Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Факультет «Информационных технологий и анализа больших данных»

**ОТЧЕТ**

**по практической работе № 7**

**по теме «Функции и процедуры. Язык PL/pgSQL»**

**по дисциплине Б.1.1.3.4 Системы управления базами данных**

**Вариант 1**

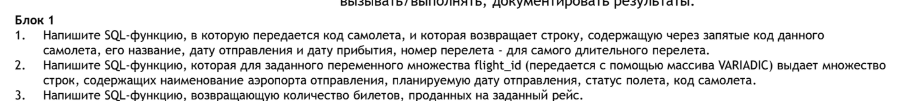
Выполнил: студент группы ИД23-1

Маслов Александр Николаевич

Проверил преподаватель: Цветков И.Е.

**Москва – 2025**

**Задание №1**

****

**Решение:**

**1.**

CREATE OR REPLACE FUNCTION get\_aircraft\_longest\_flight\_info(ac\_code CHAR(3))

RETURNS TEXT AS

$$

SELECT CONCAT\_WS(', ',

f.aircraft\_code,

ad.model,

f.scheduled\_departure,

f.scheduled\_arrival,

f.flight\_no)

FROM bookings.flights f

JOIN bookings.aircrafts\_data ad ON f.aircraft\_code = ad.aircraft\_code

WHERE f.aircraft\_code = ac\_code

ORDER BY (f.scheduled\_arrival - f.scheduled\_departure) DESC

LIMIT 1;

$$ LANGUAGE sql;

**Проверка:**

SELECT get\_aircraft\_longest\_flight\_info('319')

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Контент, сгенерированный ИИ, может содержать ошибки.

**2.**

CREATE OR REPLACE FUNCTION get\_flights\_info(VARIADIC flight\_ids INTEGER[])

RETURNS TABLE (

departure\_airport\_name TEXT,

scheduled\_departure TIMESTAMPTZ,

flight\_status VARCHAR,

aircraft\_code CHAR(3)

)

AS

$$

SELECT

(a.airport\_name ->> 'en')::TEXT,

f.scheduled\_departure,

f.status,

f.aircraft\_code

FROM bookings.flights f

JOIN bookings.airports\_data a ON f.departure\_airport = a.airport\_code

WHERE f.flight\_id = ANY(flight\_ids);

$$ LANGUAGE sql;

**Проверка:**

SELECT bookings.get\_flights\_info(550,10000,23500)

Изображение выглядит как текст, Шрифт, программное обеспечение, линия

Контент, сгенерированный ИИ, может содержать ошибки.

**3.**

CREATE OR REPLACE FUNCTION get\_ticket\_count(flight\_id\_input INTEGER)

RETURNS INTEGER AS

$$

SELECT COUNT(\*)

FROM bookings.boarding\_passes

WHERE flight\_id = flight\_id\_input;

$$ LANGUAGE sql;

**Проверка:**

SELECT bookings.get\_ticket\_count(239)

Изображение выглядит как текст, Шрифт, число, линия

Контент, сгенерированный ИИ, может содержать ошибки.

**Задание №2**

**Изображение выглядит как текст, Шрифт, снимок экрана, документ

Контент, сгенерированный ИИ, может содержать ошибки.**

**Решение:  
1.**

CREATE TABLE bookings\_copy AS TABLE bookings.bookings;

CREATE OR REPLACE PROCEDURE increase\_total\_amount(percentage NUMERIC)

AS

$$

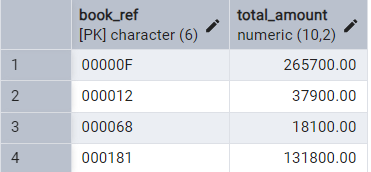
UPDATE bookings\_copy

SET total\_amount = total\_amount + (total\_amount \* percentage / 100);

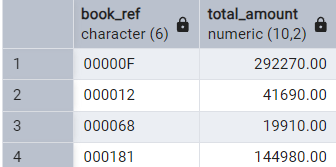
$$ LANGUAGE sql;

**Проверка:**

CALL increase\_total\_amount(10);



До выполнения процедуры

  
После выполнения процедуры

**2.**CREATE OR REPLACE FUNCTION get\_passenger\_info(

ticket CHAR(13),

OUT passenger\_id VARCHAR,

OUT full\_name TEXT,

OUT contact JSONB

)

AS

$$

SELECT

t.passenger\_id,

t.passenger\_name,

t.contact\_data

FROM bookings.tickets t

WHERE t.ticket\_no = ticket;

$$ LANGUAGE sql;

**Проверка:**

SELECT \* FROM get\_passenger\_info('0005432000987');

**Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.**

**3.**

CREATE OR REPLACE FUNCTION sum\_variadic(VARIADIC nums NUMERIC[])

RETURNS NUMERIC AS

$$

SELECT SUM(n) FROM unnest(nums) AS n;

$$ LANGUAGE sql;

**Проверка:**

SELECT sum\_variadic( 5, 3, 10)

Изображение выглядит как текст, Шрифт, линия, Прямоугольник

Контент, сгенерированный ИИ, может содержать ошибки.

**4.**

CREATE OR REPLACE FUNCTION get\_seats\_by\_fare(fare\_cond VARCHAR)

RETURNS TABLE (

aircraft\_code CHAR(3),

seat\_no VARCHAR,

fare\_conditions VARCHAR

)

AS

$$

SELECT s.aircraft\_code, s.seat\_no, s.fare\_conditions

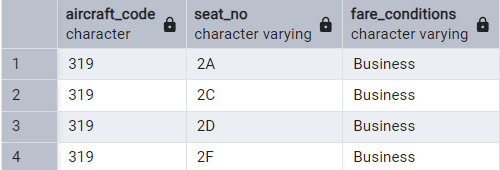
FROM bookings.seats s

WHERE s.fare\_conditions = fare\_cond;

$$ LANGUAGE sql;

**Проверка:**

SELECT \* FROM get\_seats\_by\_fare('Business')



**5.**

CREATE OR REPLACE FUNCTION get\_seats\_by\_fare\_record(fare\_cond VARCHAR)

RETURNS SETOF RECORD AS

$$

SELECT s.aircraft\_code, s.seat\_no, s.fare\_conditions

FROM bookings.seats s

WHERE s.fare\_conditions = fare\_cond;

$$ LANGUAGE sql;

**Проверка:**

SELECT \* FROM get\_seats\_by\_fare\_record('Economy') AS (

aircraft\_code CHAR(3),

seat\_no VARCHAR,

fare\_conditions VARCHAR

);

**Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.**

**Задание №3**

**Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.**

**Решение:**

**1.**

CREATE OR REPLACE FUNCTION nested\_variables\_demo()

RETURNS TEXT AS $$

DECLARE

var\_name TEXT := 'Уровень 1';

BEGIN

RAISE NOTICE 'Внешний блок: %', var\_name;

DECLARE

var\_name TEXT := 'Уровень 2';

BEGIN

RAISE NOTICE 'Вложенный блок 1: %', var\_name;

DECLARE

var\_name TEXT := 'Уровень 3';

BEGIN

RAISE NOTICE 'Вложенный блок 2: %', var\_name;

var\_name := 'Изменено на уровне 3';

RAISE NOTICE 'Вложенный блок 2 (после изменения): %', var\_name;

END;

RAISE NOTICE 'Вложенный блок 1 (после вложенного блока 2): %', var\_name;

var\_name := 'Изменено на уровне 2';

RAISE NOTICE 'Вложенный блок 1 (после изменения): %', var\_name;

END;

RAISE NOTICE 'Внешний блок (после всех вложенных блоков): %', var\_name;

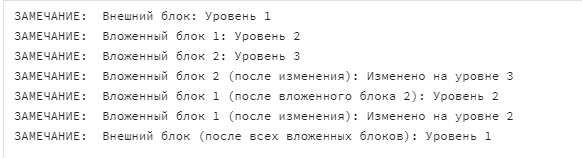
RETURN 'Демонстрация завершена';

END;

$$ LANGUAGE plpgsql;

**Проверка:**

SELECT nested\_variables\_demo();



**2.**

**Динамическая команда SELECT (ищет самолет по переданному коду)**

CREATE TABLE aircrafts\_data\_copy AS SELECT \* FROM bookings.aircrafts\_data;

CREATE OR REPLACE FUNCTION dynamic\_select\_example(p\_aircraft\_code CHAR(3))

RETURNS TABLE(model TEXT, range INT) AS $$

BEGIN

RETURN QUERY EXECUTE

'SELECT model->>''ru'' as model, range

FROM aircrafts\_data\_copy

WHERE aircraft\_code = $1'

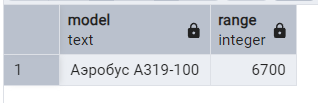
USING p\_aircraft\_code;

END;

$$ LANGUAGE plpgsql;

**Проверка:**

SELECT \* FROM dynamic\_select\_example('319')



**Динамическая команда UPDATE (обновляет статус рейса по коду)**

CREATE TABLE flights\_copy AS SELECT \* FROM bookings.flights;

CREATE OR REPLACE FUNCTION dynamic\_update\_example(p\_flight\_id INT, p\_status TEXT)

RETURNS INT AS $$

DECLARE

rows\_updated INT;

BEGIN

EXECUTE

'UPDATE flights\_copy

SET status = $1

WHERE flight\_id = $2

RETURNING 1'

INTO rows\_updated

USING p\_status, p\_flight\_id;

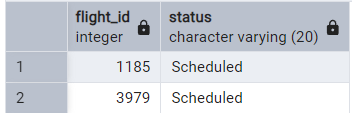
RETURN rows\_updated;

END;

$$ LANGUAGE plpgsql;

**Проверка:**

SELECT \* FROM dynamic\_update\_example(1185, 'Departed');



До выполнения функции

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Контент, сгенерированный ИИ, может содержать ошибки.

После выполнения функции

**Динамическая команда DELETE (удаляет из заданной таблицы, где есть переданное значение airport\_code)**

CREATE TABLE airports\_data\_copy AS SELECT \* FROM bookings.airports\_data;

CREATE OR REPLACE FUNCTION dynamic\_delete\_example(p\_table\_name TEXT, p\_airport\_code CHAR(3))

RETURNS INT AS $$

DECLARE

rows\_deleted INT;

BEGIN

EXECUTE

FORMAT('DELETE FROM %I

WHERE airport\_code = $1

RETURNING 1', p\_table\_name)

INTO rows\_deleted

USING p\_airport\_code;

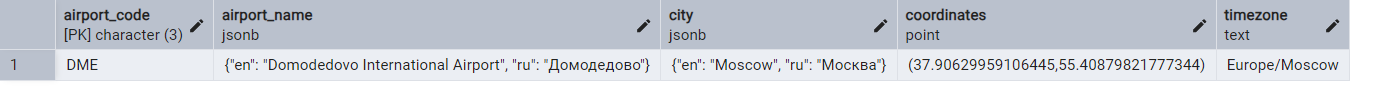
RETURN rows\_deleted;

END;

$$ LANGUAGE plpgsql;

**Проверка:**

SELECT dynamic\_delete\_example('airports\_data\_copy', 'DME');



До выполнения функции

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

После выполнения функции

**3.**

**Задание 4**

CREATE OR REPLACE FUNCTION get\_seats\_by\_fare(fare\_cond VARCHAR)

RETURNS TABLE (

aircraft\_code CHAR(3),

seat\_no VARCHAR,

fare\_conditions VARCHAR

) AS $$

BEGIN

RETURN QUERY

SELECT

s.aircraft\_code,

s.seat\_no,

s.fare\_conditions

FROM bookings.seats s

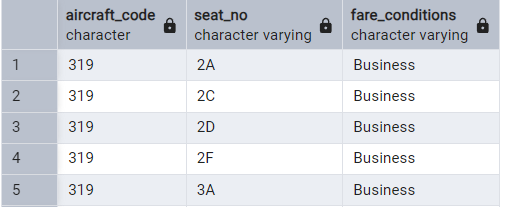
WHERE s.fare\_conditions = fare\_cond;

END;

$$ LANGUAGE plpgsql;

**Проверка:**

SELECT \* FROM get\_seats\_by\_fare('Business')



**Задание 5**

CREATE OR REPLACE FUNCTION get\_seats\_by\_fare\_record(fare\_cond VARCHAR)

RETURNS SETOF RECORD AS $$

BEGIN

RETURN QUERY

SELECT \*

FROM bookings.seats

WHERE fare\_conditions = fare\_cond;

END;

$$ LANGUAGE plpgsql;

**Проверка:**

SELECT \* FROM get\_seats\_by\_fare\_record('Economy') AS (

aircraft\_code CHAR(3),

seat\_no VARCHAR,

fare\_conditions VARCHAR

);

**Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.**

**4.**

CREATE OR REPLACE PROCEDURE find\_similar\_range\_aircrafts(p\_range\_diff INTEGER)

AS $$

DECLARE

rec1 RECORD;

rec2 RECORD;

BEGIN

RAISE NOTICE 'Пары самолетов с разницей дальности не более % км:', p\_range\_diff;

RAISE NOTICE '------------------------------------------------';

FOR rec1 IN SELECT aircraft\_code, model->>'ru' as model\_name, range

FROM bookings.aircrafts\_data

LOOP

FOR rec2 IN SELECT aircraft\_code, model->>'ru' as model\_name, range

FROM bookings.aircrafts\_data

WHERE aircraft\_code > rec1.aircraft\_code

LOOP

IF ABS(rec1.range - rec2.range) <= p\_range\_diff THEN

RAISE NOTICE '%- % (дальность: % км) и %- % (дальность: % км) - разница: % км',

rec1.aircraft\_code, rec1.model\_name, rec1.range,

rec2.aircraft\_code, rec2.model\_name, rec2.range,

ABS(rec1.range - rec2.range);

END IF;

END LOOP;

END LOOP;

RAISE NOTICE '------------------------------------------------';

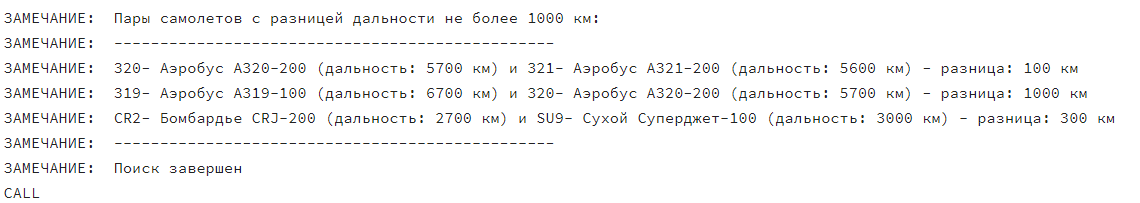
RAISE NOTICE 'Поиск завершен';

END;

$$ LANGUAGE plpgsql;

**Проверка:**

CALL find\_similar\_range\_aircrafts(1000);



**5.**

CREATE OR REPLACE FUNCTION get\_flights\_by\_departure\_city\_simple(p\_city\_name TEXT)

RETURNS TABLE(

flight\_no TEXT,

departure\_city TEXT,

arrival\_city TEXT

) AS $$

DECLARE

flight\_record RECORD;

BEGIN

FOR flight\_record IN

SELECT

f.flight\_no,

dep.city->>'ru' as departure\_city,

arr.city->>'ru' as arrival\_city

FROM

bookings.flights f

JOIN bookings.airports\_data dep ON f.departure\_airport = dep.airport\_code

JOIN bookings.airports\_data arr ON f.arrival\_airport = arr.airport\_code

WHERE

dep.city->>'ru' = p\_city\_name

LOOP

flight\_no := flight\_record.flight\_no;

departure\_city := flight\_record.departure\_city;

arrival\_city := flight\_record.arrival\_city;

RETURN NEXT;

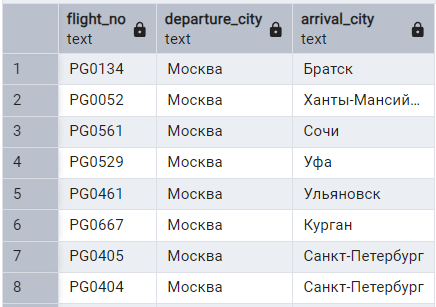
END LOOP;

END;

$$ LANGUAGE plpgsql;

**Проверка:**

SELECT \* FROM get\_flights\_by\_departure\_city\_simple('Москва')

****