
Орша                Витебская область           Беларусь
Солигорск             Минская область           Беларусь
Лида                Гродненская область         Беларусь
Новополоцк          Витебская область           Беларусь
Бобруйск            Могилевская область         Беларусь
Барань              Брестская область           Беларусь
Слоним              Гродненская область         Беларусь
Мозырь              Гомельская область          Беларусь
Пинск               Брестская область           Беларусь
Жлобин              Гомельская область          Беларусь
```

Рисунок 3.36 – Отображение данных под задание

Следовательно, данная программа работает корректно.

3.4 Исходный код программы

Исходный код программы описан в приложении А.

4. ВЫВОД

В данной лабораторной работе изучили процесс создания приложения для взаимодействия клиента с базой данных. Их основной реализации следует отметить, для внешнего подключения используются модульные компоненты (драйверы). В данном случае использовался низкоуровневый драйвер «jdbc statement».

Группа SQL-запросов: DML.

ПРИЛОЖЕНИЕ А
(обязательное)

Исходный код программы

```
/*
      Группа 130501, курсант Семёнов М.И.
      Лабораторная работа №6 по предмету «Базы данных»
      По теме «Создание приложения базы данных. Руководство пользователя!»
*/

001.  Файл «Main.java»
002.  import java.sql.SQLException;
003.  import java.util.Scanner;

004.  public class Main {
005.  public static void main(String[] args) throws SQLException,
ClassNotFoundException {
006.  System.out.println("Лабораторная работа №6: Railway-station-
databases");
007.  Scanner scanner = new Scanner(System.in);
008.  System.out.print("Введите порт для подключения: ");
009.  String port = scanner.nextLine();
010.  System.out.print("Введите название базы данных: ");
011.  String nameDataBase = scanner.nextLine();
012.  System.out.print("Введите пользователя для входа: ");
013.  String userName = scanner.nextLine();
014.  System.out.print("Введите пароль для подключения: ");
015.  String password = scanner.nextLine();
016.  ConnectorDataBases connectorDataBases = new ConnectorDataBases(port,
nameDataBase, userName, password);
017.  UIUXInterface uiuxInterface = new
UIUXInterface(connectorDataBases.getStatement());
018.  System.out.println("Программа успешно завершена!");
019.  }
020.  }

021.  Файл «UIUXInterface.java»
022.  import java.sql.SQLException;
023.  import java.sql.Statement;
024.  import java.util.Scanner;

025.  public class UIUXInterface {
026.  private Scanner scanner;
027.  private Statement statement;
028.  public UIUXInterface(Statement statement) throws SQLException {
029.  scanner = new Scanner(System.in);
030.  this.statement = statement;
031.  showMenu();
032.  }
033.  void showMenu() throws SQLException {
034.  boolean flag = true;
035.  int choice;
036.  do{
037.  System.out.print(PatternMenu.menuShowInformation());
038.  choice = scanner.nextInt();
039.  switch (choice){
040.  case 1:{
041.  showChoiceTableMenu();
042.  }break;
```

```
043.  case 2:{
044.  showInfoLabWorks();
045.  }break;
046.  case 3: {
047.  flag = false;
048.  }break;
049.  default:{
050.  System.out.println("Выбрано неверное действие, попробуйте попытку
снова");
051.  }
052.  }

053.  }while(flag);
054.  }

055.  void showChoiceTableMenu() throws SQLException {
056.  while(true){
057.  System.out.print(PatternMenu.showTableInformation());
058.  int choice = scanner.nextInt();
059.  if(choice != 9) showTableMenu(choice,
PatternMenu.returnNameTable(choice));
060.  else break;
061.  }

062.  }

063.  void showInfoLabWorks() throws SQLException {
064.  int choice;
065.  boolean flag = true;
066.  do{
067.  System.out.print(PatternMenu.showLaboratoryInformation());
068.  choice = scanner.nextInt();
069.  switch (choice){
070.  case 31: flag = false; break;
071.  default:
072.  if(choice >=1 && choice <= 30) PatternMenu.labWorkRequest(choice,
statement);
073.  else System.out.println("Вы указали неверное действие, попробуйте
снова!");
074.  }
075.  }while(flag);
076.  }
077.  void showTableMenu(int choice, String tableName) throws SQLException {
078.  boolean flag = true;
079.  do{
080.  System.out.print(PatternMenu.showTableInfo(tableName));
081.  choice = scanner.nextInt();
082.  switch(choice){
083.  case 1:{
084.  PatternMenu.showTableData(statement,tableName);
085.  }break;
086.  case 2:{
087.  PatternMenu.deleteRowTable(statement, tableName);
088.  System.out.println("Данные таблицы успешно удалены!");
089.  }break;
090.  case 3:{
091.  PatternMenu.changeRowsInformation(statement, tableName);
092.  System.out.println("Данные успешно изменены!");
093.  }break;
094.  case 4:{
095.  PatternMenu.addRowsInTable(statement, tableName);
096.  System.out.println("Данные в таблицу успешно добавлены!");
```

```
097.  }break;
098.  case 5:{
099.  flag = false;
100.  }break;
101.  default:
102.  System.out.println("Неправильный выбор!");
103.  }
104.  }while (flag);
105.  }
106.  }                    System.out.println("Данные в таблицу успешно
добавлены!");
107.  }break;
108.  case 5:{
109.  flag = false;
110.  }break;
111.  default:
112.  System.out.println("Неправильный выбор!");
113.  }
114.  }while (flag);
115.  }
116.  }
```

117. Файл «ConnectorDataBases.java»

```
118.  import java.sql.*;
119.  import java.util.Properties;

120.  public class ConnectorDataBases {

121.  private Statement statement;
122.  public ConnectorDataBases(String port, String nameDB, String user,
String password) throws ClassNotFoundException, SQLException {
123.  System.out.println("================================");
124.  Class.forName("org.postgresql.Driver");
125.  String url = "jdbc:postgresql://localhost:" + port +"/" + nameDB;
126.  Properties autorization = new Properties();
127.  autorization.put("user", user);

128.  autorization.put("password", password);
129.  Connection connection = DriverManager.getConnection(url, autorization);
130.  System.out.println("Подключение к базе данных Railway-station произошло
успешно!");
131.  statement = connection.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
132.  System.out.println("================================");
133.  }

134.  public Statement getStatement() {
135.  return statement;
136.  }
137.  }
```

138. Файл «PatternMenu.java»

```
139.  import org.postgresql.jdbc.EscapeSyntaxCallMode;

140.  import javax.print.attribute.standard.MediaSize;
141.  import java.sql.ResultSet;
142.  import java.sql.SQLException;
143.  import java.sql.SQLOutput;
144.  import java.sql.Statement;
145.  import java.util.Objects;
146.  import java.util.Scanner;
```

```
147.  public class PatternMenu {
148.  private static String schemaName = "Railway-station";
149.  static String menuShowInformation(){
150.  return "Основные действия программы: \n" +
151.  "[1] Работа с данными таблицами. \n" +
152.  "[2] Отображение результатов лабораторной №4 и №5 работы. \n" +
153.  "[3] Завершение программы! \n" +
154.  "Ваш вариант действия: ";
155.  }
156.  static String showTableInformation(){
157.  return "Выберите с какой таблицей вы хотите работать: \n" +
158.  "[1] -> CITY \n" +
159.  "[2] -> CONDUCTOR \n" +
160.  "[3] -> PASSENGER \n" +
161.  "[4] -> PASSENGER&TRAIN \n" +
162.  "[5] -> RAILWAY_CARRIAGE \n" +
163.  "[6] -> STATION \n" +
164.  "[7] -> TRAIN \n" +
165.  "[8] -> TRIP \n" +
166.  "[9] Вернуться назад \n" +
167.  "Ваш выбор: ";
168.  }

169.  static String showTableInfo(String tableName){
170.  return "Действие с таблицей [" + tableName + "]\n"+
171.  "[1] -> Просмотр данных таблицы.\n" +
172.  "[2] -> Удаление данных таблицы.\n" +
173.  "[3] -> Редактирование данных.\n" +
174.  "[4] -> Добавление данных в таблицу.\n" +
175.  "[5] Вернуться назад \n" +
176.  "Ваш выбор: ";
177.  }
178.  static String returnNameTable(int choice){
179.  return switch (choice){
180.  case 1: yield "CITY";
181.  case 2: yield "CONDUCTOR";
182.  case 3: yield "PASSENGER";
183.  case 4: yield "PASSENGER&TRAIN";
184.  case 5: yield "RAILWAY_CARRIAGE";
185.  case 6: yield "STATION";
186.  case 7: yield "TRAIN";
187.  case 8: yield "TRIP";
188.  default:
189.  throw new IllegalStateException("Unexpected value: " + choice);
190.  };
191.  }

192.  static void showTableData(Statement statement, String tableName) throws
SQLException {
193.  ResultSet resultSet = statement.executeQuery("SELECT * FROM \"" +
schemaName + "\"." + "\"" +tableName + "\";");
194.  while(resultSet.next()){
195.  showLineInformation(resultSet, tableName);
196.  }

197.  }
198.  static void showLineInformation(ResultSet rs, String tableName) throws
SQLException {
199.  if(Objects.equals(tableName, "CITY")){
200.  String CITY_NAME = rs.getString("CITY_NAME");
201.  String REGION = rs.getString("REGION");
202.  String COUNTRY = rs.getString("COUNTRY");
```

```
203.  System.out.printf("%-15s %20s %20s%n", CITY_NAME, REGION, COUNTRY);
204.  }
205.  if(tableName.equals("CONDUCTOR")){
206.  int PERSONAL_NUMBER = rs.getInt("PERSONAL_NUMBER");
207.  String SURNAME = rs.getString("SURNAME");
208.  String NAME = rs.getString("NAME");
209.  String MIDDLE_NAME = rs.getString("MIDDLE_NAME");
210.  System.out.printf("%-15d %20s %20s %20s%n", PERSONAL_NUMBER, SURNAME,
NAME, MIDDLE_NAME);
211.  }
212.  if(tableName.equals("PASSENGER")){
213.  String NUMBER_PASSPORT = rs.getString("NUMBER_PASSPOER");
214.  String SURNAME = rs.getString("SURNAME");
215.  String NAME = rs.getString("NAME");
216.  String MIDDLE_NAME = rs.getString("MIDDLE_NAME");
217.  int PERSONAL_NUMBER_FK = rs.getInt("PERSONAL_NUMBER_FK");
218.  System.out.printf("%-15s %15s %15s %15s %15d%n", NUMBER_PASSPORT,
SURNAME, NAME,  MIDDLE_NAME, PERSONAL_NUMBER_FK);
219.  }
220.  if(tableName.equals("PASSENGER&TRAIN")){
221.  String NUMBER_PASSPORT_FK_PK = rs.getString("NUMBER_PASSPORT_FK_PK");
222.  String TRAIN_ID_PK_FK = rs.getString("TRAIN_ID_PK_FK");
223.  System.out.printf("%-10s %15s%n", NUMBER_PASSPORT_FK_PK,
TRAIN_ID_PK_FK);
224.  }
225.  if(tableName.equals("RAILWAY_CARRIAGE")){
226.  int RAILWAY_CARRIAGE_ID = rs.getInt("RAILWAY_CARRIAGE_ID");
227.  int PLACE = rs.getInt("PLACE");
228.  int NUMBER_CARRIAGE = rs.getInt("NUMBER_CARRIAGE");
229.  String TYPE_CARRIAGE = rs.getString("TYPE_CARRIAGE");
230.  int TRAIN_ID_FK = rs.getInt("TRAIN_ID_FK");
231.  System.out.printf("%-10d %10d %15d %10s %10d%n", RAILWAY_CARRIAGE_ID,
PLACE, NUMBER_CARRIAGE, TYPE_CARRIAGE, TRAIN_ID_FK);
232.  }
233.  if(tableName.equals("STATION")){
234.  int STATION_ID = rs.getInt("STATION_ID");
235.  String STATION_NAME = rs.getString("STATION_NAME");
236.  int COUNT_PATH = rs.getInt("COUNT_PATH");
237.  float SQUARE = rs.getFloat("SQUARE");
238.  String CITY_NAME_FK = rs.getString("CITY_NAME_FK");
239.  System.out.printf("%-10d %25s %10d %15f %15s%n", STATION_ID,
STATION_NAME, COUNT_PATH, SQUARE, CITY_NAME_FK);
240.  }
241.  if(tableName.equals("TRAIN")){
242.  int TRAIN_ID = rs.getInt("TRAIN_ID");
243.  String STATION_DEPARTURE = rs.getString("STATION_DEPARTURE");
244.  String STATION_ARRIVAL = rs.getString("STATION_ARRIVAL");
245.  String FUEL_TYPE = rs.getString("FUEL_TYPE");
246.  int  STATION_ID_FK = rs.getInt("STATION_ID_FK");
247.  System.out.printf("%-10d %20s %20s %15s %10d%n", TRAIN_ID,
STATION_DEPARTURE, STATION_ARRIVAL, FUEL_TYPE, STATION_ID_FK);
248.  }
249.  if(tableName.equals("TRIP")){
250.  int TRAIN_ID_PK_FK = rs.getInt("TRAIN_ID_PK_FK");
251.  int RAILWAY_CARRIAGE_ID_PK_FK = rs.getInt("RAILWAY_CARRIAGE_ID_PK_FK");
252.  int PERSONAL_NUMBER_PK_FK = rs.getInt("PERSONAL_NUMBER_PK_FK");
253.  String DATA = rs.getDate("DATA").toString();
254.  String TIME = rs.getTime("TIME").toString();
255.  System.out.printf("%-5d %10d %10d %15s %15s%n", TRAIN_ID_PK_FK,
RAILWAY_CARRIAGE_ID_PK_FK, PERSONAL_NUMBER_PK_FK, DATA, TIME);
256.  }
257.  }
```

```
258.  static String showLaboratoryInformation(){
259.  return "[Запросы лабораторной работы №4] \n" +
260.  "[1] -> Вывести все табличные данные с использованием оператора SELECT
для таблицы <CITY>\n" +
261.  "[2] -> Вывести данные столбцов <STATION_DEPARTURE> и <STATION_ARRIVAL>
для таблицы <TRAIN>\n"+
262.  "[3] -> Вывести из таблицы <RAILWAY_CARRIAGE> вагоны, в которых
количество мест для сидений превышает 30\n" +
263.  "[4] -> Вывести имена проводников из таблицы <CONDUCTOR> без
повторений\n" +
264.  "[5] -> Вывести первые 10 значений таблицы <PASSENGER&TRAIN>\n" +
265.  "[6] -> Вывести таблицу <STATION> с информацией, сколько площади
приходится на один пути (примечание: рассчитывается как общая площадь
поделить на кол-во путей)\n" +
266.  "[7] -> Реализовать вывод таблицы <PASSENGER> с объединением данных
фамилии, имени и отчества в одну колонку\n" +
267.  "[8] -> Выполнить отсортированный вывод таблицы <TRIP> по времени\n" +
268.  "[9] -> Вывести информацию о поездах изз таблицы <TRAIN>, у которых тип
топлива <Газ> или <Электричество>\n" +
269.  "[10] -> Вывести вагоны, количество мест которых от 30 до 50 в таблицы
<RAILWAY_CARRIAGE>\n" +
270.  "[11] -> Вывести информацию о городах, которые входят в Могилевскую и
Брестскую область исходя данных таблицы <CITY>\n" +
271.  "[12] -> Соединить связанные данные таблиц <CITY> и <STATION>\n" +
272.  "[13] -> Вывести всех пассажиров из таблицы <PASSENGER>, фамилия
которых начинается на букву <C>\n" +
273.  "[14] -> Вывести личные номера проводников, которые задействованы на
поездку в месяце феврале и между 8:00:00 и 19:00:00 дня таблицы <TRIP>\n" +
274.  "[15] -> Выполнить вывод вагон с количеством мест более 40 в порядке
убывания таблицы <RAILWAY_CARRIAGE>\n" +
275.  "[Запросы лабораторной работы %5]\n" +
276.  "[16] -> Вывести средние значения количества путей и площади,
соответствующих колонок <COUNT_PATH> и <SQUARE>, в таблице <STATION>\n" +
277.  "[17] -> Вывести количество вагонов, у которых количество мест больше,
чем 30 в таблицы <RAILWAY_CARRIAGE>\n" +
278.  "[18] -> Вывести максимальное количество путей и максимальную площадь
станции, согласно данным таблицы <STATION>\n" +
279.  "[19] -> Выполнить минимальный персональный номер, минимальное
количество символов в имени и фамилии из всей таблицы, выданный проводнику,
согласно таблице <CONDUCTOR>\n" +
280.  "[20] -> Найти суммарное количество путей и суммарно занимаемую площадь
на станциях, согласно таблице <STATION>\n" +
281.  "[21] -> Выполнить группировку данных таблице <CITY> по областям и
вывести количество городов в каждой области\n" +
282.  "[22] -> Вывести информацию количестве пассажиров, к которым относятся
к проводниками в то время как у проводника должно быть больше двух
пассажиров. Согласно таблице <PASSENGER>\n" +
283.  "[23] -> Вывести информацию о проводниках, у которых персональный номер
больше среднего значения по всей таблице <CONDUCTOR>\n" +
284.  "[24] -> Найти медианные значения площади, количества путей согласно
таблице <STATION>\n" +
285.  "[25] -> Вывести всю информацию о поездках после 2023 года марта,
согласно данным таблицы <TRIP>\n" +
286.  "[26] -> Вывести ФИО пассажиров, которые принадлежат проводникам с
личным номером от 500000 до 800000, и есть ли фамилия <Лебедев> в исходной
таблице <PASSENGER>\n" +
287.  "[27] -> Вывести информацию о поездках, которые не осуществляются между
датами от <2023-03-05> до <2023-09-12>, согласно таблице <TRIP>\n" +
288.  "[28] -> Вывести данные о вагонах купе, где количество мест больше 20 и
информацию о типах <Плацкарт>, количество мест, где больше 20\n" +
289.  "[29] -> Вывести информацию об поездках, которые прошли в дни кроме
месяца март\n" +
```

```
290.  "[30] -> Вывести поезда, в которых количество мест больше 20, но без
типа вагона <Плацкарт>\n" +
291.  "[31] Вернуться назад.\n" +
292.  "Ваш вариант действия: ";
293.  }


294.  //Для строго вывода лабораторной работы №5 №6
295.  public static void labWorkShowRequest(int choice, ResultSet resultSet)
throws SQLException {
296.  switch (choice) {
297.  case 2: {
298.  while (resultSet.next()) {
299.  String STATION_DEPARTURE = resultSet.getString("STATION_DEPARTURE");
300.  String STATION_ARRIVAL = resultSet.getString("STATION_ARRIVAL");
301.  System.out.printf("%-20s %20s%n", STATION_ARRIVAL, STATION_DEPARTURE);
302.  }
303.  }
304.  break;

305.  case 3: {
306.  while (resultSet.next()) {
307.  int RAILWAY_CARRIAGE_ID = resultSet.getInt("RAILWAY_CARRIAGE_ID");
308.  int PLACE = resultSet.getInt("PLACE");
309.  int NUMBER_CARRIAGE = resultSet.getInt("NUMBER_CARRIAGE");
310.  String TYPE_CARRIAGE = resultSet.getString("TYPE_CARRIAGE");
311.  int TRAIN_ID_FK = resultSet.getInt("TRAIN_ID_FK");
312.  System.out.printf("%-10d %10d %15d %10s %10d%n", RAILWAY_CARRIAGE_ID,
PLACE, NUMBER_CARRIAGE, TYPE_CARRIAGE, TRAIN_ID_FK);
313.  }
314.  }
315.  break;
316.  case 4: {
317.  while (resultSet.next()) {
318.  String NAME = resultSet.getString("NAME");
319.  System.out.println(NAME);
320.  }
321.  }
322.  break;
323.  case 5: {
324.  while (resultSet.next()) {
325.  String NUMBER_PASSPORT_FK_PK = resultSet.getString("fast");
326.  String TRAIN_ID_PK_FK = resultSet.getString("TRAIN_ID_PK_FK");
327.  System.out.printf("%-10s %15s%n", NUMBER_PASSPORT_FK_PK,
TRAIN_ID_PK_FK);
328.  }
329.  }
330.  break;
331.  case 6: {
332.  while (resultSet.next()) {
333.  int STATION_ID = resultSet.getInt("STATION_ID");
334.  String STATION_NAME = resultSet.getString("STATION_NAME");
335.  int COUNT_PATH = resultSet.getInt("COUNT_PATH");
336.  float SQUARE = resultSet.getFloat("SQUARE");
337.  String CITY_NAME_FK = resultSet.getString("CITY_NAME_FK");
338.  int PATH_SQUARE = resultSet.getInt("PATH_SQUARE");
339.  System.out.printf("%-10d %25s %10d %15f %15s %15d%n", STATION_ID,
STATION_NAME, COUNT_PATH, SQUARE, CITY_NAME_FK, PATH_SQUARE);
340.  }
341.  }
342.  break;
343.  case 7: {
```

```
344.  while (resultSet.next()) {
345.  String NUMBER_PASSPOER = resultSet.getString("NUMBER_PASSPOER");
346.  int PERSONAL_NUMBER_FK = resultSet.getInt("PERSONAL_NUMBER_FK");
347.  String FIO = resultSet.getString("FIO");
348.  System.out.printf("%-15s %20d %40s%n", NUMBER_PASSPOER,
PERSONAL_NUMBER_FK, FIO);
349.  }
350.  }
351.  break;
352.  case 8: {
353.  while (resultSet.next()) {
354.  int TRAIN_ID_PK_FK = resultSet.getInt("TRAIN_ID_PK_FK");
355.  int RAILWAY_CARRIAGE_ID_PK_FK =
resultSet.getInt("RAILWAY_CARRIAGE_ID_PK_FK");
356.  int PERSONAL_NUMBER_PK_FK = resultSet.getInt("PERSONAL_NUMBER_PK_FK");
357.  String DATA = resultSet.getDate("DATA").toString();
358.  String TIME = resultSet.getTime("TIME").toString();
359.  System.out.printf("%-5d %10d %10d %15s %15s%n", TRAIN_ID_PK_FK,
RAILWAY_CARRIAGE_ID_PK_FK, PERSONAL_NUMBER_PK_FK, DATA, TIME);
360.  }
361.  }
362.  break;
363.  case 9: {
364.  while (resultSet.next()) {
365.  int TRAIN_ID = resultSet.getInt("TRAIN_ID");
366.  String STATION_DEPARTURE = resultSet.getString("STATION_DEPARTURE");
367.  String STATION_ARRIVAL = resultSet.getString("STATION_ARRIVAL");
368.  String FUEL_TYPE = resultSet.getString("FUEL_TYPE");
369.  int STATION_ID_FK = resultSet.getInt("STATION_ID_FK");
370.  System.out.printf("%-10d %20s %20s %15s %10d%n", TRAIN_ID,
STATION_DEPARTURE, STATION_ARRIVAL, FUEL_TYPE, STATION_ID_FK);
371.  }
372.  }
373.  break;
374.  case 11: {
375.  while (resultSet.next()) {
376.  String CITY_NAME = resultSet.getString("CITY_NAME");
377.  String REGION = resultSet.getString("REGION");
378.  String COUNTRY = resultSet.getString("COUNTRY");
379.  System.out.printf("%-15s %20s %20s%n", CITY_NAME, REGION, COUNTRY);
380.  }
381.  }
382.  break;
383.  case 12: {
384.  while (resultSet.next()) {
385.  int STATION_ID = resultSet.getInt("STATION_ID");
386.  String STATION_NAME = resultSet.getString("STATION_NAME");
387.  int COUNT_PATH = resultSet.getInt("COUNT_PATH");
388.  float SQUARE = resultSet.getFloat("SQUARE");
389.  String CITY_NAME_FK = resultSet.getString("CITY_NAME_FK");
390.  System.out.printf("%-10d %25s %10d %15f %15s%n", STATION_ID,
STATION_NAME, COUNT_PATH, SQUARE, CITY_NAME_FK);
391.  }
392.  }break;
393.  case 13:{
394.  while (resultSet.next()){
395.  String NUMBER_PASSPORT = resultSet.getString("NUMBER_PASSPOER");
396.  String SURNAME = resultSet.getString("SURNAME");
397.  String NAME = resultSet.getString("NAME");
398.  String MIDDLE_NAME = resultSet.getString("MIDDLE_NAME");
399.  int PERSONAL_NUMBER_FK = resultSet.getInt("PERSONAL_NUMBER_FK");
```

```
400.  System.out.printf("%-15s %15s %15s %15s %15d%n", NUMBER_PASSPORT,
SURNAME, NAME,  MIDDLE_NAME, PERSONAL_NUMBER_FK);
401.  }
402.  }break;
403.  case 14:{
404.  while (resultSet.next()){
405.  int PERSONAL_NUMBER_PK_FK = resultSet.getInt("PERSONAL_NUMBER_PK_FK");
406.  System.out.println(PERSONAL_NUMBER_PK_FK);
407.  }
408.  }
409.  case 16:{
410.  while (resultSet.next()){
411.  int count_path_average = resultSet.getInt("count_path_average");
412.  int square_average = resultSet.getInt("square_average");
413.  System.out.printf("%-15d %15d%n", count_path_average, square_average);
414.  }
415.  }break;
416.  case 17:{
417.  resultSet.next();
418.  System.out.println(resultSet.getInt("count"));
419.  }break;
420.  case 18: {
421.  resultSet.next();
422.  System.out.println(resultSet.getInt("MAX_COUNT_PATH") + "         " +
resultSet.getInt("MAX_SQUARE"));
423.  }break;
424.  case 19: {
425.  resultSet.next();
426.  System.out.println(resultSet.getInt("MIN_PERSONAL_NUMBER") + "      " +
resultSet.getInt("MIN_LENGTH_NAME") + "         " +
resultSet.getInt("MIN_LENGTH_SURNAME"));
427.  }break;
428.  case 20:{
429.  resultSet.next();
430.  System.out.println(resultSet.getInt("SUM_SQUARE") + "         " +
resultSet.getInt("SUM_COUNT_PATH"));
431.  }break;
432.  case 21:{
433.  while(resultSet.next()){
434.  String REGION = resultSet.getString("REGION");
435.  int count = resultSet.getInt("count");
436.  System.out.printf("%-30s %15d%n", REGION, count);
437.  }
438.  }break;
439.  case 22:{
440.  while (resultSet.next()){
441.  int PERSONAL_NUMBER_FK = resultSet.getInt("PERSONAL_NUMBER_FK");
442.  int count = resultSet.getInt("count");
443.  System.out.printf("%-15d %15d%n", PERSONAL_NUMBER_FK, count);
444.  }
445.  }break;
446.  case 23:{
447.  while (resultSet.next()){
448.  int PERSONAL_NUMBER = resultSet.getInt("PERSONAL_NUMBER");
449.  String SURNAME = resultSet.getString("SURNAME");
450.  String NAME = resultSet.getString("NAME");
451.  String MIDDLE_NAME = resultSet.getString("MIDDLE_NAME");
452.  System.out.printf("%-15d %20s %20s %20s%n", PERSONAL_NUMBER, SURNAME,
NAME, MIDDLE_NAME);
453.  }
454.  }break;
455.  case 24:{
```

```
456.  while (resultSet.next()){
457.  int MEDIAN_PATH = resultSet.getInt("MEDIAN_PATH");
458.  float MEDIAN_SQUARE = resultSet.getFloat("MEDIAN_SQUARE");
459.  System.out.println(MEDIAN_SQUARE + "            " + MEDIAN_PATH);
460.  }
461.  }break;
462.  case 25:{

463.  }break;
464.  }


465.  }
466.  public static void labWorkRequest(int choice, Statement statement)
throws SQLException {
467.  ResultSet result = null;
468.  if(choice == 1){
469.  showTableData(statement, "CITY");
470.  }
471.  else if(choice == 2){
472.  result = statement.executeQuery("SELECT \"STATION_DEPARTURE\",
\"STATION_ARRIVAL\" FROM \"" + schemaName +"\".\"TRAIN\"");
473.  }
474.  else if(choice == 3){
475.  result = statement.executeQuery("SELECT * FROM \"" + schemaName +
"\".\"RAILWAY_CARRIAGE\" WHERE \"PLACE\">30;");
476.  }
477.  else if(choice == 4){
478.  result = statement.executeQuery("SELECT DISTINCT \"NAME\" FROM \"" +
schemaName + "\".\"CONDUCTOR\";");
479.  }
480.  else if(choice == 5){
481.  result = statement.executeQuery("SELECT * FROM \"" + schemaName +
"\".\"PASSENGER&TRAIN\" OPTION(FAST) LIMIT 10");
482.  }
483.  else if(choice == 6){
484.  result = statement.executeQuery("SELECT *, \"SQUARE\"/\"COUNT_PATH\" AS
\"PATH_SQUARE\" FROM \"" + schemaName + "\".\"STATION\";");
485.  }
486.  else if(choice == 7){
487.  result = statement.executeQuery("SELECT \"NUMBER_PASSPOER\",
\"PERSONAL_NUMBER_FK\", CONCAT(\"SURNAME\", \'  \', \"NAME\", \' \',
\"MIDDLE_NAME\") AS \"FIO\" FROM \"" + schemaName + "\".\"PASSENGER\";");
488.  }
489.  else if(choice == 8){
490.  result = statement.executeQuery("SELECT * FROM \"" + schemaName +
"\".\"TRIP\" ORDER BY \"TIME\";");
491.  }
492.  else if(choice == 9){
493.  result = statement.executeQuery("SELECT * FROM \""+schemaName +
"\".\"TRAIN\" WHERE (\"FUEL_TYPE\" = \'Газ\' OR \"FUEL_TYPE\" =
\'Электричество\');");
494.  }
495.  else if(choice == 10){
496.  result = statement.executeQuery("SELECT * FROM \"" + schemaName +
"\".\"RAILWAY_CARRIAGE\" WHERE \"PLACE\" BETWEEN 30 AND 50;");
497.  choice = 3;
498.  }
499.  else if(choice == 11){
500.  result = statement.executeQuery("SELECT * FROM \"" + schemaName +
"\".\"CITY\" WHERE \"REGION\" IN (\'Могилевская область\', \'Брестская
область\');");
```

```
501.  }
502.  else if(choice == 12){
503.  result = statement.executeQuery("SELECT * FROM \"" + schemaName +
"\".\"STATION\" INNER JOIN \"" + schemaName + "\".\"CITY\" ON
\"STATION\".\"CITY_NAME_FK\" = \"CITY\".\"CITY_NAME\"; ");
504.  }
505.  else if(choice == 13){
506.  result = statement.executeQuery("SELECT * FROM \""+ schemaName +
"\".\"PASSENGER\" WHERE \"SURNAME\" LIKE \'C%\';");
507.  }
508.  else if(choice == 14){
509.  result = statement.executeQuery("SELECT \"PERSONAL_NUMBER_PK_FK\" FROM
\"" + schemaName + "\".\"TRIP\" WHERE EXTRACT(MONTH FROM \"DATA\") = 2 AND
\"TIME\" BETWEEN \'8:00:00\' AND \'19:00:00\';");
510.  }
511.  else if(choice == 15){
512.  result = statement.executeQuery("SELECT * FROM \"" + schemaName +
"\".\"RAILWAY_CARRIAGE\" WHERE \"PLACE\" > 40 ORDER BY \"PLACE\" DESC;");
513.  choice = 3;
514.  }
515.  else if(choice == 16){
516.  result = statement.executeQuery("SELECT AVG(\"COUNT_PATH\") AS
\"count_path_average\", AVG(\"SQUARE\") AS \"square_average\" FROM \"" +
schemaName + "\".\"STATION\";");
517.  }
518.  else if(choice == 17){
519.  result = statement.executeQuery("SELECT COUNT(*) FROM \"" + schemaName
+ "\".\"RAILWAY_CARRIAGE\" WHERE \"PLACE\" > 30;");
520.  }
521.  else if(choice == 18){
522.  result = statement.executeQuery("SELECT MAX(\"COUNT_PATH\") AS
\"MAX_COUNT_PATH\", MAX(\"SQUARE\") AS \"MAX_SQUARE\" FROM \"" + schemaName +
"\".\"STATION\";");
523.  }
524.  else if(choice == 19){
525.  result = statement.executeQuery("SELECT MIN(\"PERSONAL_NUMBER\") AS
\"MIN_PERSONAL_NUMBER\", MIN(LENGTH(\"NAME\")) AS \"MIN_LENGTH_NAME\",
MIN(LENGTH(\"SURNAME\")) AS \"MIN_LENGTH_SURNAME\" FROM \"" + schemaName +
"\".\"CONDUCTOR\";");
526.  }
527.  else if(choice == 20){
528.  result = statement.executeQuery("SELECT SUM(\"SQUARE\") AS
\"SUM_SQUARE\", SUM(\"COUNT_PATH\") AS \"SUM_COUNT_PATH\" FROM \"" +
schemaName + "\".\"STATION\";" );
529.  }
530.  else if(choice == 21){
531.  result = statement.executeQuery("SELECT \"REGION\", COUNT(DISTINCT
\"CITY_NAME\") FROM \""+ schemaName + "\".\"CITY\" GROUP BY \"REGION\";");
532.  }
533.  else if(choice == 22){
534.  result = statement.executeQuery("SELECT \"PERSONAL_NUMBER_FK\",
COUNT(*) FROM \"" + schemaName + "\".\"PASSENGER\" GROUP BY
\"PERSONAL_NUMBER_FK\" HAVING COUNT(*) > 2;");
535.  }
536.  else if(choice == 23){
537.  result = statement.executeQuery("SELECT * FROM \"" + schemaName +
"\".\"CONDUCTOR\" WHERE \"PERSONAL_NUMBER\" > (SELECT
AVG(\"PERSONAL_NUMBER\") FROM \"" + schemaName + "\".\"CONDUCTOR\");");
538.  }
539.  else if(choice == 24){
540.  result = statement.executeQuery("SELECT percentile_cont(0.5) WITHIN
GROUP (ORDER BY \"COUNT_PATH\") AS \"MEDIAN_PATH\", percentile_cont(0.5)
```

```
WITHIN GROUP (ORDER BY \"SQUARE\") AS \"MEDIAN_SQUARE\" FROM \""+ schemaName
+ "\".\"STATION\";");
541.  }
542.  else if(choice == 25){
543.  result = statement.executeQuery("SELECT * FROM \""+ schemaName +
"\".\"TRIP\" WHERE \"DATA\" = ANY (SELECT \"DATA\" FROM \"" +schemaName +
"\".\"TRIP\" WHERE \"DATA\" > \'2023-04-04\');");
544.  choice = 8;
545.  }
546.  else if(choice == 26){
547.  result = statement.executeQuery("SELECT * FROM \"" + schemaName +
"\".\"PASSENGER\" WHERE \"PERSONAL_NUMBER_FK\" BETWEEN 500000 AND 800000 AND
EXISTS(SELECT * FROM \"" + schemaName + "\".\"PASSENGER\" WHERE \"SURNAME\"
LIKE \'Лебедев\');");
548.  choice = 13;
549.  }
550.  else if(choice == 27){
551.  result = statement.executeQuery("SELECT * FROM \"" + schemaName +
"\".\"TRIP\" WHERE \"DATA\" <> ALL (SELECT \"DATA\" FROM \"" +schemaName +
"\".\"TRIP\" WHERE \"DATA\" BETWEEN \'2023-03-05\' AND \'2023-09-12\');");
552.  choice = 8;
553.  }
554.  else if(choice == 28){
555.  result = statement.executeQuery("SELECT * FROM \"" + schemaName +
"\".\"RAILWAY_CARRIAGE\" WHERE \"TYPE_CARRIAGE\" = \'Плацкарт\' AND\"PLACE\"
> 25 UNION SELECT * FROM \""+ schemaName + "\".\"RAILWAY_CARRIAGE\" WHERE
\"TYPE_CARRIAGE\" LIKE \'Купе\' AND \"PLACE\" > 20");
556.  choice = 3;
557.  }
558.  else if(choice == 29){
559.  result = statement.executeQuery("SELECT * FROM \"" + schemaName +
"\".\"TRIP\" INTERSECT SELECT * FROM \"" + schemaName + "\".\"TRIP\" WHERE
\"DATA\" NOT BETWEEN \'2023-03-01\' AND \'2023-04-01\';");
560.  choice = 8;
561.  }
562.  else if(choice == 30){
563.  result = statement.executeQuery("SELECT * FROM \"" + schemaName +
"\".\"RAILWAY_CARRIAGE\" WHERE \"PLACE\" > 20 EXCEPT SELECT * FROM \"" +
schemaName + "\".\"RAILWAY_CARRIAGE\" WHERE \"TYPE_CARRIAGE\" LIKE
\'Плацкарт\';");
564.  choice = 3;
565.  }
566.  labWorkShowRequest(choice, result);
567.  }

568.  public static void deleteRowTable(Statement statement, String
tableName) throws SQLException {
569.  Scanner scanner = new Scanner(System.in);
570.  if(Objects.equals(tableName, "CITY")){
571.  System.out.print("Введите название города для удаления: ");
572.  String cityName = scanner.nextLine();
573.  statement.executeUpdate("DELETE FROM \"" + schemaName + "\".\"" +
tableName + "\" WHERE \"CITY_NAME\" = \'" + cityName + "\';");
574.  }
575.  else if(Objects.equals(tableName, "CONDUCTOR")){
576.  System.out.print("Введите персональный номер проводника: ");
577.  int personalNumber = scanner.nextInt();
578.  statement.executeUpdate("DELETE FROM \"" + schemaName + "\".\"" +
tableName + "\" WHERE \"PERSONAL_NUMBER\" = " + personalNumber + ";");
579.  }
580.  else if(Objects.equals(tableName, "PASSENGER")){
581.  System.out.print("Введите номер паспорта: ");
```

```
582.  String numberPassport = scanner.nextLine();
583.  statement.executeUpdate("DELETE FROM \"" + schemaName + "\".\"" +
tableName + "\" WHERE \"NUMBER_PASSPOER\" = \'" + numberPassport + "\';");
584.  }
585.  else if(Objects.equals(tableName, "PASSENGER&TRAIN")){
586.  System.out.print("Введите номер паспорта: ");
587.  String numberPassport = scanner.nextLine();
588.  statement.executeUpdate("DELETE FROM \"" + schemaName + "\".\"" +
tableName + "\" WHERE \"NUMBER_PASSPORT_FK_PK\" = \'" + numberPassport +
"\';");
589.  }
590.  else if(Objects.equals(tableName, "RAILWAY_CARRIAGE")){
591.  System.out.print("Введите номер вагона: ");
592.  int numberRailwayCarriage = scanner.nextInt();
593.  statement.executeUpdate("DELETE FROM \"" + schemaName + "\".\"" +
tableName + "\" WHERE \"RAILWAY_CARRIAGE_ID\" = " + numberRailwayCarriage +
";");
594.  }
595.  else if(Objects.equals(tableName, "STATION")){
596.  System.out.print("Введите номер станции: ");
597.  int numberStation = scanner.nextInt();
598.  statement.executeUpdate("DELETE FROM \"" + schemaName + "\".\"" +
tableName +"\" WHERE \"STATION_ID\" = " + numberStation + ";");
599.  }
600.  else if(Objects.equals(tableName, "TRAIN")){
601.  System.out.print("Введите номер поезда: ");
602.  int numberTrain = scanner.nextInt();
603.  statement.executeUpdate("DELETE FROM \"" + schemaName + "\".\"" +
tableName + "\" WHERE \"TRAIN_ID\" = " + numberTrain + ";");
604.  }
605.  else if(Objects.equals(tableName, "TRIP")){
606.  System.out.print("Введите номера поезда: ");
607.  int numberTrain = scanner.nextInt();
608.  statement.executeUpdate("DELETE FROM \"" + schemaName + "\".\"" +
tableName + "\" WHERE \"TRAIN_ID\" = " + numberTrain + ";");
609.  }
610.  }

611.  public static void changeRowsInformation(Statement statement, String
tableName) throws SQLException {
612.  Scanner scanner = new Scanner(System.in);
613.  if(Objects.equals(tableName, "CITY")){
614.  System.out.print("Введите название города: ");
615.  String cityName = scanner.nextLine();
616.  System.out.print("Введите новый регион: ");
617.  String REGION = scanner.nextLine();
618.  System.out.print("Введите новую страну: ");
619.  String COUNTRY = scanner.nextLine();
620.  statement.executeUpdate("UPDATE \"" + schemaName + "\".\"" + tableName
+ "\" SET \"REGION\" = \'" + REGION + "\', \"COUNTRY\" = \'" + COUNTRY + "\'
WHERE \"" + "CITY_NAME" + "\" = \'" + cityName + "\';");
621.  }
622.  else if(Objects.equals(tableName, "CONDUCTOR")){
623.  System.out.print("Введите персональный номер проводника: ");
624.  int personalNumber = scanner.nextInt();
625.  System.out.print("Введите имя:");
626.  String NAME = scanner.nextLine();
627.  System.out.print("Введите фамилию: ");
628.  String SURNAME = scanner.nextLine();
629.  System.out.print("Введите отчество: ");
630.  String MIDDLE_NAME = scanner.nextLine();
```

```
631.  statement.executeUpdate("UPDATE \"" + schemaName + "\".\"" + tableName
+ "\" SET \"NAME\" = \'" + NAME + "\', SET \"SURNAME\" = \'" + SURNAME + "\'
SET \"MIDDLE_NAME\" = \'" + MIDDLE_NAME + "\' WHERE \"" + "PESONAL_NUMBER" +
"\" = " + personalNumber + ";");
632.  }
633.  else if(Objects.equals(tableName, "PASSENGER")){
634.  System.out.print("Введите номер паспорта: ");
635.  String numberPassport = scanner.nextLine();
636.  System.out.print("Введите имя:");
637.  String NAME = scanner.nextLine();
638.  System.out.print("Введите фамилию: ");
639.  String SURNAME = scanner.nextLine();
640.  System.out.print("Введите отчество: ");
641.  String MIDDLE_NAME = scanner.nextLine();
642.  System.out.print("Введите персональный номер проводника: ");
643.  int PERSONAL_NUMBER_FK = scanner.nextInt();
644.  statement.executeUpdate("UPDATE \"" +schemaName + "\".\"" + tableName +
"\" SET \"SURNAME\" = \'" + SURNAME + "\', SET \"NAME\" = \'" + NAME + "\',
SET \"MIDDLE_NAME\" = \'" + MIDDLE_NAME + "\', SET \"PERSONAL_NUMBER_FK\" = "
+ PERSONAL_NUMBER_FK + " WHERE \"NUMBER_PASSPOER\" = \'" + numberPassport +
"\';");

645.  }
646.  else if(Objects.equals(tableName, "PASSENGER&TRAIN")){
647.  System.out.print("Введите номер паспорта: ");
648.  String numberPassport = scanner.nextLine();
649.  System.out.print("Введите номер вагона: ");
650.  int trainId = scanner.nextInt();
651.  statement.executeUpdate("UPDATE \"" + schemaName +
"\".\"PASSENGER&TRAIN\" SET \"TRAIN_ID_PK_FK\" = " + trainId +" WHERE
\"NUMBER_PASSPORT_FK_PK\" = \'" + numberPassport + "\';" );
652.  }
653.  else if(Objects.equals(tableName, "RAILWAY_CARRIAGE")){
654.  System.out.print("Введите номер вагона: ");
655.  int numberRailwayCarriage = scanner.nextInt();
656.  System.out.print("Введите количество мест");
657.  int PLACE = scanner.nextInt();
658.  System.out.print("Введите номер вагона: ");
659.  int number_carriage = scanner.nextInt();
660.  System.out.print("Введите тип вагона: ");
661.  String typeCarriage = scanner.nextLine();
662.  System.out.print("Введите номер поезда к которому он относится: ");
663.  int numberTrain = scanner.nextInt();
664.  statement.executeUpdate("UPDATE \"" + schemaName + "\".\"" + tableName
+ "\" SET \"PLACE\" = " + PLACE + " ,SET \"NUMBER_CARRIAGE\" = " +
number_carriage + " ,SET \"TYPE_CARRIAGE\" = \'" + typeCarriage + "\' ,SET
\"TRAIN_ID_FK\" = " + numberTrain + " WHERE \"RAILWAY_CARRIAGE_ID\" = " +
numberRailwayCarriage +";");
665.  }
666.  else if(Objects.equals(tableName, "STATION")){
667.  System.out.print("Введите номер станции: ");
668.  int numberStation = scanner.nextInt();
669.  System.out.print("Введите название станции: ");
670.  String stationName = scanner.nextLine();
671.  System.out.print("Введите количество путей: ");
672.  int countPath = scanner.nextInt();
673.  System.out.print("Введите площадь станции: ");
674.  float square = scanner.nextFloat();
675.  System.out.print("Введите город: ");
676.  String cityName = scanner.nextLine();
677.  statement.executeUpdate("UPDATE \"" + schemaName + "\".\"" + tableName
+ "\" SET \"STATION_NAME\" = \'" + stationName + "\', SET \"COUNT_PATH\" = "
```

```
+ countPath + ", SET \"SQUARE\" = " + square +" ,SET \"CITY_NAME_FK\" = \'" +
cityName + "\' WHERE \"STATION_ID\" = " + numberStation + ";");
678.  }
679.  else if(Objects.equals(tableName, "TRAIN")){
680.  System.out.print("Введите номер поезда: ");
681.  int numberTrain = scanner.nextInt();
682.  System.out.print("Введите станцию отправления: ");
683.  String stationDeparture = scanner.nextLine();
684.  System.out.println("Введите станцию прибытия: ");
685.  String stationArrival = scanner.nextLine();
686.  System.out.print("Введите тип топлива: ");
687.  String fuelType = scanner.nextLine();
688.  System.out.println("Введите станцию: ");
689.  int stationId = scanner.nextInt();
690.  statement.executeUpdate("UPDATE \"" + schemaName + "\".\""+ tableName +
"\" SET \"STATION_DEPARTURE\" = \'" + stationDeparture + "\', SET
\"STATION_ARRIVAL\" = \'" + stationArrival + "\', SET \"FUEL_TYPE\" = \'" +
fuelType + "\', SET \"STATION_ID_FK\" = " + stationId + " WHERE \"TRAIN_ID\"
= " + numberTrain + ";");
691.  }
692.  else if(Objects.equals(tableName, "TRIP")){
693.  System.out.print("Введите номера поезда: ");
694.  int numberTrain = scanner.nextInt();
695.  System.out.print("Введите дату поездки");
696.  String data = scanner.nextLine();
697.  System.out.print("Введите время поездки:");
698.  String time = scanner.nextLine();
699.  statement.executeUpdate("UPDATE \"" + schemaName + "\".\"" + tableName
+ "\" SET \"DATA\" = \'" + data + "\', SET \"TIME\" = \'" + time + "\' WHERE
\"TRAIN_ID_PK_FK\" = " +numberTrain +";");
700.  }
701.  }
702.  public static void addRowsInTable(Statement statement, String
tableName) throws SQLException {
703.  Scanner scanner = new Scanner(System.in);
704.  if(Objects.equals(tableName, "CITY")){
705.  System.out.print("Введите название нового города: ");
706.  String cityName = scanner.nextLine();
707.  System.out.print("Введите новый регион: ");
708.  String REGION = scanner.nextLine();
709.  System.out.print("Введите новую страну: ");
710.  String COUNTRY = scanner.nextLine();
711.  statement.executeUpdate("INSERT INTO \"" + schemaName + "\".\"" +
tableName + "\" (\"CITY_NAME\", \"REGION\", \"COUNTRY\") VALUES (\'" +
cityName + "\', \'" + REGION + "\', \'" + COUNTRY + "\');");
712.  }
713.  else if(Objects.equals(tableName, "CONDUCTOR")){
714.  System.out.print("Введите персональный номер проводника: ");
715.  int personalNumber = scanner.nextInt();
716.  System.out.print("Введите имя:");
717.  String NAME = scanner.nextLine();
718.  System.out.print("Введите фамилию: ");
719.  String SURNAME = scanner.nextLine();
720.  System.out.print("Введите отчество: ");
721.  String MIDDLE_NAME = scanner.nextLine();
722.  statement.executeUpdate("INSERT INTO \"" + schemaName + "\".\"" +
tableName + "\" (\"PERSONAL_NUMBER\", \"SURNAME\", \"NAME\", \"MIDDLE_NAME\")
VALUES (" + personalNumber + ", \'" + SURNAME + "\', \'" + NAME + "\', \'" +
MIDDLE_NAME + "\');");
723.  }
724.  else if(Objects.equals(tableName, "PASSENGER")){
725.  System.out.print("Введите номер паспорта: ");
```

```
726.  String numberPassport = scanner.nextLine();
727.  System.out.print("Введите имя:");
728.  String NAME = scanner.nextLine();
729.  System.out.print("Введите фамилию: ");
730.  String SURNAME = scanner.nextLine();
731.  System.out.print("Введите отчество: ");
732.  String MIDDLE_NAME = scanner.nextLine();
733.  System.out.print("Введите персональный номер проводника: ");
734.  int PERSONAL_NUMBER_FK = scanner.nextInt();
735.  statement.executeUpdate("INSERT INTO \"" + schemaName + "\".\"" +
tableName + "\" (\"NUMBER_PASSPOER\", \"SURNAME\", \"NAME\", \"MIDDLE_NAME\",
\"PERSONAL_NUMBER_FK\") VALUES (\'" + numberPassport + "\', \'" + SURNAME +
"\', \'" + NAME + "\', \'" + MIDDLE_NAME + "\', " + PERSONAL_NUMBER_FK +
");");
736.  }
737.  else if(Objects.equals(tableName, "PASSENGER&TRAIN")){
738.  System.out.print("Введите номер паспорта: ");
739.  String numberPassport = scanner.nextLine();
740.  System.out.print("Введите номер вагона: ");
741.  int trainId = scanner.nextInt();
742.  statement.executeUpdate("INSERT INTO \"" + schemaName + "\".\"" +
tableName + "\" (\"NUMBER_PASSPORT_FK_PK\", \"TRAIN_ID_PK_FK\") VALUES (\'" +
numberPassport + "\', " + trainId + ");");
743.  }
744.  else if(Objects.equals(tableName, "RAILWAY_CARRIAGE")){
745.  System.out.print("Введите номер вагона: ");
746.  int numberRailwayCarriage = scanner.nextInt();
747.  System.out.print("Введите количество мест");
748.  int PLACE = scanner.nextInt();
749.  System.out.print("Введите номер вагона: ");
750.  int number_carriage = scanner.nextInt();
751.  System.out.print("Введите тип вагона: ");
752.  String typeCarriage = scanner.nextLine();
753.  System.out.print("Введите номер поезда к которому он относится: ");
754.  int numberTrain = scanner.nextInt();
755.  statement.executeUpdate("INSERT INTO \"" + schemaName + "\".\"" +
tableName + "\" (\"RAILWAY_CARRIAGE_ID\", \"PLACE\", \"NUMBER_CARRIAGE\",
\"TYPE_CARRIAGE\", \"TRAIN_ID_FK\") VALUES (" + numberRailwayCarriage + ", "
+ PLACE + ", " + number_carriage + ", \'" + typeCarriage + "\', " +
numberTrain + ");");
756.  }
757.  else if(Objects.equals(tableName, "STATION")){
758.  System.out.print("Введите номер станции: ");
759.  int numberStation = scanner.nextInt();
760.  System.out.print("Введите название станции: ");
761.  String stationName = scanner.nextLine();
762.  System.out.print("Введите количество путей: ");
763.  int countPath = scanner.nextInt();
764.  System.out.print("Введите площадь станции: ");
765.  float square = scanner.nextFloat();
766.  System.out.print("Введите город: ");
767.  String cityName = scanner.nextLine();
768.  statement.executeUpdate("INSERT INTO \"" + schemaName + "\".\"" +
tableName + "\" (\"STATION_ID\", \"COUNT_PATH\", \"SQUARE\",
\"STATION_NAME\", \"CITY_NAME_FK\") VALUES (" + numberStation + ", " +
countPath + ", " + square + ", \'" + stationName + "\', \'" + cityName +
"\');");
769.  }
770.  else if(Objects.equals(tableName, "TRAIN")){
771.  System.out.print("Введите номер поезда: ");
772.  int numberTrain = scanner.nextInt();
773.  System.out.print("Введите станцию отправления: ");
```

```
774.  String stationDeparture = scanner.nextLine();
775.  System.out.println("Введите станцию прибытия: ");
776.  String stationArrival = scanner.nextLine();
777.  System.out.print("Введите тип топлива: ");
778.  String fuelType = scanner.nextLine();
779.  System.out.println("Введите станцию: ");
780.  int stationId = scanner.nextInt();
781.  statement.executeUpdate("INSERT INTO \"" + schemaName + "\".\"" +
tableName + "\" (\"TRAIN_ID\", \"STATION_DEPARTURE\", \"STATION_ARRIVAL\",
\"FUEL_TYPE\", \"STATION_ID_FK\") VALUES (" + numberTrain + ", \'" +
stationDeparture + "\', \'" + stationArrival + "\', \'" + fuelType + "\', "+
stationId + ");");
782.  }
783.  else if(Objects.equals(tableName, "TRIP")){
784.  System.out.print("Введите номера поезда: ");
785.  int numberTrain = scanner.nextInt();
786.  System.out.print("Введите дату поездки");
787.  String data = scanner.nextLine();
788.  System.out.print("Введите время поездки:");
789.  String time = scanner.nextLine();
790.  System.out.print("Введите ID поезда: ");
791.  int trainID = scanner.nextInt();
792.  System.out.print("Введите ID вагона: ");
793.  int railwayID = scanner.nextInt();
794.  System.out.print("Введите персональный номер: ");
795.  int personalnumber = scanner.nextInt();
796.  statement.executeUpdate("INSERT INTO \"" + schemaName + "\".\"" +
tableName + "\" (\"TRAIN_ID_PK_FK\", \"RAILWAY_CARRIAGE_ID_PK_FK\",
\"PERSONAL_NUMBER_PK_FK\", \"DATA\", \"TIME\") VALUES (" +trainID+", " +
railwayID + ", " + personalnumber+", \'"+ data +"\', \'" +time +  "\');");
797.  }
798.  }
799.  }
```