Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Новосибирский государственный технический университет»

Кафедра вычислительной техники

**Отчет**

По лабораторной работе №6-7

По дисциплине: «Базы данных»

Функции в PostgreSQL. Проектирование структуры БД.

Вариант 10

Выполнили: Павлюк А.С. Преподаватель: Харюткина С. А.

Казанцев К. О.

Группа: АВТ-008

Факультет: АВТ

Новосибирск, 2022

Оглавление

[Цель работы 3](#_Toc113728084)

[Задание 3](#_Toc113728085)

[Структура базы данных 5](#_Toc113728086)

[Ход работы 5](#_Toc113728087)

[Вывод 12](#_Toc113728088)

[Приложение 12](#_Toc113728089)

Цель работы

Изучить правила создания функций. Приобрести практические навыки создания функций в среде PostgreSQL. Научиться проектировать БД в целом на основе поставленного задания с использованием всех полученных ранее базовых знаний и навыков.

Задание

Ознакомиться с теоретическими сведениями о возможностях создания пользовательских функций в PostgreSQL. Разработать БД в соответствии с индивидуальным заданием. Создать функции, реализующие интерфейс для работы с базой данных. Проверить работоспособность функций путем выполнения этих функций с параметрами, обеспечивающими как успешное выполнение функции, так и невыполнение функции.

Обязательные требования к БД:

1) Наличие таблиц-справочников и таблиц, использующих справочники. Предусмотреть сохранение ссылочной целостности для таблиц, использующих таблицы-справочники.

2) Предусмотреть следующие роли:

а) оператор БД (пополнение справочников)

б) пользователь БД (основная работа с БД, с ограничениями для некоторого вида операций)

в) аналитик (разрешено выполнение запросов и функций, не изменяющих данные в БД)

г) администратор БД (просмотр протокола операций, любые изменения БД)

3) Действия, изменяющие БД пользователем с любой ролью протоколируются в таблице-журнале операций.

4) Для всех запросов необходимо создать индексы (для гарантированного использования индексов можно использовать отключение параметра enable\_seqscan в текущей сессии)

Создать функции, реализующие интерфейс для работы с базой данных. Проверить работоспособность функций путем выполнения этих функций с параметрами, обеспечивающими как успешное выполнение функции, так и невыполнение функции.

*Вариант 10:* база данных туристического агентства. Должна содержать следующие данные: информацию об имеющихся турах, информацию о руководителях тура, данные о выполненных турах. Предусмотреть анализ следующих показателей: выдать рекомендации для клиента с учетом его пола, возраста и выполненных туров на основании общей статистики, популярность туров в зависимости от вида тура (автобусный, железнодорожный, авиа), среднее время длительности тура в зависимости от сезона (например, лето, осень, зима, весна), стоимость тура (средняя, максимальная, минимальная) в зависимости от города, анализ спроса туров среди путешественников местных и иногородних.

Структура базы данных

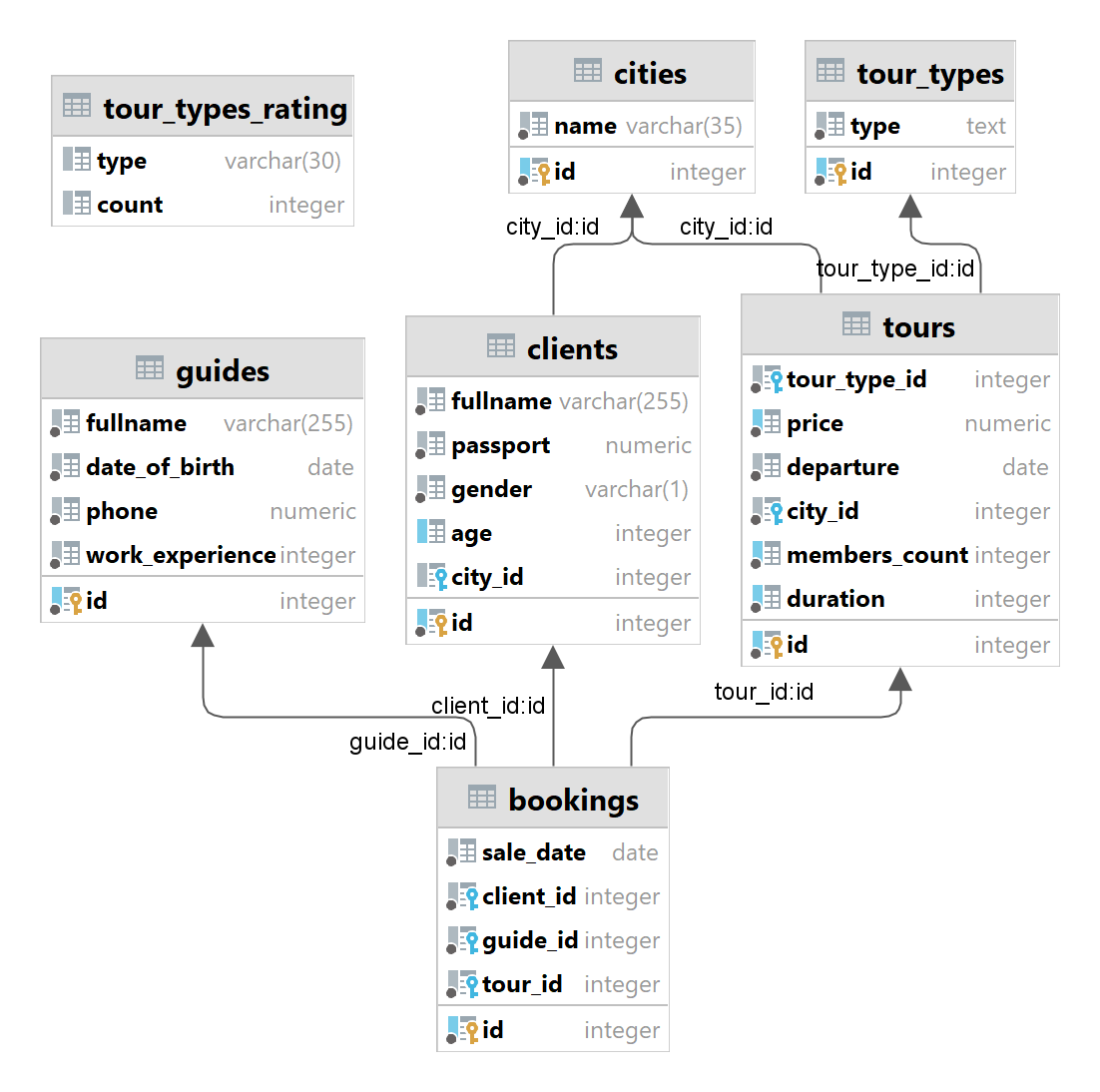


Рисунок 1. Структура базы данных

Ход работы

Начнем проектирование базы данных с создания таблиц:

1. Справочник городов
2. Типы туров
3. Список туров
4. Список руководителей туров
5. Список заказов
6. Список клиентов
7. Рейтинг туров по типу

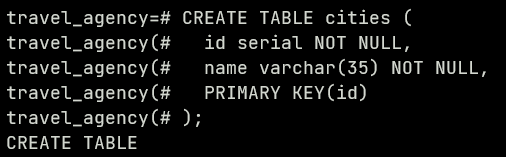


Рисунок 2. Справочник городов

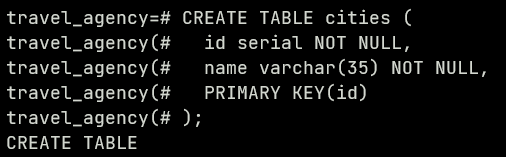


Рисунок 3. Типы туров

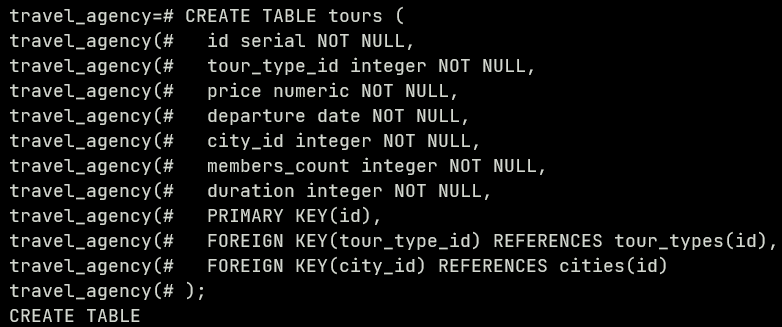


Рисунок 4. Список туров

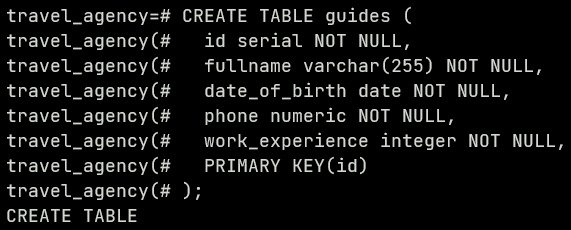


Рисунок 5. Список гидов

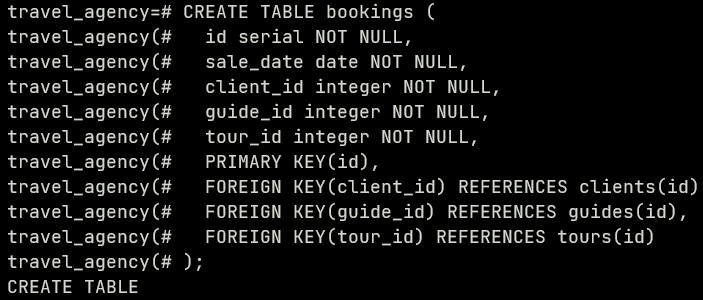


Рисунок 6. Список заказов

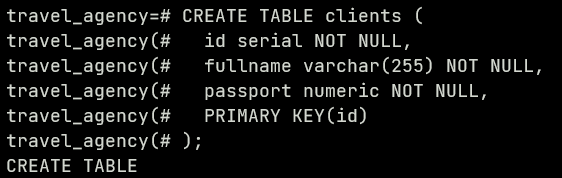


Рисунок 7. Список клиентов

Далее заполним каждую таблицу тестовыми данными.

Перейдем к функциям. Их можно разделить на 2 типа:

1. Основные действия с таблицами (добавление, удаление сущностей);
2. Предназначенные для получения анализа той или иной характеристики агентства.

К первому типу было создано 8 функций:

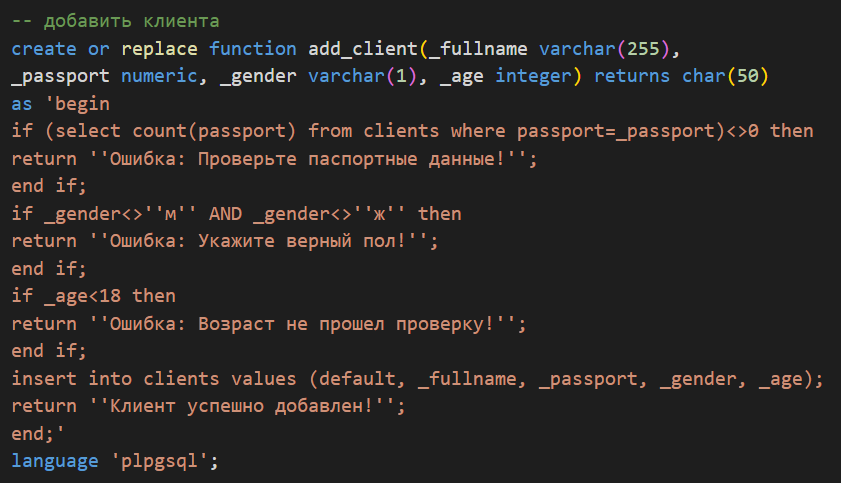


Рисунок 8. Функция добавления клиента

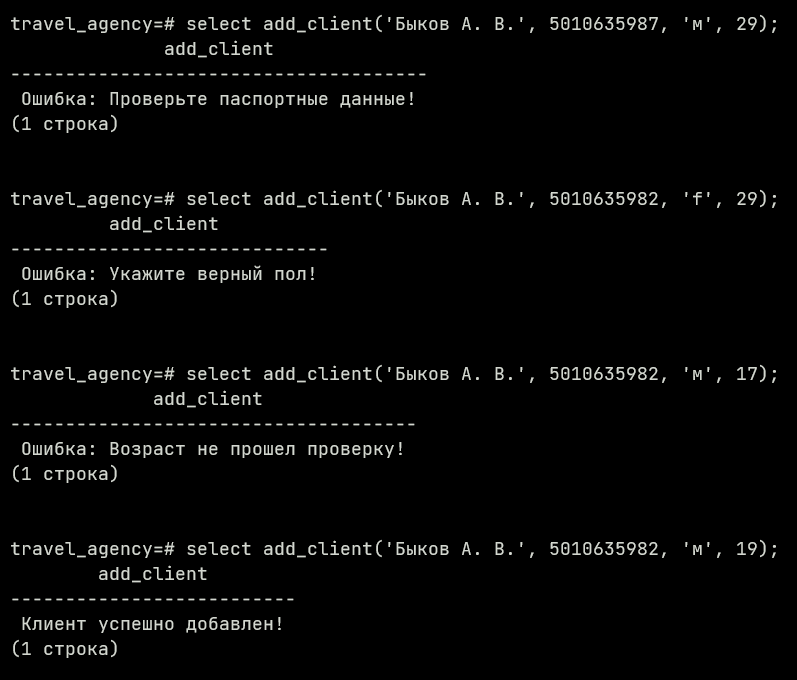


Рисунок 9. Тест функции добавления клиентов

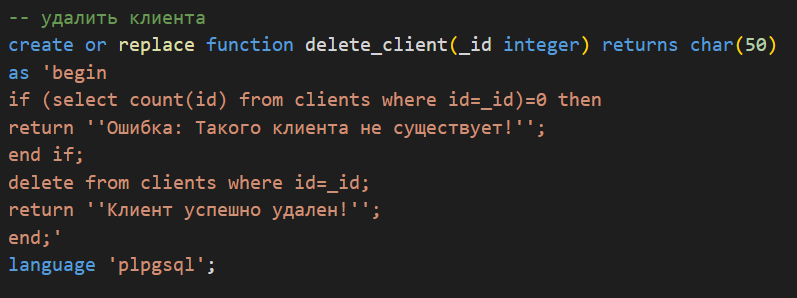


Рисунок 10. Функция удаления клиента

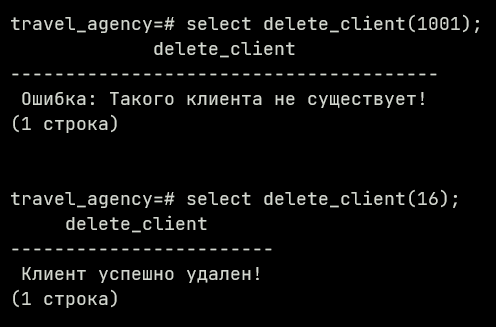


Рисунок 11. Тест функции удаления клиента

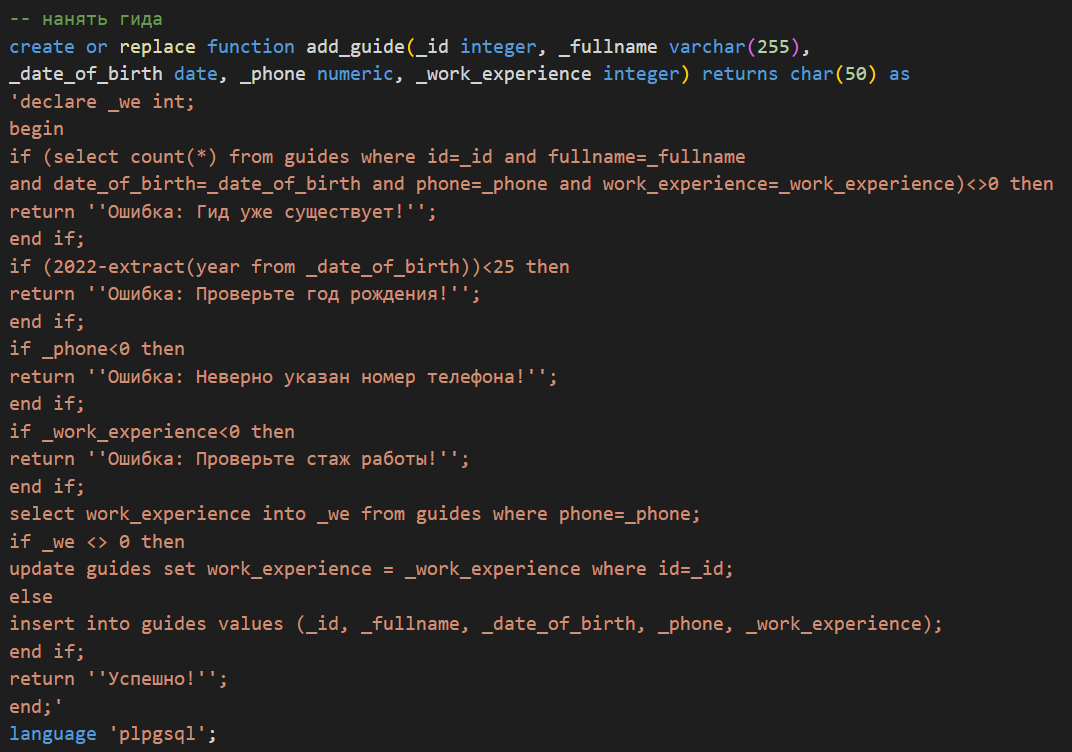


Рисунок 12. Функция добавления гида

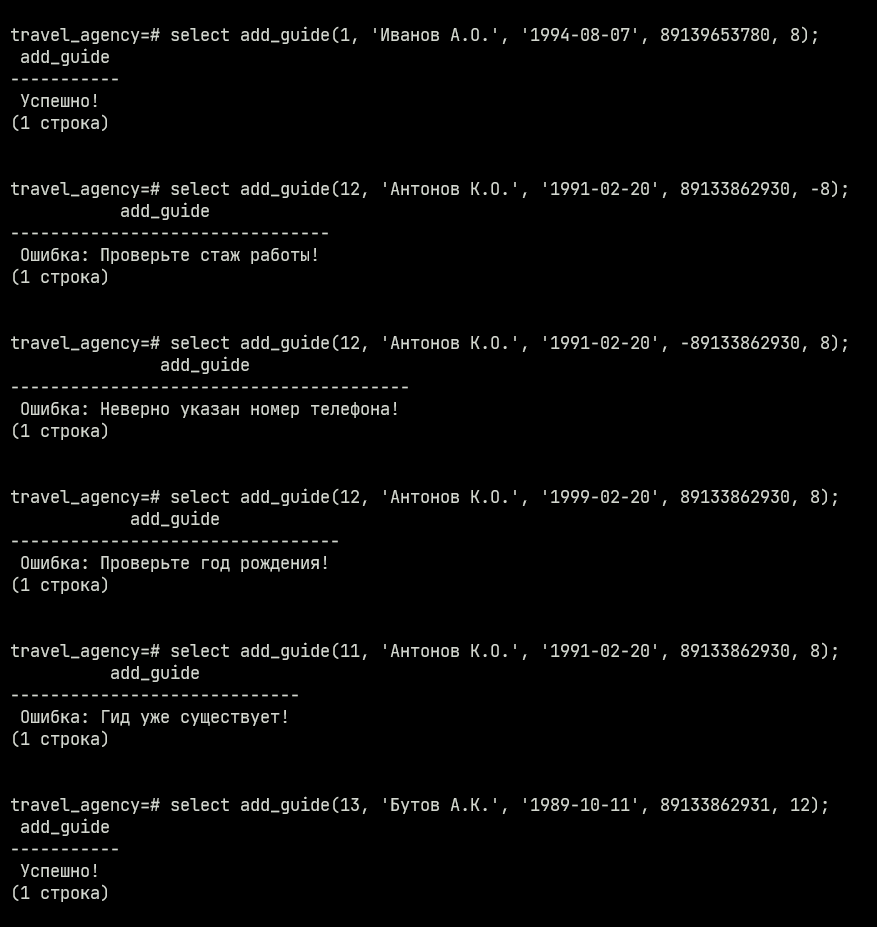


Рисунок 13. Тест функции добавления гида

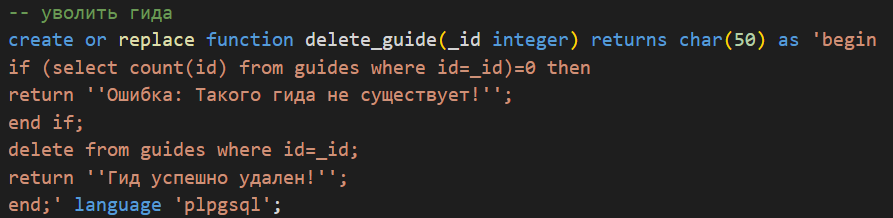


Рисунок 14. Функция удаления гида

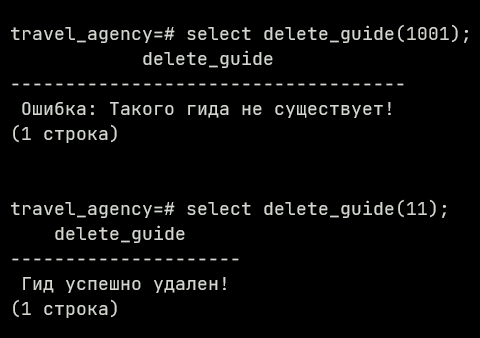


Рисунок 15. Тест функции удаления гида

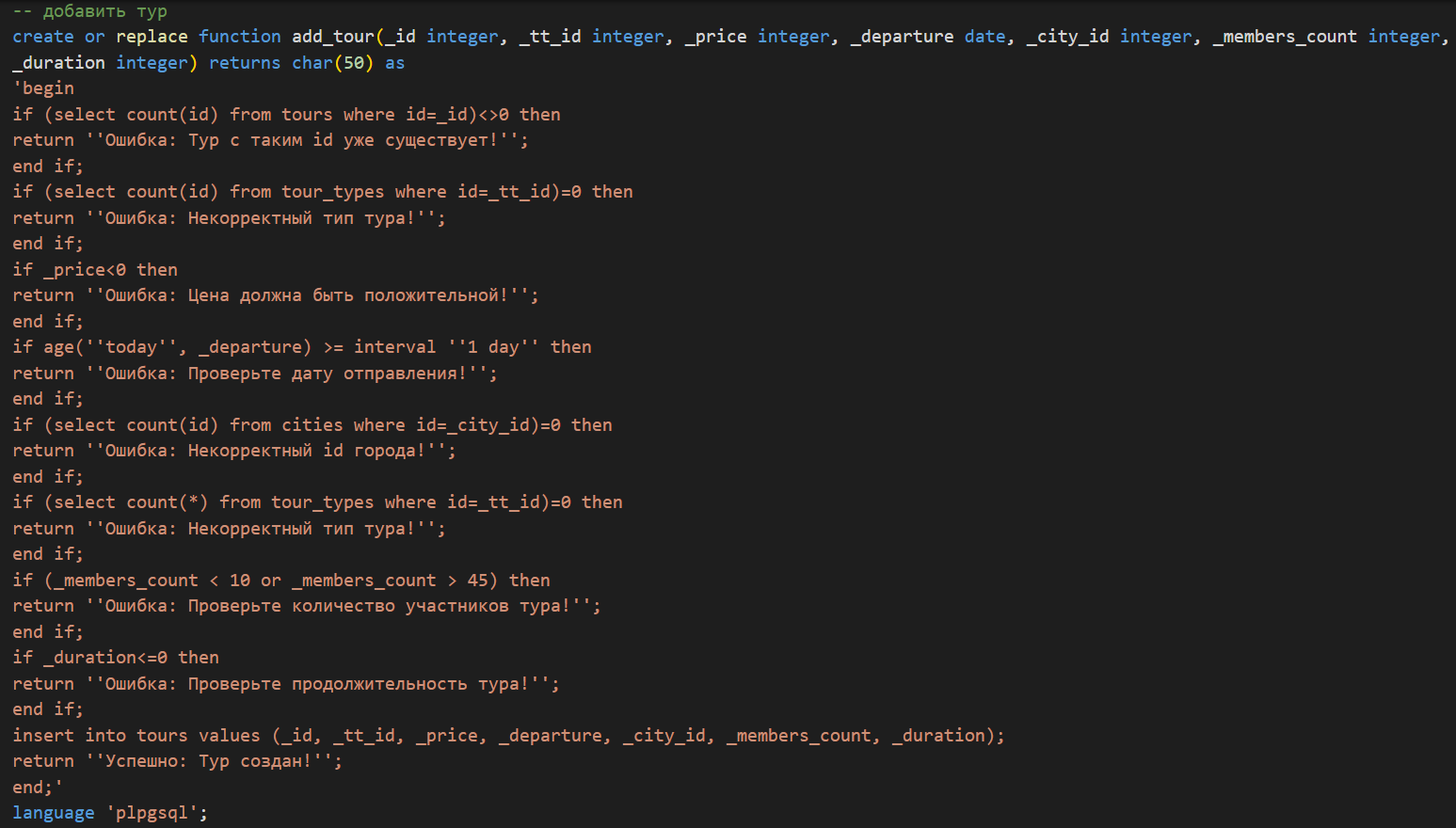


Рисунок 16. Функция добавления тура

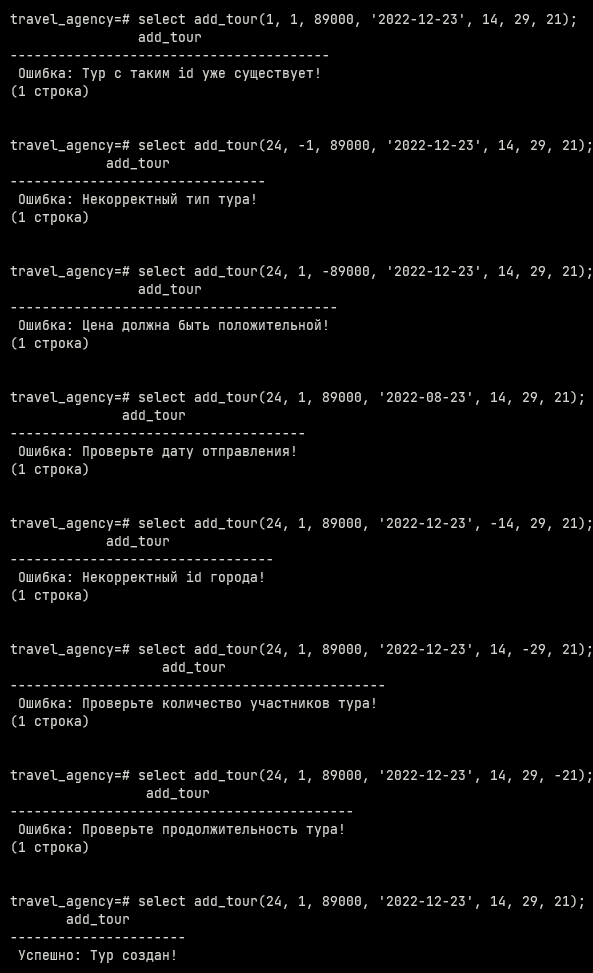


Рисунок 17. Тест функции добавления тура

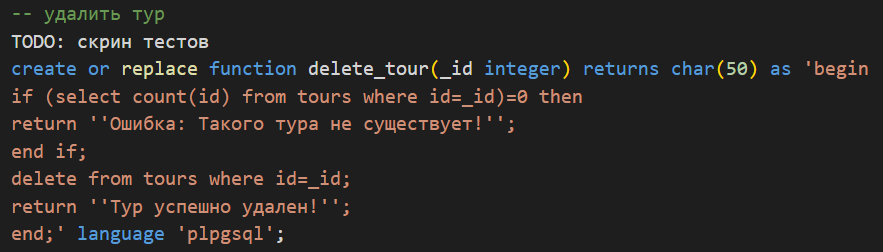


Рисунок 18. Функция удаления тура

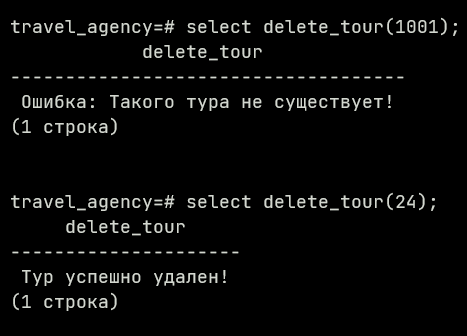


Рисунок 19. Тест функции удаления тура

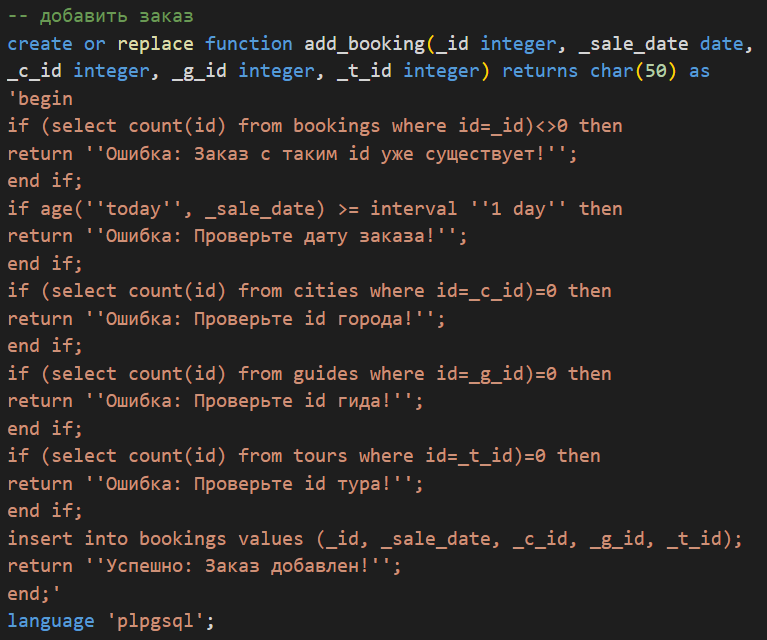


Рисунок 20. Функция удаления заказа

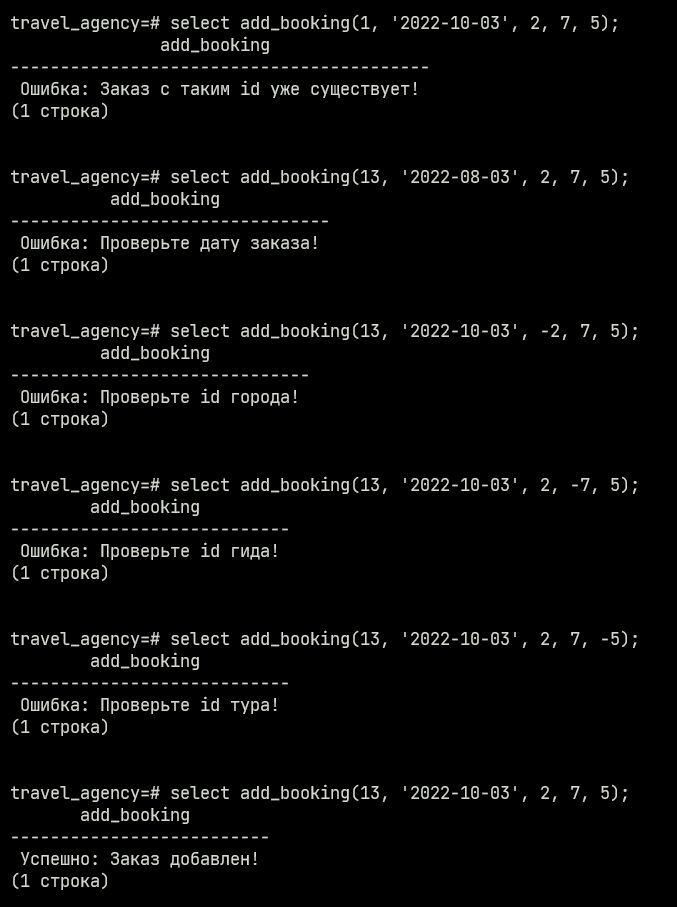


Рисунок 21. Тест функции добавления заказа

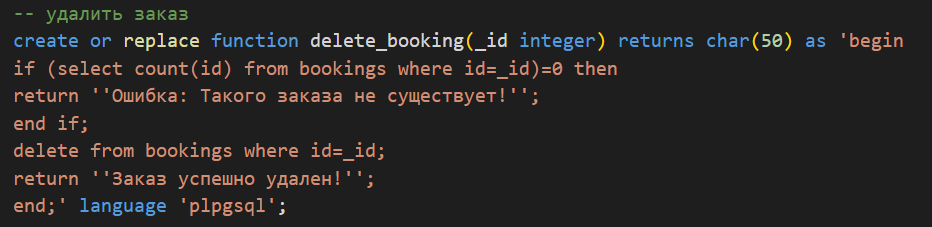


Рисунок 22. Функция удаления заказа

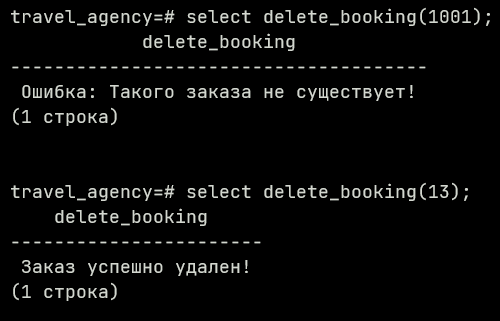


Рисунок 23. Тест функции удаления заказа

Перейдем ко второму типу функций.

Рекомендации для клиента выдаются с учетом его пола и возраста. В итоговой выборке присутствуют только туры, пол клиентов в которых совпадает с полом клиента, а возраст отличается максимум на 8 лет.

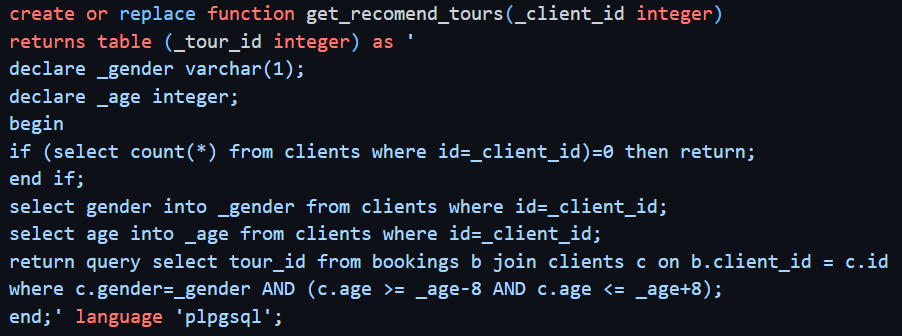


Рисунок 24. Рекомендации для клиента

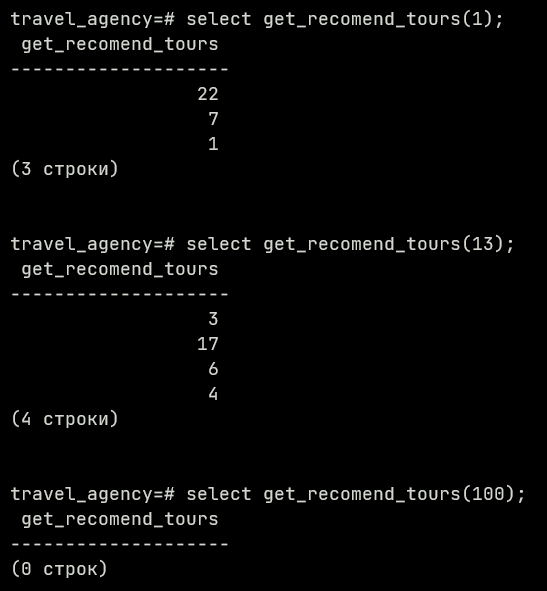


Рисунок 25. Тест функции получения рекомендованных туров

Популярность каждого вида туров считалась количеством туров с каждым видом. Результаты заносятся и обновляются в таблицу **tour\_types\_rating**.

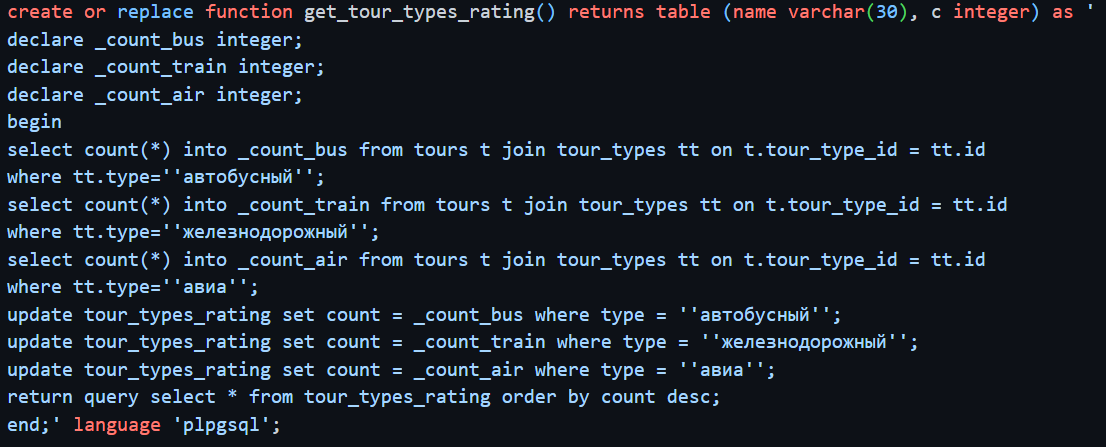


Рисунок 26. Функция получения рейтинга типов туров

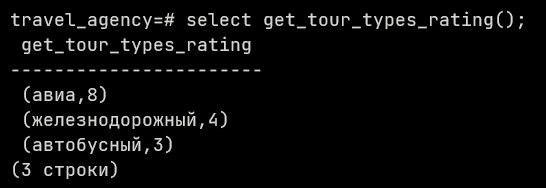


Рисунок 27. Тест функции получения рейтинга по турам

Далее создадим функцию получения среднего числа дней в зависимости от сезона.

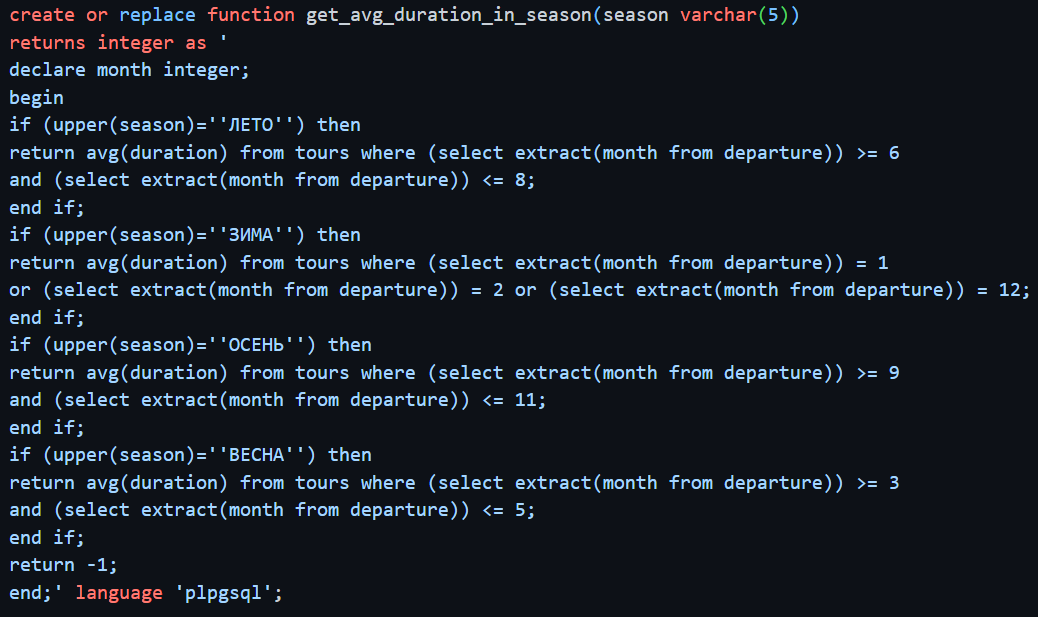


Рисунок 28. Функция получения среднего числа дней по сезонам

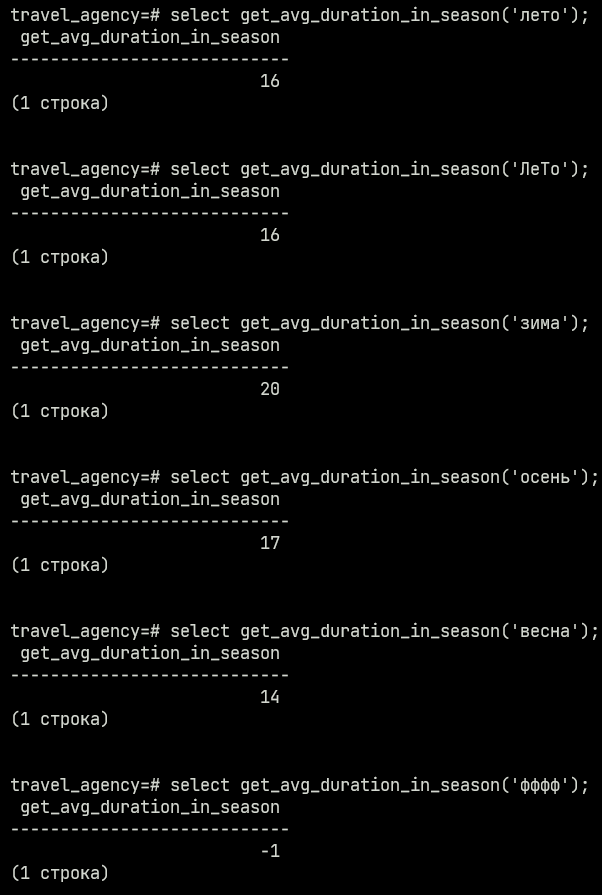


Рисунок 29. Тест функции средней продолжительности

Теперь напишем функцию получения значений цен туров в зависимости от города.



Рисунок 30. Функция получения цен по городу

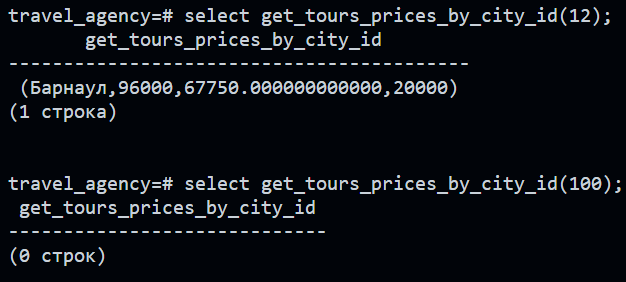


Рисунок 31. Тест функции получения стоимостей

Далее напишем функцию получения спроса среди местных и иногородних клиентов.

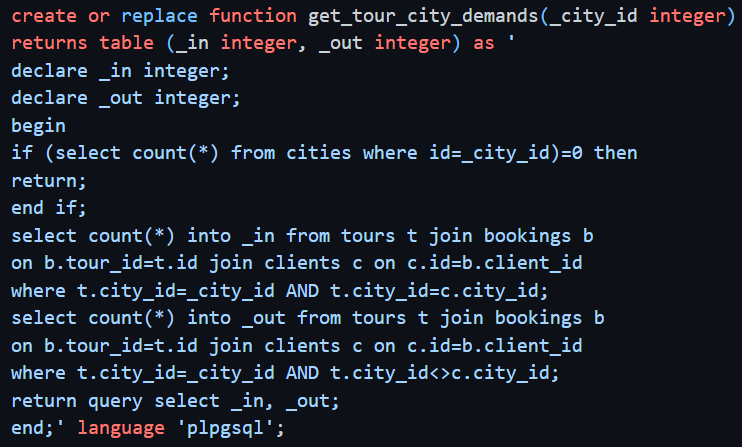


Рисунок 32. Спрос туров среди местных и иногородних

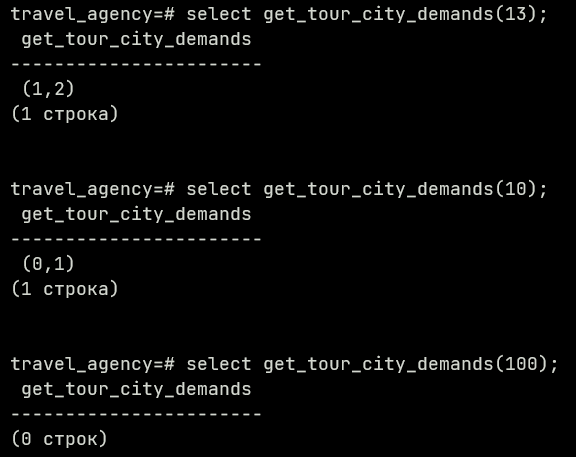


Рисунок 33. Тест функции спроса

Стоит добавить индексы для полей:

* Clients.age
* Tours.duration
* Tours.price
* Tours.members\_count

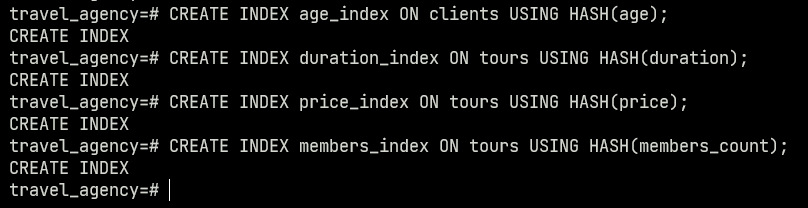


Рисунок 34. Создание индексов

Перейдем к созданию ролей согласно заданию.

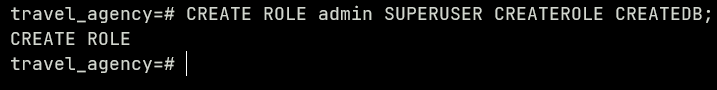


Рисунок 35. Создание роли администратора

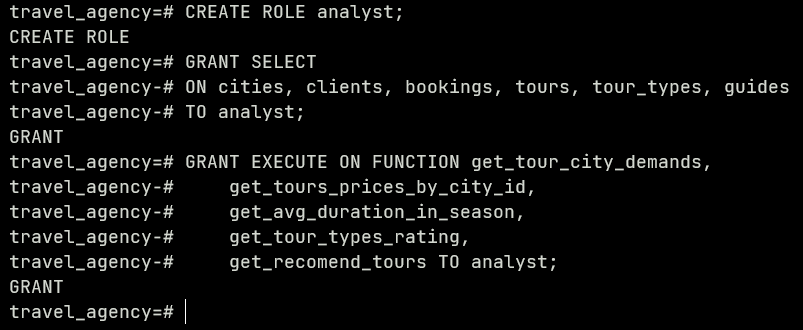


Рисунок 36. Создание роли аналитика

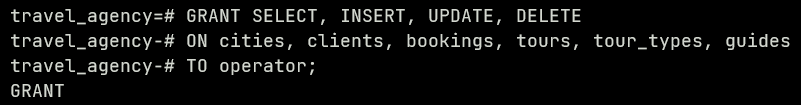


Рисунок 37. Создание роли оператора

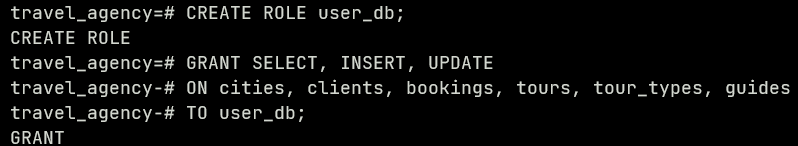


Рисунок 38. Создание роли пользователя

Вывод

В результате выполнения лабораторной работы были изучены правила создания функций. Приобретены практические навыки создания функций в среде PostgreSQL, проектирования БД в целом на основе поставленного задания с использованием всех полученных ранее базовых знаний и навыков.

Приложение

CREATE DATABASE travel\_agency OWNER nstu;

CREATE TABLE cities (

id serial NOT NULL,

name varchar(35) NOT NULL,

PRIMARY KEY(id)

);

CREATE TABLE guides (

id serial NOT NULL,

fullname varchar(255) NOT NULL,

date\_of\_birth date NOT NULL,

phone numeric NOT NULL,

work\_experience integer NOT NULL,

PRIMARY KEY(id)

);

CREATE TABLE tour\_types (

id serial NOT NULL,

type text NOT NULL,

PRIMARY KEY(id)

);

CREATE TABLE clients (

id serial NOT NULL,

fullname varchar(255) NOT NULL,

passport numeric NOT NULL,

gender varchar(1) NOT NULL DEFAULT 'м',

age integer,

PRIMARY KEY(id)

);

CREATE TABLE tours (

id serial NOT NULL,

tour\_type\_id integer NOT NULL,

price numeric NOT NULL,

departure date NOT NULL,

city\_id integer NOT NULL,

members\_count integer NOT NULL,

duration integer NOT NULL,

PRIMARY KEY(id),

FOREIGN KEY(tour\_type\_id) REFERENCES tour\_types(id),

FOREIGN KEY(city\_id) REFERENCES cities(id)

);

CREATE TABLE bookings (

id serial NOT NULL,

sale\_date date NOT NULL,

client\_id integer NOT NULL,

guide\_id integer NOT NULL,

tour\_id integer NOT NULL,

PRIMARY KEY(id),

FOREIGN KEY(client\_id) REFERENCES clients(id),

FOREIGN KEY(guide\_id) REFERENCES guides(id),

FOREIGN KEY(tour\_id) REFERENCES tours(id)

);

-- Общий функционал для работы с бд

-- добавить клиента OK

create or replace function add\_client(

\_fullname varchar(255),

\_passport numeric,

\_gender varchar(1),

\_age integer

) returns char(50) as 'begin

if (select count(passport) from clients where passport=\_passport)<>0 then

return ''Ошибка: Проверьте паспортные данные!'';

end if;

if \_gender<>''м'' AND \_gender<>''ж'' then

return ''Ошибка: Укажите верный пол!'';

end if;

if \_age<18 then

return ''Ошибка: Возраст не прошел проверку!'';

end if;

insert into clients values (default, \_fullname, \_passport, \_gender, \_age);

return ''Клиент успешно добавлен!'';

end;' language 'plpgsql';

-- удалить клиента OK

create or replace function delete\_client(\_id integer) returns char(50) as 'begin

if (select count(id) from clients where id=\_id)=0 then

return ''Ошибка: Такого клиента не существует!'';

end if;

delete from clients where id=\_id;

delete from bookings where client\_id=\_id;

return ''Клиент успешно удален!'';

end;' language 'plpgsql';

-- нанять гида ОК

create or replace function add\_guide(

\_id integer,

\_fullname varchar(255),

\_date\_of\_birth date,

\_phone numeric,

\_work\_experience integer

) returns char(50) as 'declare \_we int;

begin

if (select count(\*) from guides where id=\_id and fullname=\_fullname

and date\_of\_birth=\_date\_of\_birth and phone=\_phone and work\_experience=\_work\_experience)<>0 then

return ''Ошибка: Гид уже существует!'';

end if;

if (2022-extract(year from \_date\_of\_birth))<25 then

return ''Ошибка: Проверьте год рождения!'';

end if;

if \_phone<0 then

return ''Ошибка: Неверно указан номер телефона!'';

end if;

if \_work\_experience<0 then

return ''Ошибка: Проверьте стаж работы!'';

end if;

select work\_experience into \_we from guides where phone=\_phone;

if \_we <> 0 then

update guides set work\_experience = \_work\_experience where id=\_id;

else

insert into guides values (\_id, \_fullname, \_date\_of\_birth, \_phone, \_work\_experience);

end if;

return ''Успешно!'';

end;' language 'plpgsql';

-- уволить гида ОК

create or replace function delete\_guide(\_id integer) returns char(50) as 'begin

if (select count(id) from guides where id=\_id)=0 then

return ''Ошибка: Такого гида не существует!'';

end if;

delete from guides where id=\_id;

delete from bookings where guide\_id=\_id;

return ''Гид успешно удален!'';

end;' language 'plpgsql';

-- добавить тур ОК

create or replace function add\_tour(

\_id integer,

\_tt\_id integer,

\_price integer,

\_departure date,

\_city\_id integer,

\_members\_count integer,

\_duration integer

) returns char(50) as 'begin

if (select count(id) from tours where id=\_id)<>0 then

return ''Ошибка: Тур с таким id уже существует!'';

end if;

if (select count(id) from tour\_types where id=\_tt\_id)=0 then

return ''Ошибка: Некорректный тип тура!'';

end if;

if \_price<0 then

return ''Ошибка: Цена должна быть положительной!'';

end if;

if age(''today'', \_departure) >= interval ''1 day'' then

return ''Ошибка: Проверьте дату отправления!'';

end if;

if (select count(id) from cities where id=\_city\_id)=0 then

return ''Ошибка: Некорректный id города!'';

end if;

if (select count(\*) from tour\_types where id=\_tt\_id)=0 then

return ''Ошибка: Некорректный тип тура!'';

end if;

if (\_members\_count < 10 or \_members\_count > 45) then

return ''Ошибка: Проверьте количество участников тура!'';

end if;

if \_duration<=0 then

return ''Ошибка: Проверьте продолжительность тура!'';

end if;

insert into tours values (\_id, \_tt\_id, \_price, \_departure, \_city\_id, \_members\_count, \_duration);

return ''Успешно: Тур создан!'';

end;' language 'plpgsql';

-- удалить тур ОК

create or replace function delete\_tour(\_id integer) returns char(50) as 'begin

if (select count(id) from tours where id=\_id)=0 then

return ''Ошибка: Такого тура не существует!'';

end if;

delete from tours where id=\_id;

delete from bookings where tour\_id=\_id;

return ''Тур успешно удален!'';

end;' language 'plpgsql';

-- добавить заказ ОК

create or replace function add\_booking(

\_id integer,

\_sale\_date date,

\_c\_id integer,

\_g\_id integer,

\_t\_id integer

) returns char(50) as 'begin

if (select count(id) from bookings where id=\_id)<>0 then

return ''Ошибка: Заказ с таким id уже существует!'';

end if;

if age(''today'', \_sale\_date) >= interval ''1 day'' then

return ''Ошибка: Проверьте дату заказа!'';

end if;

if (select count(id) from cities where id=\_c\_id)=0 then

return ''Ошибка: Проверьте id города!'';

end if;

if (select count(id) from guides where id=\_g\_id)=0 then

return ''Ошибка: Проверьте id гида!'';

end if;

if (select count(id) from tours where id=\_t\_id)=0 then

return ''Ошибка: Проверьте id тура!'';

end if;

insert into bookings values (\_id, \_sale\_date, \_c\_id, \_g\_id, \_t\_id);

return ''Успешно: Заказ добавлен!'';

end;' language 'plpgsql';

-- удалить заказ ОК

create or replace function delete\_booking(\_id integer) returns char(50) as 'begin

if (select count(id) from bookings where id=\_id)=0 then

return ''Ошибка: Такого заказа не существует!'';

end if;

delete from bookings where id=\_id;

return ''Заказ успешно удален!'';

end;' language 'plpgsql';

-- выдать рекомендации для клиента с учетом его

-- пола, возраста и выполненных туров на основании общей статистики

--

-- идея: выдавать для М(Ж) только туры из заказов, у которых клиет М(Ж) и возраст +-8 лет. ОК

create or replace function get\_recomend\_tours(\_client\_id integer) returns table (\_tour\_id integer) as '

declare \_gender varchar(1);

declare \_age integer;

begin

if (select count(\*) from clients where id=\_client\_id)=0 then return;

end if;

select gender into \_gender from clients where id=\_client\_id;

select age into \_age from clients where id=\_client\_id;

return query select tour\_id from bookings b join clients c on b.client\_id = c.id

where c.gender=\_gender AND (c.age >= \_age-8 AND c.age <= \_age+8);

end;' language 'plpgsql';

--

-- популярность туров в зависимости от вида тура (автобусный,

-- железнодорожный, авиа)

--

-- идея: собрать кол-во туров (рейтинг) каждого вида и вывести по возрастанию кол-ва

--

CREATE TABLE tour\_types\_rating(varchar(30), integer);

INSERT INTO tour\_types\_rating

values ('автобусный', 0),

('железнодорожный', 0),

('авиа', 0);

create or replace function get\_tour\_types\_rating() returns table (name varchar(30), c integer) as '

declare \_count\_bus integer;

declare \_count\_train integer;

declare \_count\_air integer;

begin

select count(\*) into \_count\_bus from tours t join tour\_types tt on t.tour\_type\_id = tt.id

where tt.type=''автобусный'';

select count(\*) into \_count\_train from tours t join tour\_types tt on t.tour\_type\_id = tt.id

where tt.type=''железнодорожный'';

select count(\*) into \_count\_air from tours t join tour\_types tt on t.tour\_type\_id = tt.id

where tt.type=''авиа'';

update tour\_types\_rating set count = \_count\_bus where type = ''автобусный'';

update tour\_types\_rating set count = \_count\_train where type = ''железнодорожный'';

update tour\_types\_rating set count = \_count\_air where type = ''авиа'';

return query select \* from tour\_types\_rating order by count desc;

end;' language 'plpgsql';

-- среднее время длительности тура в зависимости от сезона

-- (например, лето, осень, зима, весна),

--

-- идея: вывести среднее число дней. входной параместр - сезон.

--

create or replace function get\_avg\_duration\_in\_season(season varchar(5)) returns integer as '

declare month integer;

begin

if (upper(season)=''ЛЕТО'') then

return avg(duration) from tours where (select extract(month from departure)) >= 6

and (select extract(month from departure)) <= 8;

end if;

if (upper(season)=''ЗИМА'') then

return avg(duration) from tours where (select extract(month from departure)) = 1

or (select extract(month from departure)) = 2 or (select extract(month from departure)) = 12;

end if;

if (upper(season)=''ОСЕНЬ'') then

return avg(duration) from tours where (select extract(month from departure)) >= 9

and (select extract(month from departure)) <= 11;

end if;

if (upper(season)=''ВЕСНА'') then

return avg(duration) from tours where (select extract(month from departure)) >= 3

and (select extract(month from departure)) <= 5;

end if;

return -1;

end;' language 'plpgsql';

-- стоимость тура (средняя, максимальная, минимальная)

-- в зависимости от города

--

create or replace function get\_tours\_prices\_by\_city\_id(\_city\_id integer) returns table (

city varchar(35),

max numeric,

avg numeric,

min numeric

) as '

declare city varchar(30);

declare max numeric;

declare avg numeric;

declare min numeric;

begin

if (select count(id) from tours where city\_id=\_city\_id)=0 then

return;

end if;

select name into city from cities where id=\_city\_id;

select MAX(price) into max from tours where city\_id=\_city\_id;

select AVG(price) into avg from tours where city\_id=\_city\_id;

select MIN(price) into min from tours where city\_id=\_city\_id;

return query

select city, max, avg, min;

end;' language 'plpgsql';

-- анализ спроса туров среди путешественников местных и иногородних

--

-- идея: добавить столбец id города для клиента. входной параметр - id города. выходные - число туров с местными и число туров с иногородници клиентами.

ALTER TABLE clients

ADD COLUMN city\_id integer REFERENCES cities(id);

UPDATE clients

SET city\_id = round(1 + random() \* 15);

create or replace function get\_tour\_city\_demands(\_city\_id integer) returns table (\_in integer, \_out integer) as '

declare \_in integer;

declare \_out integer;

begin

if (select count(\*) from cities where id=\_city\_id)=0 then

return;

end if;

select count(\*) into \_in from tours t join bookings b

on b.tour\_id=t.id join clients c on c.id=b.client\_id

where t.city\_id=\_city\_id AND t.city\_id=c.city\_id;

select count(\*) into \_out from tours t join bookings b

on b.tour\_id=t.id join clients c on c.id=b.client\_id

where t.city\_id=\_city\_id AND t.city\_id<>c.city\_id;

return query select \_in, \_out;

end;' language 'plpgsql';

-- индексы

-- наверное стоит добавить индесы только для полей:

-- clients.age, tours.duration, tours.price, tours.members\_count.

CREATE INDEX age\_index ON clients USING HASH(age);

CREATE INDEX duration\_index ON tours USING HASH(duration);

CREATE INDEX price\_index ON tours USING HASH(price);

CREATE INDEX members\_index ON tours USING HASH(members\_count);

-- роли

-- оператор

CREATE ROLE operator;

GRANT SELECT,

INSERT,

UPDATE,

DELETE ON cities,

clients,

bookings,

tours,

tour\_types,

guides TO operator;

-- пользователь

CREATE ROLE user\_db;

GRANT SELECT,

INSERT,

UPDATE ON cities,

clients,

bookings,

tours,

tour\_types,

guides TO user\_db;

-- аналитик

CREATE ROLE analyst;

GRANT SELECT ON cities,

clients,

bookings,

tours,

tour\_types,

guides TO analyst;

GRANT EXECUTE ON FUNCTION get\_tour\_city\_demands,

get\_tours\_prices\_by\_city\_id,

get\_avg\_duration\_in\_season,

get\_tour\_types\_rating,

get\_recomend\_tours TO analyst;

-- админ

CREATE ROLE admin SUPERUSER CREATEROLE CREATEDB;