--Федеральное государственное образовательное бюджетное

учреждение высшего образования

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

**(Финансовый университет)**

Факультет «Информационных технологий и анализа больших данных»

**ОТЧЕТ**

**по контрольной работе №2**

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

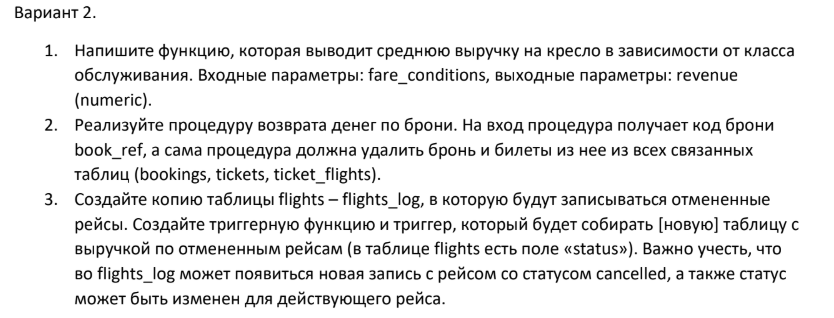
**Вариант 2**

Выполнил: студент группы ИД23-1

Маслов Александр Николаевич

Проверил преподаватель: Цветков И.Е.

**Москва – 2025**

**Задание:  
**

**1.**

CREATE OR REPLACE FUNCTION get\_avg\_revenue\_per\_seat(fare\_conditions varchar)

RETURNS numeric AS $$

BEGIN

RETURN (

SELECT AVG(tf.amount)

FROM ticket\_flights tf

WHERE tf.fare\_conditions = get\_avg\_revenue\_per\_seat.fare\_conditions

);

END;

$$ LANGUAGE plpgsql;

**Проверка:**

SELECT bookings.get\_avg\_revenue\_per\_seat(

'Economy'

)

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Контент, сгенерированный ИИ, может содержать ошибки.

**2.**

CREATE OR REPLACE PROCEDURE refund\_booking(p\_book\_ref varchar)

LANGUAGE plpgsql AS $$

BEGIN

DELETE FROM boarding\_passes

WHERE ticket\_no IN (

SELECT ticket\_no FROM tickets

WHERE book\_ref = p\_book\_ref

);

DELETE FROM ticket\_flights

WHERE ticket\_no IN (

SELECT ticket\_no FROM tickets

WHERE book\_ref = p\_book\_ref

);

DELETE FROM tickets

WHERE book\_ref = p\_book\_ref;

DELETE FROM bookings

WHERE book\_ref = p\_book\_ref;

COMMIT;

END;

$$;

**Проверка:**

SELECT \* FROM bookings.bookings

ORDER BY book\_ref ASC LIMIT 100

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

CALL bookings.refund\_booking(

"00000F"

);

SELECT \* FROM bookings.bookings

ORDER BY book\_ref ASC LIMIT 100

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

**3.**

CREATE TABLE flights\_log (

flight\_id integer PRIMARY KEY,

flight\_no character(6) NOT NULL,

scheduled\_departure timestamptz NOT NULL,

scheduled\_arrival timestamptz NOT NULL,

departure\_airport character(3) NOT NULL,

arrival\_airport character(3) NOT NULL,

status character varying(20) NOT NULL,

aircraft\_code character(3) NOT NULL,

actual\_departure timestamptz,

actual\_arrival timestamptz,

revenue numeric(10, 2)

);

CREATE OR REPLACE FUNCTION log\_cancelled\_flights()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.status = 'Cancelled' AND (OLD.status IS NULL OR OLD.status != 'Cancelled') THEN

INSERT INTO flights\_log (

flight\_id, flight\_no, scheduled\_departure, scheduled\_arrival,

departure\_airport, arrival\_airport, status, aircraft\_code,

actual\_departure, actual\_arrival, revenue

)

SELECT

NEW.flight\_id,

NEW.flight\_no,

NEW.scheduled\_departure,

NEW.scheduled\_arrival,

NEW.departure\_airport,

NEW.arrival\_airport,

NEW.status,

NEW.aircraft\_code,

NEW.actual\_departure,

NEW.actual\_arrival,

COALESCE((

SELECT SUM(tf.amount)

FROM ticket\_flights tf

WHERE tf.flight\_id = NEW.flight\_id

), 0)

;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER track\_cancelled\_flights

AFTER INSERT OR UPDATE ON flights

FOR EACH ROW

EXECUTE FUNCTION log\_cancelled\_flights();

**Проверка:**

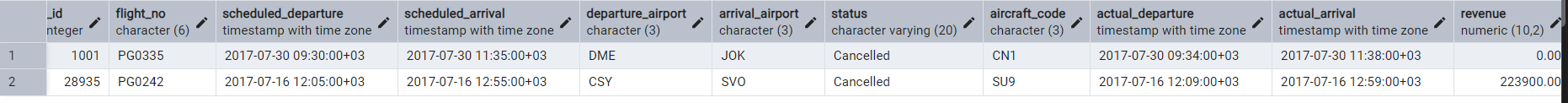
UPDATE flights

SET status = 'Cancelled'

WHERE flight\_id = 28935;

SELECT \* FROM bookings.flights\_log

ORDER BY flight\_id ASC LIMIT 100

****