Изображение выглядит как эмблема, символ, герб, нашивка

Автоматически созданное описание

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА - Российский технологический университет»**  **РТУ МИРЭА** |

**Институт** Искусственного Интеллекта (ИИИ)

**Кафедра** Промышленной Информатики

**ОТЧЁТ ПО ПРАКТИЧЕСКИМ РАБОТАМ**

**по дисциплине**

**«Разработка баз данных»**

Студент группы:ИКБО-04-22 \_\_Кликушин В. И.\_ *(Ф. И.О. студента)*

Преподаватель \_\_Зайцев И. Ю.\_\_

*(Ф.И.О. преподавателя)*

|  |  |  |
| --- | --- | --- |
|  |  |  |

Москва 2024

# СОДЕРЖАНИЕ

[1 ПРАКТИЧЕСКАЯ РАБОТА №1 5](#_Toc184815433)

[1.1 Создание базы данных 5](#_Toc184815434)

[1.2 Создание таблиц 6](#_Toc184815435)

[1.2.1 Таблица client 7](#_Toc184815436)

[1.2.2 Таблица supplier 7](#_Toc184815437)

[1.2.3 Таблица seller 8](#_Toc184815438)

[1.2.4 Таблица order\_status 9](#_Toc184815439)

[1.2.5 Таблица brand 10](#_Toc184815440)

[1.2.6 Таблица model 10](#_Toc184815441)

[1.2.7 Таблица color 11](#_Toc184815442)

[1.2.8 Таблица drive 12](#_Toc184815443)

[1.2.9 Таблица engine\_type 12](#_Toc184815444)

[1.2.10 Таблица engine 13](#_Toc184815445)

[1.2.11 Таблица car 14](#_Toc184815446)

[1.2.12 Таблица car\_order 15](#_Toc184815447)

[1.2.13 Таблица contract 16](#_Toc184815448)

[1.3 Заполнение таблиц 17](#_Toc184815449)

[1.4 Результат создания и заполнения базы данных 21](#_Toc184815450)

[2 ПРАКТИЧЕСКАЯ РАБОТА №2 22](#_Toc184815451)

[2.1 Выборка данных 22](#_Toc184815452)

[2.2 Выборка данных с сортировкой 27](#_Toc184815453)

[2.3 Операторы изменения данных 27](#_Toc184815454)

[3 ПРАКТИЧЕСКАЯ РАБОТА №3 29](#_Toc184815455)

[3.1 Резервная копия базы данных 29](#_Toc184815456)

[3.2 Выборка данных согласно операциям реляционной алгебры 29](#_Toc184815457)

[3.2.1 Операция проекции 29](#_Toc184815458)

[3.2.2 Операция селекции 31](#_Toc184815459)

[3.2.3 Операция соединения 32](#_Toc184815460)

[3.2.4 Операция объединения 35](#_Toc184815461)

[3.2.5 Операция пересечения 37](#_Toc184815462)

[3.2.6 Операция разности 39](#_Toc184815463)

[3.2.7 Операция группировки 40](#_Toc184815464)

[3.2.8 Операция сортировки 42](#_Toc184815465)

[3.2.9 Операция деления 43](#_Toc184815466)

[3.2.10 Создание представления 44](#_Toc184815467)

[3.3 Хранимые процедуры, функции, триггеры 44](#_Toc184815468)

[3.3.1 Процедуры 44](#_Toc184815469)

[3.3.2 Функции 46](#_Toc184815470)

[3.3.3 Триггеры 48](#_Toc184815471)

[4 ПРАКТИЧЕСКАЯ РАБОТА №4 49](#_Toc184815472)

[4.1 Агрегатные функции 49](#_Toc184815473)

[4.2 Ранжирующие функции 51](#_Toc184815474)

[4.3 Функции смещения 53](#_Toc184815475)

[ЗАКЛЮЧЕНИЕ 55](#_Toc184815476)

[ПРИЛОЖЕНИЯ 56](#_Toc184815477)

# 1 ПРАКТИЧЕСКАЯ РАБОТА №1

**Цель работы**: создать базу данных и таблицы в ней по выбранной теме на основе разработанных моделей.

**Выбранная тема**: «Автосалон».

На Рисунке 1.1 представлена физическая модель спроектированной базы данных.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 1.1 – Физическая модель базы данных

## 1.1 Создание базы данных

Запрос на создание базы данных с названием «auto\_show» с дальнейшим переключением в неё представлен в Листинге 1.1.1.

Листинг 1.1.1 – Запрос на создание базы данных и переключение в нее

CREATE DATABASE auto\_show;

\c auto\_show

На Рисунке 1.1.1 представлен результат отработки запроса.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1.1.1 – Создание базы данных

Дальнейшей подключение к созданной базе данных осуществляется согласно Рисунку 1.1.2.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1.1.2 – Подключение к созданной базе данных

После создания база данных не имеет никаких отношений (Рисунок 1.1.3).

Изображение выглядит как текст, снимок экрана, черный, Шрифт

Автоматически созданное описание

Рисунок 1.1.3 – Отсутствие отношений в созданной базе данных

Теперь в этой базе данных необходимо создать таблицы и заполнить их данными.

## 1.2 Создание таблиц

Полный скрипт создания таблиц представлен в Приложении А.1.

### 1.2.1 Таблица client

На Рисунке 1.2.1.1 представлен запрос на создание таблицы client (клиент).

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рисунок 1.2.1.1 – Запрос на создание таблицы client

Таблица содержит следующие поля:

* client\_id – уникальный идентификатор клиента, первичный ключ;
* firstname – имя клиента, строковый тип данных;
* surname – фамилия клиента, строковый тип данных;
* patronymic – отчество клиента, строковый тип данных;
* passport – паспорт клиента, представлен последовательностью из десяти цифр (четыре цифры серия и шесть цифр номер).

Структура таблицы client представлена на Рисунке 1.2.1.2.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1.2.1.2 - Структура таблицы client

### 1.2.2 Таблица supplier

На Рисунке 1.2.2.1 представлен запрос на создание таблицы supplier (поставщик).

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рисунок 1.2.2.1 – Запрос на создание таблицы supplier

Таблица содержит следующие поля:

* supplier\_id – уникальный идентификатор поставщика, первичный ключ;
* email – почта поставщика, строковый тип данных;
* country – страна поставщика, строковый тип данных.

Структура таблицы supplier представлена на Рисунке 1.2.2.2.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1.2.2.2 - Структура таблицы supplier

### 1.2.3 Таблица seller

На Рисунке 1.2.3.1 представлен запрос на создание таблицы seller (продавец).

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 1.2.3.1 – Запрос на создание таблицы seller

Таблица содержит следующие поля:

* seller\_id – уникальный идентификатор продавца, первичный ключ;
* firstname – имя продавца, строковый тип данных;
* surname – фамилия продавца, строковый тип данных;
* patronymic – отчество продавца, строковый тип данных;
* phone\_number – номер телефона продавца, строковый тип данных;
* email – почта продавца, строковый тип данных.

Структура таблицы seller представлена на Рисунке 1.2.3.2.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рисунок 1.2.3.2 - Структура таблицы seller

### 1.2.4 Таблица order\_status

На Рисунке 1.2.4.1 представлен запрос на создание таблицы order\_status (статус заказа).

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рисунок 1.2.4.1 – Запрос на создание таблицы order\_status

Таблица содержит следующие поля:

* status\_id – уникальный идентификатор статуса заказа, первичный ключ;
* status – статус заказа, строковый тип данных.

Структура таблицы order\_status представлена на Рисунке 1.2.4.2.

Изображение выглядит как текст, снимок экрана, черный, Шрифт

Автоматически созданное описание

Рисунок 1.2.4.2 - Структура таблицы order\_status

### 1.2.5 Таблица brand

На Рисунке 1.2.5.1 представлен запрос на создание таблицы brand (марка автомобиля).

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 1.2.5.1 – Запрос на создание таблицы brand

Таблица содержит следующие поля:

* brand\_id – уникальный идентификатор марки автомобиля, первичный ключ;
* brand – марка автомобиля, строковый тип данных.

Структура таблицы brand представлена на Рисунке 1.2.5.2.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рисунок 1.2.5.2 - Структура таблицы brand

### 1.2.6 Таблица model

На Рисунке 1.2.6.1 представлен запрос на создание таблицы model (модель автомобиля).

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рисунок 1.2.6.1 – Запрос на создание таблицы model

Таблица содержит следующие поля:

* model\_id – уникальный идентификатор модели автомобиля, первичный ключ;
* brand\_id – идентификатор марки автомобиля, внешний ключ;
* model – модель автомобиля, строковый тип данных.

Структура таблицы model представлена на Рисунке 1.2.6.2.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1.2.6.2 - Структура таблицы model

### 1.2.7 Таблица color

На Рисунке 1.2.7.1 представлен запрос на создание таблицы color (цвет).

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рисунок 1.2.7.1 – Запрос на создание таблицы color

Таблица содержит следующие поля:

* color\_id – уникальный идентификатор цвета, первичный ключ;
* color – цвет, строковый тип данных.

Структура таблицы color представлена на Рисунке 1.2.7.2.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рисунок 1.2.7.2 - Структура таблицы color

### 1.2.8 Таблица drive

На Рисунке 1.2.8.1 представлен запрос на создание таблицы drive (привод).

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рисунок 1.2.8.1 – Запрос на создание таблицы drive

Таблица содержит следующие поля:

* drive\_id – уникальный идентификатор привода, первичный ключ;
* drive\_type – привод, строковый тип данных.

Структура таблицы drive представлена на Рисунке 1.2.8.2.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рисунок 1.2.8.2 - Структура таблицы drive

### 1.2.9 Таблица engine\_type

На Рисунке 1.2.9.1 представлен запрос на создание таблицы engine\_type (тип двигателя).

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рисунок 1.2.9.1 – Запрос на создание таблицы engine\_type

Таблица содержит следующие поля:

* engine\_type\_id – уникальный идентификатор типа двигателя, первичный ключ;
* engine\_type – тип двигателя, строковый тип данных.

Структура таблицы engine\_type представлена на Рисунке 1.2.9.2.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рисунок 1.2.9.2 - Структура таблицы engine\_type

### 1.2.10 Таблица engine

На Рисунке 1.2.10.1 представлен запрос на создание таблицы engine (двигатель).

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рисунок 1.2.10.1 – Запрос на создание таблицы engine

Таблица содержит следующие поля:

* engine\_id – уникальный идентификатор двигателя, первичный ключ;
* engine\_power – мощность двигателя, целый тип данных;
* engine\_capacity – объем двигателя, целый тип данных;
* battery\_capacity – емкость батареи, целый тип данных;
* engine\_type\_id – идентификатор типа двигателя, внешний ключ.

Структура таблицы engine представлена на Рисунке 1.2.10.2.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рисунок 1.2.10.2 - Структура таблицы engine

### 1.2.11 Таблица car

На Рисунке 1.2.11.1 представлен запрос на создание таблицы car (автомобиль).

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рисунок 1.2.11.1 – Запрос на создание таблицы car

Таблица содержит следующие поля:

* car\_id – уникальный идентификатор автомобиля, первичный ключ;
* model\_id – идентификатор модели автомобиля, внешний ключ;
* drive\_id – идентификатор типа привода, внешний ключ;
* price – стоимость автомобиля, целый тип данных;
* mileage – пробег автомобиля в километрах, целый тип данных;
* vehicle\_year – год производства автомобиля, целый тип данных;
* color\_id – идентификатор цвета, внешний ключ;
* engine\_id – идентификатор двигателя, внешний ключ.

Структура таблицы car представлена на Рисунке 1.2.11.2.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1.2.11.2 - Структура таблицы car

### 1.2.12 Таблица car\_order

На Рисунке 1.2.12.1 представлен запрос на создание таблицы car\_order (заказ).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1.2.12.1 – Запрос на создание таблицы car\_order

Таблица содержит следующие поля:

* order\_id – уникальный идентификатор заказа, первичный ключ;
* status\_id – идентификатор статуса заказа, внешний ключ;
* order\_date – дата заказа, тип данных date;
* car\_id – идентификатор автомобиля, внешний ключ;
* client\_id – идентификатор клиента, внешний ключ;
* supplier\_id – идентификатор поставщика, внешний ключ.

Структура таблицы car\_order представлена на Рисунке 1.2.12.2.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1.2.12.2 - Структура таблицы car\_order

### 1.2.13 Таблица contract

На Рисунке 1.2.13.1 представлен запрос на создание таблицы contract (договор).

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рисунок 1.2.13.1 – Запрос на создание таблицы contract

Таблица содержит следующие поля:

* contract\_id – уникальный идентификатор договора, первичный ключ;
* seller\_id – идентификатор продавца, внешний ключ;
* order\_id – идентификатор заказа, внешний ключ.

Структура таблицы contract представлена на Рисунке 1.2.13.2.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рисунок 1.2.13.2 - Структура таблицы contract

## 1.3 Заполнение таблиц

Добавление записей в некоторые поля таблицы осуществляется, используя синтаксическую конструкцию, представленную в Листинге 1.3.1.

Листинг 1.3.1 – Синтаксическая конструкция для добавления записей в таблицу

INSERT INTO имя\_таблицы ('имя\_столбца', 'имя\_столбца') VALUES ('значение\_первого\_столбца','значение\_второго\_столбца');

Полный скрипт заполнения таблиц представлен в Приложении А.2.

Заполненная таблица client представлена на Рисунке 1.3.1.

Изображение выглядит как текст, снимок экрана, меню, Шрифт

Автоматически созданное описание

Рисунок 1.3.1 – Заполненная таблица client

Заполненная таблица supplier представлена на Рисунке 1.3.2.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рисунок 1.3.2 – Заполненная таблица supplier

Заполненная таблица seller представлена на Рисунке 1.3.3.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 1.3.3 – Заполненная таблица seller

Заполненная таблица order\_status представлена на Рисунке 1.3.4.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1.3.4 – Заполненная таблица order\_status

Заполненная таблица brand представлена на Рисунке 1.3.5.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1.3.5 – Заполненная таблица brand

Заполненная таблица model представлена на Рисунке 1.3.6.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 1.3.6 – Заполненная таблица model

Заполненная таблица color представлена на Рисунке 1.3.7.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рисунок 1.3.7 – Заполненная таблица color

Заполненная таблица drive представлена на Рисунке 1.3.8.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1.3.8 – Заполненная таблица drive

Заполненная таблица engine\_type представлена на Рисунке 1.3.9.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1.3.9 – Заполненная таблица engine\_type

Заполненная таблица engine представлена на Рисунке 1.3.10.

Изображение выглядит как текст, снимок экрана, диаграмма, черный

Автоматически созданное описание

Рисунок 1.3.10 – Заполненная таблица engine

Заполненная таблица car представлена на Рисунке 1.3.11.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1.3.11 – Заполненная таблица car

Заполненная таблица car\_order представлена на Рисунке 1.3.12.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1.3.12 – Заполненная таблица car\_order

Заполненная таблица contract представлена на Рисунке 1.3.13.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 1.3.13 – Заполненная таблица contract

## 1.4 Результат создания и заполнения базы данных

После проделанных запросов создана база данных auto\_show с 13 таблицами, каждая из которых была заполнена несколькими записями. Структура базы данных представлена на Рисунке 1.4.1.

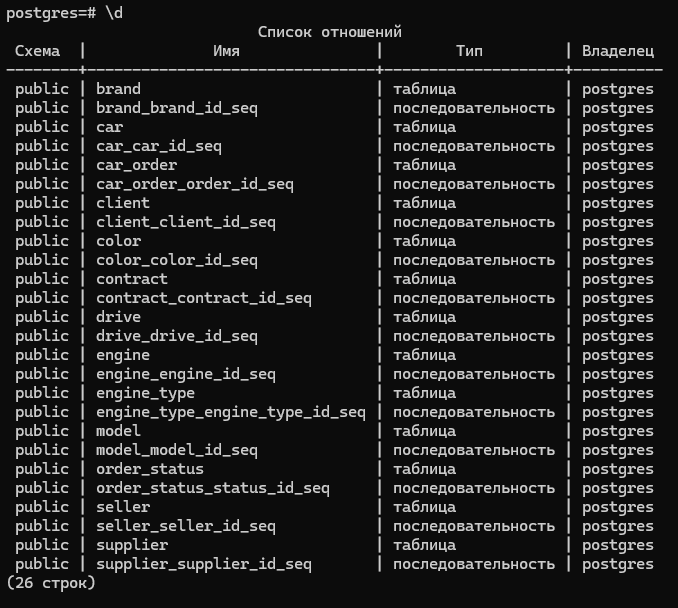


Рисунок 1.4.1 – Структура базы данных

# 2 ПРАКТИЧЕСКАЯ РАБОТА №2

**Цель работы**: изучить и создать выборку и сортировку данных, изучить и применить операторы для изменения данных в таблицах.

## 2.1 Выборка данных

Для просмотра записей в таблицах используется оператор SELECT.

Выполнены запросы на выборку данных с использованием оператора WHERE, использующие основные условия выбора.

Полный скрипт запросов, выполненных в рамках этой практической работы, представлен в Приложении Б.

1. **Оператор «=».**

На Рисунке 2.1.1 представлен запрос на выборку записей из таблицы client, у которых значение поля client\_id равно 2.

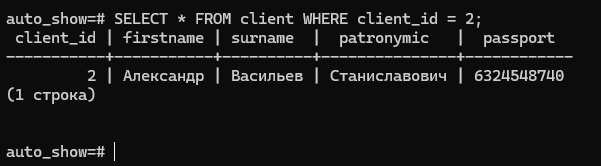


Рисунок 2.1.1 – Запрос на выборку с оператором «=»

1. **Оператор «!=».**

На Рисунке 2.1.2 представлен запрос на выборку записей из таблицы engine, у которых значение поля engine\_power не равно 1500.

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Автоматически созданное описание

Рисунок 2.1.2 – Запрос на выборку с оператором «!=»

1. **Оператор «>».**

На Рисунке 2.1.3 представлен запрос на выборку записей из таблицы supplier, у которых значение поля supplier\_id больше 3.

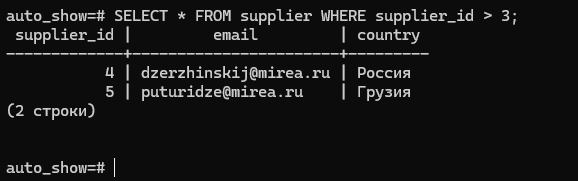


Рисунок 2.1.3 – Запрос на выборку с оператором «>»

1. **Оператор «>=».**

На Рисунке 2.1.4 представлен запрос на выборку записей из таблицы client, у которых значение поля passport больше или равно 6324500000.

Изображение выглядит как текст, снимок экрана, меню, Шрифт

Автоматически созданное описание

Рисунок 2.1.4 – Запрос на выборку с оператором «>=»

1. **Оператор «<».**

На Рисунке 2.1.5 представлен запрос на выборку записей из таблицы car, у которых значение поля price меньше 40000000.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рисунок 2.1.5 – Запрос на выборку с оператором «<»

1. **Оператор «<=».**

На Рисунке 2.1.6 представлен запрос на выборку записей из таблицы engine, у которых значение поля engine\_capacity меньше или равно 5,2.

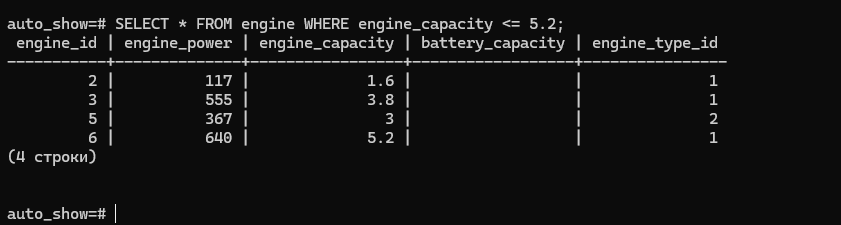


Рисунок 2.1.6 – Запрос на выборку с оператором «<=»

1. **Оператор «IS NOT NULL».**

На Рисунке 2.1.7 представлен запрос на выборку записей из таблицы engine, у которых значение поля engine\_capacity не пустое.

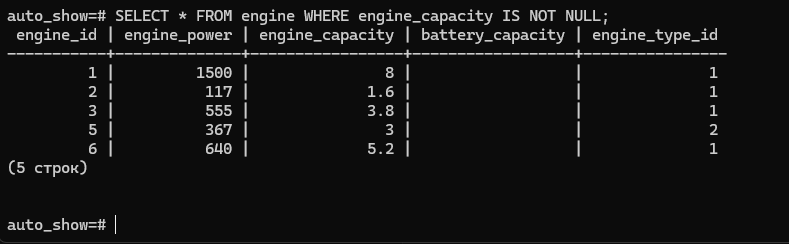


Рисунок 2.1.7 – Запрос на выборку с оператором «IS NOT NULL»

1. **Оператор «IS NULL».**

На Рисунке 2.1.8 представлен запрос на выборку записей из таблицы engine, у которых значение поля engine\_capacity пустое.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рисунок 2.1.8 - Запрос на выборку с оператором «IS NULL»

1. **Оператор «BETWEEN».**

На Рисунке 2.1.9 представлен запрос на выборку записей из таблицы car, у которых значение поля price находится в диапазоне от 1000000 до 20000000.

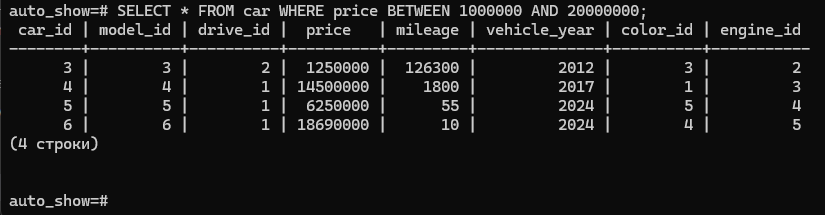


Рисунок 2.1.9 – Запрос на выборку с оператором «BETWEEN»

1. **Оператор «IN».**

На Рисунке 2.1.10 представлен запрос на выборку записей из таблицы car\_order, у которых значение поля status\_id равно 1 или 2.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рисунок 2.1.10 - Запрос на выборку с оператором «IN»

1. **Оператор «NOT IN».**

На Рисунке 2.1.11 представлен запрос на выборку записей из таблицы seller, у которых значение поля seller\_id не равно 2, 4 или 6.

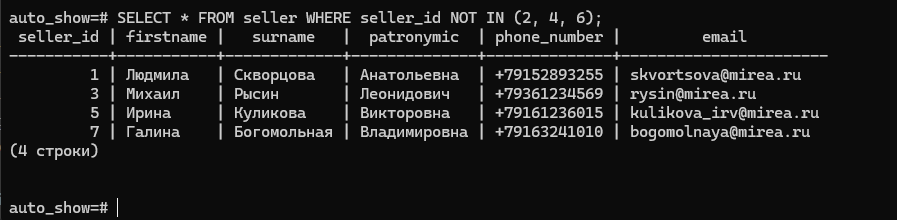


Рисунок 2.1.11 – Запрос на выборку с оператором «NOT IN»

1. **Оператор «LIKE».**

На Рисунке 2.1.12 представлен запрос на выборку записей из таблицы client, у которых значение поля surname начинается с буквы «К».

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 2.1.12 – Запрос на выборку с оператором «LIKE»

1. **Оператор «NOT LIKE».**

На Рисунке 2.1.13 представлен запрос на выборку записей из таблицы seller, у которых значение поля patronymic не содержит сочетания букв «вн».

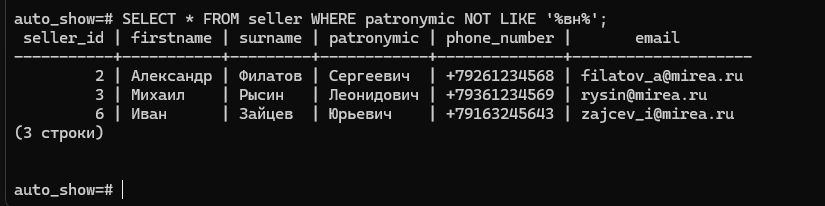


Рисунок 2.1.13 – Запрос на выборку с оператором «NOT LIKE»

## 2.2 Выборка данных с сортировкой

На Рисунке 2.2.1 представлен запрос на выборку данных из таблицы car, отсортированных по возрастанию цены.

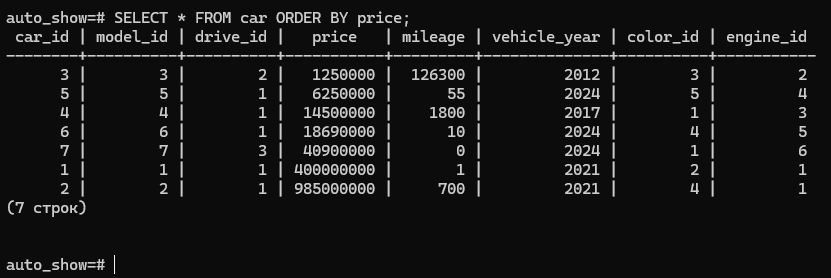


Рисунок 2.2.1 – Запрос на выборку с сортировкой по возрастанию

На Рисунке 2.2.2 представлен запрос на выборку данных из таблицы car\_order, отсортированных по убыванию даты покупки.

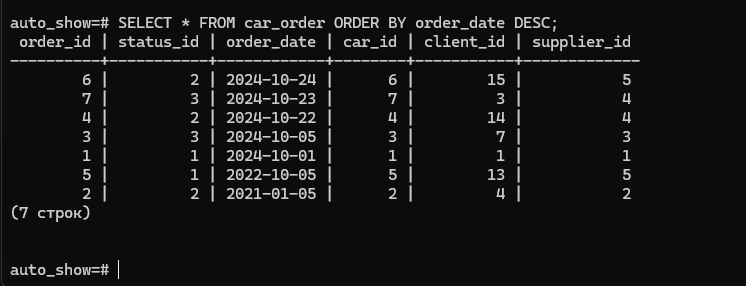


Рисунок 2.2.2 – Запрос на выборку с сортировкой по убыванию

## 2.3 Операторы изменения данных

На Рисунке 2.3.1 представлен запрос на добавление дополнительного поля number\_sold\_cars (тип - SMALLINT), отвечающего за количество проданных автомобилей, в таблицу seller.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 2.3.1 – Запрос на добавление поля в таблицу

На Рисунке 2.3.2 представлен запрос на обновления поля surname в таблице client для клиента с идентификатором, равным 14.

Изображение выглядит как текст, меню, снимок экрана

Автоматически созданное описание

Рисунок 2.3.2 – Запрос на обновление данных в таблице

# 3 ПРАКТИЧЕСКАЯ РАБОТА №3

**Цель работы**: ознакомиться с тем, как выполняется перенос базы данных на другой сервер и её бэкап; выполнить запросы выборки, реализующие различные операции реляционной алгебры; создать хранимые процедуры, функции и триггеры.

## 3.1 Резервная копия базы данных

Резервная копия базы данных создаётся с помощью команды, представленной в Листинге 3.1.1.

Листинг 3.1.1. – Команда на создание резервной копии базы данных

pg\_dump -d auto\_show -U postgres -f pract\_backup.sql

Параметры:

* auto\_show – название базы данных;
* postgres – имя пользователя;
* pract\_backup.sql – указанный файл, куда будет отправлен вывод.

Содержимое файла pract\_backup.sql представлено в Приложении В.1.

## 3.2 Выборка данных согласно операциям реляционной алгебры

Выполненные запросы выборки, демонстрирующие различные операции реляционной алгебры представлены в Приложении В.2.

### 3.2.1 Операция проекции

Осуществляется выбор только части полей таблицы, т.е. производится вертикальная выборка данных.

На Рисунке 3.2.1.1 представлен запрос на выборку названий всех автомобильных брендов из таблицы brand.

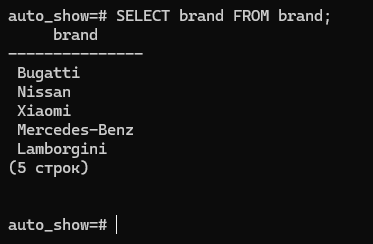


Рисунок 3.2.1.1 – Запрос на выборку всех названий брендов

На Рисунке 3.2.1.2 представлен запрос на выборку всех фамилий клиентов.

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание

Рисунок 3.2.1.2 – Запрос на выборку всех фамилий клиентов

На Рисунке 3.2.1.3 представлен запрос на выборку всех цен автомобилей.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.2.1.3 – Запрос на выборку всех цен автомобилей

### 3.2.2 Операция селекции

Осуществляется горизонтальная выборка – в результат попадают только записи, удовлетворяющие условию.

На Рисунке 3.2.2.1 представлен запрос на выборку записей из таблицы seller, у которых фамилия продавца начинается с буквы «З».

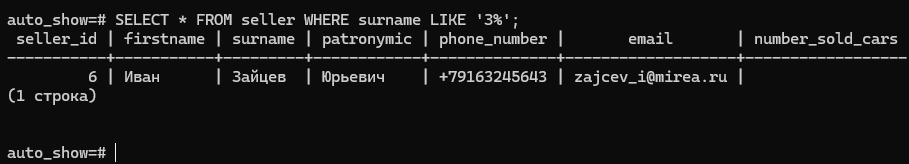


Рисунок 3.2.2.1 – Запрос на выборку записей из таблицы seller с условием на фамилию продавца

На Рисунке 3.2.2.2 представлен запрос на выборку записей из таблицы car, у которых стоимость автомобиля превышает 5000000.

Изображение выглядит как текст, снимок экрана, черный, Шрифт

Автоматически созданное описание

Рисунок 3.2.2.2 – Запрос на выборку записей из таблицы car с условием на цену автомобиля

На Рисунке 3.2.2.3 представлен запрос на выборку записей из таблицы supplier, у которых значение поля country равно «Россия».

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.2.2.3 – Запрос на выборку записей из таблицы supplier с условием на страну

### 3.2.3 Операция соединения

Выделяется декартово произведение и на его основе соединение по условию, а также естественное соединение (по одноименным полям или равенству полей с одинаковым смыслом).

На Рисунке 3.2.3.1 представлен запрос, который демонстрирует соединение таблиц car\_order и client с условием, что client\_id из car\_order совпадает с client\_id из client.

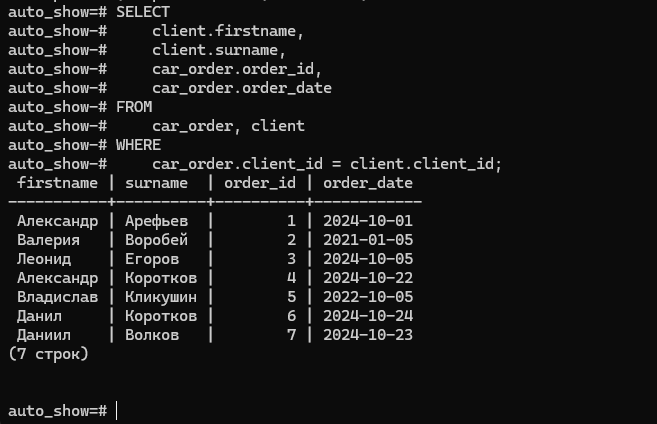


Рисунок 3.2.3.1 – Запрос, реализующий операцию соединения

Этот же запрос с применением оператора INNER JOIN представлен на Рисунке 3.2.3.2.

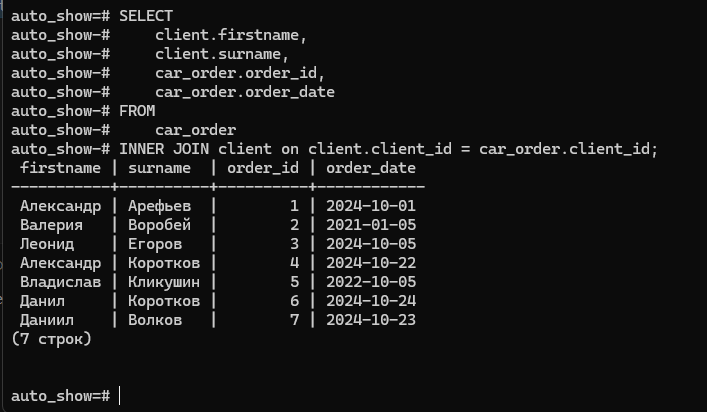


Рисунок 3.2.3.2 – Запрос, реализующий операцию соединения с использованием оператора INNER JOIN

На Рисунке 3.2.3.3 представлен запрос, который реализует естественное соединение таблиц car и car\_order по общему полю.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 3.2.3.3 – Запрос, реализующий операцию естественного соединения

На Рисунке 3.2.3.4 представлен запрос, который выполняет условное соединение трех таблицы: car, model, color по соответствующим полям и фильтрует автомобили по оранжевому цвету.

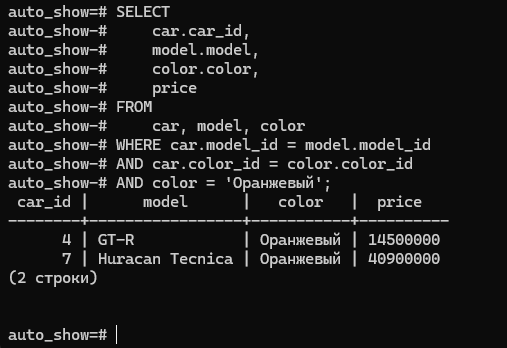


Рисунок 3.2.3.4 – Запрос, реализующий операцию соединения трех таблиц

Этот же запрос с применением операторов INNER JOIN, NATURAL JOIN представлен на Рисунке 3.2.3.5.

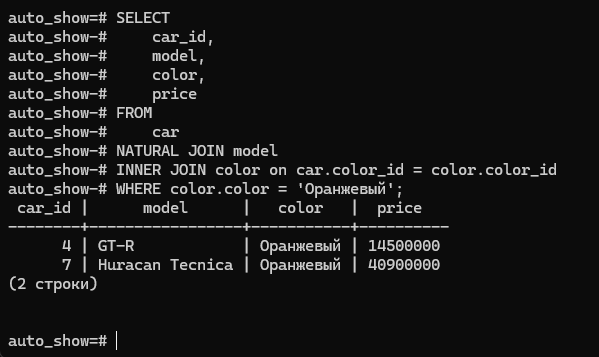


Рисунок 3.2.3.5 – Запрос, реализующий операцию соединения трех таблиц с использованием оператором INNER JOIN, NATURAL JOIN

### 3.2.4 Операция объединения

На Рисунке 3.2.4.1 представлен запрос, который объединяет заказы с состоянием «Ожидается» и «Отправлен». Объединение задано с помощью логического оператора «OR».

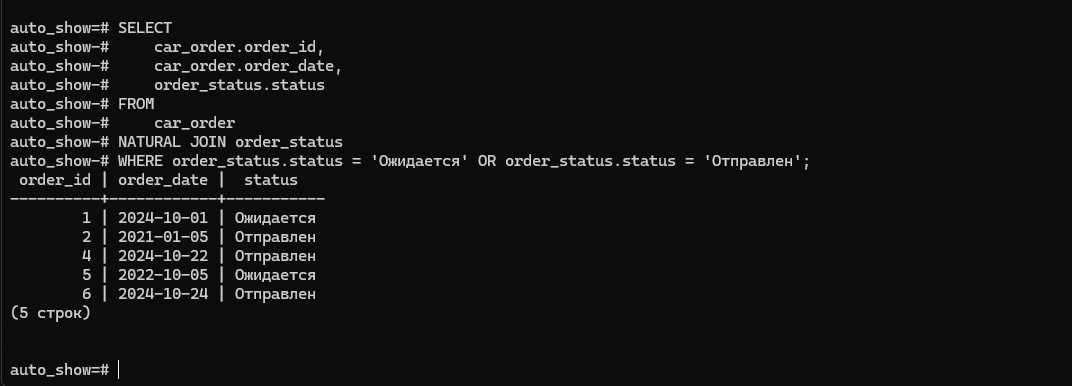


Рисунок 3.2.4.1 – Запрос, реализующий операцию объединения с использованием логического оператора OR

Этот же запрос с использованием оператора UNION представлен на Рисунке 3.2.4.2.

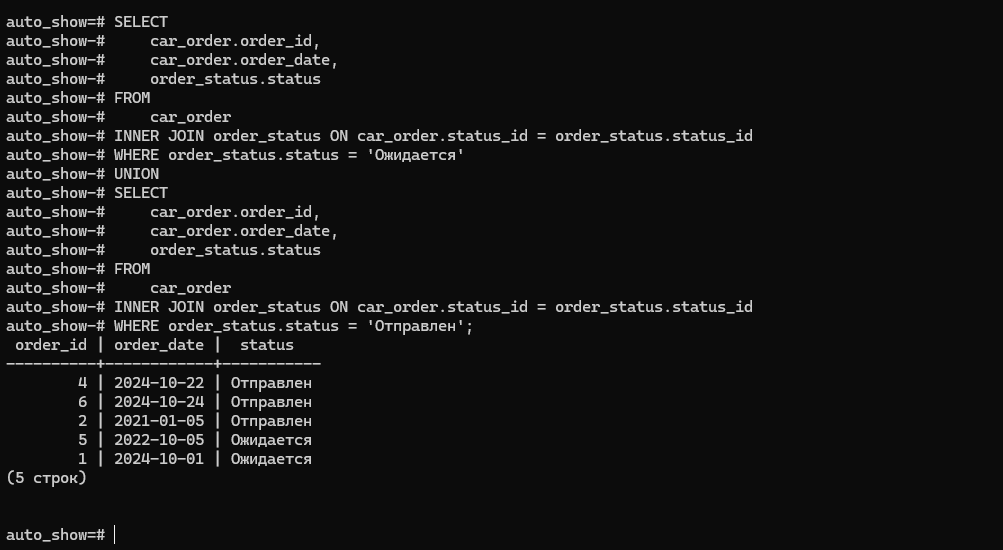


Рисунок 3.2.4.2 - Запрос, реализующий операцию объединения с использованием оператора UNION

На Рисунке 3.2.4.3 представлен запрос, который объединяет заказы с автомобилями различных брендов.

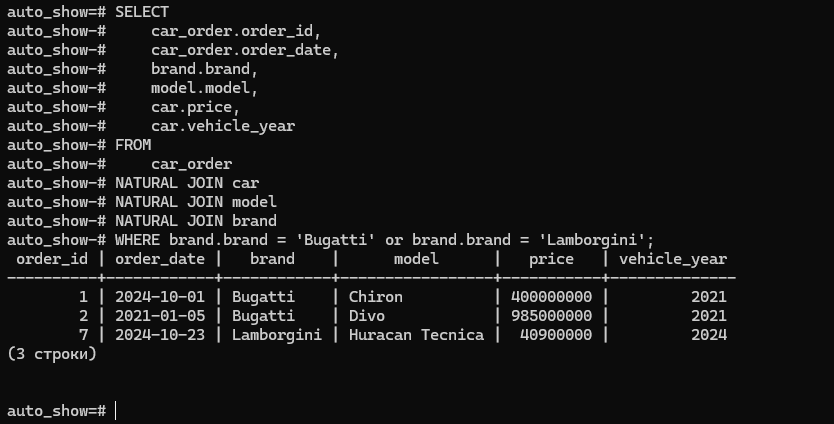


Рисунок 3.2.4.3 – Запрос, реализующий операцию объединения с использованием логического оператора OR

Запрос переписан с использованием оператора UNION (Рисунок 3.2.4.4).



Рисунок 3.2.4.4 - Запрос, реализующий операцию объединения с использованием оператора UNION

### 3.2.5 Операция пересечения

В простых случаях эту операцию можно описать с помощью логической операции AND. В более сложных случаях эта операция определяется чаще всего с помощью подзапроса и ключевого слова EXISTS, которое показывает наличие похожего элемента во множестве, которое задается подзапросом.

На Рисунке 3.2.5.1 представлен запрос, который извлекает клиентов, у которых есть хотя бы один заказ со статусом «Ожидается» и хотя бы один заказ со статусом «Доставлен».

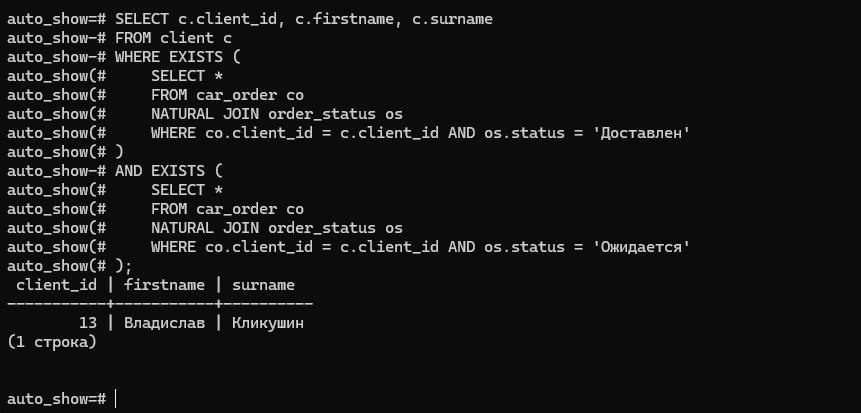


Рисунок 3.2.5.1 – Запрос, реализующий операцию пересечения

Запрос переписан с использованием оператора INTERSECT (Рисунок 3.2.5.2).

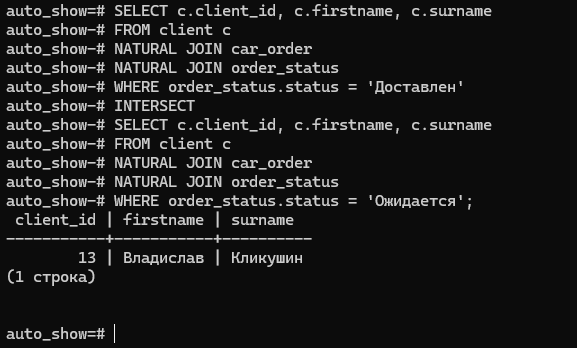


Рисунок 3.2.5.2 - Запрос, реализующий операцию пересечения с использованием оператора INTERSECT

На Рисунке 3.2.5.3 представлен запрос, который возвращает продавцов, одновременно продавших автомобили мощностью менее 700 лошадиных сил и более 700 лошадиных сил.

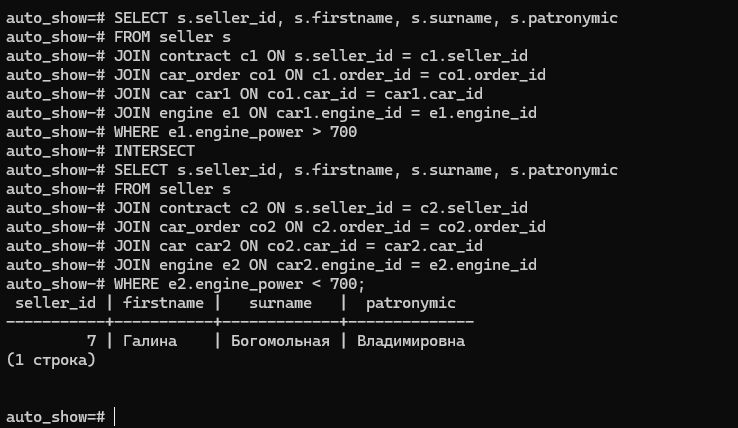


Рисунок 3.2.5.3 - Запрос, реализующий операцию пересечения с использованием оператора INTERSECT

Этот запрос с использованием функции EXISTS представлен на Рисунке 3.2.5.4.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.2.5.4 - Запрос, реализующий операцию пересечения

### 3.2.6 Операция разности

На Рисунке 3.2.6.1 представлен запрос, который находит всех продавцов, заключивших договор на продажу автомобиля в 2024 году, но не заключавших таких договоров в 2023 году.

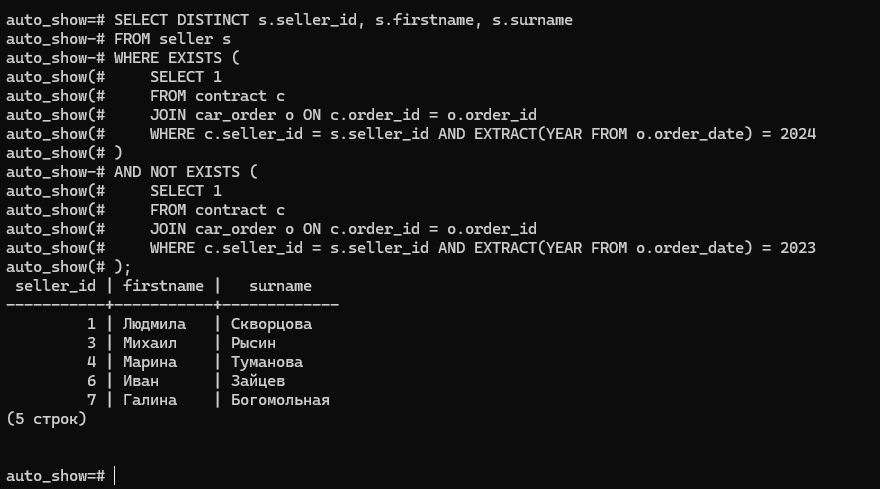


Рисунок 3.2.6.1 – Запрос, реализующий операцию разности

Этот запрос с использованием оператора EXCEPT представлен на Рисунке 3.2.6.2.

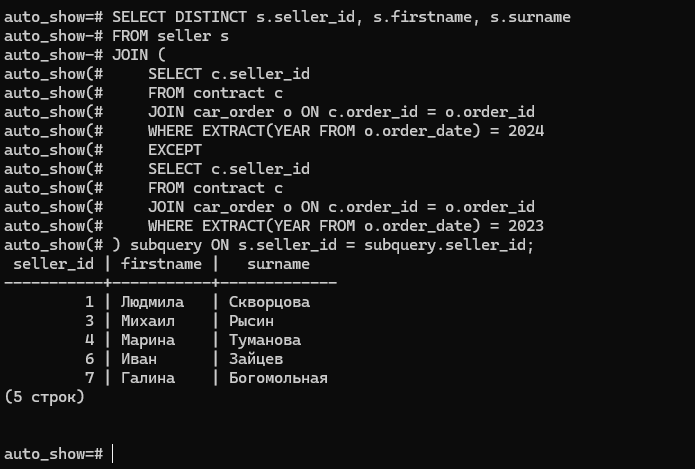


Рисунок 3.2.6.2 - Запрос, реализующий операцию разности с использованием оператора EXCEPT

### 3.2.7 Операция группировки

Эта операция связана со своеобразной сверткой таблицы по полям группировки. Помимо полей группировки результат запроса может содержать итоговые агрегирующие функции по группам (COUNT, SUM, AVG, MAX, MIN).

На Рисунке 3.2.7.1 представлен запрос, находящий суммарную стоимость всех автомобилей, сгруппированных по бренду.

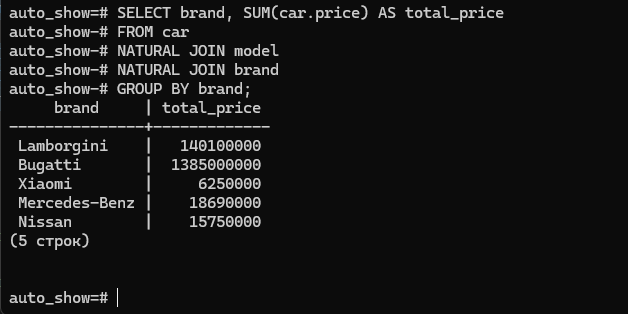


Рисунок 3.2.7.1 – Запрос, реализующий операцию группировки с использованием агрегирующей функции SUM

На Рисунке 3.2.7.2 представлен запрос, который вычисляет средний пробег автомобилей каждого года выпуска.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.2.7.2 – Запрос, реализующий операцию группировки с использованием агрегирующей функции AVG

На Рисунке 3.2.7.3 представлен запрос, который вычисляет количество проданных автомобилей каждым продавцом.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.2.7.3 – Запрос, реализующий операцию группировки с использованием агрегирующей функции COUNT

На Рисунке 3.2.7.4 представлен запрос, который реализует группировку по модели, чтобы отобразить максимальную и минимальную стоимость автомобилей этой модели.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.2.7.4 – Запрос, реализующий операцию группировки с использованием агрегирующих функций MIN, MAX

### 3.2.8 Операция сортировки

На Рисунке 3.2.8.1 представлен запрос, который считает количество брендов для каждой модели и сортирует бренды по количеству моделей в порядке убывания.

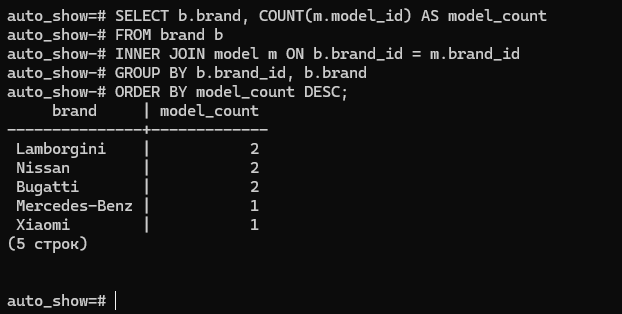


Рисунок 3.2.8.1 – Запрос, реализующий операцию сортировки по убыванию

На Рисунке 3.2.8.2 представлен запрос, который сортирует автомобили по мощности двигателя и пробегу.

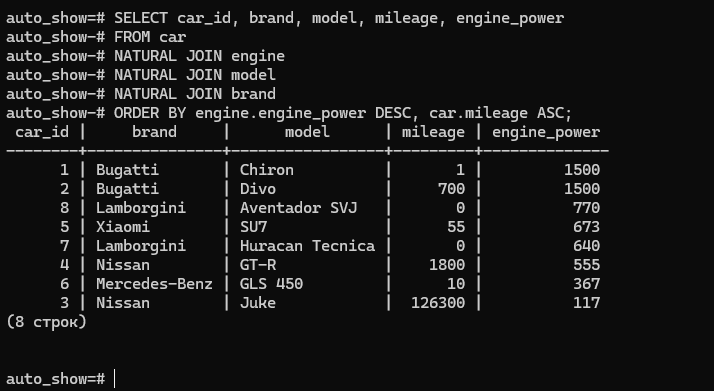


Рисунок 3.2.8.2 - Запрос, реализующий операцию сортировки по убыванию

На Рисунке 3.2.8.3 представлен запрос, реализующий сортировку поставщиков по количеству отправленных и доставленных автомобилей.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.2.8.3 - Запрос, реализующий операцию сортировки по убыванию

### 3.2.9 Операция деления

Это самая нетривиальная операция реляционной алгебры, которая обычно применяется тогда, когда требуется найти все записи первой таблицы, которые соединяются естественным образом со всеми записями второй таблицы.

На Рисунке 3.2.9.1 представлен запрос, который находит поставщиков, предоставивших автомобили со всеми типами двигателя. В Таблицы были внесены дополнительные записи.

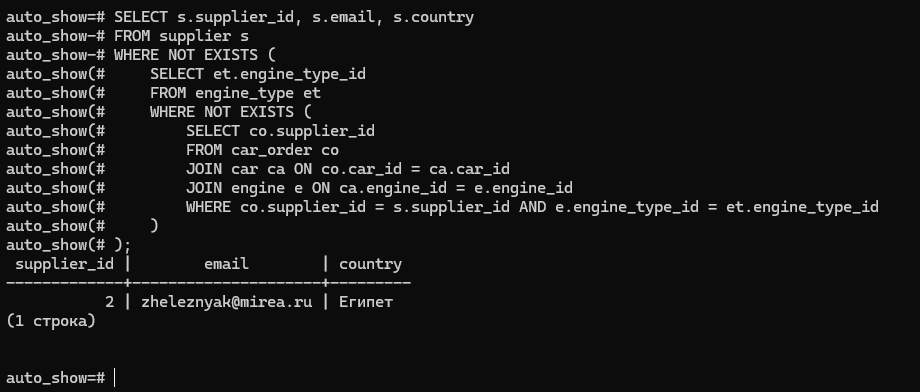


Рисунок 3.2.9.1 – Запрос, реализующий операцию деления

### 3.2.10 Создание представления

Создано представление, которое хранит информацию о клиенте и заказе автомобиля (Рисунок 3.2.10.1).

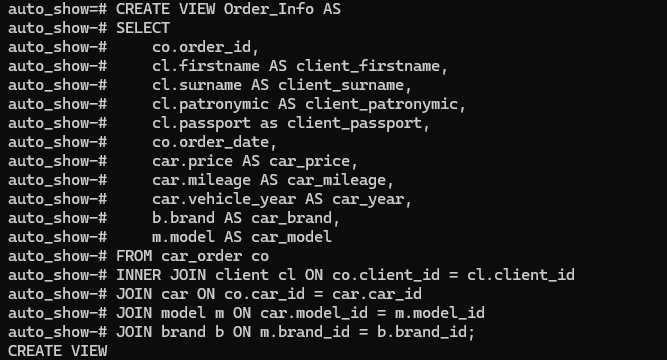


Рисунок 3.2.10.1 – Создание представления

На Рисунке 3.2.10.2 представлен запрос на выборку данных из представления, фамилии клиентов которых начинаются на букву «К».

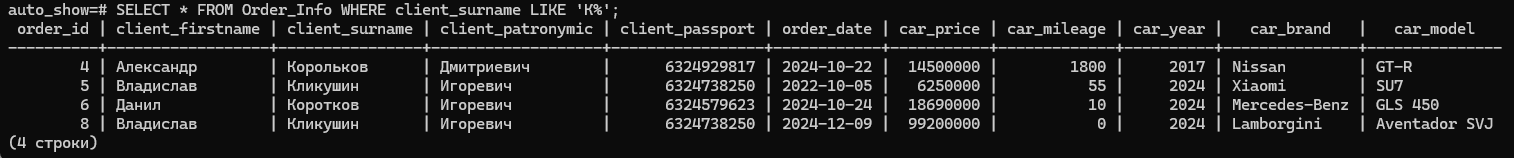


Рисунок 3.2.10.2 – Выполнение выборки из представления

## 3.3 Хранимые процедуры, функции, триггеры

Коды всех процедур, функций и триггеров представлены в Листинге В.3.

### 3.3.1 Процедуры

Добавлена процедура, которая позволяет добавить нового клиента в таблицу client (Рисунок 3.3.1.1).

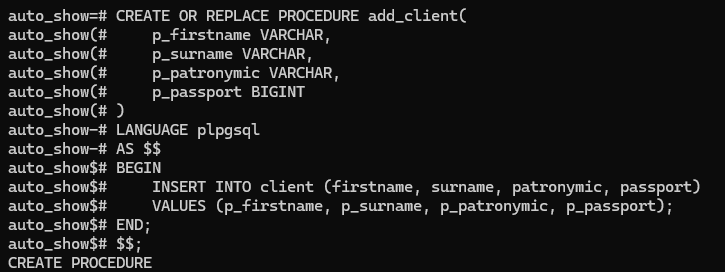


Рисунок 3.3.1.1 – Процедура на добавление нового клиента

На Рисунке 3.3.1.2 представлен запрос на выборку клиентов с именем на букву «Л» до применения процедуры.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.3.1.2 – Выборка клиентов с именем на букву «Л»

На Рисунке 3.3.1.3 произведен вызов процедуры и показана отображена таблица client.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рисунок 3.3.1.3 – Вызов процедуры добавления клиента и выборка клиентов с именем на букву «Л»

На Рисунке 3.3.1.4 представлен вызов процедуры, которая позволяет удалить клиента.

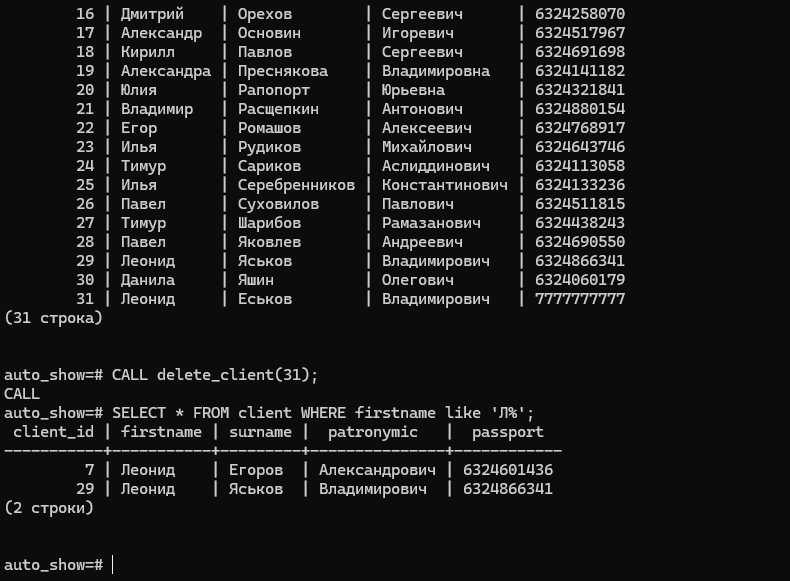


Рисунок 3.3.1.4 - Вызов процедуры удаления клиента и выборка клиентов с именем на букву «Л»

### 3.3.2 Функции

Написана функция, вычисляющая количество заказов с каждым статусом (Рисунок 3.3.2.1).

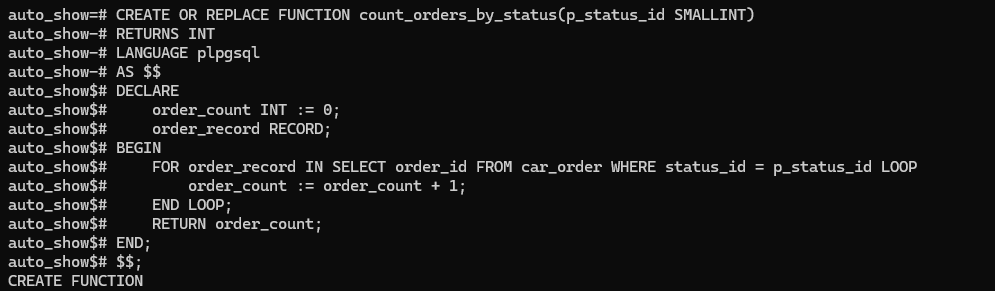


Рисунок 3.3.2.1 – Функция, которая подсчитывает количество заказов с определённым статусом

Вызов функции представлен на Рисунке 3.3.2.2.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рисунок 3.3.2.2 – Вызов функции count\_orders\_by\_status

Необходимо явно указывать тип аргумента при вызове функции в данном случае.

Также написана функция, которая позволяет вычислить стоимость всех заказов для определенного клиента (Рисунок 3.3.2.3).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.3.2.3 – Функция, которая вычисляет общую стоимость заказов определенного клиента

Вызов этой функции представлен на Рисунке 3.3.2.4.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 3.3.2.4 - Вызов функции total\_order\_cost\_by\_client

### 3.3.3 Триггеры

На Рисунке 3.3.3.1 представлена процедура для триггера, который защищает данные от изменения года выпуска на некорректный у существующих записей.

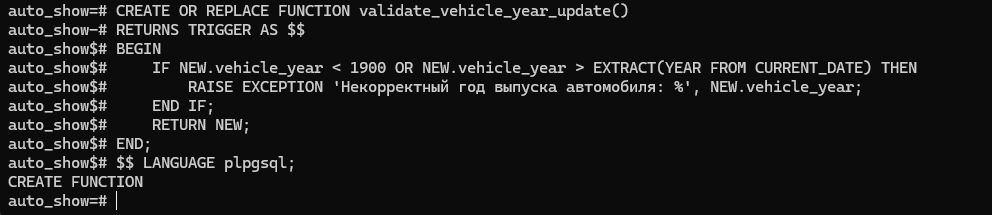


Рисунок 3.3.3.1 – Процедура для триггера

На Рисунке 3.3.3.2 представлен сам триггер, срабатывающий на операцию UPDATE для таблицы car.

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рисунок 3.3.3.2 – Триггер на операцию UPDATE для таблицы car

На Рисунке 3.3.3.3 представлена отработка триггера при попытки изменить год выпуска на недопустимый.

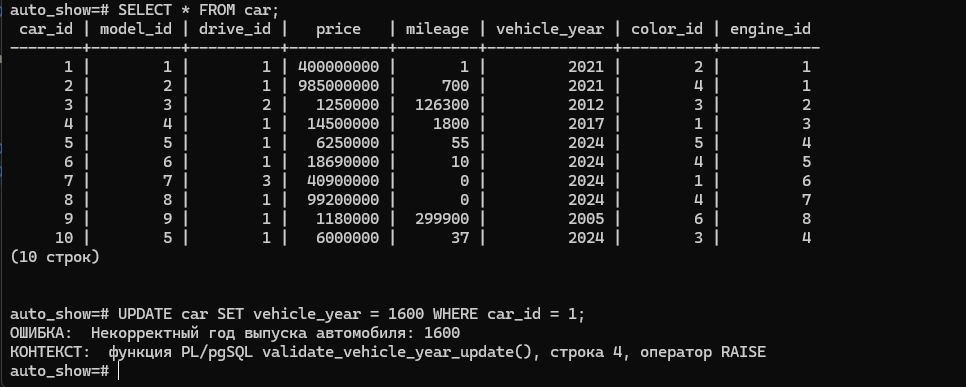


Рисунок 3.3.3.3 – Отработка триггера

# 4 ПРАКТИЧЕСКАЯ РАБОТА №4

**Цель работы:** изучить синтаксис оконных функций и научиться их применять.

## 4.1 Агрегатные функции

На Рисунке 4.1.1 представлен запрос, который отображает суммарную стоимость поставок из каждой страны.

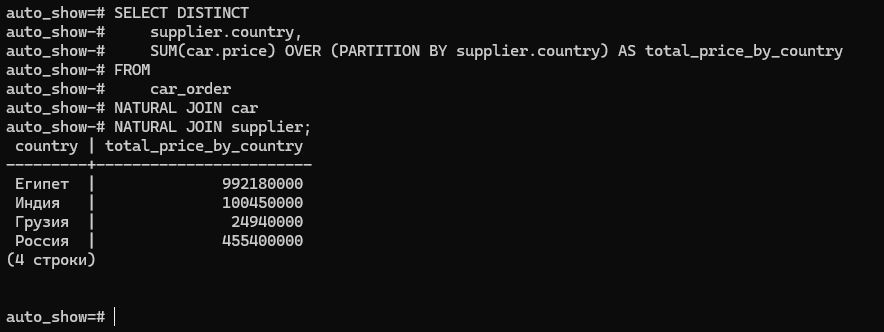


Рисунок 4.1.1 – Выборка с агрегирующей оконной функцией SUM

На Рисунке 4.1.2 представлен запрос, рассчитывающий среднюю мощность двигателей для автомобилей с каждым типом привода. Строки в данном случае группируются по типу привода.

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание

Рисунок 4.1.2 - Выборка с агрегирующей оконной функцией AVG

На Рисунке 4.1.3 представлен запрос, определяющий количество заказов в каждую конкретную дату.

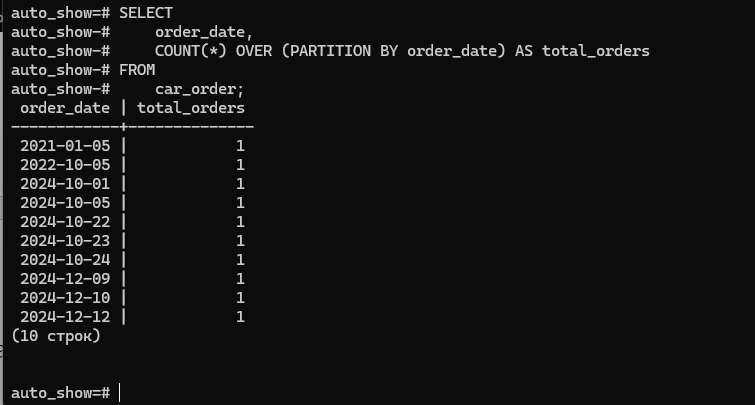


Рисунок 4.1.3 - Выборка с агрегирующей оконной функцией COUNT

На Рисунке 4.1.4 представлен запрос, определяющий минимальную цену автомобиля для каждого бренда.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 4.1.4 - Выборка с агрегирующей оконной функцией MIN

На Рисунке 4.1.5 представлен запрос, который показывает максимальную мощность двигателя для автомобиля с каждым типом привода.



Рисунок 4.1.5 - Выборка с агрегирующей оконной функцией MAX

## 4.2 Ранжирующие функции

На Рисунке 4.2.1 представлен запрос, который нумерует заказы для каждого клиента.

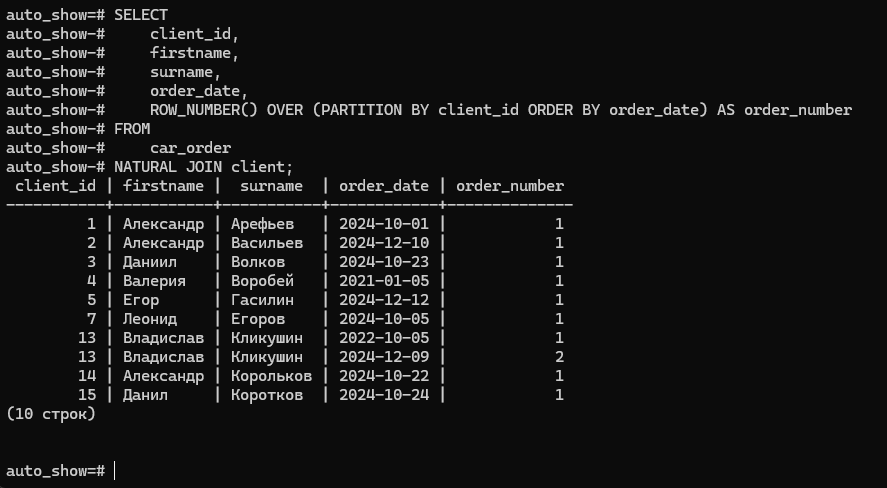


Рисунок 4.2.1 - Выборка с ранжирующей оконной функцией ROW\_NUMBER

На Рисунке 4.2.2 представлен запрос, который ранжирует автомобили по возрасту.

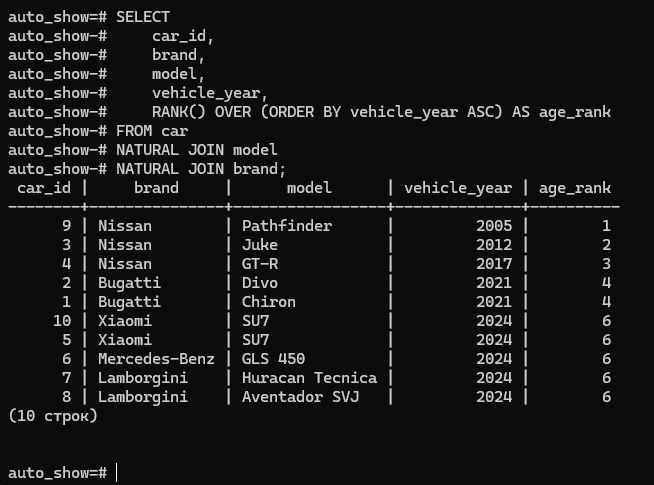


Рисунок 4.2.2 - Выборка с ранжирующей оконной функцией RANK

На Рисунке 4.2.3 представлен запрос, который осуществляет ранжирование автомобилей по цене.

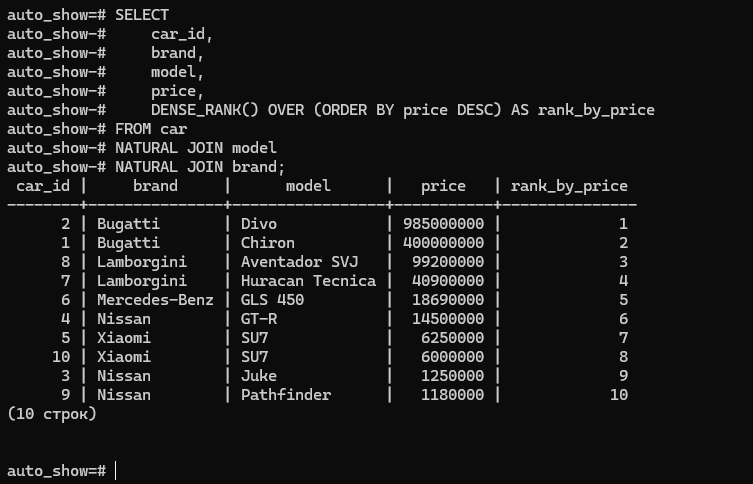


Рисунок 4.2.3 - Выборка с ранжирующей оконной функцией DENSE\_RANK

На Рисунке 4.2.4 представлен запрос, разделяющий автомобили по году выпуска на две группы.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 4.2.4 - Выборка с ранжирующей оконной функцией NTILE

## 4.3 Функции смещения

На Рисунке 4.3.1 представлен запрос, который определяет разницу в цене автомобиля с его следующим автомобилем (по возрастанию цены).

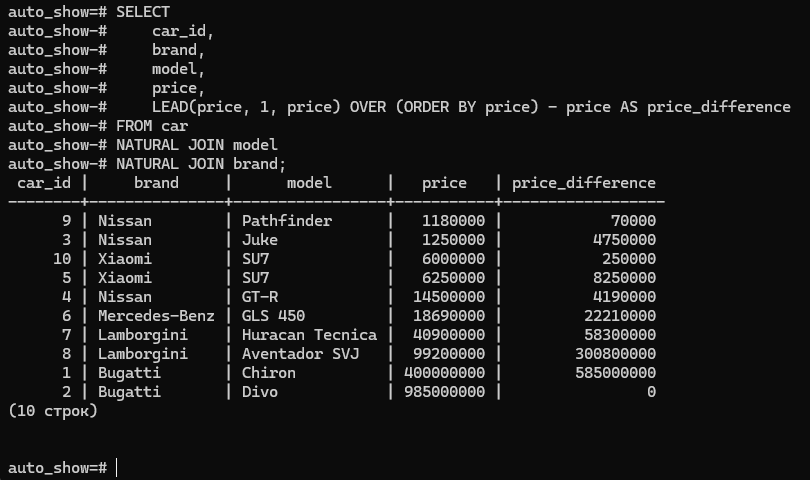


Рисунок 4.3.1 – Выборка с использованием функции смещения LEAD

На Рисунке 4.3.2 представлен запрос, который определяет разницу в цене автомобиля с предыдущим более дешевым автомобилем.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 4.3.2 – Выборка с использованием функции смещения LAG

На Рисунке 4.3.3 представлен запрос, который для каждого клиента определяет самую дорогую купленную машину и самую дешевую.

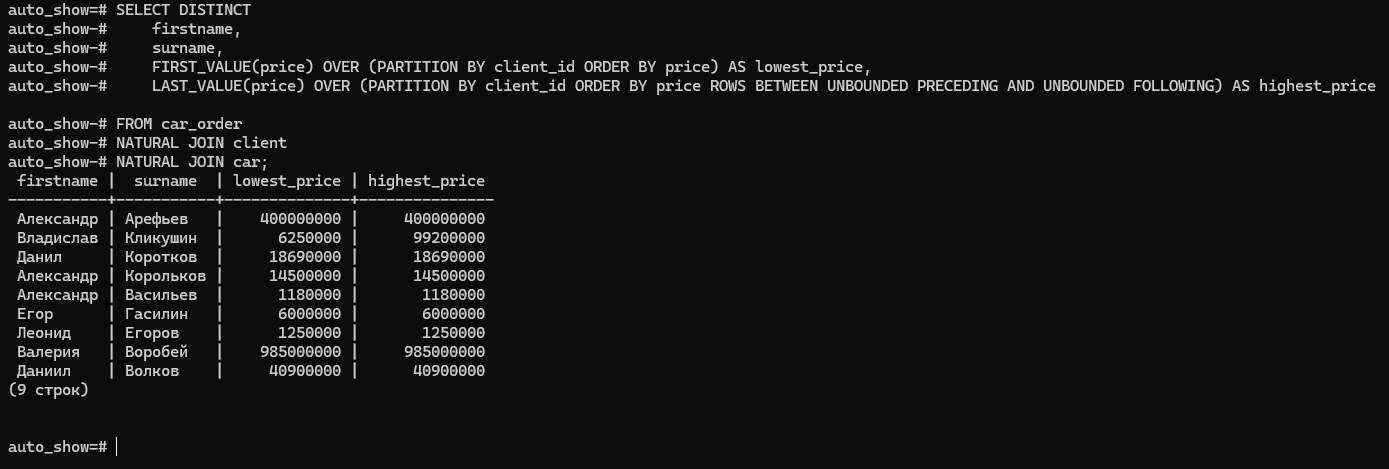


Рисунок 4.3.3 - Выборка с использованием функций смещения LAST\_VALUE и FIRST\_VALUE

# ЗАКЛЮЧЕНИЕ

В ходе выполнения практических работ освоены навыки работы с СУБД PostgreSQL, создана база данных со структурой, описанной в физической модели, сделаны запросы на её заполнение. Изучены операторы выборки и модификации данных, применение операторов выборки для выполнения операций реляционной алгебры. В базе данных созданы хранимые процедуры, функции и триггеры для упрощения работы. Также изучены оконные функции.

# ПРИЛОЖЕНИЯ

Приложение А.1 — Скрипт создания таблиц.

Приложение А.2 — Скрипт заполнения таблиц.

Приложение Б — Скрипты практической работы №2.

Приложение В.1 — Резервная копия базы данных.

Приложение В.2 — Скрипты, реализующие операции реляционной алгебры.

Приложение В.3 — Хранимые процедуры, функции, триггеры.

Приложение Г.1 — Скрипты для агрегатных функций.

Приложение Г.2 — Скрипты для ранжирующих функций.

Приложение Г.3 — Скрипты для функций смещения.

### Приложение А.1

Скрипт создания таблиц

Листинг А.1 – Скрипт создания таблиц

CREATE TABLE client(

    client\_id SERIAL PRIMARY KEY,

    firstname VARCHAR (50) NOT NULL,

    surname VARCHAR (50) NOT NULL,

    patronymic VARCHAR (50) NOT NULL,

    passport BIGINT NOT NULL UNIQUE

);

CREATE TABLE supplier(

    supplier\_id BIGSERIAL PRIMARY KEY,

    email VARCHAR(100) NOT NULL UNIQUE,

    country VARCHAR(50) NOT NULL

);

CREATE TABLE seller(

    seller\_id BIGSERIAL PRIMARY KEY,

    firstname VARCHAR (50) NOT NULL,

    surname VARCHAR (50) NOT NULL,

    patronymic VARCHAR (50) NOT NULL,

    phone\_number VARCHAR(20) NOT NULL UNIQUE,

    email VARCHAR(100) NOT NULL UNIQUE

);

CREATE TABLE order\_status(

    status\_id SMALLSERIAL PRIMARY KEY,

    status VARCHAR(20) NOT NULL

);

CREATE TABLE brand(

    brand\_id SERIAL PRIMARY KEY,

    brand VARCHAR(100) NOT NULL

);

CREATE TABLE model(

    model\_id SERIAL PRIMARY KEY,

    brand\_id INT REFERENCES brand(brand\_id) ON DELETE CASCADE NOT NULL,

    model VARCHAR(100) NOT NULL

);

CREATE TABLE color(

    color\_id SERIAL PRIMARY KEY,

    color VARCHAR(50) NOT NULL UNIQUE

);

CREATE TABLE drive(

    drive\_id SMALLSERIAL PRIMARY KEY,

    drive\_type VARCHAR(50) NOT NULL

);

CREATE TABLE engine\_type(

    engine\_type\_id SMALLSERIAL PRIMARY KEY,

    engine\_type VARCHAR(20) NOT NULL UNIQUE

);

CREATE TABLE engine(

    engine\_id BIGSERIAL PRIMARY KEY,

Окончание Листинга А.1

    engine\_power BIGINT NOT NULL,  --(мощность, л.с)

    engine\_capacity REAL, --(объем, null для электро)

    battery\_capacity INT, --(емкость батареи, null для бензина)

    engine\_type\_id SMALLINT REFERENCES engine\_type(engine\_type\_id) ON DELETE CASCADE NOT NULL

);

CREATE TABLE car(

    car\_id SERIAL PRIMARY KEY,

    model\_id SMALLINT REFERENCES model(model\_id) ON DELETE CASCADE NOT NULL,

    drive\_id SMALLINT REFERENCES drive(drive\_id) ON DELETE CASCADE NOT NULL,

    price BIGINT NOT NULL,

    mileage INT NOT NULL,

    vehicle\_year SMALLINT NOT NULL,

    color\_id SMALLINT REFERENCES color(color\_id) ON DELETE CASCADE NOT NULL,

    engine\_id BIGINT REFERENCES engine(engine\_id) ON DELETE CASCADE NOT NULL

);

CREATE TABLE car\_order(

    order\_id BIGSERIAL PRIMARY KEY,

    status\_id SMALLINT REFERENCES order\_status(status\_id) ON DELETE CASCADE NOT NULL,

    order\_date DATE NOT NULL,

    car\_id INT REFERENCES car(car\_id) ON DELETE CASCADE NOT NULL,

    client\_id INT REFERENCES client(client\_id) ON DELETE CASCADE NOT NULL,

    supplier\_id INT REFERENCES supplier(supplier\_id) ON DELETE CASCADE NOT NULL

);

CREATE TABLE contract(

    contract\_id BIGSERIAL PRIMARY KEY,

    seller\_id BIGINT REFERENCES seller(seller\_id) ON DELETE CASCADE NOT NULL,

    order\_id BIGINT REFERENCES car\_order(order\_id) ON DELETE CASCADE NOT NULL

);

### Приложение А.2

Скрипт заполнения таблиц

Листинг А.2 – Скрипт заполнения таблиц

INSERT INTO client (firstname, surname, patronymic, passport) VALUES

    ('Александр', 'Арефьев', 'Михайлович', 6324170051),

    ('Александр', 'Васильев', 'Станиславович', 6324548740),

    ('Даниил', 'Волков', 'Андреевич', 6324733970),

    ('Валерия', 'Воробей', 'Глебовна', 6324890496),

    ('Егор', 'Гасилин', 'Денисович', 6324275675),

    ('Ольга', 'Довбуш', 'Александровна', 6324602322),

    ('Леонид', 'Егоров', 'Александрович', 6324601436),

    ('Николай', 'Заковряшин', 'Михайлович', 6324587891),

    ('Эдуард', 'Исаков', 'Вячеславович', 6324156944),

    ('Александр', 'Калмыков', 'Михайлович', 6324057936),

    ('Софья', 'Капитонова', 'Романовна', 6324458970),

    ('Арина', 'Карева', 'Александровна', 6324262081),

    ('Владислав', 'Кликушин', 'Игоревич', 6324738250),

    ('Александр', 'Корольков', 'Дмитриевич', 6324929817),

    ('Данил', 'Коротков', 'Игоревич', 6324579623),

    ('Дмитрий', 'Орехов', 'Сергеевич', 6324258070),

    ('Александр', 'Основин', 'Игоревич', 6324517967),

    ('Кирилл', 'Павлов', 'Сергеевич', 6324691698),

    ('Александра', 'Преснякова', 'Владимировна', 6324141182),

    ('Юлия', 'Рапопорт', 'Юрьевна', 6324321841),

    ('Владимир', 'Расщепкин', 'Антонович', 6324880154),

    ('Егор', 'Ромашов', 'Алексеевич', 6324768917),

    ('Илья', 'Рудиков', 'Михайлович', 6324643746),

    ('Тимур', 'Сариков', 'Аслиддинович', 6324113058),

    ('Илья', 'Серебренников', 'Константинович', 6324133236),

    ('Павел', 'Суховилов', 'Павлович', 6324511815),

    ('Тимур', 'Шарибов', 'Рамазанович', 6324438243),

    ('Павел', 'Яковлев', 'Андреевич', 6324690550),

    ('Леонид', 'Яськов', 'Владимирович', 6324866341),

    ('Данила', 'Яшин', 'Олегович', 6324060179);

INSERT INTO supplier (email, country) VALUES

    ('akatev@mirea.ru', 'Россия'),

    ('zheleznyak@mirea.ru', 'Египет'),

    ('sorokin\_a@mirea.ru', 'Индия'),

    ('dzerzhinskij@mirea.ru', 'Россия'),

    ('puturidze@mirea.ru', 'Грузия');

INSERT INTO seller (firstname, surname, patronymic, phone\_number, email) VALUES

    ('Людмила', 'Скворцова', 'Анатольевна', '+79152893255', 'skvortsova@mirea.ru'),

    ('Александр', 'Филатов', 'Сергеевич', '+79261234568', 'filatov\_a@mirea.ru'),

    ('Михаил', 'Рысин', 'Леонидович', '+79361234569', 'rysin@mirea.ru'),

    ('Марина', 'Туманова', 'Борисовна', '+79161236879', 'tumanova@mirea.ru'),

    ('Ирина', 'Куликова', 'Викторовна', '+79161236015', 'kulikova\_irv@mirea.ru'),

    ('Иван', 'Зайцев', 'Юрьевич', '+79163245643', 'zajcev\_i@mirea.ru'),

    ('Галина', 'Богомольная', 'Владимировна', '+79163241010', 'bogomolnaya@mirea.ru');

INSERT INTO order\_status (status) VALUES

    ('Ожидается'),

    ('Отправлен'),

    ('Доставлен');

Продолжение Листинга А.2

INSERT INTO brand (brand) VALUES

    ('Bugatti'),

    ('Nissan'),

    ('Xiaomi'),

    ('Mercedes-Benz'),

    ('Lamborgini');

INSERT INTO model (brand\_id, model) VALUES

    (1, 'Chiron'),

    (1, 'Divo'),

    (2, 'Juke'),

    (2, 'GT-R'),

    (3, 'SU7'),

    (4, 'GLS 450'),

    (5, 'Huracan Tecnica');

INSERT INTO color (color) VALUES

    ('Оранжевый'),

    ('Синий'),

    ('Белый'),

    ('Чёрный'),

    ('Голубой');

INSERT INTO drive (drive\_type) VALUES

    ('Полный'),

    ('Передний'),

    ('Задний');

INSERT INTO engine\_type (engine\_type) VALUES

    ('Бензин'),

    ('Дизель'),

    ('Электро');

INSERT INTO engine (engine\_power, engine\_capacity, battery\_capacity, engine\_type\_id) VALUES

    (1500, 8, NULL, 1),

    (117, 1.6, NULL, 1),

    (555, 3.8, NULL, 1),

    (673, NULL, 495, 3),

    (367, 3, NULL, 2),

    (640, 5.2, NULL, 1);

INSERT INTO car (model\_id, drive\_id, price, mileage, vehicle\_year, color\_id, engine\_id) VALUES

    (1, 1, 400000000, 1, 2021, 2, 1),

    (2, 1, 985000000, 700, 2021, 4, 1),

    (3, 2, 1250000, 126300, 2012, 3, 2),

    (4, 1, 14500000, 1800, 2017, 1, 3),

    (5, 1, 6250000, 55, 2024, 5, 4),

    (6, 1, 18690000, 10, 2024, 4, 5),

    (7, 3, 40900000, 0, 2024, 1, 6);

INSERT INTO car\_order (status\_id, order\_date, car\_id, client\_id, supplier\_id) VALUES

    (1, '2024-10-01', 1, 1, 1),

    (2, '2021-01-05', 2, 4, 2),

    (3, '2024-10-05', 3, 7, 3),

    (2, '2024-10-22', 4, 14, 4),

    (1, '2022-10-05', 5, 13, 5),

    (2, '2024-10-24', 6, 15, 5),

    (3, '2024-10-23', 7, 3, 4);

Окончание Листинга А.2

INSERT INTO contract (seller\_id, order\_id) VALUES

    (1, 1),

    (2, 2),

    (3, 3),

    (4, 4),

    (5, 5),

    (6, 6),

    (7, 7);

### Приложение Б

Скрипты практической работы №2

Листинг Б – Скрипты практической работы №2

SELECT \* FROM client WHERE client\_id = 2;

SELECT \* FROM engine WHERE engine\_power != 1500;

SELECT \* FROM supplier WHERE supplier\_id > 3;

SELECT \* FROM client WHERE passport >= 6324500000;

SELECT \* FROM car WHERE price < 40000000;

SELECT \* FROM engine WHERE engine\_capacity <= 5.2;

SELECT \* FROM engine WHERE engine\_capacity IS NOT NULL;

SELECT \* FROM engine WHERE engine\_capacity IS NULL;

SELECT \* FROM car WHERE price BETWEEN 1000000 AND 20000000;

SELECT \* FROM car\_order WHERE status\_id IN (1, 2);

SELECT \* FROM seller WHERE seller\_id NOT IN (2, 4, 6);

SELECT \* FROM client WHERE surname LIKE 'К%';

SELECT \* FROM seller WHERE patronymic NOT LIKE '%вн%';

SELECT \* FROM car ORDER BY price;

SELECT \* FROM car\_order ORDER BY order\_date DESC;

UPDATE client SET surname = 'Коротков' WHERE client\_id = 14;

ALTER TABLE seller ADD COLUMN number\_sold\_cars SMALLINT;

SELECT \* FROM brand WHERE brand BETWEEN 'Bugatti' AND 'Lamborghini';

SELECT \* FROM car ORDER BY drive\_id, price;

### Приложение В.1

Резервная копия базы данных

Листинг В.1 – Резервная копия базы данных

--

-- PostgreSQL database dump

--

-- Dumped from database version 13.11

-- Dumped by pg\_dump version 13.11

SET statement\_timeout = 0;

SET lock\_timeout = 0;

SET idle\_in\_transaction\_session\_timeout = 0;

SET client\_encoding = 'UTF8';

SET standard\_conforming\_strings = on;

SELECT pg\_catalog.set\_config('search\_path', '', false);

SET check\_function\_bodies = false;

SET xmloption = content;

SET client\_min\_messages = warning;

SET row\_security = off;

SET default\_tablespace = '';

SET default\_table\_access\_method = heap;

--

-- Name: brand; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.brand (

    brand\_id integer NOT NULL,

    brand character varying(100) NOT NULL

);

ALTER TABLE public.brand OWNER TO postgres;

--

-- Name: brand\_brand\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.brand\_brand\_id\_seq

    AS integer

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER TABLE public.brand\_brand\_id\_seq OWNER TO postgres;

--

-- Name: brand\_brand\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.brand\_brand\_id\_seq OWNED BY public.brand.brand\_id;

--

-- Name: car; Type: TABLE; Schema: public; Owner: postgres

Продолжение Листинга В.1

--

CREATE TABLE public.car (

    car\_id integer NOT NULL,

    model\_id smallint NOT NULL,

    drive\_id smallint NOT NULL,

    price bigint NOT NULL,

    mileage integer NOT NULL,

    vehicle\_year smallint NOT NULL,

    color\_id smallint NOT NULL,

    engine\_id bigint NOT NULL

);

ALTER TABLE public.car OWNER TO postgres;

--

-- Name: car\_car\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.car\_car\_id\_seq

    AS integer

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER TABLE public.car\_car\_id\_seq OWNER TO postgres;

--

-- Name: car\_car\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.car\_car\_id\_seq OWNED BY public.car.car\_id;

--

-- Name: car\_order; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.car\_order (

    order\_id bigint NOT NULL,

    status\_id smallint NOT NULL,

    order\_date date NOT NULL,

    car\_id integer NOT NULL,

    client\_id integer NOT NULL,

    supplier\_id integer NOT NULL

);

ALTER TABLE public.car\_order OWNER TO postgres;

--

-- Name: car\_order\_order\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.car\_order\_order\_id\_seq

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

Продолжение Листинга В.1

ALTER TABLE public.car\_order\_order\_id\_seq OWNER TO postgres;

--

-- Name: car\_order\_order\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.car\_order\_order\_id\_seq OWNED BY public.car\_order.order\_id;

--

-- Name: client; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.client (

    client\_id integer NOT NULL,

    firstname character varying(50) NOT NULL,

    surname character varying(50) NOT NULL,

    patronymic character varying(50) NOT NULL,

    passport bigint NOT NULL

);

ALTER TABLE public.client OWNER TO postgres;

--

-- Name: client\_client\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.client\_client\_id\_seq

    AS integer

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER TABLE public.client\_client\_id\_seq OWNER TO postgres;

--

-- Name: client\_client\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.client\_client\_id\_seq OWNED BY public.client.client\_id;

--

-- Name: color; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.color (

    color\_id integer NOT NULL,

    color character varying(50) NOT NULL

);

ALTER TABLE public.color OWNER TO postgres;

--

-- Name: color\_color\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.color\_color\_id\_seq

Продолжение Листинга В.1

    AS integer

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER TABLE public.color\_color\_id\_seq OWNER TO postgres;

--

-- Name: color\_color\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.color\_color\_id\_seq OWNED BY public.color.color\_id;

--

-- Name: contract; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.contract (

    contract\_id bigint NOT NULL,

    seller\_id bigint NOT NULL,

    order\_id bigint NOT NULL

);

ALTER TABLE public.contract OWNER TO postgres;

--

-- Name: contract\_contract\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.contract\_contract\_id\_seq

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER TABLE public.contract\_contract\_id\_seq OWNER TO postgres;

--

-- Name: contract\_contract\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.contract\_contract\_id\_seq OWNED BY public.contract.contract\_id;

--

-- Name: drive; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.drive (

    drive\_id smallint NOT NULL,

    drive\_type character varying(50) NOT NULL

);

ALTER TABLE public.drive OWNER TO postgres;

Продолжение Листинга В.1

--

-- Name: drive\_drive\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.drive\_drive\_id\_seq

    AS smallint

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER TABLE public.drive\_drive\_id\_seq OWNER TO postgres;

--

-- Name: drive\_drive\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.drive\_drive\_id\_seq OWNED BY public.drive.drive\_id;

--

-- Name: engine; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.engine (

    engine\_id bigint NOT NULL,

    engine\_power bigint NOT NULL,

    engine\_capacity real,

    battery\_capacity integer,

    engine\_type\_id smallint NOT NULL

);

ALTER TABLE public.engine OWNER TO postgres;

--

-- Name: engine\_engine\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.engine\_engine\_id\_seq

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER TABLE public.engine\_engine\_id\_seq OWNER TO postgres;

--

-- Name: engine\_engine\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.engine\_engine\_id\_seq OWNED BY public.engine.engine\_id;

--

-- Name: engine\_type; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.engine\_type (

    engine\_type\_id smallint NOT NULL,

Продолжение Листинга В.1

    engine\_type character varying(20) NOT NULL

);

ALTER TABLE public.engine\_type OWNER TO postgres;

--

-- Name: engine\_type\_engine\_type\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.engine\_type\_engine\_type\_id\_seq

    AS smallint

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER TABLE public.engine\_type\_engine\_type\_id\_seq OWNER TO postgres;

--

-- Name: engine\_type\_engine\_type\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.engine\_type\_engine\_type\_id\_seq OWNED BY public.engine\_type.engine\_type\_id;

--

-- Name: model; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.model (

    model\_id integer NOT NULL,

    brand\_id integer NOT NULL,

    model character varying(100) NOT NULL

);

ALTER TABLE public.model OWNER TO postgres;

--

-- Name: model\_model\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.model\_model\_id\_seq

    AS integer

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER TABLE public.model\_model\_id\_seq OWNER TO postgres;

--

-- Name: model\_model\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.model\_model\_id\_seq OWNED BY public.model.model\_id;

Продолжение Листинга В.1

--

-- Name: order\_status; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.order\_status (

    status\_id smallint NOT NULL,

    status character varying(20) NOT NULL

);

ALTER TABLE public.order\_status OWNER TO postgres;

--

-- Name: order\_status\_status\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.order\_status\_status\_id\_seq

    AS smallint

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER TABLE public.order\_status\_status\_id\_seq OWNER TO postgres;

--

-- Name: order\_status\_status\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.order\_status\_status\_id\_seq OWNED BY public.order\_status.status\_id;

--

-- Name: seller; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.seller (

    seller\_id bigint NOT NULL,

    firstname character varying(50) NOT NULL,

    surname character varying(50) NOT NULL,

    patronymic character varying(50) NOT NULL,

    phone\_number character varying(20) NOT NULL,

    email character varying(100) NOT NULL,

    number\_sold\_cars smallint

);

ALTER TABLE public.seller OWNER TO postgres;

--

-- Name: seller\_seller\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.seller\_seller\_id\_seq

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

Продолжение Листинга В.1

ALTER TABLE public.seller\_seller\_id\_seq OWNER TO postgres;

--

-- Name: seller\_seller\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.seller\_seller\_id\_seq OWNED BY public.seller.seller\_id;

--

-- Name: supplier; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.supplier (

    supplier\_id bigint NOT NULL,

    email character varying(100) NOT NULL,

    country character varying(50) NOT NULL

);

ALTER TABLE public.supplier OWNER TO postgres;

--

-- Name: supplier\_supplier\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.supplier\_supplier\_id\_seq

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER TABLE public.supplier\_supplier\_id\_seq OWNER TO postgres;

--

-- Name: supplier\_supplier\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.supplier\_supplier\_id\_seq OWNED BY public.supplier.supplier\_id;

--

-- Name: brand brand\_id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.brand ALTER COLUMN brand\_id SET DEFAULT nextval('public.brand\_brand\_id\_seq'::regclass);

--

-- Name: car car\_id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.car ALTER COLUMN car\_id SET DEFAULT nextval('public.car\_car\_id\_seq'::regclass);

--

-- Name: car\_order order\_id; Type: DEFAULT; Schema: public; Owner: postgres

--

Продолжение Листинга В.1

ALTER TABLE ONLY public.car\_order ALTER COLUMN order\_id SET DEFAULT nextval('public.car\_order\_order\_id\_seq'::regclass);

--

-- Name: client client\_id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.client ALTER COLUMN client\_id SET DEFAULT nextval('public.client\_client\_id\_seq'::regclass);

--

-- Name: color color\_id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.color ALTER COLUMN color\_id SET DEFAULT nextval('public.color\_color\_id\_seq'::regclass);

--

-- Name: contract contract\_id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.contract ALTER COLUMN contract\_id SET DEFAULT nextval('public.contract\_contract\_id\_seq'::regclass);

--

-- Name: drive drive\_id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.drive ALTER COLUMN drive\_id SET DEFAULT nextval('public.drive\_drive\_id\_seq'::regclass);

--

-- Name: engine engine\_id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.engine ALTER COLUMN engine\_id SET DEFAULT nextval('public.engine\_engine\_id\_seq'::regclass);

--

-- Name: engine\_type engine\_type\_id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.engine\_type ALTER COLUMN engine\_type\_id SET DEFAULT nextval('public.engine\_type\_engine\_type\_id\_seq'::regclass);

--

-- Name: model model\_id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.model ALTER COLUMN model\_id SET DEFAULT nextval('public.model\_model\_id\_seq'::regclass);

--

-- Name: order\_status status\_id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.order\_status ALTER COLUMN status\_id SET DEFAULT nextval('public.order\_status\_status\_id\_seq'::regclass);

--

Продолжение Листинга В.1

-- Name: seller seller\_id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.seller ALTER COLUMN seller\_id SET DEFAULT nextval('public.seller\_seller\_id\_seq'::regclass);

--

-- Name: supplier supplier\_id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.supplier ALTER COLUMN supplier\_id SET DEFAULT nextval('public.supplier\_supplier\_id\_seq'::regclass);

--

-- Data for Name: brand; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.brand (brand\_id, brand) FROM stdin;

1   Bugatti

2   Nissan

3   Xiaomi

4   Mercedes-Benz

5   Lamborghini

\.

--

-- Data for Name: car; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.car (car\_id, model\_id, drive\_id, price, mileage, vehicle\_year, color\_id, engine\_id) FROM stdin;

1   1   1   400000000   1   2021    2   1

2   2   1   985000000   700 2021    4   1

3   3   2   1250000 126300  2012    3   2

4   4   1   14500000    1800    2017    1   3

5   5   1   6250000 55  2024    5   4

6   6   1   18690000    10  2024    4   5

7   7   3   40900000    0   2024    1   6

\.

--

-- Data for Name: car\_order; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.car\_order (order\_id, status\_id, order\_date, car\_id, client\_id, supplier\_id) FROM stdin;

1   1   2024-10-01  1   1   1

2   2   2021-01-05  2   4   2

3   3   2024-10-05  3   7   3

4   2   2024-10-22  4   14  4

5   1   2022-10-05  5   13  5

6   2   2024-10-24  6   15  5

7   3   2024-10-23  7   3   4

\.

--

-- Data for Name: client; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.client (client\_id, firstname, surname, patronymic, passport) FROM stdin;

Продолжение Листинга В.1

2   Александр   Васильев    Станиславович   6324548740

3   Даниил  Волков  Андреевич   6324733970

4   Валерия Воробей Глебовна    6324890496

5   Егор    Гасилин Денисович   6324275675

6   Ольга   Довбуш  Александровна   6324602322

7   Леонид  Егоров  Александрович   6324601436

8   Николай Заковряшин  Михайлович  6324587891

9   Эдуард  Исаков  Вячеславович    6324156944

10  Александр   Калмыков    Михайлович  6324057936

11  Софья   Капитонова  Романовна   6324458970

12  Арина   Карева  Александровна   6324262081

13  Владислав   Кликушин    Игоревич    6324738250

16  Дмитрий Орехов  Сергеевич   6324258070

17  Александр   Основин Игоревич    6324517967

18  Кирилл  Павлов  Сергеевич   6324691698

19  Александра  Преснякова  Владимировна    6324141182

20  Юлия    Рапопорт    Юрьевна 6324321841

21  Владимир    Расщепкин   Антонович   6324880154

22  Егор    Ромашов Алексеевич  6324768917

23  Илья    Рудиков Михайлович  6324643746

24  Тимур   Сариков Аслиддинович    6324113058

25  Илья    Серебренников   Константинович  6324133236

26  Павел   Суховилов   Павлович    6324511815

27  Тимур   Шарибов Рамазанович 6324438243

28  Павел   Яковлев Андреевич   6324690550

29  Леонид  Яськов  Владимирович    6324866341

30  Данила  Яшин    Олегович    6324060179

15  Данил   Корольков   Игоревич    6324579623

14  Александр   Коротков    Дмитриевич  6324929817

1   Александр   Арефьев Михайлович  1234567890

\.

--

-- Data for Name: color; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.color (color\_id, color) FROM stdin;

1   Оранжевый

2   Синий

3   Белый

4   Чёрный

5   Голубой

\.

--

-- Data for Name: contract; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.contract (contract\_id, seller\_id, order\_id) FROM stdin;

1   1   1

2   2   2

3   3   3

4   4   4

5   5   5

6   6   6

7   7   7

\.

--

-- Data for Name: drive; Type: TABLE DATA; Schema: public; Owner: postgres

--

Продолжение Листинга В.1

COPY public.drive (drive\_id, drive\_type) FROM stdin;

1   Полный

2   Передний

3   Задний

\.

--

-- Data for Name: engine; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.engine (engine\_id, engine\_power, engine\_capacity, battery\_capacity, engine\_type\_id) FROM stdin;

1   1500    8   \N  1

2   117 1.6 \N  1

3   555 3.8 \N  1

4   673 \N  495 3

5   367 3   \N  2

6   640 5.2 \N  1

\.

--

-- Data for Name: engine\_type; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.engine\_type (engine\_type\_id, engine\_type) FROM stdin;

1   Бензин

2   Дизель

3   Электро

\.

--

-- Data for Name: model; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.model (model\_id, brand\_id, model) FROM stdin;

1   1   Chiron

2   1   Divo

3   2   Juke

4   2   GT-R

5   3   SU7

6   4   GLS 450

7   5   Huracan Tecnica

\.

--

-- Data for Name: order\_status; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.order\_status (status\_id, status) FROM stdin;

1   Ожидается

2   Отправлен

3   Доставлен

\.

--

-- Data for Name: seller; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.seller (seller\_id, firstname, surname, patronymic, phone\_number,

Продолжение Листинга В.1

email, number\_sold\_cars) FROM stdin;

1   Людмила Скворцова   Анатольевна +79152893255    skvortsova@mirea.ru \N

2   Александр   Филатов Сергеевич   +79261234568    filatov\_a@mirea.ru  \N

3   Михаил  Рысин   Леонидович  +79361234569    rysin@mirea.ru  \N

4   Марина  Туманова    Борисовна   +79161236879    tumanova@mirea.ru   \N

5   Ирина   Куликова    Викторовна  +79161236015    kulikova\_irv@mirea.ru   \N

6   Иван    Зайцев  Юрьевич +79163245643    zajcev\_i@mirea.ru   \N

7   Галина  Богомольная Владимировна    +79163241010    bogomolnaya@mirea.ru    \N

\.

--

-- Data for Name: supplier; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.supplier (supplier\_id, email, country) FROM stdin;

1   akatev@mirea.ru Россия

2   zheleznyak@mirea.ru Египет

3   sorokin\_a@mirea.ru  Индия

4   dzerzhinskij@mirea.ru   Россия

5   puturidze@mirea.ru  Грузия

\.

--

-- Name: brand\_brand\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.brand\_brand\_id\_seq', 5, true);

--

-- Name: car\_car\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.car\_car\_id\_seq', 7, true);

--

-- Name: car\_order\_order\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.car\_order\_order\_id\_seq', 7, true);

--

-- Name: client\_client\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.client\_client\_id\_seq', 30, true);

--

-- Name: color\_color\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.color\_color\_id\_seq', 5, true);

--

-- Name: contract\_contract\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.contract\_contract\_id\_seq', 7, true);

Продолжение Листинга В.1

--

-- Name: drive\_drive\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.drive\_drive\_id\_seq', 3, true);

--

-- Name: engine\_engine\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.engine\_engine\_id\_seq', 6, true);

--

-- Name: engine\_type\_engine\_type\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.engine\_type\_engine\_type\_id\_seq', 3, true);

--

-- Name: model\_model\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.model\_model\_id\_seq', 7, true);

--

-- Name: order\_status\_status\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.order\_status\_status\_id\_seq', 3, true);

--

-- Name: seller\_seller\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.seller\_seller\_id\_seq', 7, true);

--

-- Name: supplier\_supplier\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.supplier\_supplier\_id\_seq', 5, true);

--

-- Name: brand brand\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.brand

    ADD CONSTRAINT brand\_pkey PRIMARY KEY (brand\_id);

--

-- Name: car\_order car\_order\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.car\_order

Продолжение Листинга В.1

    ADD CONSTRAINT car\_order\_pkey PRIMARY KEY (order\_id);

--

-- Name: car car\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.car

    ADD CONSTRAINT car\_pkey PRIMARY KEY (car\_id);

--

-- Name: client client\_passport\_key; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.client

    ADD CONSTRAINT client\_passport\_key UNIQUE (passport);

--

-- Name: client client\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.client

    ADD CONSTRAINT client\_pkey PRIMARY KEY (client\_id);

--

-- Name: color color\_color\_key; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.color

    ADD CONSTRAINT color\_color\_key UNIQUE (color);

--

-- Name: color color\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.color

    ADD CONSTRAINT color\_pkey PRIMARY KEY (color\_id);

--

-- Name: contract contract\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.contract

    ADD CONSTRAINT contract\_pkey PRIMARY KEY (contract\_id);

--

-- Name: drive drive\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.drive

    ADD CONSTRAINT drive\_pkey PRIMARY KEY (drive\_id);

--

-- Name: engine engine\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.engine

    ADD CONSTRAINT engine\_pkey PRIMARY KEY (engine\_id);

Продолжение Листинга В.1

--

-- Name: engine\_type engine\_type\_engine\_type\_key; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.engine\_type

    ADD CONSTRAINT engine\_type\_engine\_type\_key UNIQUE (engine\_type);

--

-- Name: engine\_type engine\_type\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.engine\_type

    ADD CONSTRAINT engine\_type\_pkey PRIMARY KEY (engine\_type\_id);

--

-- Name: model model\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.model

    ADD CONSTRAINT model\_pkey PRIMARY KEY (model\_id);

--

-- Name: order\_status order\_status\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.order\_status

    ADD CONSTRAINT order\_status\_pkey PRIMARY KEY (status\_id);

--

-- Name: seller seller\_email\_key; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.seller

    ADD CONSTRAINT seller\_email\_key UNIQUE (email);

--

-- Name: seller seller\_phone\_number\_key; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.seller

    ADD CONSTRAINT seller\_phone\_number\_key UNIQUE (phone\_number);

--

-- Name: seller seller\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.seller

    ADD CONSTRAINT seller\_pkey PRIMARY KEY (seller\_id);

--

-- Name: supplier supplier\_email\_key; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.supplier

    ADD CONSTRAINT supplier\_email\_key UNIQUE (email);

Продолжение Листинга В.1

--

-- Name: supplier supplier\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.supplier

    ADD CONSTRAINT supplier\_pkey PRIMARY KEY (supplier\_id);

--

-- Name: car car\_color\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.car

    ADD CONSTRAINT car\_color\_id\_fkey FOREIGN KEY (color\_id) REFERENCES public.color(color\_id) ON DELETE CASCADE;

--

-- Name: car car\_drive\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.car

    ADD CONSTRAINT car\_drive\_id\_fkey FOREIGN KEY (drive\_id) REFERENCES public.drive(drive\_id) ON DELETE CASCADE;

--

-- Name: car car\_engine\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.car

    ADD CONSTRAINT car\_engine\_id\_fkey FOREIGN KEY (engine\_id) REFERENCES public.engine(engine\_id) ON DELETE CASCADE;

--

-- Name: car car\_model\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.car

    ADD CONSTRAINT car\_model\_id\_fkey FOREIGN KEY (model\_id) REFERENCES public.model(model\_id) ON DELETE CASCADE;

--

-- Name: car\_order car\_order\_car\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.car\_order

    ADD CONSTRAINT car\_order\_car\_id\_fkey FOREIGN KEY (car\_id) REFERENCES public.car(car\_id) ON DELETE CASCADE;

--

-- Name: car\_order car\_order\_client\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.car\_order

    ADD CONSTRAINT car\_order\_client\_id\_fkey FOREIGN KEY (client\_id) REFERENCES

Окончание Листинга В.1

public.client(client\_id) ON DELETE CASCADE;

--

-- Name: car\_order car\_order\_status\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.car\_order

    ADD CONSTRAINT car\_order\_status\_id\_fkey FOREIGN KEY (status\_id) REFERENCES public.order\_status(status\_id) ON DELETE CASCADE;

--

-- Name: car\_order car\_order\_supplier\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.car\_order

    ADD CONSTRAINT car\_order\_supplier\_id\_fkey FOREIGN KEY (supplier\_id) REFERENCES public.supplier(supplier\_id) ON DELETE CASCADE;

--

-- Name: contract contract\_order\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.contract

    ADD CONSTRAINT contract\_order\_id\_fkey FOREIGN KEY (order\_id) REFERENCES public.car\_order(order\_id) ON DELETE CASCADE;

--

-- Name: contract contract\_seller\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.contract

    ADD CONSTRAINT contract\_seller\_id\_fkey FOREIGN KEY (seller\_id) REFERENCES public.seller(seller\_id) ON DELETE CASCADE;

--

-- Name: engine engine\_engine\_type\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.engine

    ADD CONSTRAINT engine\_engine\_type\_id\_fkey FOREIGN KEY (engine\_type\_id) REFERENCES public.engine\_type(engine\_type\_id) ON DELETE CASCADE;

--

-- Name: model model\_brand\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.model

    ADD CONSTRAINT model\_brand\_id\_fkey FOREIGN KEY (brand\_id) REFERENCES public.brand(brand\_id) ON DELETE CASCADE;

--

-- PostgreSQL database dump complete

--

### Приложение В.2

Скрипты, реализующие операции реляционной алгебры

Листинг В.2 – Скрипты, реализующие операции реляционной алгебры

-- 1. Операция проекции. Осуществляется выбор только части полей таблицы, т.е. производится вертикальная выборка данных.

SELECT brand FROM brand;

SELECT price FROM car;

SELECT surname FROM client;

-- 2. Операция селекции. Осуществляется горизонтальная выборка – в результат попадают только записи, удовлетворяющие условию.

SELECT \* FROM seller WHERE surname LIKE 'З%';

SELECT \* FROM car WHERE price > 5000000;

-- 3. Операции соединения. Здесь следует выделить декартово произведение и на его основе соединение по условию, а также естественное соединение (по одноименным полям или равенству полей с одинаковым смыслом).

SELECT

    client.firstname,

    client.surname,

    car\_order.order\_id,

    car\_order.order\_date

FROM

    car\_order, client

WHERE

    car\_order.client\_id = client.client\_id;

SELECT

    client.firstname,

    client.surname,

    car\_order.order\_id,

    car\_order.order\_date

FROM

    car\_order

INNER JOIN client on client.client\_id = car\_order.client\_id;

SELECT

    car\_order.order\_id,

    car\_order.order\_date,

    car.price,

    car.mileage

FROM

    car\_order

NATURAL JOIN car;

SELECT

    car.car\_id,

    model.model,

    color.color,

    price

FROM

    car, model, color

WHERE car.model\_id = model.model\_id

AND car.color\_id = color.color\_id

AND color = 'Оранжевый';

SELECT

    car\_id,

    model,

Продолжение Листинга В.2

    color,

    price

FROM

    car

NATURAL JOIN model

INNER JOIN color on car.color\_id = color.color\_id

WHERE color.color = 'Оранжевый';

-- 4. Операция объединения.

SELECT

    car\_order.order\_id,

    car\_order.order\_date,

    order\_status.status

FROM

    car\_order

NATURAL JOIN order\_status

WHERE order\_status.status = 'Ожидается' OR order\_status.status = 'Отправлен';

SELECT

    car\_order.order\_id,

    car\_order.order\_date,

    order\_status.status

FROM

    car\_order

INNER JOIN order\_status ON car\_order.status\_id = order\_status.status\_id

WHERE order\_status.status = 'Ожидается'

UNION

SELECT

    car\_order.order\_id,

    car\_order.order\_date,

    order\_status.status

FROM

    car\_order

INNER JOIN order\_status ON car\_order.status\_id = order\_status.status\_id

WHERE order\_status.status = 'Отправлен';

SELECT

    car\_order.order\_id,

    car\_order.order\_date,

    brand.brand,

    model.model,

    car.price,

    car.vehicle\_year

FROM

    car\_order

NATURAL JOIN car

NATURAL JOIN model

NATURAL JOIN brand

WHERE brand.brand = 'Bugatti' or brand.brand = 'Lamborgini';

SELECT

    car\_order.order\_id,

    car\_order.order\_date,

    brand.brand,

    model.model,

    car.price,

    car.vehicle\_year

FROM

    car\_order

NATURAL JOIN car

NATURAL JOIN model

NATURAL JOIN brand

Продолжение Листинга В.2

WHERE brand.brand = 'Bugatti'

UNION

SELECT

    car\_order.order\_id,

    car\_order.order\_date,

    brand.brand,

    model.model,

    car.price,

    car.vehicle\_year

FROM

    car\_order

NATURAL JOIN car

NATURAL JOIN model

NATURAL JOIN brand

WHERE brand.brand = 'Lamborgini';

-- 5. Операция пересечения.

SELECT c.client\_id, c.firstname, c.surname

FROM client c

WHERE EXISTS (

    SELECT \*

    FROM car\_order co

    NATURAL JOIN order\_status os

    WHERE co.client\_id = c.client\_id AND os.status = 'Доставлен'

)

AND EXISTS (

    SELECT \*

    FROM car\_order co

    NATURAL JOIN order\_status os

    WHERE co.client\_id = c.client\_id AND os.status = 'Ожидается'

);

SELECT c.client\_id, c.firstname, c.surname

FROM client c

NATURAL JOIN car\_order

NATURAL JOIN order\_status

WHERE order\_status.status = 'Доставлен'

INTERSECT

SELECT c.client\_id, c.firstname, c.surname

FROM client c

NATURAL JOIN car\_order

NATURAL JOIN order\_status

WHERE order\_status.status = 'Ожидается';

SELECT s.seller\_id, s.firstname, s.surname, s.patronymic

FROM seller s

JOIN contract c1 ON s.seller\_id = c1.seller\_id

JOIN car\_order co1 ON c1.order\_id = co1.order\_id

JOIN car car1 ON co1.car\_id = car1.car\_id

JOIN engine e1 ON car1.engine\_id = e1.engine\_id

WHERE e1.engine\_power > 700

INTERSECT

SELECT s.seller\_id, s.firstname, s.surname, s.patronymic

FROM seller s

JOIN contract c2 ON s.seller\_id = c2.seller\_id

JOIN car\_order co2 ON c2.order\_id = co2.order\_id

JOIN car car2 ON co2.car\_id = car2.car\_id

JOIN engine e2 ON car2.engine\_id = e2.engine\_id

WHERE e2.engine\_power < 700;

SELECT s.seller\_id, s.firstname, s.surname

FROM seller s

Продолжение Листинга В.2

WHERE EXISTS (

    SELECT 1

    FROM contract c1

    JOIN car\_order co1 ON c1.order\_id = co1.order\_id

    JOIN car car1 ON co1.car\_id = car1.car\_id

    JOIN engine e1 ON car1.engine\_id = e1.engine\_id

    WHERE c1.seller\_id = s.seller\_id AND e1.engine\_power > 700

)

AND EXISTS (

    SELECT 1

    FROM contract c2

    JOIN car\_order co2 ON c2.order\_id = co2.order\_id

    JOIN car car2 ON co2.car\_id = car2.car\_id

    JOIN engine e2 ON car2.engine\_id = e2.engine\_id

    WHERE c2.seller\_id = s.seller\_id AND e2.engine\_power < 700

);

-- 6. Операция разности. Эта операция также определяется часто с помощью подзапроса с ключевым словом NOT EXISTS, которое показывает отсутствие элемента во множестве, задаваемом подзапросом.

SELECT DISTINCT s.seller\_id, s.firstname, s.surname

FROM seller s

WHERE EXISTS (

    SELECT 1

    FROM contract c

    JOIN car\_order o ON c.order\_id = o.order\_id

    WHERE c.seller\_id = s.seller\_id AND EXTRACT(YEAR FROM o.order\_date) = 2024

)

AND NOT EXISTS (

    SELECT 1

    FROM contract c

    JOIN car\_order o ON c.order\_id = o.order\_id

    WHERE c.seller\_id = s.seller\_id AND EXTRACT(YEAR FROM o.order\_date) = 2023

);

SELECT DISTINCT s.seller\_id, s.firstname, s.surname

FROM seller s

JOIN (

    SELECT c.seller\_id

    FROM contract c

    JOIN car\_order o ON c.order\_id = o.order\_id

    WHERE EXTRACT(YEAR FROM o.order\_date) = 2024

    EXCEPT

    SELECT c.seller\_id

    FROM contract c

    JOIN car\_order o ON c.order\_id = o.order\_id

    WHERE EXTRACT(YEAR FROM o.order\_date) = 2023

) subquery ON s.seller\_id = subquery.seller\_id;

-- 7. Операция группировки. Эта операция связана со своеобразной сверткой таблицы по полям группировки. Помимо полей группировки результат запроса может содержать итоговые агрегирующие функции по группам (COUNT, SUM, AVG, MAX, MIN).

SELECT brand, SUM(car.price) AS total\_price

FROM car

NATURAL JOIN model

NATURAL JOIN brand

GROUP BY brand;

SELECT c.vehicle\_year, AVG(c.mileage) AS avg\_mileage

FROM car c

GROUP BY c.vehicle\_year;

SELECT s.firstname, s.surname, COUNT(\*) AS cars\_sold

Продолжение Листинга В.2

FROM seller s

JOIN contract c ON s.seller\_id = c.seller\_id

GROUP BY s.seller\_id, s.firstname, s.surname;

SELECT m.model, MAX(c.price) AS max\_price, MIN(c.price) AS min\_price

FROM car c

JOIN model m ON c.model\_id = m.model\_id

GROUP BY m.model;

-- 8. Операция сортировки.

SELECT b.brand, COUNT(m.model\_id) AS model\_count

FROM brand b

INNER JOIN model m ON b.brand\_id = m.brand\_id

GROUP BY b.brand\_id, b.brand

ORDER BY model\_count DESC;

SELECT car\_id, brand, model, mileage, engine\_power

FROM car

NATURAL JOIN engine

NATURAL JOIN model

NATURAL JOIN brand

ORDER BY engine.engine\_power DESC, car.mileage ASC;

SELECT s.country, s.email, COUNT(car\_order.order\_id) AS cars\_delivered

FROM supplier s

NATURAL JOIN car\_order

WHERE car\_order.status\_id IN (2, 3)

GROUP BY s.supplier\_id, s.country, s.email

ORDER BY cars\_delivered DESC;

-- 9. Операция деления.

SELECT s.supplier\_id, s.email, s.country

FROM supplier s

WHERE NOT EXISTS (

    SELECT et.engine\_type\_id

    FROM engine\_type et

    WHERE NOT EXISTS (

        SELECT co.supplier\_id

        FROM car\_order co

        JOIN car ca ON co.car\_id = ca.car\_id

        JOIN engine e ON ca.engine\_id = e.engine\_id

        WHERE co.supplier\_id = s.supplier\_id AND e.engine\_type\_id = et.engine\_type\_id

    )

);

-- 10. Создание представления.

CREATE VIEW Order\_Info AS

SELECT

    co.order\_id,

    cl.firstname AS client\_firstname,

    cl.surname AS client\_surname,

    cl.patronymic AS client\_patronymic,

    cl.passport as client\_passport,

    co.order\_date,

    car.price AS car\_price,

    car.mileage AS car\_mileage,

    car.vehicle\_year AS car\_year,

    b.brand AS car\_brand,

    m.model AS car\_model

FROM car\_order co

INNER JOIN client cl ON co.client\_id = cl.client\_id

JOIN car ON co.car\_id = car.car\_id

Окончание Листинга В.2

JOIN model m ON car.model\_id = m.model\_id

JOIN brand b ON m.brand\_id = b.brand\_id;

SELECT \* FROM Order\_Info WHERE client\_surname LIKE 'К%';

### Приложение В.3

Хранимые процедуры, функции, триггеры

Листинг В.3 – Хранимые процедуры, функции, триггеры

--Процедура, которая добавляет нового клиента в таблицу client

CREATE OR REPLACE PROCEDURE add\_client(

    p\_firstname VARCHAR,

    p\_surname VARCHAR,

    p\_patronymic VARCHAR,

    p\_passport BIGINT

)

LANGUAGE plpgsql

AS $$

BEGIN

    INSERT INTO client (firstname, surname, patronymic, passport)

    VALUES (p\_firstname, p\_surname, p\_patronymic, p\_passport);

END;

$$;

CALL add\_client('Леонид', 'Еськов', 'Владимирович', 7777777777);

-- Процедура, которая удаляет клиента по ID.

CREATE OR REPLACE PROCEDURE delete\_client(

    p\_client\_id INT

)

LANGUAGE plpgsql

AS $$

BEGIN

    DELETE FROM client WHERE client\_id = p\_client\_id;

END;

$$;

-- Функция для подсчёта заказов каждого статуса.

CREATE OR REPLACE FUNCTION count\_orders\_by\_status(p\_status\_id SMALLINT)

RETURNS INT

LANGUAGE plpgsql

AS $$

DECLARE

    order\_count INT := 0;

    order\_record RECORD;

BEGIN

    FOR order\_record IN SELECT order\_id FROM car\_order WHERE status\_id = p\_status\_id LOOP

        order\_count := order\_count + 1;

    END LOOP;

    RETURN order\_count;

END;

$$;

SELECT count\_orders\_by\_status(1);

-- Функция для подсчёта общей стоимости покупок автомобилей для определённого клиента.

CREATE OR REPLACE FUNCTION total\_order\_cost\_by\_client(p\_client\_id INT)

RETURNS BIGINT

LANGUAGE plpgsql

AS $$

DECLARE

    total\_cost BIGINT := 0;

    order\_record RECORD;

BEGIN

    FOR order\_record IN

        SELECT c.price FROM car\_order co

Продолжение Листинга В.3

        JOIN car c ON co.car\_id = c.car\_id

        WHERE co.client\_id = p\_client\_id

    LOOP

        total\_cost := total\_cost + order\_record.price;

    END LOOP;

    RETURN total\_cost;

END;

$$;

SELECT total\_order\_cost\_by\_client(13);

-- Триггер, который проверяет год выпуска автомобиля

CREATE OR REPLACE FUNCTION validate\_vehicle\_year\_update()

RETURNS TRIGGER AS $$

BEGIN

    IF NEW.vehicle\_year < 1900 OR NEW.vehicle\_year > EXTRACT(YEAR FROM CURRENT\_DATE) THEN

        RAISE EXCEPTION 'Некорректный год выпуска автомобиля: %', NEW.vehicle\_year;

    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER before\_car\_update

BEFORE UPDATE ON car

FOR EACH ROW

EXECUTE FUNCTION validate\_vehicle\_year\_update();

### Приложение Г.1

Скрипты для агрегатных функций

Листинг Г.1 – Скрипты для агрегатных функций

-- 1. Агрегатная функция SUM

SELECT DISTINCT

    supplier.country,

    SUM(car.price) OVER (PARTITION BY supplier.country) AS total\_price\_by\_country

FROM

    car\_order

NATURAL JOIN car

NATURAL JOIN supplier;

-- 2. Агрегатная функция AVG

SELECT

    drive.drive\_type,

    engine.engine\_power,

    AVG(engine.engine\_power) OVER (PARTITION BY drive.drive\_type) AS avg\_power\_by\_drive

FROM

    car

NATURAL JOIN drive

NATURAL JOIN engine;

-- 3. Агрегатная функция COUNT

SELECT

    order\_date,

    COUNT(\*) OVER (PARTITION BY order\_date) AS total\_orders

FROM

    car\_order;

-- 4. Агрегатная функция MIN

SELECT

    brand.brand,

    car.price,

    MIN(car.price) OVER (PARTITION BY brand.brand\_id) AS min\_price\_by\_brand

FROM

    car

INNER JOIN model ON car.model\_id = model.model\_id

INNER JOIN brand ON model.brand\_id = brand.brand\_id;

-- 5. Агрегатная функция MAX

SELECT

    drive.drive\_type,

    engine.engine\_power,

    MAX(engine.engine\_power) OVER (PARTITION BY drive.drive\_type) AS max\_power\_by\_drive

FROM

    car

NATURAL JOIN drive

NATURAL JOIN engine;

### Приложение Г.2

Скрипты для ранжирующих функций

Листинг Г.2 – Скрипты для ранжирующих функций

-- 6. Ранжирующая функция ROW\_NUMBER

SELECT

    client\_id,

    firstname,

    surname,

    order\_date,

    ROW\_NUMBER() OVER (PARTITION BY client\_id ORDER BY order\_date) AS order\_number

FROM

    car\_order

NATURAL JOIN client;

-- 7. Ранжирующая функция RANK

SELECT

    car\_id,

    brand,

    model,

    vehicle\_year,

    RANK() OVER (ORDER BY vehicle\_year ASC) AS age\_rank

FROM car

NATURAL JOIN model

NATURAL JOIN brand;

-- 8. Ранжирующая функция DENSE\_RANK

SELECT

    car\_id,

    brand,

    model,

    price,

    DENSE\_RANK() OVER (ORDER BY price DESC) AS rank\_by\_price

FROM car

NATURAL JOIN model

NATURAL JOIN brand;

-- 9. Ранжирующая функция NTILE

SELECT

    car\_id,

    brand,

    model,

    vehicle\_year,

    NTILE(2) OVER (ORDER BY vehicle\_year ASC) AS year\_group

FROM car

NATURAL JOIN model

NATURAL JOIN brand;

### Приложение Г.3

Скрипты для функций смещения

Листинг Г.3 – Скрипты для функций смещения

-- 10. Функция смещения LEAD

SELECT

    car\_id,

    brand,

    model,

    price,

    LEAD(price, 1, price) OVER (ORDER BY price) - price AS price\_difference

FROM car

NATURAL JOIN model

NATURAL JOIN brand;

-- 11. Функция смещения LAG

SELECT

    car\_id,

    brand,

    model,

    price,

    LAG(price, 1, price) OVER (ORDER BY price DESC) - price AS price\_difference

FROM car

NATURAL JOIN model

NATURAL JOIN brand;

-- 12. Функция смещения FIRST\_VALUE() и LAST\_VALUE()

SELECT DISTINCT

    firstname,

    surname,

    FIRST\_VALUE(price) OVER (PARTITION BY client\_id ORDER BY price) AS lowest\_price,

    LAST\_VALUE(price) OVER (PARTITION BY client\_id ORDER BY price ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS highest\_price

FROM car\_order

NATURAL JOIN client

NATURAL JOIN car;