Проект: итерация 4

Велегурина Елена Б05-121

Пункт 11. Создать не менее 2 триггеров. Логика согласовывается с семинаристом.

1. Триггер для таблицы shop.supply, проверяющий при вставке или обновлении записи, что дата начала поставки меньше даты конца поставки.

```
CREATE OR REPLACE FUNCTION check_supply_dates() RETURNS TRIGGER AS
$$
BEGIN
   IF NEW.date_from >= NEW.date_to THEN
       RAISE EXCEPTION 'Supply start date must be earlier than end
date';
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_supply_dates
   BEFORE INSERT OR UPDATE
   ON shop.supply
   FOR EACH ROW
EXECUTE FUNCTION check_supply_dates();
```

Проверим работу этого триггера, попытавшись вставить поставку, в которой дата начала больше даты конца:

```
INSERT INTO shop.supply
VALUES (12, '2022-04-10 15:25:33', '2021-04-11 12:58:37',
       'Novosibirskaya oblast, Novosibirsk, Vybornaya Ul., bld.
106, appt. 186');
```

Результат:

```
[P0001] ERROR: Supply start date must be earlier than end date
Где: PL/pgSQL function check_supply_dates() line 4 at RAISE
```

Итоговая таблица supply также не изменилась.

2. Триггер, который при добавлении новой поставки в таблицу supply, будет вычитать 1 из amount купленного ковра.

```
CREATE OR REPLACE FUNCTION update_carpet_amount() RETURNS TRIGGER
AS
$$
BEGIN
```

```
    UPDATE shop.carpet
    SET amount = amount - 1
    WHERE carpet_id = NEW.carpet_id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_carpet_amount
    AFTER INSERT
    ON shop.carpet_supply
    FOR EACH ROW
EXECUTE FUNCTION update_carpet_amount();
```

Проверим этот триггер, добавив 12 поставку, в которой был куплен 10 тип ковра. Его количество на складе должно уменьшиться на 1.

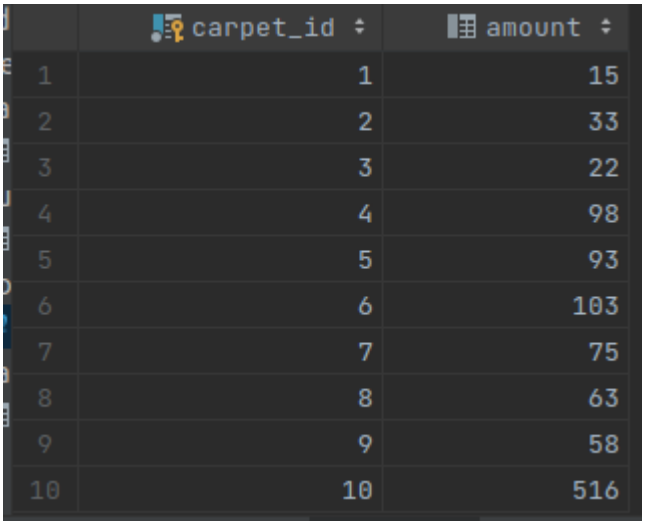
```
INSERT INTO shop.supply
VALUES (12, '2022-04-10 15:25:33', '2023-04-11 12:58:37',
        'Novosibirskaya oblast, Novosibirsk, Vybornaya Ul., bld.
106, appt. 186');

SELECT carpet_id, amount
FROM shop.carpet;
```

Результат:

Таблица до запроса:

	carpet_id	amount
1	1	15
2	2	33
3	3	22
4	4	98
5	5	93
6	6	103
7	7	75
8	8	63
9	9	58
10	10	517

Таблица после запроса:



	carpet_id	amount
1	1	15
2	2	33
3	3	22
4	4	98
5	5	93
6	6	103
7	7	75
8	8	63
9	9	58
10	10	516

3. Триггер для таблицы shop.carpet, проверяющий при вставке или обновлении записи, что производитель этого ковра существует в таблице shop.producer.

```
CREATE OR REPLACE FUNCTION check_producer_exists() RETURNS TRIGGER
AS
$$
BEGIN
   IF NOT EXISTS(SELECT 1
                 FROM shop.producer
                 WHERE producer_id = NEW.producer_id)
   THEN
       RAISE EXCEPTION 'Producer with ID % does not exist',
NEW.producer_id;
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_producer_exists
   BEFORE INSERT OR UPDATE
   ON shop.carpet
   FOR EACH ROW
EXECUTE FUNCTION check_producer_exists();
```

Проверим этот триггер, добавив ковёр от 15-го поставщика, которого не существует.

```
INSERT INTO shop.carpet
VALUES (11, 15, 'Wrong carpet', 15000, 1);
```

Результат:

```
[P0001] ERROR: Producer with ID 15 does not exist
Где: PL/pgSQL function check_producer_exists() line 4 at RAISE
```

Пункт 12. Используя любимый язык программирования и библиотеку, сгенерировать данные и с их помощью вставить данные в уже оформленную БД. Теми же инструментами извлечь данные (из таблицы на выбор), возможно, предварительно агрегированные средствами СУБД, и провести анализ (графики/heatmap - на ваше усмотрение).

Для генерации данных будем использовать библиотеку Faker. Создадим данные для таблицы shop.carpet, заполнив ее случайно сгенерированными значениями для producer_id, name, price и amount. Кроме того, создадим данные для таблицы shop.producer и привяжем их к соответствующим коврам в таблице shop.carpet через внешний ключ producer_id. Добавим для каждого ковра подробное описание в carpet_description. Более того, используем try-except и rollback для обработки ошибок.

```
import os
import pandas as pd
import psycopg2 as pg
from faker import Faker
from dataclasses import dataclass


@dataclass
class Credentials:
    dbname: str = "pg_db"
    host: str = "127.0.0.1"
    port: int = 5432
    user: str = "postgres"
    password: str = "postgres"


def psycopg2_conn_string():
    return f"""
        dbname='{os.getenv("DBNAME", Credentials.dbname)}'
        user='{os.getenv("DBUSER", Credentials.user)}'
        host='{os.getenv("DBHOST", Credentials.host)}'
        port='{os.getenv("DBPORT", Credentials.port)}'
        password='{os.getenv("DBPASSWORD", Credentials.password)}'
"""


def set_connection():
    return pg.connect(psycopg2_conn_string())


if __name__ == '__main__':
    conn = set_connection()
```

```
    cur = conn.cursor()

    # Инициализация Faker
    fake = Faker()

    # Сгенерировать данные для таблицы shop.producer
    for i in range(10):
        producer_name = fake.company()
        producer_country = fake.country()
        try:
             cur.execute("INSERT INTO shop.producer (name, country)
VALUES (%s, %s)",
                         (producer_name, producer_country))
            conn.commit()
        except pg.IntegrityError:
            # Обрабатываем ошибку
            conn.rollback()

    # Получить список всех производителей
    cur.execute("SELECT producer_id FROM shop.producer")
    producer_ids = [row[0] for row in cur.fetchall()]

    # Сгенерировать данные для таблицы shop.carpet
    for i in range(100):
        # Вставка данных в таблицу shop.carpet
                                          carpet_producer_id      =
fake.random_element(elements=producer_ids)
        carpet_name = fake.text(max_nb_chars=50)
        carpet_price = fake.pyint(min_value=10, max_value=1000)
        carpet_amount = fake.pyint(min_value=0, max_value=100)
        try:
                 cur.execute("INSERT INTO shop.carpet (producer_id,
name, price, amount) VALUES (%s, %s, %s, %s)",
                                    (carpet_producer_id,  carpet_name,
carpet_price, carpet_amount))
            conn.commit()
        except pg.IntegrityError:
            # Обрабатываем ошибку
            conn.rollback()

        # Получение текущего значения для carpet_id
        cur.execute("SELECT currval('shop.carpet_carpet_id_seq')")
        carpet_id = cur.fetchone()[0]

        # Вставка данных в таблицу shop.carpet_description
        carpet_category = fake.word()
        carpet_length = fake.pyint(min_value=10, max_value=100)
```

```
        carpet_width = fake.pyint(min_value=10, max_value=100)
        carpet_colour = fake.color_name()
        try:
            cur.execute(
                    "INSERT INTO shop.carpet_description (carpet_id,
category, length, width, colour) VALUES (%s, %s, %s, %s, %s)",
                        (carpet_id, carpet_category, carpet_length,
carpet_width, carpet_colour))
            conn.commit()
        except pg.IntegrityError:
            # Обрабатываем ошибку
            conn.rollback()

    # Закрытие соединения с БД
    cur.close()
    conn.close()
```

Результат:

producer:

24	Leonard Group	Ethiopia
25	Arellano, Hernandez and Marquez	Azerbaijan
26	Page-Jones	Suriname
27	Cobb PLC	Saint Kitts and Nev…
28	Ross-Green	Comoros
29	Walsh, Sampson and Douglas	Bangladesh
30	Sweeney and Sons	Panama
31	Duncan Group	Sierra Leone
32	Keith, Gates and Giles	Antigua and Barbuda
33	Miller, Whitehead and Baker	Netherlands Antilles

carpet:

232	33	Customer foreign Mrs cultur…	504	41
233	25	Return generation charge ba…	854	61
234	26	Town say think simple someo…	65	83
235	4	Reality born piece agree in…	972	14
236	33	Federal range idea. They or…	806	27
237	30	Teacher white population im…	45	60
238	1	Lose red last some miss wid…	313	91
239	24	Body room field.	46	4
240	27	Night relationship everyone…	495	21
241	27	Usually behind trade relati…	520	20
242	3	Break our until will miss a…	220	15
243	1	Successful action when cent…	519	12
244	31	Myself could kitchen appear.	768	1
245	3	Expect nearly organization …	109	6
246	26	Ten research political purp…	540	45

carpet_description:

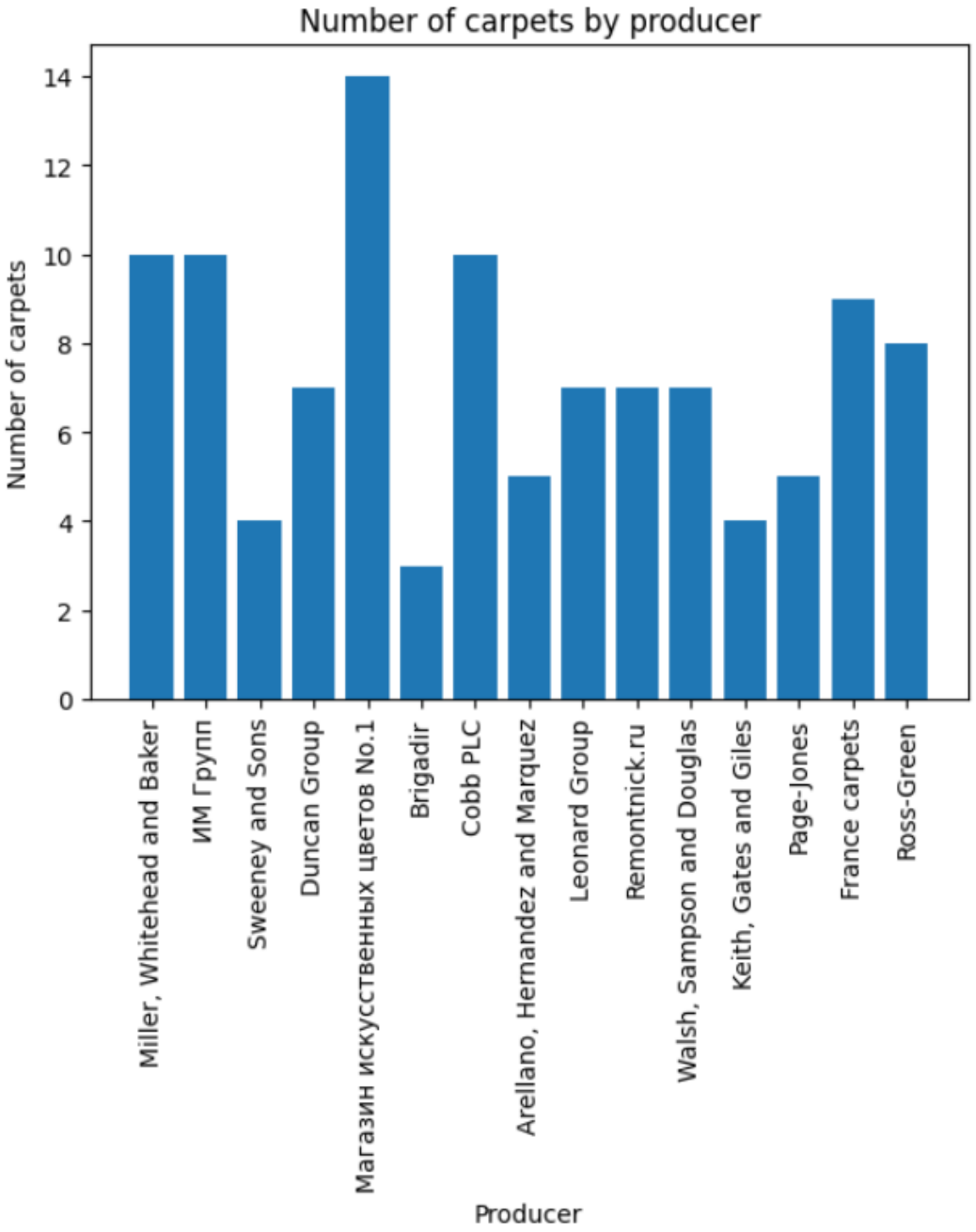
214	while	69	78	LightYellow
215	Mrs	53	14	MidnightBlue
216	experience	12	19	OliveDrab
217	others	15	20	Tomato
218	much	89	100	Plum
219	remember	27	85	RosyBrown
220	piece	42	10	LavenderBlush
221	exactly	99	18	DarkOrchid
222	prove	10	13	Azure
223	mother	88	61	FireBrick
224	rule	35	91	MintCream
225	station	76	60	Olive
226	agent	53	10	OldLace
227	financial	75	71	SeaShell
228	feel	55	57	Gray
229	these	65	21	DarkSlateBlue
230	read	52	91	Azure

Теперь, когда данные сгенерированы и добавлены в БД, выберем некоторые данные из таблицы shop.carpet и проведём анализ. Используя библиотеку matplotlib для создания графика, создадим график количества ковров по каждому производителю, отображая данные с помощью столбчатой диаграммы.

```
import os
import pandas as pd
import psycopg2 as pg
from faker import Faker
from dataclasses import dataclass
import matplotlib.pyplot as plt


@dataclass
class Credentials:
    dbname: str = "pg_db"
    host: str = "127.0.0.1"
    port: int = 5432
    user: str = "postgres"
    password: str = "postgres"


def psycopg2_conn_string():
    return f"""
        dbname='{os.getenv("DBNAME", Credentials.dbname)}'
        user='{os.getenv("DBUSER", Credentials.user)}'
```

```
        host='{os.getenv("DBHOST", Credentials.host)}'
        port='{os.getenv("DBPORT", Credentials.port)}'
        password='{os.getenv("DBPASSWORD", Credentials.password)}'
"""


def set_connection():
    return pg.connect(psycopg2_conn_string())


if __name__ == '__main__':
    conn = set_connection()

    cur = conn.cursor()
    # Получить количество ковров по каждому производителю
    cur.execute(
        "SELECT p.name, count(c.carpet_id) "
        "FROM shop.producer p "
        "JOIN shop.carpet c ON p.producer_id = c.producer_id "
        "GROUP BY p.name")
    rows = cur.fetchall()

    # Создать столбчатую диаграмму
    names = [row[0] for row in rows]
    values = [row[1] for row in rows]
    plt.bar(names, values)

    plt.title("Number of carpets by producer")
    plt.xlabel("Producer")
    plt.ylabel("Number of carpets")
    plt.xticks(rotation=90)
    plt.show()

    # Закрытие соединения с БД
    cur.close()
    conn.close()
```

Результат:



Далее изучим, из каких стран наши поставщики. Это будет удобно увидеть на круговой диаграмме.

```
import os
import pandas as pd
import psycopg2 as pg
from faker import Faker
from dataclasses import dataclass
import matplotlib.pyplot as plt


@dataclass
class Credentials:
```

```
    dbname: str = "pg_db"
    host: str = "127.0.0.1"
    port: int = 5432
    user: str = "postgres"
    password: str = "postgres"


def psycopg2_conn_string():
    return f"""
        dbname='{os.getenv("DBNAME", Credentials.dbname)}'
        user='{os.getenv("DBUSER", Credentials.user)}'
        host='{os.getenv("DBHOST", Credentials.host)}'
        port='{os.getenv("DBPORT", Credentials.port)}'
        password='{os.getenv("DBPASSWORD", Credentials.password)}'
"""


def set_connection():
    return pg.connect(psycopg2_conn_string())


if __name__ == '__main__':
    conn = set_connection()

    cur = conn.cursor()

      # SQL-запрос для получения количества ковров, произведенных
каждой страной производителем
    query = """
    SELECT p.country, count(c.carpet_id)
    FROM shop.producer p
    LEFT JOIN shop.carpet c ON p.producer_id = c.producer_id
    GROUP BY p.country;
    """

    # Выполнение SQL-запроса и получение результатов
    cur.execute(query)
    rows = cur.fetchall()

    # Построение круговой диаграммы
    labels = [row[0] for row in rows]
    sizes = [row[1] for row in rows]
    plt.pie(sizes, labels=labels, autopct='%1.1f%%')
    plt.title("Страны производителей ковров")
    plt.axis('equal')
    plt.show()

    # Закрытие соединения с БД
```

```
    cur.close()
    conn.close()
```

Результат: