МИНИСТЕРСТВО НАУКИ И ВЫШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРТСВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**Кафедра вычислительной техники**



**Курсовая работа**

**по предмету «Базы данных»**

Разработка базы данных «Велосипеды»

Группа: АВТ-042

Студент: Антонов А.Р.

Преподаватель: Трошина Г. В.

Вариант: 80

Новосибирск, 2022

**Оглавление**

[1. Задание 3](#_Toc121256325)

[2. Введение 5](#_Toc121256326)

[Актуальность 5](#_Toc121256327)

[Цель 5](#_Toc121256328)

[3. Диаграмма связей таблиц 6](#_Toc121256329)

[4. Структуры таблиц базы данных 7](#_Toc121256330)

[5. Содержание таблиц баз данных 1](#_Toc121256331)0

[6. Руководство пользователя 1](#_Toc121256332)4

[7. Тексты запросов и примеры работ 1](#_Toc121256333)6

[8. Анализ результатов и выводы 5](#_Toc121256334)1

[Список использованных источников 5](#_Toc121256335)2

# 

# 1. Задание

Разработать базу данных «Велосипеды». Сформировать несколько таблиц. Предусмотреть: ввод данных, редактирование, просмотр данных. Обязательные требования к базе данных: наличие таблиц-справочников и таблиц, использующих справочники; предусмотреть следующие роли: оператор, пользователь и администратор базы данных. Реализовать следующие типы запросов с использованием нескольких таблиц:

1. Для каждого вида велосипеда (горный, шоссейный, детский, тандем и т.д.) указать сведения о нем (наименование, год выпуска, место изготовления, цена, поставщик, производитель и т.п.).
2. Для каждого вида велосипеда выдать список, отсортированный: по году выпуска, по поставщику, по стоимости.
3. Найти самый дорогой велосипед, самый дешевый, среднюю стоимость.
4. Найти велосипеды с ценой свыше 10000 р. (и любая другая сумма, т.е. предусмотреть ввод цены с клавиатуры).
5. Найти количество велосипедов, выпущенных за определенный период (месяц, 3 месяца, 6 месяцев) среднюю стоимость, за этот же период – самый дорогой велосипед, самый дешевый.
6. Найти долю велосипедов, поступивших из заданного города (выбор города) от общего числа велосипедов.
7. Найти все велосипеды с заданной датой выпуска (ввод даты).
8. Найти все велосипеды заданного поставщика (ввод), чья стоимость находится в заданных пределах (ввод интервала).
9. Найти долю велосипедов, поступивших от заданного поставщика (ввод поставщика) от общего числа поставщиков.
10. Найти все велосипеды заданного года выпуска чья стоимость больше заданной (ввод стоимости).
11. Найти все велосипеды заданного производителя (выбор).
12. Найти долю велосипедов, проданных за определенный период (ввод периода) от общего времени продажи.
13. Найти все велосипеды, поступившие от заданного поставщика (ввод поставщика), чья стоимость больше, чем средняя стоимость велосипедов, поступивших из заданной страны (ввод страны).
14. Найти долю дешевых велосипедов (чья стоимость меньше заданной, ввод стоимости), проданных заданному клиенту (ввод клиента), и в целом.
15. Найти среднюю стоимость велосипедов, проданных за определенный промежуток времени (ввод интервала).
16. Найти все велосипеды, чья стоимость выше, чем средняя стоимость велосипедов заданного производителя (выбор).

# 2. Введение

## Актуальность

Любая современная организация не может обойтись без базы данных. Это учебные заведения, банки, магазины, заводы, любые предприятия и государственные учреждения. Они используют их для перевода данных в электронный вид и объединения данных, а также оперативного доступа к ним. Это позволяет экономить время и средства на затраты.

Актуальность данной работы заключается в том, что разработанную базу данных велосипедов при расширении функционала можно интегрировать в систему продажи данного вида товаров и использовать в коммерческих целях.

## Цель

Разработать базу данных, которая будет содержать в себе информацию об велосипедах и продаже.

# 3. Диаграмма связей таблиц

Ниже предоставлена диаграмма связей, на которой отображены таблицы и их поля в базе данных.

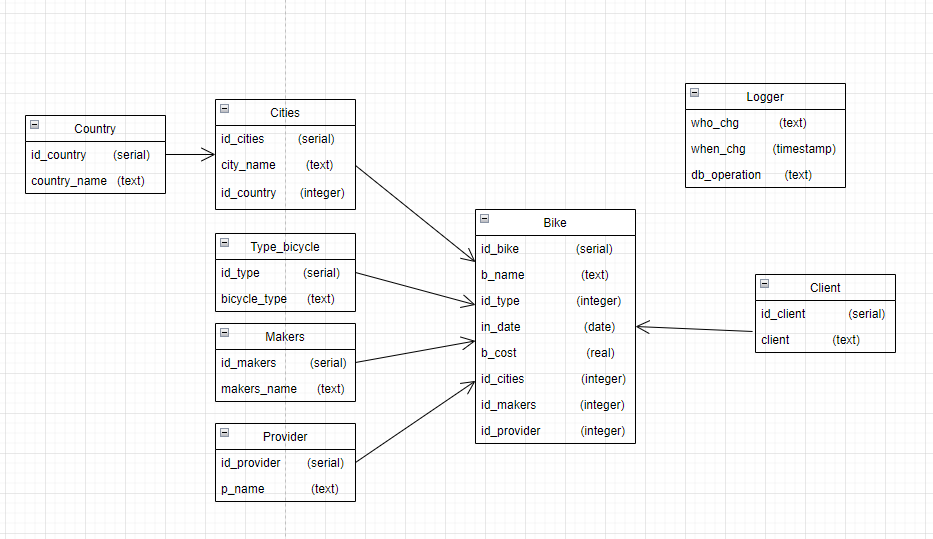


Рисунок 3.1 – UML-диаграма отношения таблиц в базе данных.

# 

# 4. Структуры таблиц базы данных

Ниже предоставлены скриншоты, на которых отображена информация об структурах таблиц базы данных.

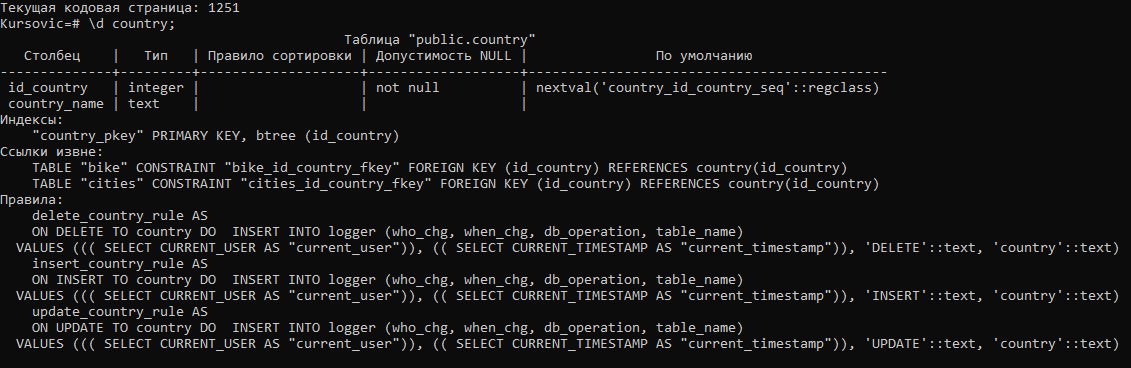


Рисунок 4.1 – Структура таблицы-справочника стран (country).

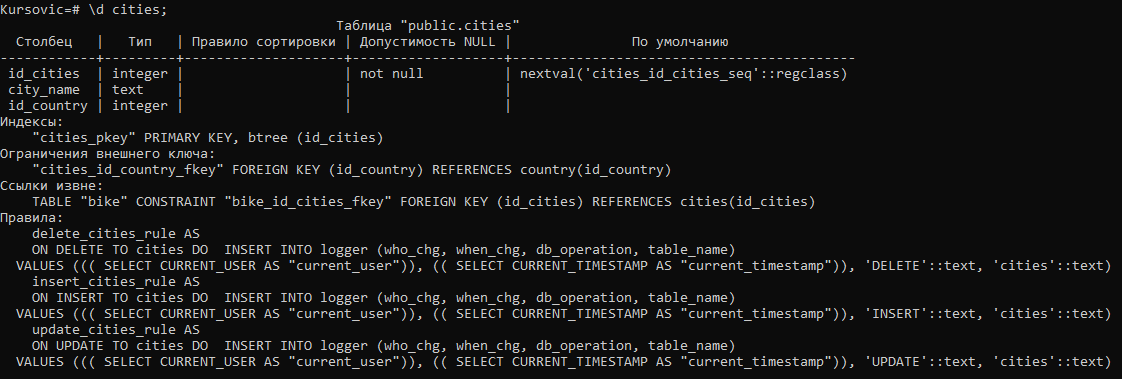


Рисунок 4.2 – Структура таблицы городов (cities).

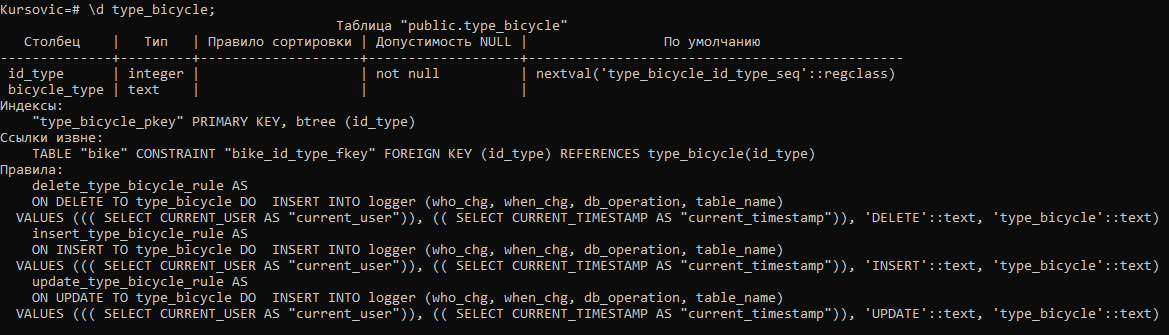


Рисунок 4.3 – Структура таблицы-справочника типов велосипедов (type\_bicycle).

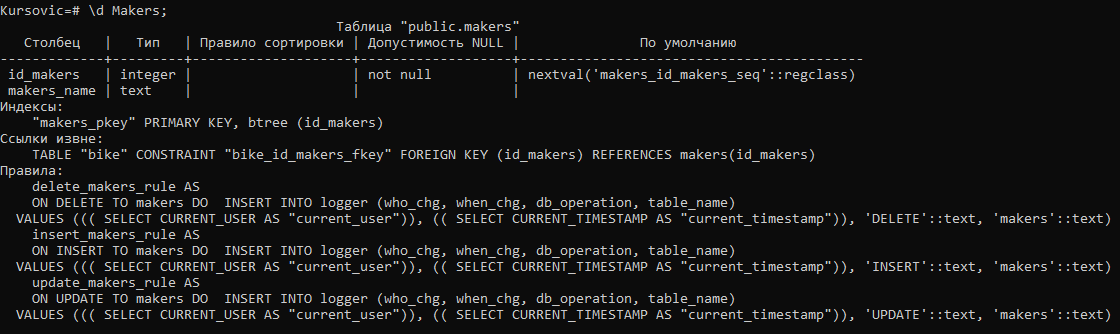


Рисунок 4.4 – Структура таблицы-справочника производителей (makers).

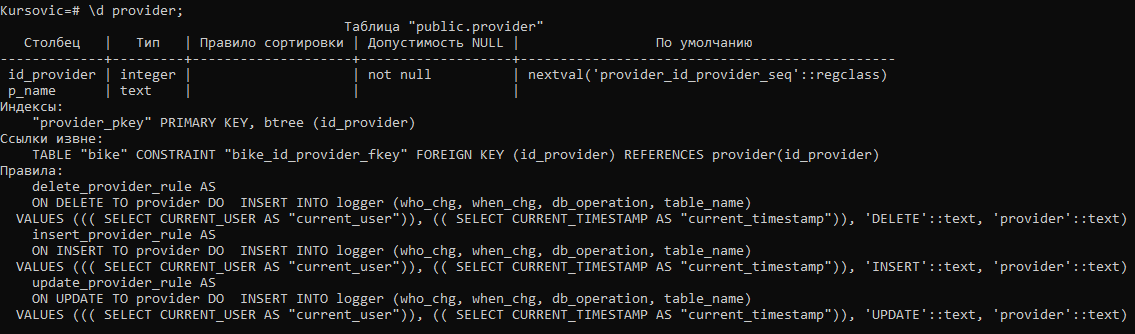


Рисунок 4.5 – Структура таблицы - справочника поставщик (providers).

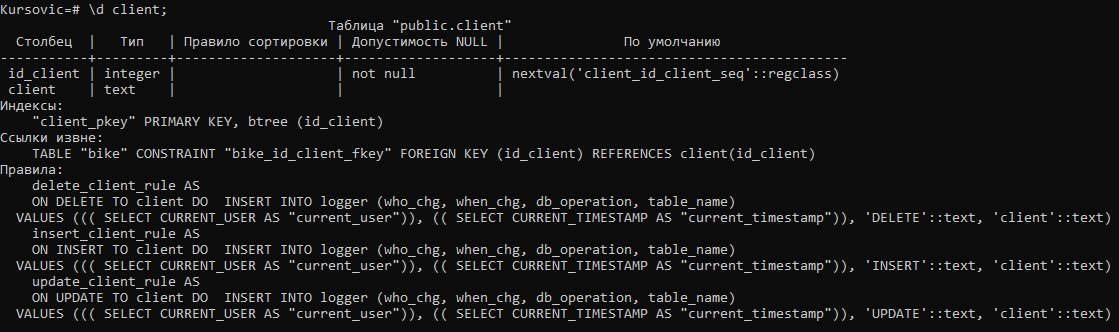


Рисунок 4.6 – Структура таблицы - справочника клиент (client).

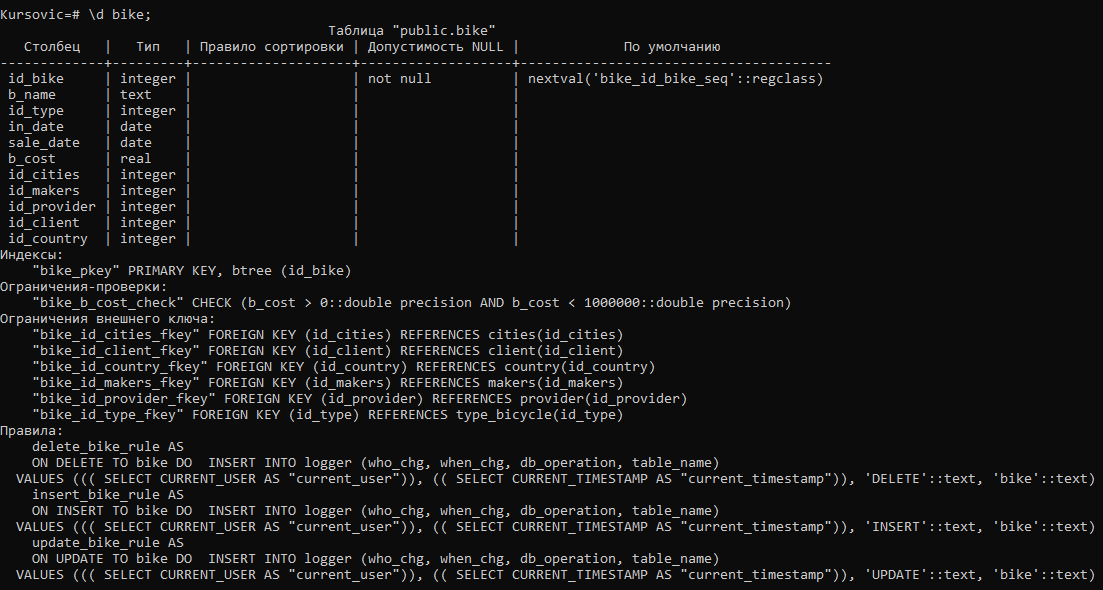


Рисунок 4.7 – Структура таблицы велосипедов (bike).

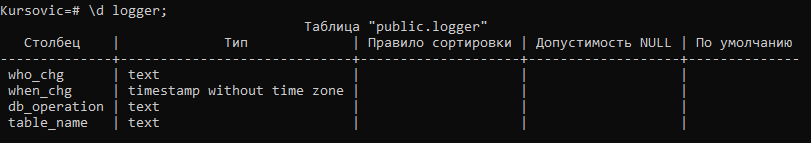


Рисунок 4.8 – Структура таблицы логгирования (logger).

# 5. Содержание таблиц баз данных

Ниже предоставлены скриншоты, на которых отображена информация, содержащаяся в таблицах базы данных.



Рисунок 5.1 – Содержание таблицы - справочник стран (country).

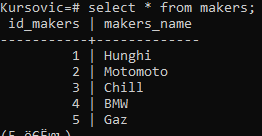


Рисунок 5.2 – Содержание таблицы - справочник производителей (makers).

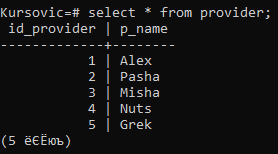


Рисунок 5.3– Содержание таблицы - справочник поставщиков (provider).



Рисунок 5.4 – Содержание таблицы городов (cities).

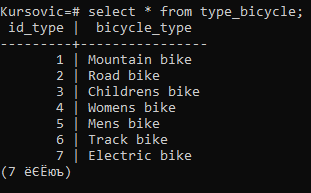


Рисунок 5.5 – Содержание таблицы - справочник типов велосипедов (type\_bicycle).

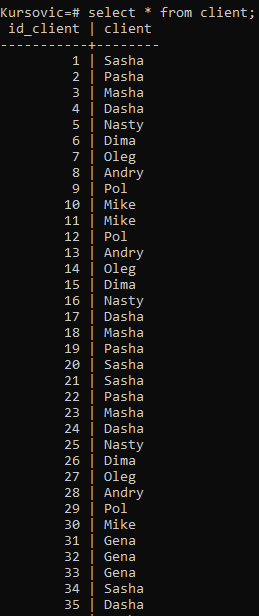


Рисунок 5.6 – Содержание таблицы-справочник клиентов (client).

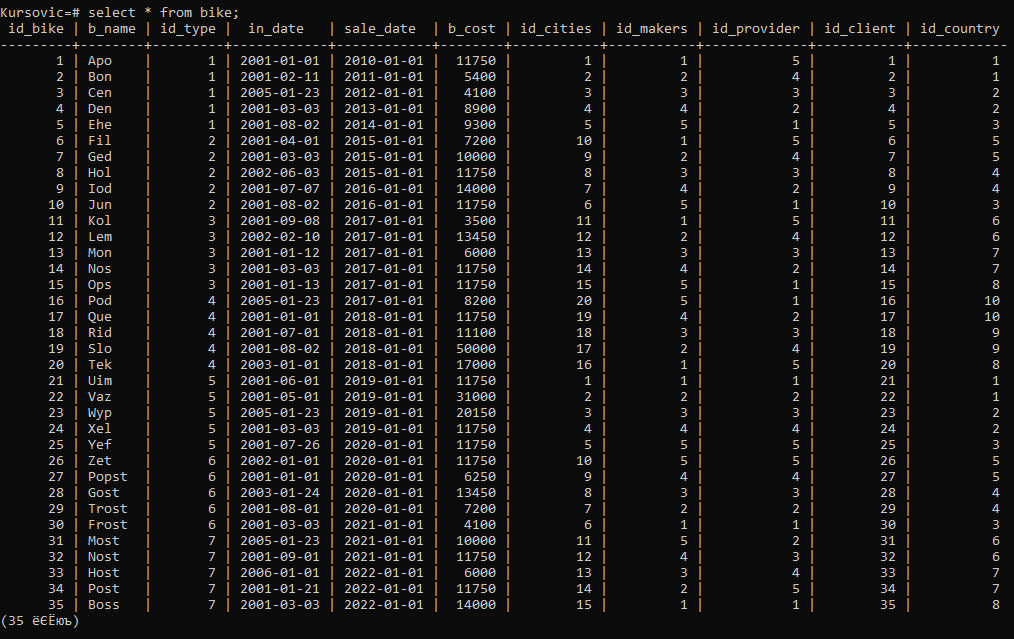


Рисунок 5.7 – Содержание таблицы велосипедов (bike).

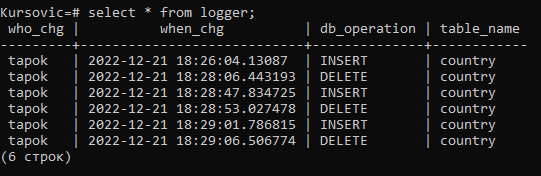


Рисунок 5.8 – Содержание таблицы-журнала операций (logger).

# 6. Руководство пользователя

База данных приборов для освещения предусматривает наличие трех ролей:

* Пользователь (user\_course) – есть права на то, чтобы использовать оператор SELECT на всех таблицах и созданных функциях-запросов; также INSERT и UPDATE на всех таблицах, кроме таблицы-журнала операций logger.
* Оператор (operator\_course) – есть права на то, чтобы использовать оператор SELECT на всех таблицах и созданных функциях-запросов; также INSERT и UPDATE на таблице-журнале logger.
* Администратор (admin\_course) – является суперпользователем с возможностью создавать базы данных и роли, также единственный пользователь, который может удалять данные в таблицах.

Ниже на скриншотах приведены примеры ограничений и возможностей для пользователей.

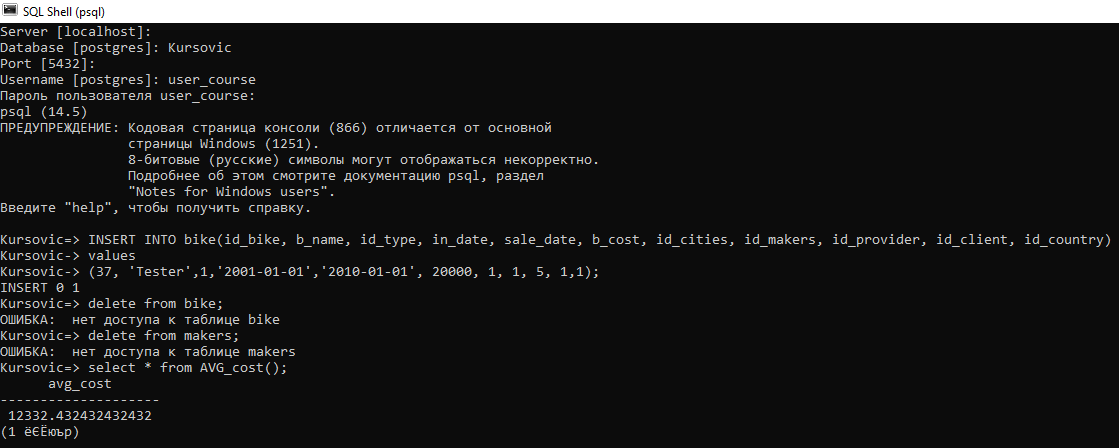


Рисунок 6.1 – Демонстрация возможностей пользователя (user\_course).

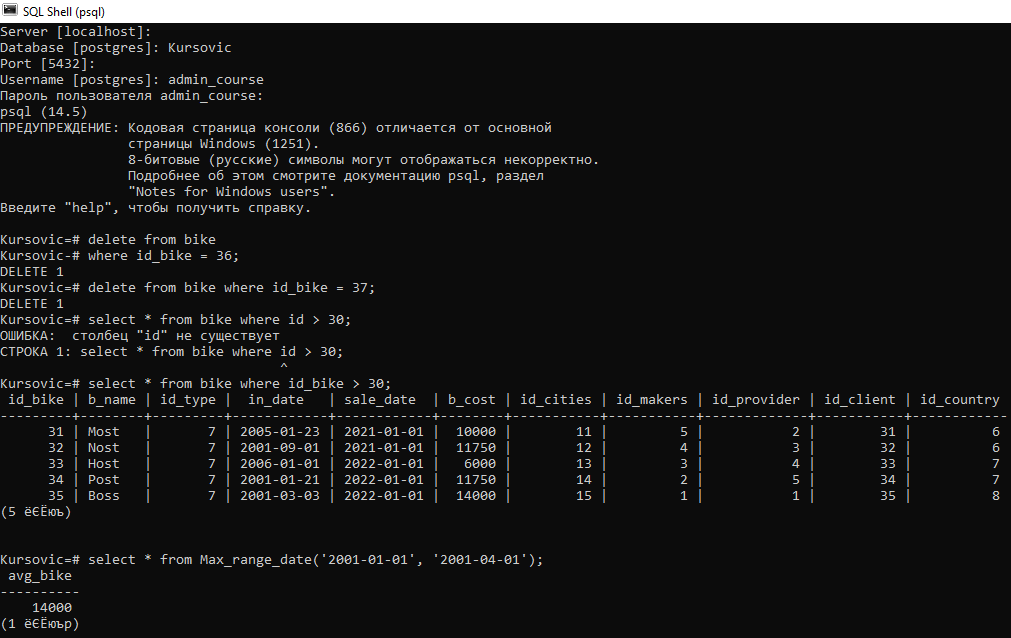


Рисунок 6.2 – Демонстрация возможностей администратора (admin\_course).

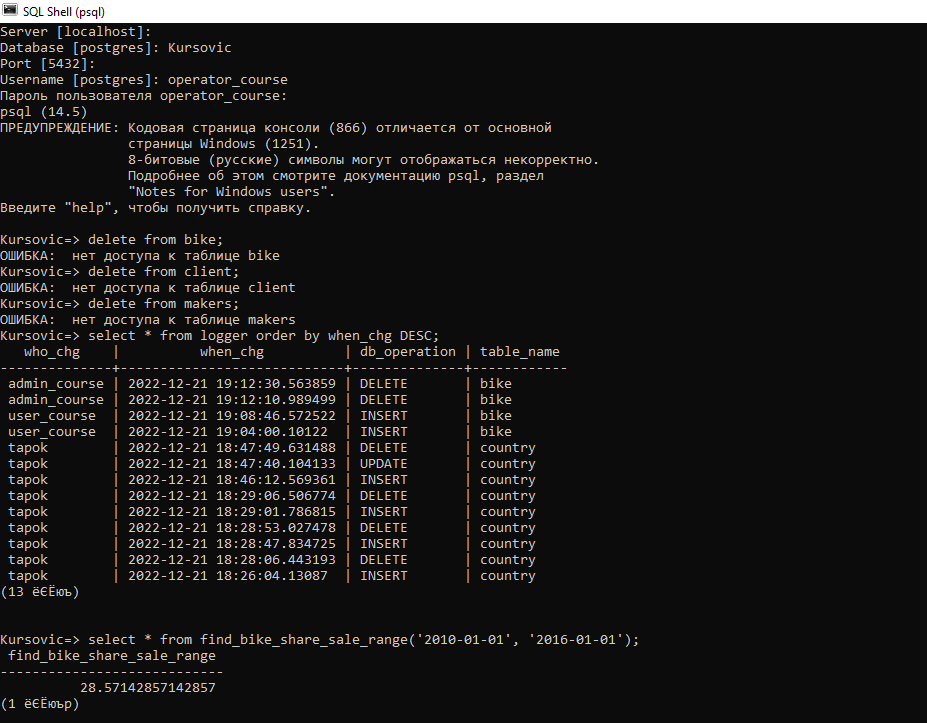


Рисунок 6.3 – Демонстрация возможностей оператора (operator\_course).

# 7. Тексты запросов и примеры работ

***1 пункт***

В данном пункте производилось создание правил для работы таблицы-журнала logger; создание ролей, которые описаны в разделе 6, и назначение им ограничений.

CREATE TABLE logger(

who\_chg TEXT,

when\_chg TIMESTAMP,

db\_operation TEXT,

table\_name TEXT);

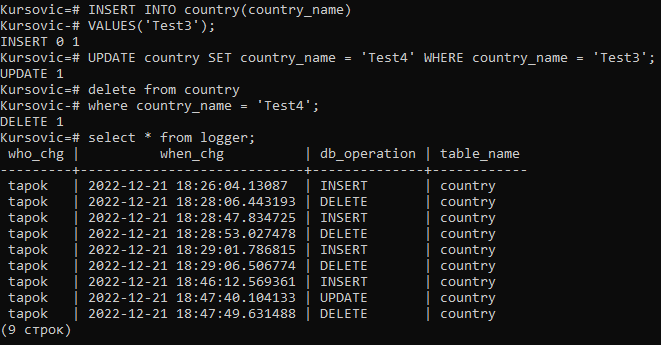


Рисунок 7.1 – демонстрация работы процедуры логгирования.

CREATE USER user\_course PASSWORD 'user';

CREATE USER operator\_course PASSWORD 'operator';

CREATE USER admin\_course WITH SUPERUSER CREATEDB CREATEROLE PASSWORD 'admin';

GRANT SELECT ON ALL TABLES IN SCHEMA PUBLIC TO PUBLIC;

GRANT INSERT, UPDATE ON TABLE country, cities, type\_bicycle, provider, bike, client, makers TO user\_course, admin\_course;

GRANT ALL ON ALL TABLES IN SCHEMA PUBLIC TO admin\_course;

GRANT INSERT, UPDATE, SELECT ON TABLE logger TO operator\_course;

GRANT EXECUTE ON FUNCTION show\_in\_date, show\_provider\_name, range\_date, show\_cost, Max\_cost, Min\_cost,

AVG\_cost, find\_cost, range\_date, avg\_range\_date, Max\_range\_date, Min\_range\_date, find\_bicycle\_shares, find\_by\_in\_date,

find\_by\_range\_cost\_provider, find\_provider\_shares, find\_by\_in\_date\_cost, find\_makers\_bike, find\_bike\_share\_sale\_range,

find\_bike\_expensive\_avg\_by\_provider\_country, find\_bike\_share\_cheaper\_client, find\_bike\_avg\_sale\_range, find\_bike\_expensive\_avg\_by\_makers

TO user\_course, admin\_course, operator\_course;

REVOKE ALL PRIVILEGES ON TABLE logger FROM user\_course;

REVOKE ALL PRIVILEGES ON TABLE country, cities, type\_bicycle, provider, makers, bike, client FROM operator\_course;

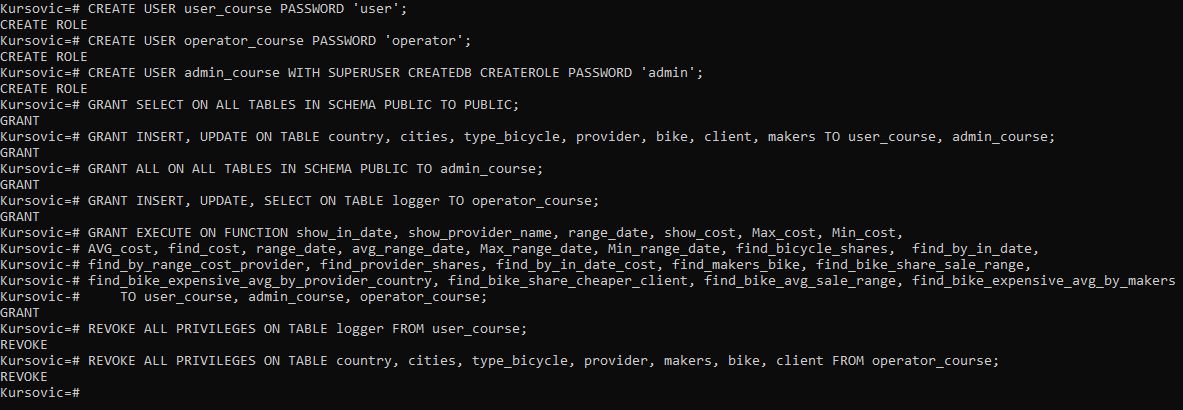


Рисунок 7.2 – Результат создания ролей и ограничений для них.

CREATE RULE insert\_country\_rule AS ON INSERT TO country

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'INSERT', 'country');

CREATE RULE update\_country\_rule AS ON UPDATE TO country

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'UPDATE', 'country');

CREATE RULE delete\_country\_rule AS ON DELETE TO country

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'DELETE', 'country');

CREATE RULE insert\_cities\_rule AS ON INSERT TO cities

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'INSERT', 'cities');

CREATE RULE update\_cities\_rule AS ON UPDATE TO cities

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'UPDATE', 'cities');

CREATE RULE delete\_cities\_rule AS ON DELETE TO cities

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'DELETE', 'cities');

CREATE RULE insert\_type\_bicycle\_rule AS ON INSERT TO type\_bicycle

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'INSERT', 'type\_bicycle');

CREATE RULE update\_type\_bicycle\_rule AS ON UPDATE TO type\_bicycle

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'UPDATE', 'type\_bicycle');

CREATE RULE delete\_type\_bicycle\_rule AS ON DELETE TO type\_bicycle

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'DELETE', 'type\_bicycle');

CREATE RULE insert\_provider\_rule AS ON INSERT TO provider

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'INSERT', 'provider');

CREATE RULE update\_provider\_rule AS ON UPDATE TO provider

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'UPDATE', 'provider');

CREATE RULE delete\_provider\_rule AS ON DELETE TO provider

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'DELETE', 'provider');

CREATE RULE insert\_makers\_rule AS ON INSERT TO makers

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'INSERT', 'makers');

CREATE RULE update\_makers\_rule AS ON UPDATE TO makers

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'UPDATE', 'makers');

CREATE RULE delete\_makers\_rule AS ON DELETE TO makers

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'DELETE', 'makers');

CREATE RULE insert\_client\_rule AS ON INSERT TO client

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'INSERT', 'client');

CREATE RULE update\_client\_rule AS ON UPDATE TO client

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'UPDATE', 'client');

CREATE RULE delete\_client\_rule AS ON DELETE TO client

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'DELETE', 'client');

CREATE RULE insert\_bike\_rule AS ON INSERT TO bike

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'INSERT', 'bike');

CREATE RULE update\_bike\_rule AS ON UPDATE TO bike

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'UPDATE', 'bike');

CREATE RULE delete\_bike\_rule AS ON DELETE TO bike

DO INSERT INTO logger

VALUES((SELECT CURRENT\_USER), (SELECT current\_timestamp), 'DELETE', 'bike');

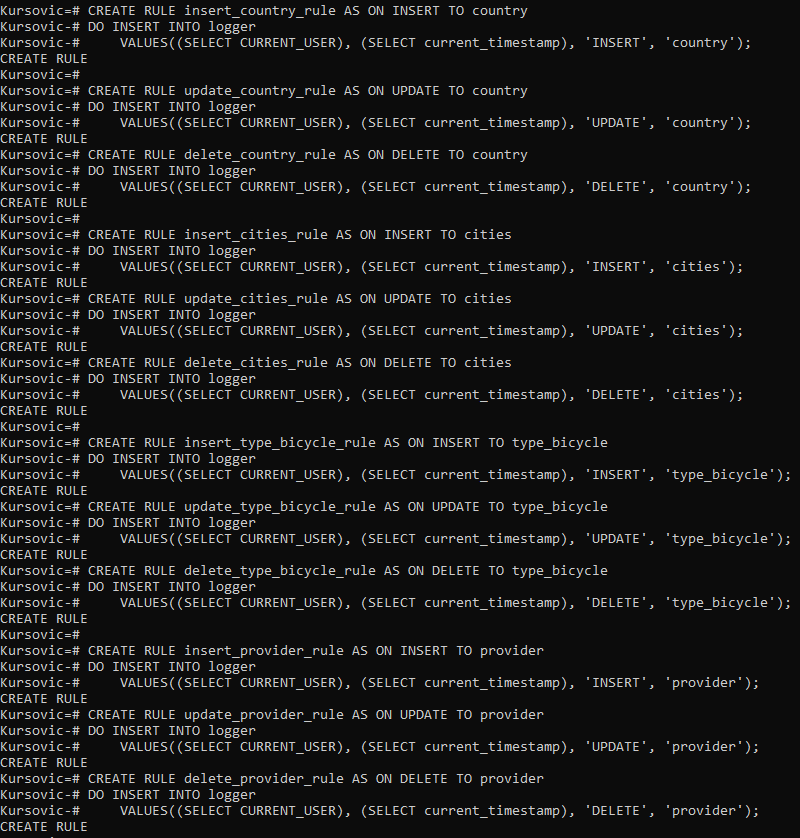


Рисунок 7.3 – Результат создания правил для работы таблицы-журнала операций для таблиц country, cities, type\_bicycle, provider (для остальных таблиц результат аналогичен).

## 2 пункт

В пунктах 2-16 производилось создание и тестирование функций, которые содержат в себе запросы, в соответствии с заданием.

Для каждого вида велосипеда выдать список, отсортированный: по году выпуска, по поставщику, по стоимости.

CREATE OR REPLACE FUNCTION show\_in\_date()

RETURNS TABLE(

id\_bike int,

b\_name text,

id\_type int,

in\_date date,

b\_cost real,

id\_cities int,

id\_makers int,

id\_provider int

)

AS $$

BEGIN

RETURN QUERY SELECT \* FROM bike ORDER BY in\_date;

END;

$$ LANGUAGE plpgsql;

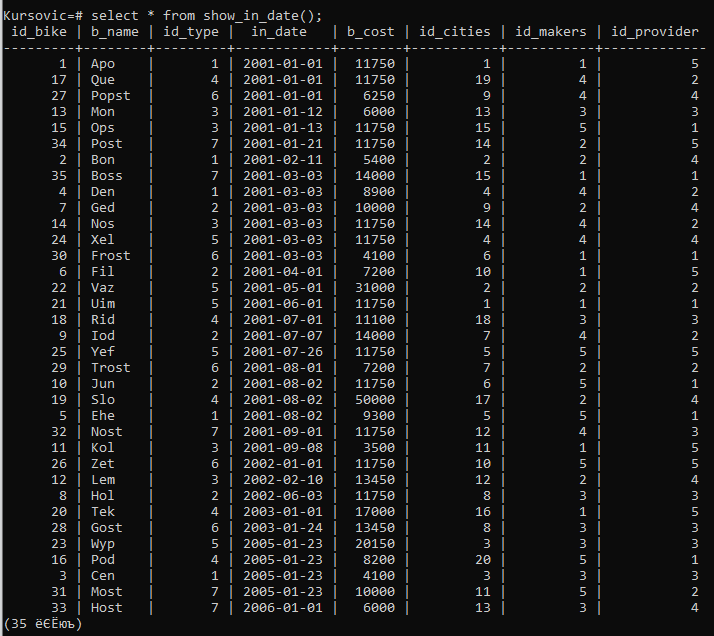


Рисунок 7.4 – Результат выполнение запроса из пункта 2, сортировка по году выпуска.

CREATE OR REPLACE FUNCTION show\_provider\_name()

RETURNS TABLE(

id\_bike int,

b\_name text,

p\_name text

)

AS $$

BEGIN

RETURN QUERY SELECT bike.id\_bike, bike.b\_name, provider.p\_name AS provider\_name FROM bike

JOIN provider ON bike.id\_provider = provider.id\_provider

ORDER BY provider.p\_name;

END;

$$ LANGUAGE plpgsql;

select \* from show\_provider\_name();

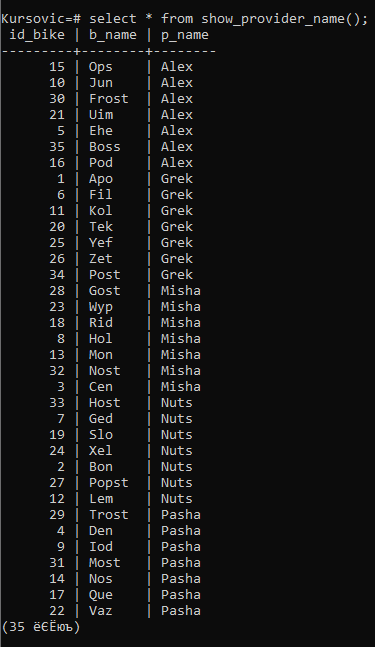


Рисунок 7.5 – Результат выполнение запроса из пункта 2, сортировка по названию поставщика.

CREATE OR REPLACE FUNCTION show\_cost()

RETURNS TABLE(

id\_bike int,

b\_name text,

id\_type int,

in\_date date,

b\_cost real,

id\_cities int,

id\_makers int,

id\_provider int

)

AS $$

BEGIN

RETURN QUERY SELECT \* FROM bike ORDER BY b\_cost;

END;

$$ LANGUAGE plpgsql;

select \* from show\_cost();

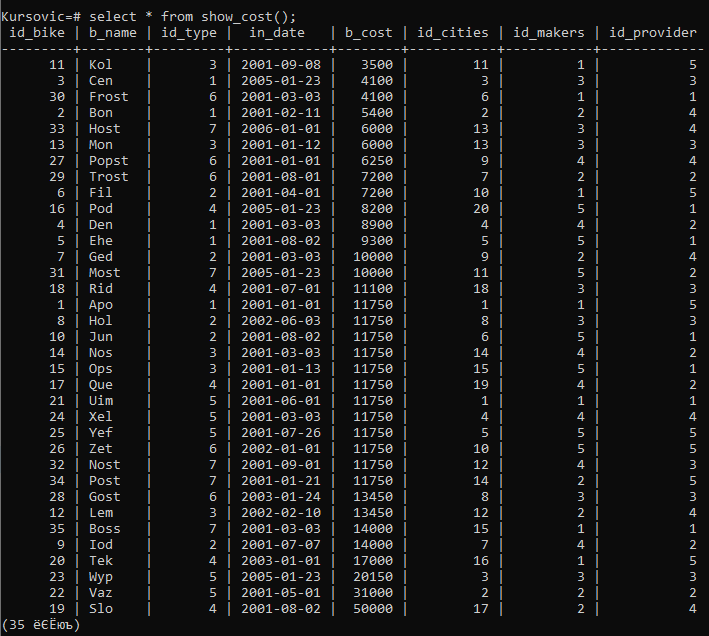


Рисунок 7.6 – Результат выполнение запроса из пункта 2, сортировка по цене.

***3 пункт***

Найти самый дорогой велосипед, самый дешевый, среднюю стоимость.

CREATE OR REPLACE FUNCTION Max\_cost()

RETURNS TABLE(

id\_bike int,

b\_name text,

id\_type int,

in\_date date,

b\_cost real,

id\_cities int,

id\_makers int,

id\_provider int

)

AS $$

BEGIN

RETURN QUERY SELECT \* FROM bike

WHERE bike.b\_cost = ALL (SELECT MAX(bike.b\_cost) FROM bike);

END;

$$ LANGUAGE plpgsql;

select \* from Max\_cost();

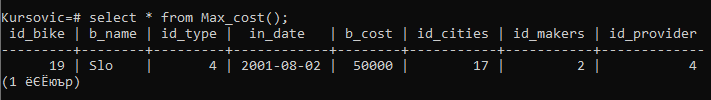


Рисунок 7.7 – Результат выполнение запроса из пункта 3, самый дорогой велосипед.

CREATE OR REPLACE FUNCTION Min\_cost()

RETURNS TABLE(

id\_bike int,

b\_name text,

id\_type int,

in\_date date,

b\_cost real,

id\_cities int,

id\_makers int,

id\_provider int

)

AS $$

BEGIN

RETURN QUERY SELECT \* FROM bike

WHERE bike.b\_cost = ALL (SELECT MIN(bike.b\_cost) FROM bike);

END;

$$ LANGUAGE plpgsql;

select \* from Min\_cost();

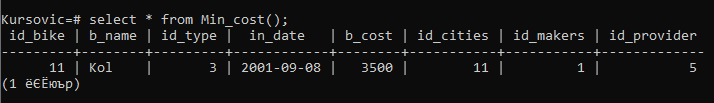


Рисунок 7.8 – Результат выполнение запроса из пункта 3, самый дешевый велосипед.

CREATE OR REPLACE FUNCTION AVG\_cost()

RETURNS TABLE(

avg\_cost double precision

)

AS $$

BEGIN

RETURN QUERY SELECT AVG(b\_cost) AS avg\_cost FROM bike;

END;

$$ LANGUAGE plpgsql;

select \* from AVG\_cost();

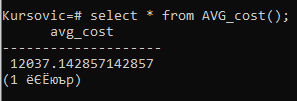


Рисунок 7.9 – Результат выполнение запроса из пункта 3, средняя стоимость велосипедов.

***4 пункт***

Найти велосипед с ценой больше заданной (предусмотреть ввод цены с клавиатуры, например 10000).

CREATE OR REPLACE FUNCTION find\_cost(int)

RETURNS TABLE(

id\_bike int,

b\_name text,

id\_type int,

in\_date date,

b\_cost real,

id\_cities int,

id\_makers int,

id\_provider int )

AS $$ BEGIN

RETURN QUERY SELECT \* FROM bike

WHERE bike.b\_cost > $1

ORDER BY bike.b\_cost; END;$$ LANGUAGE plpgsql;

select \* from find\_cost(10000);

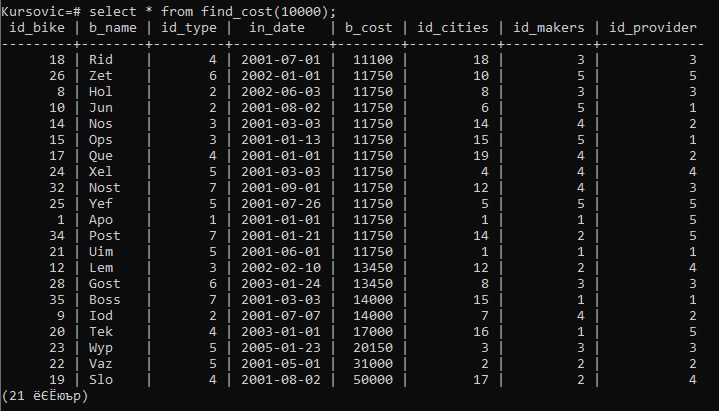


Рисунок 7.10 – Результат выполнение запроса из пункта 4, поиск велосипедов выше заданной цены.

***5 Пункт***

Найти количество велосипедов, выпущенных за определенный период (месяц, 3 месяца, 6 месяцев), среднюю стоимость, максимальную, минимальную.

CREATE OR REPLACE FUNCTION range\_date(date, date)

RETURNS TABLE(

count\_bike bigint)

AS $$

BEGIN

IF $1 > $2 THEN

RETURN QUERY SELECT count(\*) FROM bike

WHERE bike.in\_date BETWEEN $2 AND $1;

END IF;

RETURN QUERY SELECT count (\*) FROM bike

WHERE bike.in\_date BETWEEN $1 AND $2;

END;

$$ LANGUAGE plpgsql;

select \* from range\_date('2001-01-01', '2001-02-01'); - месяц

select \* from range\_date('2001-01-01', '2001-04-01'); - 3 месяца

select \* from range\_date('2001-01-01', '2001-07-01'); - 6 месяцев

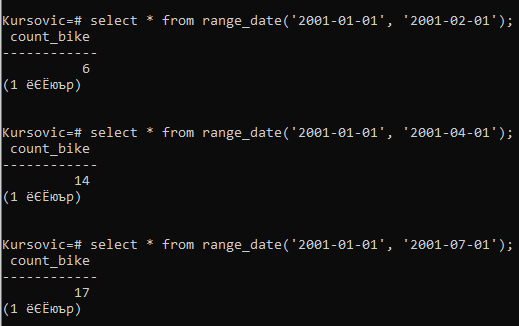


Рисунок 7.11 – Результат выполнение запроса из пункта 5, количество велосипедов, выпущенных за месяц, 3 месяца, 6 месяцев.

CREATE OR REPLACE FUNCTION avg\_range\_date(date, date)

RETURNS TABLE(

avg\_bike double precision)

AS $$

BEGIN

IF $1 > $2 THEN

RETURN QUERY SELECT AVG(b\_cost) AS avg\_cost FROM bike

WHERE bike.in\_date BETWEEN $2 AND $1;

END IF;

RETURN QUERY SELECT AVG(b\_cost) AS avg\_cost FROM bike

WHERE bike.in\_date BETWEEN $1 AND $2;

END;

$$ LANGUAGE plpgsql;

select \* from avg\_range\_date('2001-01-01', '2001-02-01'); - средняя стоимость за месяц

select \* from avg\_range\_date('2001-01-01', '2001-04-01'); - средняя стоимость за 3 месяца

select \* from avg\_range\_date('2001-01-01', '2001-07-01'); - средняя стоимость за 6 месяцев

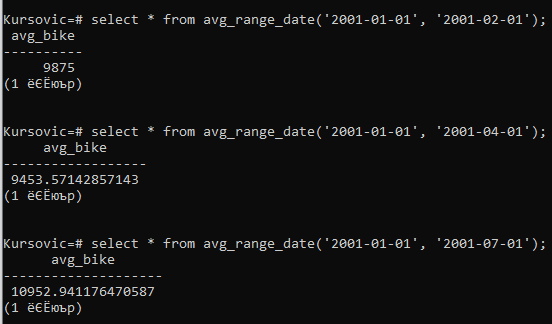


Рисунок 7.12 – Результат выполнение запроса из пункта 5, средняя стоимость велосипедов, из выпущенных за месяц, 3 месяца, 6 месяцев.

CREATE OR REPLACE FUNCTION Max\_range\_date(date, date)

RETURNS TABLE(

avg\_bike real)

AS $$

BEGIN

IF $1 > $2 THEN

RETURN QUERY SELECT Max(b\_cost) AS avg\_cost FROM bike

WHERE bike.in\_date BETWEEN $2 AND $1;

END IF;

RETURN QUERY SELECT Max(b\_cost) AS avg\_cost FROM bike

WHERE bike.in\_date BETWEEN $1 AND $2;

END;

$$ LANGUAGE plpgsql;

select \* from Max\_range\_date('2001-01-01', '2001-01-01'); максимум за месяц

select \* from Max\_range\_date('2001-01-01', '2001-04-01'); максимум за 3 месяца

select \* from Max\_range\_date('2001-01-01', '2001-07-01'); максимум за 6 месяцев

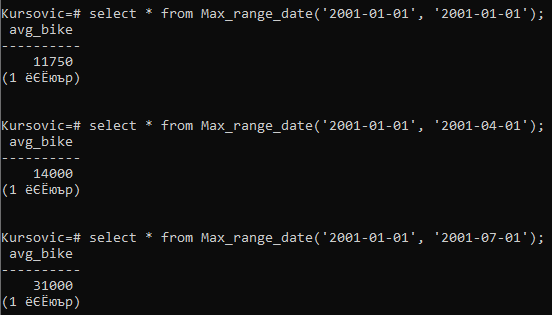


Рисунок 7.13 – Результат выполнение запроса из пункта 5, максимальная стоимость велосипеда, из выпущенных за месяц, 3 месяца, 6 месяцев.

CREATE OR REPLACE FUNCTION Min\_range\_date(date, date)

RETURNS TABLE(

avg\_bike real)

AS $$

BEGIN

IF $1 > $2 THEN

RETURN QUERY SELECT min(b\_cost) AS avg\_cost FROM bike

WHERE bike.in\_date BETWEEN $2 AND $1;

END IF;

RETURN QUERY SELECT min(b\_cost) AS avg\_cost FROM bike

WHERE bike.in\_date BETWEEN $1 AND $2;

END;

$$ LANGUAGE plpgsql;

select \* from Min\_range\_date('2001-01-01', '2001-01-01'); минимум за месяц

select \* from Min\_range\_date('2001-01-01', '2001-04-01'); минимум за 3 месяца

select \* from Min\_range\_date('2001-01-01', '2001-07-01'); минимум за 6 месяцев

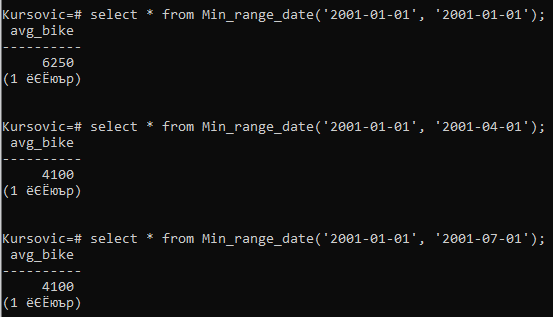


Рисунок 7.14 – Результат выполнение запроса из пункта 5, минимальная стоимость велосипеда, из выпущенных за месяц, 3 месяца, 6 месяцев.

***6 Пункт***

Найти долю велосипедов, поступивших из заданного города (выбор города) от общего числа велосипедов.

CREATE OR REPLACE FUNCTION find\_bicycle\_shares(varchar(100)) RETURNS FLOAT AS $$

DECLARE result FLOAT;

BEGIN

SELECT (COUNT(\*)::FLOAT / (SELECT COUNT(\*) FROM bike)::FLOAT \* 100) INTO result FROM bike

WHERE id\_cities = (SELECT id\_cities FROM cities WHERE city\_name = $1);

RETURN result;

END;

$$ LANGUAGE plpgsql;

select \* from find\_bicycle\_shares('Moscow');

select \* from find\_bicycle\_shares('Ankara');

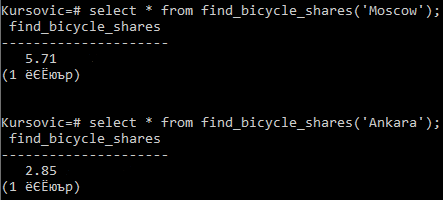


Рисунок 7.15 – Результат выполнение запроса из пункта 6, доля велосипедов поступивших из городов 'Moscow' и 'Ankara'

***7 Пункт***

Найти все велосипеды с заданной датой выпуска (ввод даты).

CREATE OR REPLACE FUNCTION find\_by\_in\_date(date)

RETURNS TABLE(

id\_bike int,

b\_name text,

id\_type int,

in\_date date,

b\_cost real,

id\_cities int,

id\_makers int,

id\_provider int

)

AS $$

BEGIN

RETURN QUERY SELECT \* FROM bike

WHERE bike.in\_date = $1;

END;

$$ LANGUAGE plpgsql;

select \* from find\_by\_in\_date('2001-01-01');

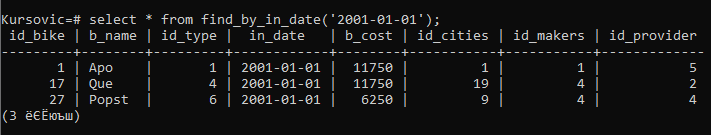


Рисунок 7.16 – Результат выполнение запроса из пункта 7, поиск велосипедов с заданной датой выпуска

***8 Пункт***

Найти все велосипеды заданного поставщика (ввод), чья стоимость находится в заданных пределах (ввод интервала).

CREATE OR REPLACE FUNCTION find\_by\_range\_cost\_provider(int, int, varchar(100))

RETURNS TABLE(

id\_bike int,

b\_name text,

id\_type int,

in\_date date,

b\_cost real,

id\_cities int,

id\_makers int,

id\_provider int

)

AS $$

BEGIN

IF $1 > $2 THEN

RETURN QUERY SELECT bike.id\_bike, bike.b\_name, bike.id\_type, bike.in\_date, bike.b\_cost, bike.id\_provider,

bike.id\_makers, bike.id\_cities FROM bike

JOIN provider ON provider.id\_provider = bike.id\_provider

WHERE bike.b\_cost BETWEEN $2 AND $1

AND (SELECT lower(provider.p\_name)) = (SELECT lower($3)); END IF;

RETURN QUERY SELECT bike.id\_bike, bike.b\_name, bike.id\_type, bike.in\_date, bike.b\_cost, bike.id\_provider,

bike.id\_makers, bike.id\_cities FROM bike

JOIN provider ON provider.id\_provider = bike.id\_provider

WHERE bike.b\_cost BETWEEN $1 AND $2

AND (SELECT lower(provider.p\_name)) = (SELECT lower($3));END;

$$ LANGUAGE plpgsql;

select \* from find\_by\_range\_cost\_provider(1000, 10000, 'Pasha');

select \* from find\_by\_range\_cost\_provider(20000, 1000, 'Alex');

select \* from find\_by\_range\_cost\_provider(1000, 13000, 'Misha');

select \* from find\_by\_range\_cost\_provider(1000, 50000, 'Grek');

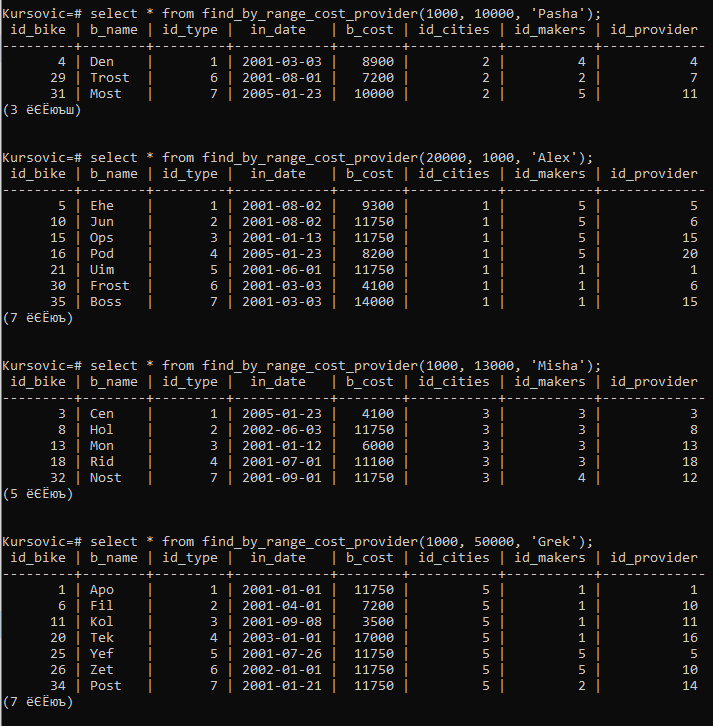


Рисунок 7.17 – Результат выполнение запроса из пункта 8, поиск велосипедов заданного поставщика в пределах заданной стоимости

***9 Пункт***

Найти долю велосипедов, поступивших от заданного поставщика (ввод поставщика) от общего числа поставщиков.

CREATE OR REPLACE FUNCTION find\_provider\_shares(varchar(100)) RETURNS FLOAT AS $$ DECLARE result FLOAT; BEGIN

SELECT (COUNT(\*)::FLOAT / (SELECT COUNT(\*) FROM bike)::FLOAT \* 100) INTO result FROM bike

WHERE bike.id\_provider = (SELECT id\_provider FROM provider WHERE p\_name = $1); RETURN result; END; $$ LANGUAGE plpgsql;

select \* from find\_provider\_shares('Pasha');

select \* from find\_provider\_shares('Alex');

select \* from find\_provider\_shares('Misha');

select \* from find\_provider\_shares('Grek');

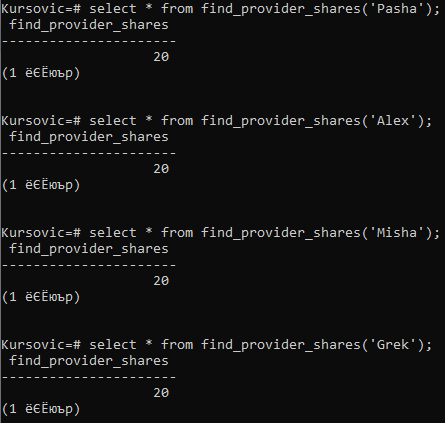


Рисунок 7.18 – Результат выполнение запроса из пункта 9, поиск доли велосипедов заданного поставщика от общегочисла поставщиков.(Равны так как при создании всем выдано одинаковое количество)

***10 Пункт***

Найти все велосипеды заданного года выпуска чья стоимость больше заданной (ввод стоимости).

CREATE OR REPLACE FUNCTION find\_by\_in\_date\_cost(date, int)

RETURNS TABLE(

id\_bike int,

b\_name text,

id\_type int,

in\_date date,

b\_cost real,

id\_cities int,

id\_makers int,

id\_provider int

)

AS $$

BEGIN

RETURN QUERY SELECT \* FROM bike

WHERE bike.in\_date = $1 and bike.b\_cost > $2 ;

END;

$$ LANGUAGE plpgsql;

select \* from find\_by\_in\_date\_cost('2001-01-01', 1000);

select \* from find\_by\_in\_date\_cost('2001-01-01', 10000);

select \* from find\_by\_in\_date\_cost('2001-04-01', 7000);

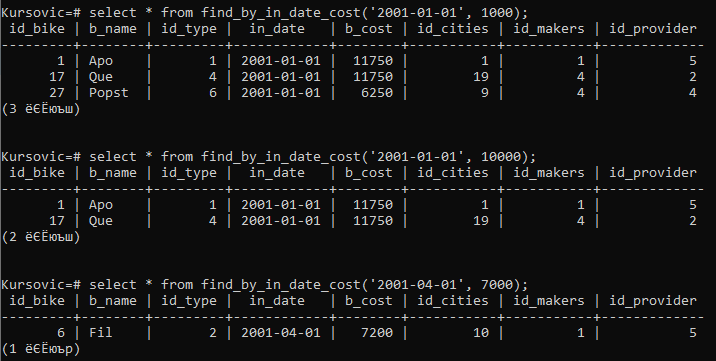
.

Рисунок 7.19 – Результат выполнение запроса из пункта 10, поиск велосипедов с заданной датой выпуска, чья стоимость больше заданной.

***11 Пункт***

Найти все велосипеды заданного производителя (выбор).

CREATE OR REPLACE FUNCTION find\_makers\_bike(text)

RETURNS TABLE(

id\_bike int,

b\_name text,

id\_type int,

in\_date date,

b\_cost real,

id\_cities int,

id\_makers int,

id\_provider int

)

AS $$

BEGIN

RETURN QUERY SELECT \* FROM bike

WHERE bike.id\_makers = (SELECT makers.id\_makers FROM Makers WHERE makers\_name = $1);

END;

$$ LANGUAGE plpgsql;

select \* from find\_makers\_bike('Hunghi');

select \* from find\_makers\_bike('Motomoto');

select \* from find\_makers\_bike('Chill');

select \* from find\_makers\_bike('BMW');

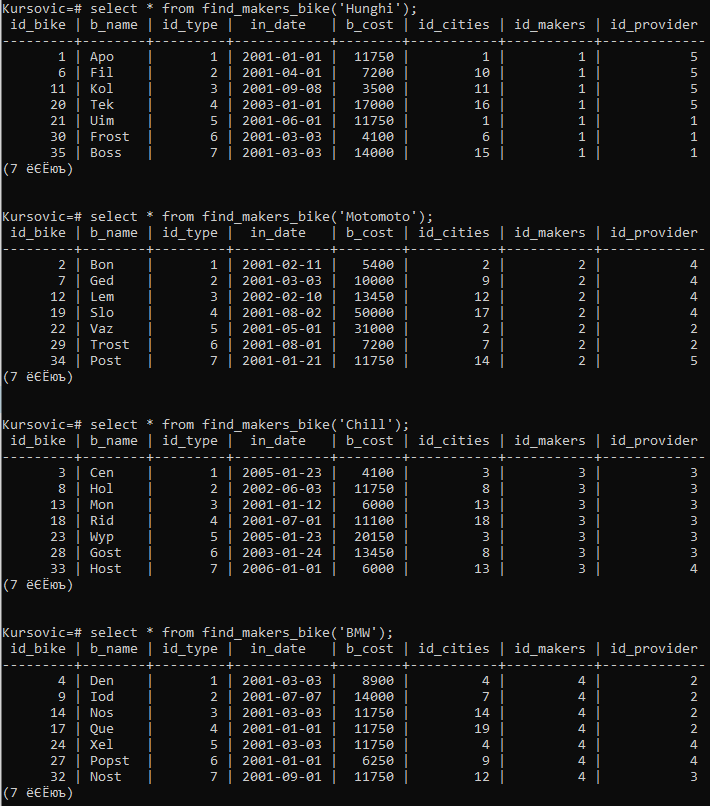


Рисунок 7.20 – Результат выполнение запроса из пункта 11, поиск велосипедов с заданным производителем.

***12 Пункт***

Найти долю велосипедов, проданных за определенный период (ввод периода) от общего времени продажи.

CREATE OR REPLACE FUNCTION find\_bike\_share\_sale\_range(date, date) RETURNS FLOAT AS $$ DECLARE result FLOAT;

BEGIN

IF $1 > $2 THEN

SELECT (COUNT(\*)::FLOAT / (SELECT COUNT(\*) FROM bike)::FLOAT \* 100) INTO result FROM bike

WHERE bike.sale\_date BETWEEN $2 AND $1;

ELSE

SELECT (COUNT(\*)::FLOAT / (SELECT COUNT(\*) FROM bike)::FLOAT \* 100) INTO result FROM bike

WHERE bike.sale\_date BETWEEN $1 AND $2;

END IF;

RETURN result;END;$$ LANGUAGE plpgsql;

select \* from find\_bike\_share\_sale\_range('2010-01-01', '2016-01-01');

select \* from find\_bike\_share\_sale\_range('2015-01-01', '2019-01-01');

select \* from find\_bike\_share\_sale\_range('2010-01-01', '2021-01-01');

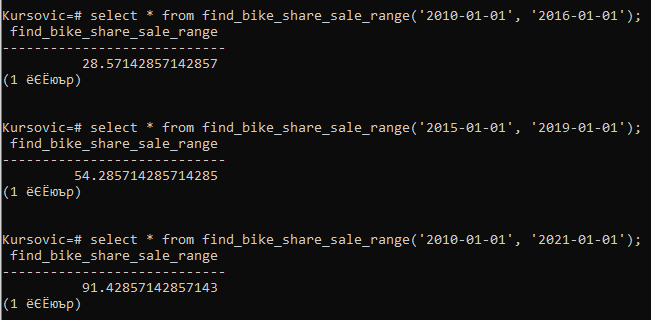


Рисунок 7.21 – Результат выполнение запроса из пункта 12, поиск доли велосипедов проданных за определенное время, от общих продаж.

***13 Пункт***

Найти все велосипеды, поступившие от заданного поставщика (ввод поставщика), чья стоимость больше, чем средняя стоимость велосипедов, поступивших из заданной страны (ввод страны).

CREATE OR REPLACE FUNCTION

find\_bike\_expensive\_avg\_by\_provider\_country(varchar(100), varchar(100))

RETURNS TABLE(

id\_bike int,

b\_name text,

id\_type int,

in\_date date,

b\_cost real,

p\_name text,

country\_name text,

average double precision)

AS $$

BEGIN

RETURN QUERY SELECT bike.id\_bike, bike.b\_name, bike.id\_type, bike.in\_date, bike.b\_cost, provider.p\_name,

country.country\_name,(SELECT AVG(bike.b\_cost) FROM bike

INNER JOIN provider ON provider.id\_provider = bike.id\_provider

INNER JOIN country ON bike.id\_country = country.id\_country

WHERE (SELECT lower(country.country\_name)) = (SELECT lower($2))) FROM bike

INNER JOIN provider ON provider.id\_provider = bike.id\_provider

INNER JOIN country ON bike.id\_country = country.id\_country

WHERE bike.b\_cost > (SELECT AVG(bike.b\_cost) FROM bike

INNER JOIN provider ON provider.id\_provider = bike.id\_provider

INNER JOIN country ON bike.id\_country = country.id\_country

WHERE (SELECT lower(country.country\_name)) = (SELECT lower($2)))

AND (SELECT lower(provider.p\_name)) = (SELECT lower($1));

END;

$$ LANGUAGE plpgsql;

select \* from find\_bike\_expensive\_avg\_by\_provider\_country('Grek', 'Russia');

select \* from find\_bike\_expensive\_avg\_by\_provider\_country('Pasha', 'France');

select \* from find\_bike\_expensive\_avg\_by\_provider\_country('Nuts', 'Japan');

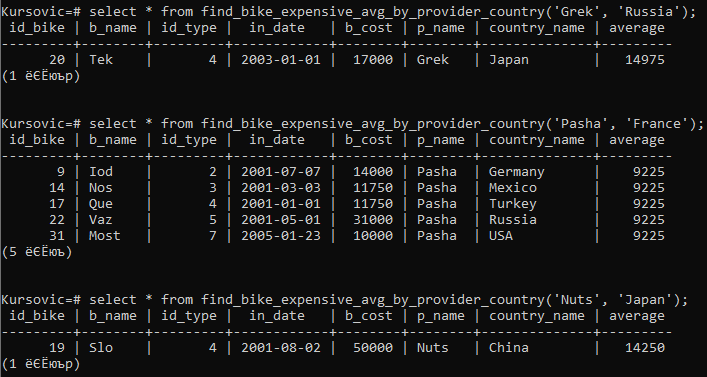


Рисунок 7.22 – Результат выполнение запроса из пункта 13, поиск велосипедов поступивших от заданного поставщика, стоимость которых больше средней стоимости велосипедов, поступивших из заданной страны.

***14 Пункт***

Найти долю дешевых велосипедов (чья стоимость меньше заданной, ввод стоимости), проданных заданному клиенту (ввод клиента), и в целом.

CREATE OR REPLACE FUNCTION find\_bike\_share\_cheaper\_client(int, varchar(100)) RETURNS FLOAT AS $$

DECLARE result FLOAT; BEGIN

SELECT (COUNT(\*)::FLOAT / (SELECT COUNT(\*) FROM bike)::FLOAT \* 100) INTO result FROM bike

JOIN client ON bike.id\_client = client.id\_client

WHERE bike.b\_cost < $1 AND (SELECT lower(client.client)) = (SELECT lower($2));

RETURN result;END;

$$ LANGUAGE plpgsql;

select \* from find\_bike\_share\_cheaper\_client(11111, 'Dasha');

select \* from find\_bike\_share\_cheaper\_client(24600, 'Sasha');

select \* from find\_bike\_share\_cheaper\_client(50000, 'Pasha');

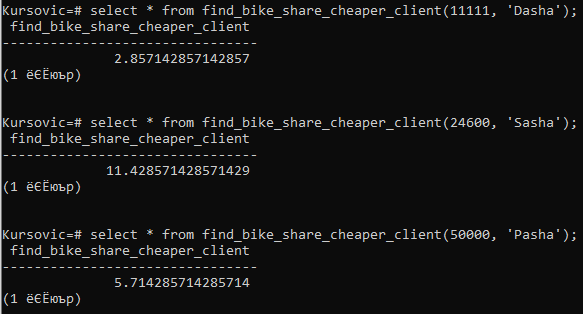


Рисунок 7.23 – Результат выполнение запроса из пункта 14, поиск доли велосипедов чья стоимость меньше заданной, поступивших заданному клиенту.

CREATE OR REPLACE FUNCTION find\_bike\_share\_cheaper(int) RETURNS FLOAT AS $$

DECLARE result FLOAT;

BEGIN

SELECT (COUNT(\*)::FLOAT / (SELECT COUNT(\*) FROM bike)::FLOAT \* 100) INTO result FROM bike

WHERE bike.b\_cost < $1;

RETURN result;

END;

$$ LANGUAGE plpgsql;

select \* from find\_bike\_share\_cheaper(11111);

select \* from find\_bike\_share\_cheaper(24600);

select \* from find\_bike\_share\_cheaper(51000);

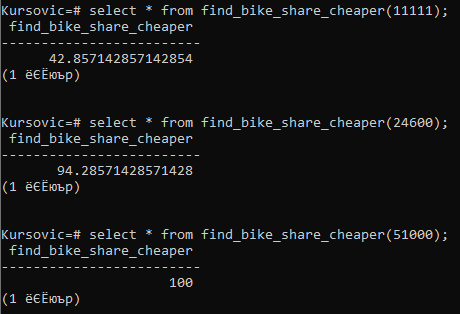


Рисунок 7.24 – Результат выполнение запроса из пункта 14, поиск доли велосипедов чья стоимость меньше заданной.

***15 Пункт***

Найти среднюю стоимость велосипедов, проданных за определенный промежуток времени (ввод интервала).

CREATE OR REPLACE FUNCTION find\_bike\_avg\_sale\_range(date, date) RETURNS FLOAT AS $$

DECLARE result NUMERIC;

BEGIN

IF $1 > $2 THEN

SELECT AVG(bike.b\_cost) INTO result FROM bike

WHERE bike.sale\_date BETWEEN $2 AND $1;

ELSE

SELECT AVG(bike.b\_cost) INTO result FROM bike

WHERE bike.sale\_date BETWEEN $1 AND $2;

END IF;

RETURN result;END;

$$ LANGUAGE plpgsql;

select \* from find\_bike\_avg\_sale\_range('2010-09-22', '2022-01-10');

select \* from find\_bike\_avg\_sale\_range('2013-01-10', '2016-09-22');

select \* from find\_bike\_avg\_sale\_range('2017-09-22', '2022-01-10');

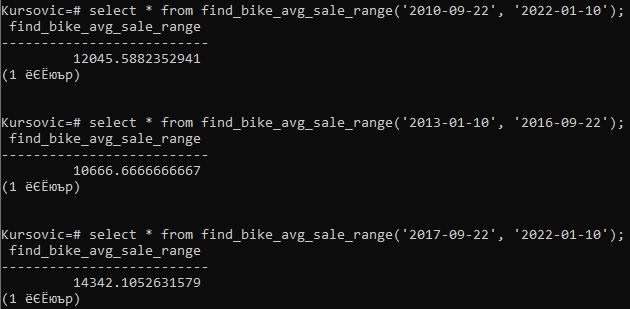


Рисунок 7.25 – Результат выполнение запроса из пункта 15, поиск средней стоимости велосипедов в интервале дат.

***16 Пункт***

Найти все велосипеды, чья стоимость выше, чем средняя стоимость велосипедов заданного производителя (выбор).

CREATE OR REPLACE FUNCTION find\_bike\_expensive\_avg\_by\_makers(varchar(100))

RETURNS TABLE(

id\_bike int,

b\_name text,

id\_type int,

in\_date date,

b\_cost real,

makers\_name text,

average double precision)

AS $$

BEGIN

RETURN QUERY SELECT bike.id\_bike, bike.b\_name, bike.id\_type, bike.in\_date, bike.b\_cost, makers.makers\_name,

(SELECT AVG(bike.b\_cost) FROM bike

JOIN makers ON makers.id\_makers = bike.id\_makers

WHERE (SELECT lower(makers.makers\_name)) = (SELECT lower($1))) FROM bike

JOIN makers ON makers.id\_makers = bike.id\_makers

WHERE bike.b\_cost > (SELECT AVG(bike.b\_cost) FROM bike

JOIN makers ON makers.id\_makers = bike.id\_makers

WHERE (SELECT lower(makers.makers\_name)) = (SELECT lower($1)));

END;

$$ LANGUAGE plpgsql;

select \* from find\_bike\_expensive\_avg\_by\_makers('Hunghi');

select \* from find\_bike\_expensive\_avg\_by\_makers('Chill');

select \* from find\_bike\_expensive\_avg\_by\_makers('Gaz');

select \* from find\_bike\_expensive\_avg\_by\_makers('Motomoto');

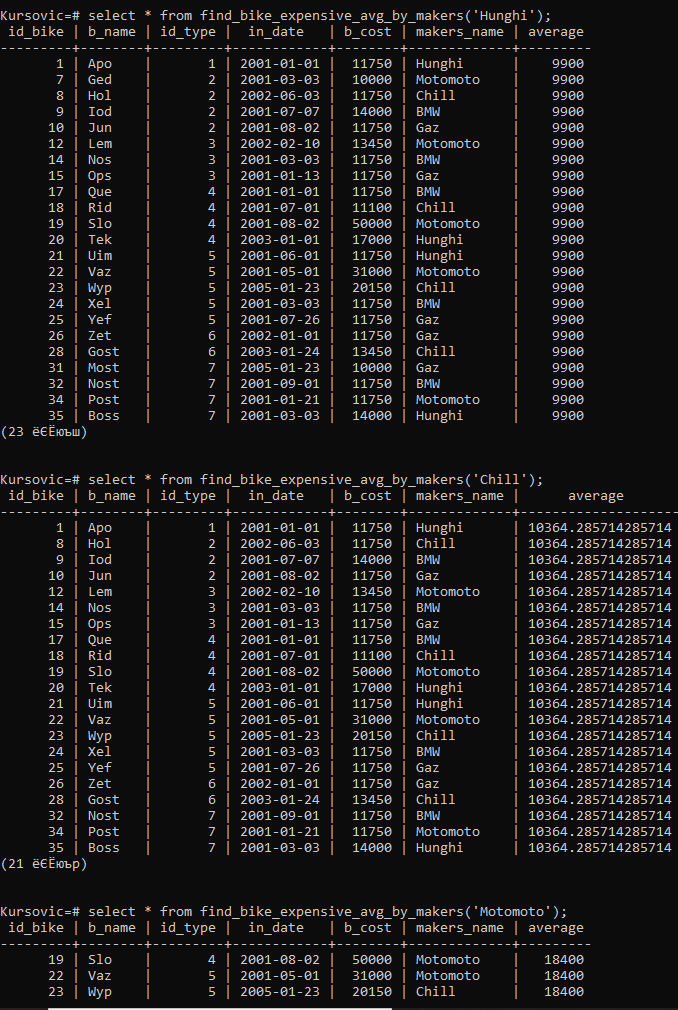


Рисунок 7.26 – Результат выполнение запроса из пункта 16, поиск велосипедов чья стоимость выше стоимости велосипедов заданного производителя.

# 8. Анализ результатов и выводы

Разработанную в ходе выполнения курсовой работы базу данных может использовать специализированный магазин по продаже велосипедов, чтобы была возможность вести учет продажи, анализировать продажи устройств, с целью оптимизации работы как самого магазина, так и сотрудников. Для работы интернет магазина подобная база данных будет жизненно необходимой, чтобы пользователи могли ознакомится с имеющимся ассортиментом и фильтровать устройства по различным категориям.

При постоянной поддержке и развитии разработанной базы данных ее актуальность для бизнеса, связанного с велосипедами, будет только расти.

# Список использованных источников

1. Документация к PostgreSQL 15.1 [Электронный ресурс]: 2022 – URL: <https://postgrespro.ru/docs/postgresql/15/index> (Дата обращения: 06.12.2022)
2. METANIT.COM - Сайт о программировании  
   <https://metanit.com/sql/postgresql/3.4.php> ( Дата обращения: 07.12.2022)