



**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 .....	4
1.1 Создание базы данных.....	4
1.2 Создание таблиц.....	5
1.2.1 Таблица client.....	6
1.2.2 Таблица supplier.....	6
1.2.3 Таблица seller.....	7
1.2.4 Таблица order_status.....	8
1.2.5 Таблица brand .....	9
1.2.6 Таблица model.....	9
1.2.7 Таблица color .....	10
1.2.8 Таблица drive .....	11
1.2.9 Таблица engine_type.....	11
1.2.10 Таблица engine.....	12
1.2.11 Таблица car.....	13
1.2.12 Таблица car_order .....	14
1.2.13 Таблица contract.....	15
1.3 Заполнение таблиц .....	16
1.4 Результат создания и заполнения базы данных .....	20
2 ПРАКТИЧЕСКАЯ РАБОТА №2 .....	21
2.1 Выборка данных.....	21
2.2 Выборка данных с сортировкой .....	26
2.3 Операторы изменения данных.....	26
3 ПРАКТИЧЕСКАЯ РАБОТА №3 .....	28
3.1 Резервная копия базы данных.....	28
3.2 Выборка данных согласно операциям реляционной алгебры.....	28
3.2.1 Операция проекции.....	28
3.2.2 Операция селекции .....	30
3.2.3 Операция соединения .....	31

3.2.4 Операция объединения .....	34
3.2.5 Операция пересечения .....	36
3.2.6 Операция разности .....	38
3.2.7 Операция группировки .....	39
3.2.8 Операция сортировки .....	41
3.2.9 Операция деления .....	42
3.2.10 Создание представления.....	43
3.3 Хранимые процедуры, функции, триггеры .....	43
3.3.1 Процедуры .....	43
3.3.2 Функции .....	45
3.3.3 Триггеры.....	47
4 ПРАКТИЧЕСКАЯ РАБОТА №4 .....	48
4.1 Агрегатные функции.....	48
4.2 Ранжирующие функции.....	50
4.3 Функции смещения .....	52
ЗАКЛЮЧЕНИЕ .....	54
ПРИЛОЖЕНИЯ.....	55

# 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.

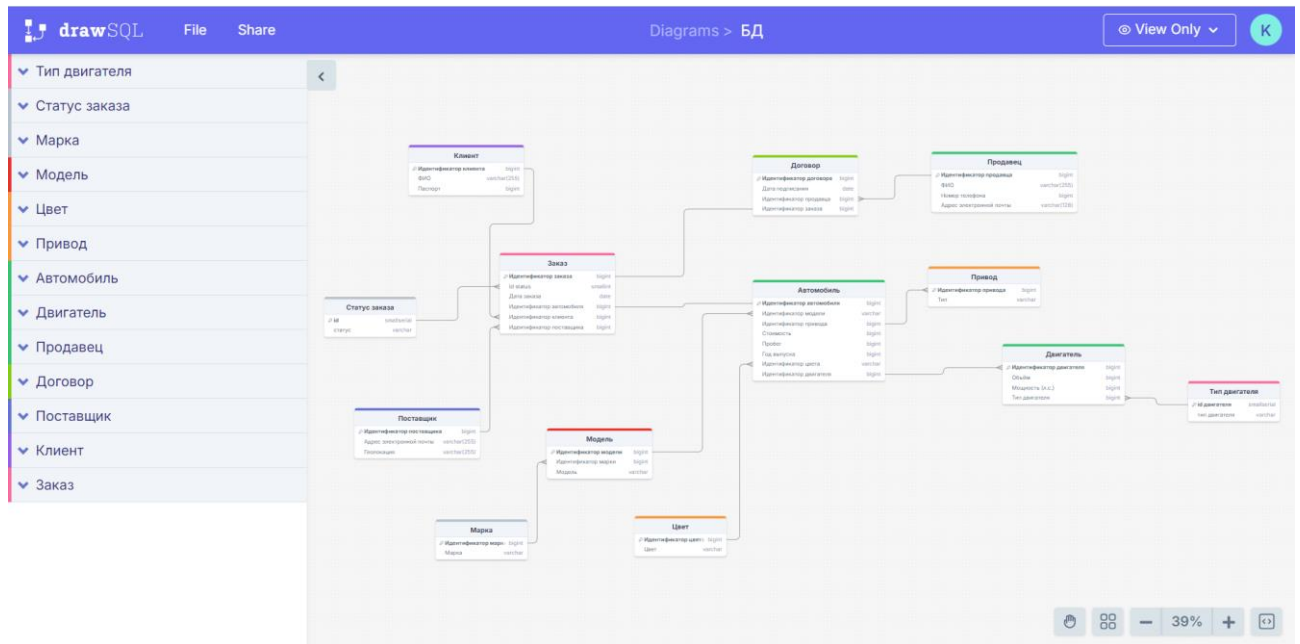


Рисунок 1.1 – Физическая модель базы данных

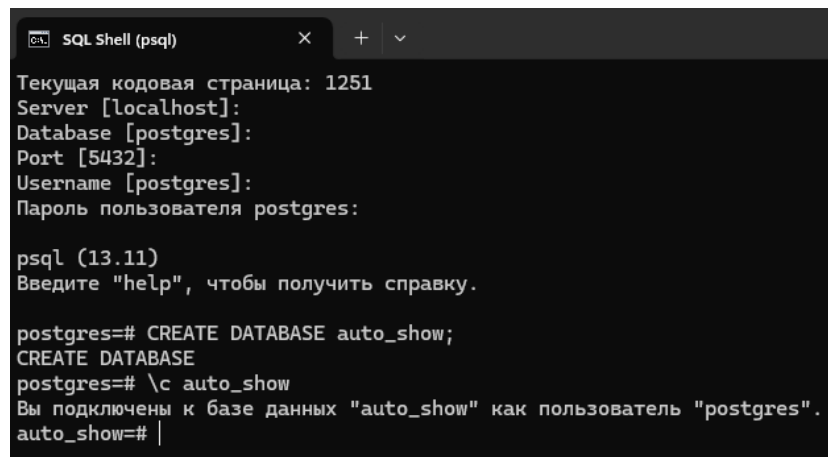
## 1.1 Создание базы данных

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 обработки запроса.



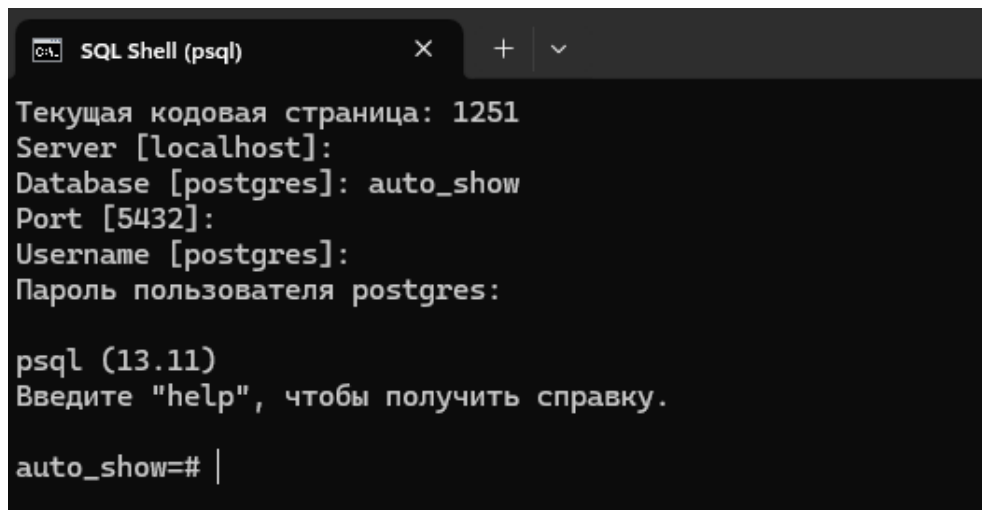
```
SQL Shell (psql)
Текущая кодовая страница: 1251
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Пароль пользователя postgres:

psql (13.11)
Введите "help", чтобы получить справку.

postgres=# CREATE DATABASE auto_show;
CREATE DATABASE
postgres=# \c auto_show
Вы подключены к базе данных "auto_show" как пользователь "postgres".
auto_show=# |
```

**Рисунок 1.1.1 – Создание базы данных**

Дальнейшей подключение к созданной базе данных осуществляется согласно Рисунку 1.1.2.



```
SQL Shell (psql)
Текущая кодовая страница: 1251
Server [localhost]:
Database [postgres]: auto_show
Port [5432]:
Username [postgres]:
Пароль пользователя postgres:

psql (13.11)
Введите "help", чтобы получить справку.

auto_show=# |
```

**Рисунок 1.1.2 – Подключение к созданной базе данных**

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



```
auto_show=# \d
Отношения не найдены.
auto_show=# |
```

**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).

```
auto_show=# CREATE TABLE client(  
auto_show(#      client_id SERIAL PRIMARY KEY,  
auto_show(#      firstname VARCHAR (50) NOT NULL,  
auto_show(#      surname VARCHAR (50) NOT NULL,  
auto_show(#      patronymic VARCHAR (50) NOT NULL,  
auto_show(#      passport BIGINT NOT NULL UNIQUE  
auto_show(# );  
CREATE TABLE  
auto_show=# |
```

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.

```
postgres=# \d client  
  
Таблица "public.client"  


| Столбец    | Тип                   | Правило сортировки | Допустимость NULL | По умолчанию                              |
|------------|-----------------------|--------------------|-------------------|-------------------------------------------|
| client_id  | integer               |                    | not null          | nextval('client_client_id_seq'::regclass) |
| firstname  | character varying(50) |                    | not null          |                                           |
| surname    | character varying(50) |                    | not null          |                                           |
| patronymic | character varying(50) |                    | not null          |                                           |
| passport   | bigint                |                    | not null          |                                           |

  
Индексы:  
"client_pkey" PRIMARY KEY, btree (client_id)  
"client_passport_key" UNIQUE CONSTRAINT, btree (passport)  
Ссылки извне:  
TABLE "car_order" CONSTRAINT "car_order_client_id_fkey" FOREIGN KEY (client_id) REFERENCES client(client_id) ON DELETE CASCADE  
  
postgres=# |
```

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).

```

auto_show=# CREATE TABLE supplier(
auto_show(#      supplier_id BIGSERIAL PRIMARY KEY,
auto_show(#      email VARCHAR(100) NOT NULL UNIQUE,
auto_show(#      country VARCHAR(50) NOT NULL
auto_show(# );
CREATE TABLE
auto_show=# |

```

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.

```

postgres=# \d supplier

```

Столбец	Тип	Правило сортировки	Допустимость NULL	По умолчанию
supplier_id	bigint		not null	nextval('supplier_supplier_id_seq'::regclass)
email	character varying(100)		not null	
country	character varying(50)		not null	

Индексы:  
 "supplier\_pkey" PRIMARY KEY, btree (supplier\_id)  
 "supplier\_email\_key" UNIQUE CONSTRAINT, btree (email)

Ссылки извне:  
 TABLE "car\_order" CONSTRAINT "car\_order\_supplier\_id\_fkey" FOREIGN KEY (supplier\_id) REFERENCES supplier(supplier\_id) ON DELETE CASCADE

```

postgres=#

```

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).

```

auto_show=# CREATE TABLE seller(
auto_show(#      seller_id BIGSERIAL PRIMARY KEY,
auto_show(#      firstname VARCHAR (50) NOT NULL,
auto_show(#      surname VARCHAR (50) NOT NULL,
auto_show(#      patronymic VARCHAR (50) NOT NULL,
auto_show(#      phone_number VARCHAR(20) NOT NULL UNIQUE,
auto_show(#      email VARCHAR(100) NOT NULL UNIQUE
auto_show(# );
CREATE TABLE
auto_show=# |

```

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.

```
postgres=# \d seller
```

Таблица "public.seller"				
Столбец	Тип	Правило сортировки	Допустимость NULL	По умолчанию
seller_id	bigint		not null	nextval('seller_seller_id_seq'::regclass)
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	

Индексы:

- "seller\_pkey" PRIMARY KEY, btree (seller\_id)
- "seller\_email\_key" UNIQUE CONSTRAINT, btree (email)
- "seller\_phone\_number\_key" UNIQUE CONSTRAINT, btree (phone\_number)

Ссылки извне:

- TABLE "contract" CONSTRAINT "contract\_seller\_id\_fkey" FOREIGN KEY (seller\_id) REFERENCES seller(seller\_id) ON DELETE CASCADE

```
postgres=#
```

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).

```
auto_show=# CREATE TABLE order_status(
auto_show(#      status_id SMALLSERIAL PRIMARY KEY,
auto_show(#      status VARCHAR(20) NOT NULL
auto_show(# );
CREATE TABLE
auto_show=#
```

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.



```

postgres=# \d order_status

```

Столбец	Тип	Правило сортировки	Допустимость NULL	По умолчанию
status_id	smallint		not null	nextval('order_status_status_id_seq'::regclass)
status	character varying(20)		not null	

```

Индексы:
    "order_status_pkey" PRIMARY KEY, btree (status_id)
Ссылки извне:
    TABLE "car_order" CONSTRAINT "car_order_status_id_fkey" FOREIGN KEY (status_id) REFERENCES order_status(status_id) ON DELETE CASCADE
postgres=#

```

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).

```

auto_show=# CREATE TABLE brand(
auto_show(#      brand_id SERIAL PRIMARY KEY,
auto_show(#      brand VARCHAR(100) NOT NULL
auto_show(# );
CREATE TABLE
auto_show=#

```

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.

```

postgres=# \d brand

```

Столбец	Тип	Правило сортировки	Допустимость NULL	По умолчанию
brand_id	integer		not null	nextval('brand_brand_id_seq'::regclass)
brand	character varying(100)		not null	

```

Индексы:
    "brand_pkey" PRIMARY KEY, btree (brand_id)
Ссылки извне:
    TABLE "model" CONSTRAINT "model_brand_id_fkey" FOREIGN KEY (brand_id) REFERENCES brand(brand_id) ON DELETE CASCADE
postgres=#

```

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).

```

auto_show=# CREATE TABLE model(
auto_show(#      model_id SERIAL PRIMARY KEY,
auto_show(#      brand_id INT REFERENCES brand(brand_id) ON DELETE CASCADE NOT NULL,
auto_show(#      model VARCHAR(100) NOT NULL
auto_show(# );
CREATE TABLE
auto_show=# |

```

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.

```

postgres=# \d model
                                Таблица "public.model"
+-----+-----+-----+-----+-----+
| Столбец |      Тип      | Правило сортировки | Допустимость NULL | По умолчанию |
+-----+-----+-----+-----+-----+
| model_id | integer       |                     | not null           | nextval('model_model_id_seq'::regclass) |
| brand_id | integer       |                     | not null           |                     |
| model    | character varying(100) |                     | not null           |                     |
+-----+-----+-----+-----+-----+
Индексы:
    "model_pkey" PRIMARY KEY, btree (model_id)
Ограничения внешнего ключа:
    "model_brand_id_fkey" FOREIGN KEY (brand_id) REFERENCES brand(brand_id) ON DELETE CASCADE
Ссылки извне:
    TABLE "car" CONSTRAINT "car_model_id_fkey" FOREIGN KEY (model_id) REFERENCES model(model_id) ON DELETE CASCADE
postgres=# |

```

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).

```

auto_show=# CREATE TABLE color(
auto_show(#      color_id SERIAL PRIMARY KEY,
auto_show(#      color VARCHAR(50) NOT NULL UNIQUE
auto_show(# );
CREATE TABLE
auto_show=# |

```

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.

```

postgres=# \d color

```

Столбец	Тип	Правило сортировки	Допустимость NULL	По умолчанию
color_id	integer		not null	nextval('color_color_id_seq'::regclass)
color	character varying(50)		not null	

```

Индексы:
    "color_pkey" PRIMARY KEY, btree (color_id)
    "color_color_key" UNIQUE CONSTRAINT, btree (color)
Ссылки извне:
    TABLE "car" CONSTRAINT "car_color_id_fkey" FOREIGN KEY (color_id) REFERENCES color(color_id) ON DELETE CASCADE
postgres=#

```

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).

```

auto_show=# CREATE TABLE drive(
auto_show=#     drive_id SMALLSERIAL PRIMARY KEY,
auto_show=#     drive_type VARCHAR(50) NOT NULL
auto_show=# );
CREATE TABLE
auto_show=#

```

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.

```

postgres=# \d drive

```

Столбец	Тип	Правило сортировки	Допустимость NULL	По умолчанию
drive_id	smallint		not null	nextval('drive_drive_id_seq'::regclass)
drive_type	character varying(50)		not null	

```

Индексы:
    "drive_pkey" PRIMARY KEY, btree (drive_id)
Ссылки извне:
    TABLE "car" CONSTRAINT "car_drive_id_fkey" FOREIGN KEY (drive_id) REFERENCES drive(drive_id) ON DELETE CASCADE
postgres=#

```

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).

```

auto_show=# CREATE TABLE engine_type(
auto_show(#      engine_type_id SMALLSERIAL PRIMARY KEY,
auto_show(#      engine_type VARCHAR(20) NOT NULL UNIQUE
auto_show(# );
CREATE TABLE
auto_show=# |

```

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.

```

postgres=# \d engine_type

```

Таблица "public.engine_type"				
Столбец	Тип	Правило сортировки	Допустимость NULL	По умолчанию
engine_type_id	smallint		not null	nextval('engine_type_engine_type_id_seq'::regclass)
engine_type	character varying(20)		not null	

Индексы:

- "engine\_type\_pkey" PRIMARY KEY, btree (engine\_type\_id)
- "engine\_type\_engine\_type\_key" UNIQUE CONSTRAINT, btree (engine\_type)

Ссылки извне:

- TABLE "engine" CONSTRAINT "engine\_engine\_type\_id\_fkey" FOREIGN KEY (engine\_type\_id) REFERENCES engine\_type(engine\_type\_id) ON DELETE CASCADE

```

postgres=# |

```

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).

```

auto_show=# CREATE TABLE engine(
auto_show(#      engine_id BIGSERIAL PRIMARY KEY,
auto_show(#      engine_power BIGINT NOT NULL, --(мощность, л.с)
auto_show(#      engine_capacity REAL, --(объем, null для электро)
auto_show(#      battery_capacity INT, --(емкость батареи, null для бензина)
auto_show(#      engine_type_id SMALLINT REFERENCES engine_type(engine_type_id) ON DELETE CASCADE NOT NULL
auto_show(# );
CREATE TABLE
auto_show=# |

```

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.

```
postgres=# \d engine
```

Таблица "public.engine"				
Столбец	Тип	Правило сортировки	Допустимость NULL	По умолчанию
engine_id	bigint		not null	nextval('engine_engine_id_seq'::regclass)
engine_power	bigint		not null	
engine_capacity	real			
battery_capacity	integer			
engine_type_id	smallint		not null	

Индексы:  
 "engine\_pkey" PRIMARY KEY, btree (engine\_id)

Ограничения внешнего ключа:  
 "engine\_engine\_type\_id\_fkey" FOREIGN KEY (engine\_type\_id) REFERENCES engine\_type(engine\_type\_id) ON DELETE CASCADE

Ссылки извне:  
 TABLE "car" CONSTRAINT "car\_engine\_id\_fkey" FOREIGN KEY (engine\_id) REFERENCES engine(engine\_id) ON DELETE CASCADE

```
postgres=# |
```

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).

```
auto_show=# CREATE TABLE car(
auto_show(#   car_id SERIAL PRIMARY KEY,
auto_show(#   model_id SMALLINT REFERENCES model(model_id) ON DELETE CASCADE NOT NULL,
auto_show(#   drive_id SMALLINT REFERENCES drive(drive_id) ON DELETE CASCADE NOT NULL,
auto_show(#   price BIGINT NOT NULL,
auto_show(#   mileage INT NOT NULL,
auto_show(#   vehicle_year SMALLINT NOT NULL,
auto_show(#   color_id SMALLINT REFERENCES color(color_id) ON DELETE CASCADE NOT NULL,
auto_show(#   engine_id BIGINT REFERENCES engine(engine_id) ON DELETE CASCADE NOT NULL
auto_show(# );
CREATE TABLE
auto_show=# |
```

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.

```
postgres=# \d car
```

Таблица "public.car"				
Столбец	Тип	Правило сортировки	Допустимость NULL	По умолчанию
car_id	integer		not null	nextval('car_car_id_seq'::regclass)
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	

Индексы:  
 "car\_pkey" PRIMARY KEY, btree (car\_id)

Ограничения внешнего ключа:  
 "car\_color\_id\_fkey" FOREIGN KEY (color\_id) REFERENCES color(color\_id) ON DELETE CASCADE  
 "car\_drive\_id\_fkey" FOREIGN KEY (drive\_id) REFERENCES drive(drive\_id) ON DELETE CASCADE  
 "car\_engine\_id\_fkey" FOREIGN KEY (engine\_id) REFERENCES engine(engine\_id) ON DELETE CASCADE  
 "car\_model\_id\_fkey" FOREIGN KEY (model\_id) REFERENCES model(model\_id) ON DELETE CASCADE

Ссылки извне:  
 TABLE "car\_order" CONSTRAINT "car\_order\_car\_id\_fkey" FOREIGN KEY (car\_id) REFERENCES car(car\_id) ON DELETE CASCADE

```
postgres=#
```

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).

```
CREATE TABLE
auto_show=# CREATE TABLE car_order(
auto_show=#     order_id BIGSERIAL PRIMARY KEY,
auto_show=#     status_id SMALLINT REFERENCES order_status(status_id) ON DELETE CASCADE NOT NULL,
auto_show=#     order_date DATE NOT NULL,
auto_show=#     car_id INT REFERENCES car(car_id) ON DELETE CASCADE NOT NULL,
auto_show=#     client_id INT REFERENCES client(client_id) ON DELETE CASCADE NOT NULL,
auto_show=#     supplier_id INT REFERENCES supplier(supplier_id) ON DELETE CASCADE NOT NULL
auto_show=# );
CREATE TABLE
auto_show=#
```

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.

```
postgres=# \d car_order
```

Таблица "public.car_order"				
Столбец	Тип	Правило сортировки	Допустимость NULL	По умолчанию
order_id	bigint		not null	nextval('car_order_order_id_seq'::regclass)
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	

Индексы:  
 "car\_order\_pkey" PRIMARY KEY, btree (order\_id)

Ограничения внешнего ключа:  
 "car\_order\_car\_id\_fkey" FOREIGN KEY (car\_id) REFERENCES car(car\_id) ON DELETE CASCADE  
 "car\_order\_client\_id\_fkey" FOREIGN KEY (client\_id) REFERENCES client(client\_id) ON DELETE CASCADE  
 "car\_order\_status\_id\_fkey" FOREIGN KEY (status\_id) REFERENCES order\_status(status\_id) ON DELETE CASCADE  
 "car\_order\_supplier\_id\_fkey" FOREIGN KEY (supplier\_id) REFERENCES supplier(supplier\_id) ON DELETE CASCADE

Ссылки извне:  
 TABLE "contract" CONSTRAINT "contract\_order\_id\_fkey" FOREIGN KEY (order\_id) REFERENCES car\_order(order\_id) ON DELETE CASCADE

```
postgres=# |
```

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).

```
auto_show=# CREATE TABLE contract(
auto_show(#   contract_id BIGSERIAL PRIMARY KEY,
auto_show(#   seller_id BIGINT REFERENCES seller(seller_id) ON DELETE CASCADE NOT NULL,
auto_show(#   order_id BIGINT REFERENCES car_order(order_id) ON DELETE CASCADE NOT NULL
auto_show(# );
CREATE TABLE
auto_show=# |
```

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.

```
postgres=# \d contract
```

Таблица "public.contract"				
Столбец	Тип	Правило сортировки	Допустимость NULL	По умолчанию
contract_id	bigint		not null	nextval('contract_contract_id_seq'::regclass)
seller_id	bigint		not null	
order_id	bigint		not null	

Индексы:  
 "contract\_pkey" PRIMARY KEY, btree (contract\_id)

Ограничения внешнего ключа:  
 "contract\_order\_id\_fkey" FOREIGN KEY (order\_id) REFERENCES car\_order(order\_id) ON DELETE CASCADE  
 "contract\_seller\_id\_fkey" FOREIGN KEY (seller\_id) REFERENCES seller(seller\_id) ON DELETE CASCADE

```
postgres=# |
```

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.

```
postgres=# SELECT * FROM client;
```

client_id	firstname	surname	patronymic	passport
1	Александр	Арефьев	Михайлович	6324170051
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
14	Александр	Корольков	Дмитриевич	6324929817
15	Данил	Коротков	Игоревич	6324579623
16	Дмитрий	Орехов	Сергеевич	6324258070
17	Александр	Основин	Игоревич	6324517967
18	Кирилл	Павлов	Сергеевич	6324691698
19	Александра	Преснякова	Владимировна	6324141182
20	Юлия	Рапорт	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

(30 строк)  
-- Далее --

Рисунок 1.3.1 – Заполненная таблица client

Заполненная таблица supplier представлена на Рисунке 1.3.2.

```
postgres=# SELECT * FROM supplier;
```

supplier_id	email	country
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

(5 строк)

Рисунок 1.3.2 – Заполненная таблица supplier



Заполненная таблица seller представлена на Рисунке 1.3.3.

```
postgres=# SELECT * FROM seller;
```

seller_id	firstname	surname	patronymic	phone_number	email
1	Людмила	Скворцова	Анатоьевна	+79152893255	skvortsova@mirea.ru
2	Александр	Филатов	Сергеевич	+79261234568	filatov_a@mirea.ru
3	Михаил	Рысин	Леонидович	+79361234569	rysin@mirea.ru
4	Марина	Туманова	Борисовна	+79161236879	tumanova@mirea.ru
5	Ирина	Куликова	Викторовна	+79161236015	kulikova_irv@mirea.ru
6	Иван	Зайцев	Юрьевич	+79163245643	zajcev_i@mirea.ru
7	Галина	Богомольная	Владимировна	+79163241010	bogomolnaya@mirea.ru

(7 строк)

```
postgres=# |
```

Рисунок 1.3.3 – Заполненная таблица seller

Заполненная таблица order\_status представлена на Рисунке 1.3.4.

```
postgres=# SELECT * FROM order_status;
```

status_id	status
1	Ожидается
2	Отправлен
3	Доставлен

(3 строки)

```
postgres=# |
```

Рисунок 1.3.4 – Заполненная таблица order\_status

Заполненная таблица brand представлена на Рисунке 1.3.5.

```
postgres=# SELECT * FROM brand;
```

brand_id	brand
1	Bugatti
2	Nissan
3	Xiaomi
4	Mercedes-Benz
5	Lamborghini

(5 строк)

```
postgres=# |
```

Рисунок 1.3.5 – Заполненная таблица brand

Заполненная таблица model представлена на Рисунке 1.3.6.

```
postgres=# SELECT * FROM model;
 model_id | brand_id |      model
-----+-----+-----
         1 |         1 | Chiron
         2 |         1 | Divo
         3 |         2 | Juke
         4 |         2 | GT-R
         5 |         3 | SU7
         6 |         4 | GLS 450
         7 |         5 | Huracan Tecnica
(7 строк)

postgres=# |
```

Рисунок 1.3.6 – Заполненная таблица model

Заполненная таблица color представлена на Рисунке 1.3.7.

```
postgres=# SELECT * FROM color;
 color_id |      color
-----+-----
         1 | Оранжевый
         2 | Синий
         3 | Белый
         4 | Чёрный
         5 | Голубой
(5 строк)

postgres=# |
```

Рисунок 1.3.7 – Заполненная таблица color

Заполненная таблица drive представлена на Рисунке 1.3.8.

```
postgres=# SELECT * FROM drive;
 drive_id | drive_type
-----+-----
         1 | Полный
         2 | Передний
         3 | Задний
(3 строки)

postgres=# |
```

Рисунок 1.3.8 – Заполненная таблица drive

Заполненная таблица engine\_type представлена на Рисунке 1.3.9.

```
postgres=# SELECT * FROM engine_type;
engine_type_id | engine_type
-----+-----
1 | Бензин
2 | Дизель
3 | Электро
(3 строки)

postgres=#
```

Рисунок 1.3.9 – Заполненная таблица engine\_type

Заполненная таблица engine представлена на Рисунке 1.3.10.

```
postgres=# SELECT * FROM engine;
engine_id | engine_power | engine_capacity | battery_capacity | engine_type_id
-----+-----+-----+-----+-----
1 | 1500 | 8 |  | 1
2 | 117 | 1.6 |  | 1
3 | 555 | 3.8 |  | 1
4 | 673 |  | 495 | 3
5 | 367 | 3 |  | 2
6 | 640 | 5.2 |  | 1
(6 строк)

postgres=#
```

Рисунок 1.3.10 – Заполненная таблица engine

Заполненная таблица car представлена на Рисунке 1.3.11.

```
postgres=# SELECT * FROM car;
car_id | model_id | drive_id | price | mileage | vehicle_year | color_id | engine_id
-----+-----+-----+-----+-----+-----+-----+-----
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
(7 строк)

postgres=#
```

Рисунок 1.3.11 – Заполненная таблица car

Заполненная таблица car\_order представлена на Рисунке 1.3.12.

```
postgres=# SELECT * FROM car_order;
order_id | status_id | order_date | car_id | client_id | supplier_id
-----+-----+-----+-----+-----+-----
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
(7 строк)

postgres=#
```

Рисунок 1.3.12 – Заполненная таблица car\_order

Заполненная таблица contract представлена на Рисунке 1.3.13.

```
postgres=# SELECT * FROM contract;
```

contract_id	seller_id	order_id
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7

(7 строк)

```
postgres=#
```

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.

```
postgres=# \d
```

Список отношений			
Схема	Имя	Тип	Владелец
public	brand	таблица	postgres
public	brand_brand_id_seq	последовательность	postgres
public	car	таблица	postgres
public	car_car_id_seq	последовательность	postgres
public	car_order	таблица	postgres
public	car_order_order_id_seq	последовательность	postgres
public	client	таблица	postgres
public	client_client_id_seq	последовательность	postgres
public	color	таблица	postgres
public	color_color_id_seq	последовательность	postgres
public	contract	таблица	postgres
public	contract_contract_id_seq	последовательность	postgres
public	drive	таблица	postgres
public	drive_drive_id_seq	последовательность	postgres
public	engine	таблица	postgres
public	engine_engine_id_seq	последовательность	postgres
public	engine_type	таблица	postgres
public	engine_type_engine_type_id_seq	последовательность	postgres
public	model	таблица	postgres
public	model_model_id_seq	последовательность	postgres
public	order_status	таблица	postgres
public	order_status_status_id_seq	последовательность	postgres
public	seller	таблица	postgres
public	seller_seller_id_seq	последовательность	postgres
public	supplier	таблица	postgres
public	supplier_supplier_id_seq	последовательность	postgres

(26 строк)

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.

```
auto_show=# SELECT * FROM client WHERE client_id = 2;
 client_id | firstname | surname | patronymic | passport
-----+-----+-----+-----+-----
          2 | Александр | Васильев | Станиславович | 6324548740
(1 строка)

auto_show=# |
```

Рисунок 2.1.1 – Запрос на выборку с оператором «=»

#### 2. Оператор «!=».

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

```
auto_show=# SELECT * FROM engine WHERE engine_power != 1500;
 engine_id | engine_power | engine_capacity | battery_capacity | engine_type_id
-----+-----+-----+-----+-----
          2 |          117 |             1.6 |                |              1
          3 |          555 |             3.8 |                |              1
          4 |          673 |                |          495    |              3
          5 |          367 |              3  |                |              2
          6 |          640 |             5.2 |                |              1
(5 строк)

auto_show=# |
```

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

### 3. Оператор «>».

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

```
auto_show=# SELECT * FROM supplier WHERE supplier_id > 3;
supplier_id |          email          | country
-----+-----+-----
          4 | dzerzhinskij@mirea.ru | Россия
          5 | puturidze@mirea.ru    | Грузия
(2 строки)

auto_show=# |
```

Рисунок 2.1.3 – Запрос на выборку с оператором «>»

### 4. Оператор «>=».

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

```
auto_show=# SELECT * FROM client WHERE passport >= 6324500000;
client_id | firstname | surname | patronymic | passport
-----+-----+-----+-----+-----
          2 | Александр | Васильев | Станиславович | 6324548740
          3 | Даниил   | Волков  | Андреевич    | 6324733970
          4 | Валерия  | Воробей | Глебовна     | 6324890496
          6 | Ольга    | Довбуш  | Александровна | 6324602322
          7 | Леонид   | Егоров  | Александрович | 6324601436
          8 | Николай  | Заковряшин | Михайлович   | 6324587891
         13 | Владислав | Кликушин | Игоревич     | 6324738250
         14 | Александр | Корольков | Дмитриевич   | 6324929817
         15 | Данил    | Коротков | Игоревич     | 6324579623
         17 | Александр | Основин | Игоревич     | 6324517967
         18 | Кирилл   | Павлов  | Сергеевич    | 6324691698
         21 | Владимир | Расщепкин | Антонович    | 6324880154
         22 | Егор     | Ромашов | Алексеевич   | 6324768917
         23 | Илья     | Рудилов  | Михайлович   | 6324643746
         26 | Павел    | Суховилов | Павлович     | 6324511815
         28 | Павел    | Яковлев  | Андреевич    | 6324690550
         29 | Леонид   | Яськов  | Владимирович | 6324866341
(17 строк)

auto_show=# |
```

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

### 5. Оператор «<».

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

```
auto_show=# SELECT * FROM car WHERE price < 40000000;
```

car_id	model_id	drive_id	price	mileage	vehicle_year	color_id	engine_id
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

(4 строки)

```
auto_show=#
```

Рисунок 2.1.5 – Запрос на выборку с оператором «<»

## 6. 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.

```
auto_show=# SELECT * FROM engine WHERE engine_capacity <= 5.2;
```

engine_id	engine_power	engine_capacity	battery_capacity	engine_type_id
2	117	1.6		1
3	555	3.8		1
5	367	3		2
6	640	5.2		1

(4 строки)

```
auto_show=#
```

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

## 7. 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.

```
auto_show=# SELECT * FROM engine WHERE engine_capacity IS NOT NULL;
```

engine_id	engine_power	engine_capacity	battery_capacity	engine_type_id
1	1500	8		1
2	117	1.6		1
3	555	3.8		1
5	367	3		2
6	640	5.2		1

(5 строк)

```
auto_show=#
```

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

## 8. Оператор «IS NULL».

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

```
auto_show=# SELECT * FROM engine WHERE engine_capacity IS NULL;
```

engine_id	engine_power	engine_capacity	battery_capacity	engine_type_id
4	673		495	3

(1 строка)

```
auto_show=#
```

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

## 9. 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.

```
auto_show=# SELECT * FROM car WHERE price BETWEEN 1000000 AND 20000000;
```

car_id	model_id	drive_id	price	mileage	vehicle_year	color_id	engine_id
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

(4 строки)

```
auto_show=#
```

Рисунок 2.1.9 – Запрос на выборку с оператором «BETWEEN»

## 10. 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.

```
auto_show=# SELECT * FROM car_order WHERE status_id IN (1, 2);
```

order_id	status_id	order_date	car_id	client_id	supplier_id
1	1	2024-10-01	1	1	1
2	2	2021-01-05	2	4	2
4	2	2024-10-22	4	14	4
5	1	2022-10-05	5	13	5
6	2	2024-10-24	6	15	5

(5 строк)

```
auto_show=#
```

Рисунок 2.1.10 - Запрос на выборку с оператором «IN»

## 11. 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.



```

auto_show=# SELECT * FROM seller WHERE seller_id NOT IN (2, 4, 6);
 seller_id | firstname | surname | patronymic | phone_number | email
-----+-----+-----+-----+-----+-----
          1 | Людмила  | Скворцова | Анатольевна | +79152893255 | skvortsova@mirea.ru
          3 | Михаил  | Рысин    | Леонидович  | +79361234569 | rysin@mirea.ru
          5 | Ирина   | Куликова | Викторовна  | +79161236015 | kulikova_irv@mirea.ru
          7 | Галина  | Богомольная | Владимировна | +79163241010 | bogomolnaya@mirea.ru
(4 строки)

auto_show=#

```

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

## 12. Оператор «LIKE».

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

```

auto_show=# SELECT * FROM client WHERE surname LIKE 'K%';
 client_id | firstname | surname | patronymic | passport
-----+-----+-----+-----+-----
         10 | Александр | Калмыков | Михайлович | 6324057936
         11 | Софья    | Капитонова | Романовна | 6324458970
         12 | Арина    | Карева   | Александровна | 6324262081
         13 | Владислав | Кликушин | Игоревич   | 6324738250
         14 | Александр | Корольков | Дмитриевич | 6324929817
         15 | Данил    | Коротков | Игоревич   | 6324579623
(6 строк)

auto_show=#

```

Рисунок 2.1.12 – Запрос на выборку с оператором «LIKE»

## 13. 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».

```

auto_show=# SELECT * FROM seller WHERE patronymic NOT LIKE '%вн%';
 seller_id | firstname | surname | patronymic | phone_number | email
-----+-----+-----+-----+-----+-----
          2 | Александр | Филатов | Сергеевич  | +79261234568 | filatov_a@mirea.ru
          3 | Михаил  | Рысин   | Леонидович | +79361234569 | rysin@mirea.ru
          6 | Иван    | Зайцев  | Юрьевич    | +79163245643 | zajcev_i@mirea.ru
(3 строки)

auto_show=#

```

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.

```
auto_show=# SELECT * FROM car ORDER BY price;
```

car_id	model_id	drive_id	price	mileage	vehicle_year	color_id	engine_id
3	3	2	1250000	126300	2012	3	2
5	5	1	6250000	55	2024	5	4
4	4	1	14500000	1800	2017	1	3
6	6	1	18690000	10	2024	4	5
7	7	3	40900000	0	2024	1	6
1	1	1	400000000	1	2021	2	1
2	2	1	985000000	700	2021	4	1

(7 строк)

```
auto_show=#
```

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.

```
auto_show=# SELECT * FROM car_order ORDER BY order_date DESC;
```

order_id	status_id	order_date	car_id	client_id	supplier_id
6	2	2024-10-24	6	15	5
7	3	2024-10-23	7	3	4
4	2	2024-10-22	4	14	4
3	3	2024-10-05	3	7	3
1	1	2024-10-01	1	1	1
5	1	2022-10-05	5	13	5
2	2	2021-01-05	2	4	2

(7 строк)

```
auto_show=#
```

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.

```

auto_show=# ALTER TABLE seller ADD COLUMN number_sold_cars SMALLINT;
ALTER TABLE
auto_show=# \d seller;

```

Столбец	Тип	Правило сортировки	Допустимость NULL	По умолчанию
seller_id	bigint		not null	nextval('seller_seller_id_seq'::regclass)
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			

```

Индексы:
    "seller_pkey" PRIMARY KEY, btree (seller_id)
    "seller_email_key" UNIQUE CONSTRAINT, btree (email)
    "seller_phone_number_key" UNIQUE CONSTRAINT, btree (phone_number)
Ссылки извне:
    TABLE "contract" CONSTRAINT "contract_seller_id_fkey" FOREIGN KEY (seller_id) REFERENCES seller(seller_id) ON DELETE CASCADE
auto_show=#

```

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.

```

auto_show=# UPDATE client SET surname = 'Коротков' WHERE client_id = 14;
UPDATE 1
auto_show=# SELECT * FROM client;

```

client_id	firstname	surname	patronymic	passport
1	Александр	Арефьев	Михайлович	6324170051
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
15	Данил	Коротков	Игоревич	6324579623
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

```

-- Далее --

```

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.

```

auto_show=# SELECT brand FROM brand;
brand
-----
Bugatti
Nissan
Xiaomi
Mercedes-Benz
Lamborghini
(5 строк)

auto_show=# |

```

Рисунок 3.2.1.1 – Запрос на выборку всех названий брендов

На Рисунке 3.2.1.2 представлен запрос на выборку всех фамилий клиентов.

```

auto_show=# SELECT surname FROM client;
surname
-----
Арефьев
Васильев
Волков
Воробей
Гасилин
Довбуш
Егоров
Заковряшин
Исаков
Калмыков
Капитонова
Карева
Кликушин
Коротков
Орехов
Основин
Павлов
Преснякова
Рапопорт
Расцепкин
Ромашов
Рудиков
Сариков
Серебренников
Суховилов
Шарибов
Яковлев
-- Далее -- |

```

Рисунок 3.2.1.2 – Запрос на выборку всех фамилий клиентов

На Рисунке 3.2.1.3 представлен запрос на выборку всех цен автомобилей.

```

auto_show=# SELECT price FROM car;
price
-----
400000000
985000000
1250000
14500000
6250000
18690000
40900000
(7 строк)

auto_show=# |

```

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».

```

auto_show=# SELECT * FROM seller WHERE surname LIKE 'З%';
seller_id | firstname | surname | patronymic | phone_number | email | number_sold_cars
-----
6 | Иван | Зайцев | Юрьевич | +79163245643 | zajcev_i@mirea.ru | 1
(1 строка)

auto_show=# |

```

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.

```

auto_show=# SELECT * FROM car WHERE price > 5000000;
car_id | model_id | drive_id | price | mileage | vehicle_year | color_id | engine_id
-----
1 | 1 | 1 | 400000000 | 1 | 2021 | 2 | 1
2 | 2 | 1 | 985000000 | 700 | 2021 | 4 | 1
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
(6 строк)

auto_show=# |

```

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».

```
auto_show=# SELECT * FROM supplier WHERE country = 'Россия';
supplier_id |          email          | country
-----+-----+-----
          1 | akatev@mirea.ru        | Россия
          4 | dzerzhinskij@mirea.ru  | Россия
(2 строки)

auto_show=# |
```

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`.

```
auto_show=# SELECT
auto_show=#     client.firstname,
auto_show=#     client.surname,
auto_show=#     car_order.order_id,
auto_show=#     car_order.order_date
auto_show=# FROM
auto_show=#     car_order, client
auto_show=# WHERE
auto_show=#     car_order.client_id = client.client_id;
firstname | surname | order_id | order_date
-----+-----+-----+-----
Александр | Арефьев |         1 | 2024-10-01
Валерия   | Воробей |         2 | 2021-01-05
Леонид    | Егоров  |         3 | 2024-10-05
Александр | Коротков |         4 | 2024-10-22
Владислав | Кликушин |         5 | 2022-10-05
Данил     | Коротков |         6 | 2024-10-24
Даниил    | Волков  |         7 | 2024-10-23
(7 строк)

auto_show=# |
```

Рисунок 3.2.3.1 – Запрос, реализующий операцию соединения

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

```

auto_show=# SELECT
auto_show=#     client.firstname,
auto_show=#     client.surname,
auto_show=#     car_order.order_id,
auto_show=#     car_order.order_date
auto_show=# FROM
auto_show=#     car_order
auto_show=# INNER JOIN client on client.client_id = car_order.client_id;
  firstname | surname | order_id | order_date
-----+-----+-----+-----
Александр | Арефьев |         1 | 2024-10-01
Валерия   | Воробей |         2 | 2021-01-05
Леонид     | Егоров  |         3 | 2024-10-05
Александр | Коротков |         4 | 2024-10-22
Владислав | Кликушин |         5 | 2022-10-05
Данил      | Коротков |         6 | 2024-10-24
Даниил     | Волков  |         7 | 2024-10-23
(7 строк)

auto_show=#

```

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.

```

auto_show=# SELECT
auto_show=#     car_order.order_id,
auto_show=#     car_order.order_date,
auto_show=#     car.price,
auto_show=#     car.mileage
auto_show=# FROM
auto_show=#     car_order
auto_show=# NATURAL JOIN car;
 order_id | order_date | price  | mileage
-----+-----+-----+-----
         1 | 2024-10-01 | 400000000 |         1
         2 | 2021-01-05 | 985000000 |         700
         3 | 2024-10-05 | 1250000  |       126300
         4 | 2024-10-22 | 14500000 |         1800
         5 | 2022-10-05 | 6250000  |          55
         6 | 2024-10-24 | 18690000 |          10
         7 | 2024-10-23 | 40900000 |           0
(7 строк)

auto_show=#

```

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.

```

auto_show=# SELECT
auto_show=#     car.car_id,
auto_show=#     model.model,
auto_show=#     color.color,
auto_show=#     price
auto_show=# FROM
auto_show=#     car, model, color
auto_show=# WHERE car.model_id = model.model_id
auto_show=# AND car.color_id = color.color_id
auto_show=# AND color = 'Оранжевый';
 car_id |      model      | color   | price
-----+-----+-----+-----
      4 | GT-R            | Оранжевый | 14500000
      7 | Huracan Tecnica | Оранжевый | 40900000
(2 строки)

auto_show=# |

```

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.

```

auto_show=# SELECT
auto_show=#     car_id,
auto_show=#     model,
auto_show=#     color,
auto_show=#     price
auto_show=# FROM
auto_show=#     car
auto_show=# NATURAL JOIN model
auto_show=# INNER JOIN color on car.color_id = color.color_id
auto_show=# WHERE color.color = 'Оранжевый';
 car_id |      model      | color   | price
-----+-----+-----+-----
      4 | GT-R            | Оранжевый | 14500000
      7 | Huracan Tecnica | Оранжевый | 40900000
(2 строки)

auto_show=# |

```

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».

```
auto_show=# SELECT
auto_show=#     car_order.order_id,
auto_show=#     car_order.order_date,
auto_show=#     order_status.status
auto_show=# FROM
auto_show=#     car_order
auto_show=# NATURAL JOIN order_status
auto_show=# WHERE order_status.status = 'Ожидается' OR order_status.status = 'Отправлен';
 order_id | order_date | status
-----+-----+-----
        1 | 2024-10-01 | Ожидается
        2 | 2021-01-05 | Отправлен
        4 | 2024-10-22 | Отправлен
        5 | 2022-10-05 | Ожидается
        6 | 2024-10-24 | Отправлен
(5 строк)
auto_show=# |
```

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.

```
auto_show=# SELECT
auto_show=#     car_order.order_id,
auto_show=#     car_order.order_date,
auto_show=#     order_status.status
auto_show=# FROM
auto_show=#     car_order
auto_show=# INNER JOIN order_status ON car_order.status_id = order_status.status_id
auto_show=# WHERE order_status.status = 'Ожидается'
auto_show=# UNION
auto_show=# SELECT
auto_show=#     car_order.order_id,
auto_show=#     car_order.order_date,
auto_show=#     order_status.status
auto_show=# FROM
auto_show=#     car_order
auto_show=# INNER JOIN order_status ON car_order.status_id = order_status.status_id
auto_show=# WHERE order_status.status = 'Отправлен';
 order_id | order_date | status
-----+-----+-----
        4 | 2024-10-22 | Отправлен
        6 | 2024-10-24 | Отправлен
        2 | 2021-01-05 | Отправлен
        5 | 2022-10-05 | Ожидается
        1 | 2024-10-01 | Ожидается
(5 строк)
auto_show=# |
```

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.

```

auto_show=# SELECT
auto_show=#     car_order.order_id,
auto_show=#     car_order.order_date,
auto_show=#     brand.brand,
auto_show=#     model.model,
auto_show=#     car.price,
auto_show=#     car.vehicle_year
auto_show=# FROM
auto_show=#     car_order
auto_show=# NATURAL JOIN car
auto_show=# NATURAL JOIN model
auto_show=# NATURAL JOIN brand
auto_show=# WHERE brand.brand = 'Bugatti' or brand.brand = 'Lamborghini';
 order_id | order_date | brand      | model      | price      | vehicle_year
-----+-----+-----+-----+-----+-----
          1 | 2024-10-01 | Bugatti    | Chiron     | 400000000  | 2021
          2 | 2021-01-05 | Bugatti    | Divo       | 985000000  | 2021
          7 | 2024-10-23 | Lamborghini | Huracan Tecnica | 409000000  | 2024
(3 строки)

auto_show=# |

```

**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).

```

auto_show=# SELECT
auto_show=#     car_order.order_id,
auto_show=#     car_order.order_date,
auto_show=#     brand.brand,
auto_show=#     model.model,
auto_show=#     car.price,
auto_show=#     car.vehicle_year
auto_show=# FROM
auto_show=#     car_order
auto_show=# NATURAL JOIN car
auto_show=# NATURAL JOIN model
auto_show=# NATURAL JOIN brand
auto_show=# WHERE brand.brand = 'Bugatti'
auto_show=# UNION
auto_show=# SELECT
auto_show=#     car_order.order_id,
auto_show=#     car_order.order_date,
auto_show=#     brand.brand,
auto_show=#     model.model,
auto_show=#     car.price,
auto_show=#     car.vehicle_year
auto_show=# FROM
auto_show=#     car_order
auto_show=# NATURAL JOIN car
auto_show=# NATURAL JOIN model
auto_show=# NATURAL JOIN brand
auto_show=# WHERE brand.brand = 'Lamborghini';
 order_id | order_date | brand      | model      | price      | vehicle_year
-----+-----+-----+-----+-----+-----
          7 | 2024-10-23 | Lamborghini | Huracan Tecnica | 409000000  | 2024
          1 | 2024-10-01 | Bugatti    | Chiron     | 400000000  | 2021
          2 | 2021-01-05 | Bugatti    | Divo       | 985000000  | 2021
(3 строки)

auto_show=# |

```

**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».

```
auto_show=# SELECT c.client_id, c.firstname, c.surname
auto_show=# FROM client c
auto_show=# WHERE EXISTS (
auto_show(# SELECT *
auto_show(# FROM car_order co
auto_show(# NATURAL JOIN order_status os
auto_show(# WHERE co.client_id = c.client_id AND os.status = 'Доставлен'
auto_show(# )
auto_show=# AND EXISTS (
auto_show(# SELECT *
auto_show(# FROM car_order co
auto_show(# NATURAL JOIN order_status os
auto_show(# WHERE co.client_id = c.client_id AND os.status = 'Ожидается'
auto_show(# );
 client_id | firstname | surname
-----+-----+-----
      13  | Владислав | Кликушин
(1 строка)

auto_show=# |
```

Рисунок 3.2.5.1 – Запрос, реализующий операцию пересечения

Запрос переписан с использованием оператора INTERSECT (Рисунок 3.2.5.2).

```
auto_show=# SELECT c.client_id, c.firstname, c.surname
auto_show=# FROM client c
auto_show=# NATURAL JOIN car_order
auto_show=# NATURAL JOIN order_status
auto_show=# WHERE order_status.status = 'Доставлен'
auto_show=# INTERSECT
auto_show=# SELECT c.client_id, c.firstname, c.surname
auto_show=# FROM client c
auto_show=# NATURAL JOIN car_order
auto_show=# NATURAL JOIN order_status
auto_show=# WHERE order_status.status = 'Ожидается';
 client_id | firstname | surname
-----+-----+-----
      13  | Владислав | Кликушин
(1 строка)

auto_show=# |
```

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.

```

auto_show=# SELECT s.seller_id, s.firstname, s.surname, s.patronymic
auto_show=# FROM seller s
auto_show=# JOIN contract c1 ON s.seller_id = c1.seller_id
auto_show=# JOIN car_order co1 ON c1.order_id = co1.order_id
auto_show=# JOIN car car1 ON co1.car_id = car1.car_id
auto_show=# JOIN engine e1 ON car1.engine_id = e1.engine_id
auto_show=# WHERE e1.engine_power > 700
auto_show=# INTERSECT
auto_show=# SELECT s.seller_id, s.firstname, s.surname, s.patronymic
auto_show=# FROM seller s
auto_show=# JOIN contract c2 ON s.seller_id = c2.seller_id
auto_show=# JOIN car_order co2 ON c2.order_id = co2.order_id
auto_show=# JOIN car car2 ON co2.car_id = car2.car_id
auto_show=# JOIN engine e2 ON car2.engine_id = e2.engine_id
auto_show=# WHERE e2.engine_power < 700;
  seller_id | firstname | surname | patronymic
-----+-----+-----+-----
          7 | Галина   | Богомольная | Владимировна
(1 строка)

auto_show=# |

```

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.

```

auto_show=# SELECT s.seller_id, s.firstname, s.surname
auto_show=# FROM seller s
auto_show=# WHERE EXISTS (
auto_show(#      SELECT 1
auto_show(#      FROM contract c1
auto_show(#      JOIN car_order co1 ON c1.order_id = co1.order_id
auto_show(#      JOIN car car1 ON co1.car_id = car1.car_id
auto_show(#      JOIN engine e1 ON car1.engine_id = e1.engine_id
auto_show(#      WHERE c1.seller_id = s.seller_id AND e1.engine_power > 700
auto_show(# )
auto_show=# AND EXISTS (
auto_show(#      SELECT 1
auto_show(#      FROM contract c2
auto_show(#      JOIN car_order co2 ON c2.order_id = co2.order_id
auto_show(#      JOIN car car2 ON co2.car_id = car2.car_id
auto_show(#      JOIN engine e2 ON car2.engine_id = e2.engine_id
auto_show(#      WHERE c2.seller_id = s.seller_id AND e2.engine_power < 700
auto_show(# );
  seller_id | firstname | surname
-----+-----+-----
          7 | Галина   | Богомольная
(1 строка)

auto_show=# |

```

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.

```
auto_show=# SELECT DISTINCT s.seller_id, s.firstname, s.surname
auto_show=# FROM seller s
auto_show=# WHERE EXISTS (
auto_show(#      SELECT 1
auto_show(#      FROM contract c
auto_show(#      JOIN car_order o ON c.order_id = o.order_id
auto_show(#      WHERE c.seller_id = s.seller_id AND EXTRACT(YEAR FROM o.order_date) = 2024
auto_show(# )
auto_show=# AND NOT EXISTS (
auto_show(#      SELECT 1
auto_show(#      FROM contract c
auto_show(#      JOIN car_order o ON c.order_id = o.order_id
auto_show(#      WHERE c.seller_id = s.seller_id AND EXTRACT(YEAR FROM o.order_date) = 2023
auto_show(# );
seller_id | firstname | surname
-----+-----+-----
1 | Людмила | Скворцова
3 | Михаил | Рысин
4 | Марина | Туманова
6 | Иван | Зайцев
7 | Галина | Богомольная
(5 строк)
auto_show=# |
```

Рисунок 3.2.6.1 – Запрос, реализующий операцию разности

Этот запрос с использованием оператора EXCEPT представлен на Рисунке 3.2.6.2.

```
auto_show=# SELECT DISTINCT s.seller_id, s.firstname, s.surname
auto_show=# FROM seller s
auto_show=# JOIN (
auto_show(#      SELECT c.seller_id
auto_show(#      FROM contract c
auto_show(#      JOIN car_order o ON c.order_id = o.order_id
auto_show(#      WHERE EXTRACT(YEAR FROM o.order_date) = 2024
auto_show(#      EXCEPT
auto_show(#      SELECT c.seller_id
auto_show(#      FROM contract c
auto_show(#      JOIN car_order o ON c.order_id = o.order_id
auto_show(#      WHERE EXTRACT(YEAR FROM o.order_date) = 2023
auto_show(# ) subquery ON s.seller_id = subquery.seller_id;
seller_id | firstname | surname
-----+-----+-----
1 | Людмила | Скворцова
3 | Михаил | Рысин
4 | Марина | Туманова
6 | Иван | Зайцев
7 | Галина | Богомольная
(5 строк)
auto_show=# |
```

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.

```
auto_show=# SELECT brand, SUM(car.price) AS total_price
auto_show=# FROM car
auto_show=# NATURAL JOIN model
auto_show=# NATURAL JOIN brand
auto_show=# GROUP BY brand;
 brand      | total_price
-----+-----
Lamborghini | 140100000
Bugatti     | 138500000
Xiaomi      | 6250000
Mercedes-Benz | 18690000
Nissan       | 15750000
(5 строк)

auto_show=# |
```

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.

```
auto_show=# SELECT c.vehicle_year, AVG(c.mileage) AS avg_mileage
auto_show=# FROM car c
auto_show=# GROUP BY c.vehicle_year;
 vehicle_year | avg_mileage
-----+-----
2017          | 1800.0000000000000000
2012          | 126300.000000000000
2024          | 16.2500000000000000
2021          | 350.5000000000000000
(4 строки)

auto_show=# |
```

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.

```

auto_show=# SELECT s.firstname, s.surname, COUNT(*) AS cars_sold
auto_show=# FROM seller s
auto_show=# JOIN contract c ON s.seller_id = c.seller_id
auto_show=# GROUP BY s.seller_id, s.firstname, s.surname;

```

firstname	surname	cars_sold
Михаил	Рысин	1
Ирина	Куликова	1
Марина	Туманова	1
Иван	Зайцев	1
Александр	Филатов	1
Галина	Богомольная	2
Людмила	Скворцова	1

(7 строк)

```

auto_show=# |

```

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.

```

auto_show=# SELECT m.model, MAX(c.price) AS max_price, MIN(c.price) AS min_price
auto_show=# FROM car c
auto_show=# JOIN model m ON c.model_id = m.model_id
auto_show=# GROUP BY m.model;

```

model	max_price	min_price
GLS 450	18690000	18690000
Huracan Tecnica	40900000	40900000
Chiron	400000000	400000000
Divo	985000000	985000000
GT-R	14500000	14500000
Juke	1250000	1250000
SU7	6250000	6250000
Aventador SVJ	99200000	99200000

(8 строк)

```

auto_show=# |

```

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.

```
auto_show=# SELECT b.brand, COUNT(m.model_id) AS model_count
auto_show=# FROM brand b
auto_show=# INNER JOIN model m ON b.brand_id = m.brand_id
auto_show=# GROUP BY b.brand_id, b.brand
auto_show=# ORDER BY model_count DESC;
```

brand	model_count
Lamborghini	2
Nissan	2
Bugatti	2
Mercedes-Benz	1
Xiaomi	1

(5 строк)

```
auto_show=# |
```

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.

```
auto_show=# SELECT car_id, brand, model, mileage, engine_power
auto_show=# FROM car
auto_show=# NATURAL JOIN engine
auto_show=# NATURAL JOIN model
auto_show=# NATURAL JOIN brand
auto_show=# ORDER BY engine.engine_power DESC, car.mileage ASC;
```

car_id	brand	model	mileage	engine_power
1	Bugatti	Chiron	1	1500
2	Bugatti	Divo	700	1500
8	Lamborghini	Aventador SVJ	0	770
5	Xiaomi	SU7	55	673
7	Lamborghini	Huracan Tecnica	0	640
4	Nissan	GT-R	1800	555
6	Mercedes-Benz	GLS 450	10	367
3	Nissan	Juke	126300	117

(8 строк)

```
auto_show=# |
```

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.

```

auto_show=# SELECT s.country, s.email, COUNT(car_order.order_id) AS cars_delivered
auto_show=# FROM supplier s
auto_show=# NATURAL JOIN car_order
auto_show=# WHERE car_order.status_id IN (2, 3)
auto_show=# GROUP BY s.supplier_id, s.country, s.email
auto_show=# ORDER BY cars_delivered DESC;
country |          email          | cars_delivered
-----+-----+-----
Индия   | sorokin_a@mirea.ru     | 2
Россия  | dzerzhinskij@mirea.ru  | 2
Египет  | zheleznyak@mirea.ru    | 1
Грузия  | puturidze@mirea.ru     | 1
(4 строки)
auto_show=# |

```

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.

```

auto_show=# SELECT s.supplier_id, s.email, s.country
auto_show=# FROM supplier s
auto_show=# WHERE NOT EXISTS (
auto_show(#       SELECT et.engine_type_id
auto_show(#       FROM engine_type et
auto_show(#       WHERE NOT EXISTS (
auto_show(#         SELECT co.supplier_id
auto_show(#         FROM car_order co
auto_show(#         JOIN car ca ON co.car_id = ca.car_id
auto_show(#         JOIN engine e ON ca.engine_id = e.engine_id
auto_show(#         WHERE co.supplier_id = s.supplier_id AND e.engine_type_id = et.engine_type_id
auto_show(#       )
auto_show(# );
supplier_id |          email          | country
-----+-----+-----
2 | zheleznyak@mirea.ru    | Египет
(1 строка)
auto_show=# |

```

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).

```
auto_show=# CREATE VIEW Order_Info AS
auto_show=# SELECT
auto_show=#     co.order_id,
auto_show=#     cl.firstname AS client_firstname,
auto_show=#     cl.surname AS client_surname,
auto_show=#     cl.patronymic AS client_patronymic,
auto_show=#     cl.passport as client_passport,
auto_show=#     co.order_date,
auto_show=#     car.price AS car_price,
auto_show=#     car.mileage AS car_mileage,
auto_show=#     car.vehicle_year AS car_year,
auto_show=#     b.brand AS car_brand,
auto_show=#     m.model AS car_model
auto_show=# FROM car_order co
auto_show=# INNER JOIN client cl ON co.client_id = cl.client_id
auto_show=# JOIN car ON co.car_id = car.car_id
auto_show=# JOIN model m ON car.model_id = m.model_id
auto_show=# JOIN brand b ON m.brand_id = b.brand_id;
CREATE VIEW
```

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».

auto_show=# SELECT * FROM Order_Info WHERE client_surname LIKE 'K%';										
order_id	client_firstname	client_surname	client_patronymic	client_passport	order_date	car_price	car_mileage	car_year	car_brand	car_model
4	Александр	Корольков	Дмитриевич	6324929817	2024-10-22	14500000	1800	2017	Nissan	GT-R
5	Владислав	Кликушин	Игоревич	6324738250	2022-10-05	6250000	55	2024	Xiaomi	SU7
6	Данил	Коротков	Игоревич	6324579623	2024-10-24	18690000	10	2024	Mercedes-Benz	GLS 450
8	Владислав	Кликушин	Игоревич	6324738250	2024-12-09	99200000	0	2024	Lamborghini	Aventador SVJ

(4 строки)

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).

```

auto_show=# CREATE OR REPLACE PROCEDURE add_client(
auto_show(#      p_firstname VARCHAR,
auto_show(#      p_surname VARCHAR,
auto_show(#      p_patronymic VARCHAR,
auto_show(#      p_passport BIGINT
auto_show(# )
auto_show=# LANGUAGE plpgsql
auto_show=# AS $$
auto_show$# BEGIN
auto_show$#      INSERT INTO client (firstname, surname, patronymic, passport)
auto_show$#      VALUES (p_firstname, p_surname, p_patronymic, p_passport);
auto_show$# END;
auto_show$# $$;
CREATE PROCEDURE

```

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.

```

auto_show=# SELECT * FROM client WHERE firstname LIKE 'Л%';
 client_id | firstname | surname | patronymic | passport
-----+-----+-----+-----+-----
          7 | Леонид   | Егоров  | Александрович | 6324601436
          29 | Леонид   | Яськов  | Владимирович  | 6324866341
(2 строки)

auto_show=# |

```

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.

```

auto_show=# CALL add_client('Леонид', 'Еськов', 'Владимирович', 777777777);
CALL
auto_show=# SELECT * FROM client WHERE firstname like 'Л%';
 client_id | firstname | surname | patronymic | passport
-----+-----+-----+-----+-----
          7 | Леонид   | Егоров  | Александрович | 6324601436
          29 | Леонид   | Яськов  | Владимирович  | 6324866341
          31 | Леонид   | Еськов  | Владимирович  | 7777777777
(3 строки)

auto_show=# |

```

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.

```

16 | Дмитрий | Орехов | Сергеевич | 6324258070
17 | Александр | Основин | Игоревич | 6324517967
18 | Кирилл | Павлов | Сергеевич | 6324691698
19 | Александра | Преснякова | Владимировна | 6324141182
20 | Юлия | Рапорт | 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 | Шариков | Рамазанович | 6324438243
28 | Павел | Яковлев | Андреевич | 6324690550
29 | Леонид | Яськов | Владимирович | 6324866341
30 | Данила | Яшин | Олегович | 6324060179
31 | Леонид | Еськов | Владимирович | 7777777777
(31 строка)

auto_show=# CALL delete_client(31);
CALL
auto_show=# SELECT * FROM client WHERE firstname like 'Л%';
 client_id | firstname | surname | patronymic | passport
-----+-----+-----+-----+-----
          7 | Леонид | Егоров | Александрович | 6324601436
         29 | Леонид | Яськов | Владимирович | 6324866341
(2 строки)

auto_show=# |

```

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).

```

auto_show=# CREATE OR REPLACE FUNCTION count_orders_by_status(p_status_id SMALLINT)
auto_show=# RETURNS INT
auto_show=# LANGUAGE plpgsql
auto_show=# AS $$
auto_show$# DECLARE
auto_show$#     order_count INT := 0;
auto_show$#     order_record RECORD;
auto_show$# BEGIN
auto_show$#     FOR order_record IN SELECT order_id FROM car_order WHERE status_id = p_status_id LOOP
auto_show$#         order_count := order_count + 1;
auto_show$#     END LOOP;
auto_show$#     RETURN order_count;
auto_show$# END;
auto_show$# $$;
CREATE FUNCTION

```

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

Вызов функции представлен на Рисунке 3.3.2.2.

```

auto_show=# SELECT count_orders_by_status(1::SMALLINT);
count_orders_by_status
-----
3
(1 строка)

auto_show=# |

```

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).

```

auto_show=# CREATE OR REPLACE FUNCTION total_order_cost_by_client(p_client_id INT)
auto_show=# RETURNS BIGINT
auto_show=# LANGUAGE plpgsql
auto_show=# AS $$
auto_show$# DECLARE
auto_show$#     total_cost BIGINT := 0;
auto_show$#     order_record RECORD;
auto_show$# BEGIN
auto_show$#     FOR order_record IN
auto_show$#         SELECT c.price FROM car_order co
auto_show$#         JOIN car c ON co.car_id = c.car_id
auto_show$#         WHERE co.client_id = p_client_id
auto_show$#     LOOP
auto_show$#         total_cost := total_cost + order_record.price;
auto_show$#     END LOOP;
auto_show$#     RETURN total_cost;
auto_show$# END;
auto_show$# $$;
CREATE FUNCTION
auto_show=# |

```

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

Вызов этой функции представлен на Рисунке 3.3.2.4.

```

auto_show=# SELECT client, total_order_cost_by_client(client_id) FROM client;
client | total_order_cost_by_client
-----|-----
(1, Александр, Арефьев, Михайлович, 6324170051) | 400000000
(2, Александр, Васильев, Станиславович, 6324548740) | 1180000
(3, Даниил, Волков, Андреевич, 6324733970) | 409000000
(4, Валерия, Воробей, Глебовна, 6324890496) | 985000000
(5, Егор, Гасилин, Денисович, 6324275675) | 6000000
(6, Ольга, Довбуш, Александровна, 6324602322) | 0
(7, Леонид, Егоров, Александрович, 6324601436) | 1250000
(8, Николай, Заковряшин, Михайлович, 6324587891) | 0
(9, Эдуард, Исаков, Вячеславович, 6324156944) | 0
(10, Александр, Калмыков, Михайлович, 6324057936) | 0
(11, Софья, Капитонова, "Романовна", 6324458970) | 0
(12, Арина, Карева, Александровна, 6324262081) | 0
(13, Владислав, Кликушин, Игоревич, 6324738250) | 105450000
(14, Александр, Корольков, Дмитриевич, 6324929817) | 14500000
(15, Данил, Коротков, Игоревич, 6324579623) | 186900000
(16, Дмитрий, Орехов, Сергеевич, 6324258070) | 0
(17, Александр, Основин, Игоревич, 6324517967) | 0
(18, Кирилл, Павлов, Сергеевич, 6324691698) | 0
(19, Александра, Преснякова, Владимировна, 6324141182) | 0
(20, Юлия, "Рапопорт", Юрьевна, 6324321841) | 0
(21, Владимир, "Расцепкин", Антонович, 6324880154) | 0
(22, Егор, "Ромашов", Алексеевич, 6324768917) | 0
(23, Илья, "Рудиков", Михайлович, 6324643746) | 0
(24, Тимур, Сариков, Аслиддинович, 6324113058) | 0
(25, Илья, Серебренников, Константинович, 6324133236) | 0
(26, Павел, Суховилов, Павлович, 6324511815) | 0
(27, Тимур, Шариков, "Рамазанович", 6324438243) | 0
-- Далее -- |

```

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.

```
auto_show=# CREATE OR REPLACE FUNCTION validate_vehicle_year_update()
auto_show=# RETURNS TRIGGER AS $$
auto_show$# BEGIN
auto_show$#     IF NEW.vehicle_year < 1900 OR NEW.vehicle_year > EXTRACT(YEAR FROM CURRENT_DATE) THEN
auto_show$#         RAISE EXCEPTION 'Некорректный год выпуска автомобиля: %', NEW.vehicle_year;
auto_show$#     END IF;
auto_show$#     RETURN NEW;
auto_show$# END;
auto_show$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
auto_show=# |
```

Рисунок 3.3.3.1 – Процедура для триггера

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

```
auto_show=# CREATE TRIGGER before_car_update
auto_show=# BEFORE UPDATE ON car
auto_show=# FOR EACH ROW
auto_show=# EXECUTE FUNCTION validate_vehicle_year_update();
CREATE TRIGGER
auto_show=# |
```

Рисунок 3.3.3.2 – Триггер на операцию UPDATE для таблицы car

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

```
auto_show=# SELECT * FROM car;
 car_id | model_id | drive_id | price  | mileage | vehicle_year | color_id | engine_id
-----+-----+-----+-----+-----+-----+-----+-----
      1 |         |         | 400000000 |      1 |          2021 |         |         1
      2 |         |         | 985000000 |     700 |          2021 |         |         1
      3 |         |         | 1250000   | 126300 |          2012 |         |         2
      4 |         |         | 145000000 |    1800 |          2017 |         |         3
      5 |         |         | 625000000 |     55 |          2024 |         |         4
      6 |         |         | 186900000 |     10 |          2024 |         |         5
      7 |         |         | 409000000 |      0 |          2024 |         |         6
      8 |         |         | 992000000 |      0 |          2024 |         |         7
      9 |         |         | 118000000 | 299900 |          2005 |         |         8
     10 |         |         | 600000000 |     37 |          2024 |         |         4
(10 строк)

auto_show=# UPDATE car SET vehicle_year = 1600 WHERE car_id = 1;
ОШИБКА: Некорректный год выпуска автомобиля: 1600
КОНТЕКСТ: функция PL/pgSQL validate_vehicle_year_update(), строка 4, оператор RAISE
auto_show=# |
```

Рисунок 3.3.3.3 – Оработка 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.

```
auto_show=# SELECT DISTINCT
auto_show=#     supplier.country,
auto_show=#     SUM(car.price) OVER (PARTITION BY supplier.country) AS total_price_by_country
auto_show=# FROM
auto_show=#     car_order
auto_show=# NATURAL JOIN car
auto_show=# NATURAL JOIN supplier;
 country | total_price_by_country
-----+-----
 Египет  |          992180000
 Индия   |         100450000
 Грузия  |          24940000
 Россия  |         455400000
(4 строки)

auto_show=# |
```

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.

```
auto_show=# SELECT
auto_show=#     drive.drive_type,
auto_show=#     engine.engine_power,
auto_show=#     AVG(engine.engine_power) OVER (PARTITION BY drive.drive_type) AS avg_power_by_drive
auto_show=# FROM
auto_show=#     car
auto_show=# NATURAL JOIN drive
auto_show=# NATURAL JOIN engine;
 drive_type | engine_power | avg_power_by_drive
-----+-----+-----
 Задний    |          640 | 640.000000000000000
 Передний  |          117 | 117.000000000000000
 Полный    |          555 | 776.500000000000000
 Полный    |          673 | 776.500000000000000
 Полный    |          367 | 776.500000000000000
 Полный    |          770 | 776.500000000000000
 Полный    |          174 | 776.500000000000000
 Полный    |         1500 | 776.500000000000000
 Полный    |          673 | 776.500000000000000
 Полный    |         1500 | 776.500000000000000
(10 строк)

auto_show=# |
```

Рисунок 4.1.2 - Выборка с агрегирующей оконной функцией AVG



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

```

auto_show=# SELECT
auto_show=#     order_date,
auto_show=#     COUNT(*) OVER (PARTITION BY order_date) AS total_orders
auto_show=# FROM
auto_show=#     car_order;
order_date | total_orders
-----+-----
2021-01-05 |          1
2022-10-05 |          1
2024-10-01 |          1
2024-10-05 |          1
2024-10-22 |          1
2024-10-23 |          1
2024-10-24 |          1
2024-12-09 |          1
2024-12-10 |          1
2024-12-12 |          1
(10 строк)
auto_show=#

```

Рисунок 4.1.3 - Выборка с агрегирующей оконной функцией COUNT

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

```

auto_show=# SELECT
auto_show=#     brand.brand,
auto_show=#     car.price,
auto_show=#     MIN(car.price) OVER (PARTITION BY brand.brand_id) AS min_price_by_brand
auto_show=# FROM
auto_show=#     car
auto_show=# INNER JOIN model ON car.model_id = model.model_id
auto_show=# INNER JOIN brand ON model.brand_id = brand.brand_id;
brand      | price  | min_price_by_brand
-----+-----+-----
Bugatti    | 400000000 | 400000000
Bugatti    | 985000000 | 400000000
Nissan     | 1180000   | 1180000
Nissan     | 1250000   | 1180000
Nissan     | 145000000 | 1180000
Xiaomi     | 6000000   | 6000000
Xiaomi     | 6250000   | 6000000
Mercedes-Benz | 18690000 | 18690000
Lamborghini | 99200000 | 40900000
Lamborghini | 40900000 | 40900000
(10 строк)
auto_show=#

```

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.

```

auto_show=# SELECT
auto_show=#     drive.drive_type,
auto_show=#     engine.engine_power,
auto_show=#     MAX(engine.engine_power) OVER (PARTITION BY drive.drive_type) AS max_power_by_drive
auto_show=# FROM
auto_show=#     car
auto_show=# NATURAL JOIN drive
auto_show=# NATURAL JOIN engine;
 drive_type | engine_power | max_power_by_drive
-----+-----+-----
Задний      |          640 |          640
Передний    |          117 |          117
Полный      |          555 |         1500
Полный      |          673 |         1500
Полный      |          367 |         1500
Полный      |          770 |         1500
Полный      |          174 |         1500
Полный      |         1500 |         1500
Полный      |          673 |         1500
Полный      |         1500 |         1500
(10 строк)

auto_show=# |

```

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.

```

auto_show=# SELECT
auto_show=#     client_id,
auto_show=#     firstname,
auto_show=#     surname,
auto_show=#     order_date,
auto_show=#     ROW_NUMBER() OVER (PARTITION BY client_id ORDER BY order_date) AS order_number
auto_show=# FROM
auto_show=#     car_order
auto_show=# NATURAL JOIN client;
 client_id | firstname | surname | order_date | order_number
-----+-----+-----+-----+-----
          1 | Александр | Арефьев | 2024-10-01 |          1
          2 | Александр | Васильев | 2024-12-10 |          1
          3 | Даниил   | Волков  | 2024-10-23 |          1
          4 | Валерия  | Воробей | 2021-01-05 |          1
          5 | Егор     | Гасилин | 2024-12-12 |          1
          7 | Леонид   | Егоров  | 2024-10-05 |          1
         13 | Владислав | Кликушин | 2022-10-05 |          1
         13 | Владислав | Кликушин | 2024-12-09 |          2
         14 | Александр | Корольков | 2024-10-22 |          1
         15 | Данил    | Коротков | 2024-10-24 |          1
(10 строк)

auto_show=# |

```

Рисунок 4.2.1 - Выборка с ранжирующей оконной функцией ROW\_NUMBER

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

```

auto_show=# SELECT
auto_show=#     car_id,
auto_show=#     brand,
auto_show=#     model,
auto_show=#     vehicle_year,
auto_show=#     RANK() OVER (ORDER BY vehicle_year ASC) AS age_rank
auto_show=# FROM car
auto_show=# NATURAL JOIN model
auto_show=# NATURAL JOIN brand;
  car_id |      brand      |      model      | vehicle_year | age_rank
-----+-----+-----+-----+-----
      9 | Nissan          | Pathfinder      | 2005         | 1
      3 | Nissan          | Juke            | 2012         | 2
      4 | Nissan          | GT-R           | 2017         | 3
      2 | Bugatti         | Divo            | 2021         | 4
      1 | Bugatti         | Chiron          | 2021         | 4
     10 | Xiaomi          | SU7             | 2024         | 6
      5 | Xiaomi          | SU7             | 2024         | 6
      6 | Mercedes-Benz  | GLS 450         | 2024         | 6
      7 | Lamborghini     | Huracan Tecnica | 2024         | 6
      8 | Lamborghini     | Aventador SVJ   | 2024         | 6
(10 строк)

auto_show=# |

```

Рисунок 4.2.2 - Выборка с ранжирующей оконной функцией RANK

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

```

auto_show=# SELECT
auto_show=#     car_id,
auto_show=#     brand,
auto_show=#     model,
auto_show=#     price,
auto_show=#     DENSE_RANK() OVER (ORDER BY price DESC) AS rank_by_price
auto_show=# FROM car
auto_show=# NATURAL JOIN model
auto_show=# NATURAL JOIN brand;
  car_id |      brand      |      model      |      price      | rank_by_price
-----+-----+-----+-----+-----
      2 | Bugatti         | Divo            | 985000000       | 1
      1 | Bugatti         | Chiron          | 400000000       | 2
      8 | Lamborghini     | Aventador SVJ   | 99200000        | 3
      7 | Lamborghini     | Huracan Tecnica | 40900000        | 4
      6 | Mercedes-Benz  | GLS 450         | 18690000        | 5
      4 | Nissan          | GT-R           | 14500000        | 6
      5 | Xiaomi          | SU7             | 6250000         | 7
     10 | Xiaomi          | SU7             | 6000000         | 8
      3 | Nissan          | Juke            | 1250000         | 9
      9 | Nissan          | Pathfinder      | 1180000         | 10
(10 строк)

auto_show=# |

```

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.

```

auto_show=# SELECT
auto_show=#     car_id,
auto_show=#     brand,
auto_show=#     model,
auto_show=#     vehicle_year,
auto_show=#     NTILE(2) OVER (ORDER BY vehicle_year ASC) AS year_group
auto_show=# FROM car
auto_show=# NATURAL JOIN model
auto_show=# NATURAL JOIN brand;

```

car_id	brand	model	vehicle_year	year_group
9	Nissan	Pathfinder	2005	1
3	Nissan	Juke	2012	1
4	Nissan	GT-R	2017	1
2	Bugatti	Divo	2021	1
1	Bugatti	Chiron	2021	1
10	Xiaomi	SU7	2024	2
5	Xiaomi	SU7	2024	2
6	Mercedes-Benz	GLS 450	2024	2
7	Lamborghini	Huracan Tecnica	2024	2
8	Lamborghini	Aventador SVJ	2024	2

(10 строк)

```

auto_show=# |

```

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).

```

auto_show=# SELECT
auto_show=#     car_id,
auto_show=#     brand,
auto_show=#     model,
auto_show=#     price,
auto_show=#     LEAD(price, 1, price) OVER (ORDER BY price) - price AS price_difference
auto_show=# FROM car
auto_show=# NATURAL JOIN model
auto_show=# NATURAL JOIN brand;

```

car_id	brand	model	price	price_difference
9	Nissan	Pathfinder	1180000	70000
3	Nissan	Juke	1250000	4750000
10	Xiaomi	SU7	6000000	250000
5	Xiaomi	SU7	6250000	8250000
4	Nissan	GT-R	14500000	4190000
6	Mercedes-Benz	GLS 450	18690000	22210000
7	Lamborghini	Huracan Tecnica	40900000	58300000
8	Lamborghini	Aventador SVJ	99200000	300800000
1	Bugatti	Chiron	400000000	585000000
2	Bugatti	Divo	985000000	0

(10 строк)

```

auto_show=# |

```

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.

```

auto_show=# SELECT
auto_show=#     car_id,
auto_show=#     brand,
auto_show=#     model,
auto_show=#     price,
auto_show=#     LAG(price, 1, price) OVER (ORDER BY price DESC) - price AS price_difference
auto_show=# FROM car
auto_show=# NATURAL JOIN model
auto_show=# NATURAL JOIN brand;

```

car_id	brand	model	price	price_difference
2	Bugatti	Divo	985000000	0
1	Bugatti	Chiron	400000000	585000000
8	Lamborghini	Aventador SVJ	99200000	300800000
7	Lamborghini	Huracan Tecnica	40900000	58300000
6	Mercedes-Benz	GLS 450	18690000	22210000
4	Nissan	GT-R	14500000	4190000
5	Xiaomi	SU7	6250000	8250000
10	Xiaomi	SU7	6000000	250000
3	Nissan	Juke	1250000	4750000
9	Nissan	Pathfinder	1180000	70000

(10 строк)

```

auto_show=#

```

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.

```

auto_show=# SELECT DISTINCT
auto_show=#     firstname,
auto_show=#     surname,
auto_show=#     FIRST_VALUE(price) OVER (PARTITION BY client_id ORDER BY price) AS lowest_price,
auto_show=#     LAST_VALUE(price) OVER (PARTITION BY client_id ORDER BY price ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS highest_price
auto_show=# FROM car_order
auto_show=# NATURAL JOIN client
auto_show=# NATURAL JOIN car;

```

firstname	surname	lowest_price	highest_price
Александр	Арефьев	400000000	400000000
Владислав	Кликушин	6250000	99200000
Данил	Коротков	18690000	18690000
Александр	Корольков	14500000	14500000
Александр	Васильев	1180000	1180000
Егор	Гасилин	6000000	6000000
Леонид	Егоров	1250000	1250000
Валерия	Воробей	985000000	985000000
Даниил	Волков	40900000	40900000

(9 строк)

```

auto_show=#

```

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', 'Сарииков', 'Аслиддинович', 6324113058),
('Илья', 'Серебреников', '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', 'Анатолевна', '+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'),
    ('Lamborghini');

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 < 400000000;
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 'K%';
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  Арёфьев  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 '3%';
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 = 'Lamborghini';

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 = 'Lamborghini';
-- 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 'K%';
```

## 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', 777777777);
-- 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;
```