Procedura

- Procedura wprowadzająca nowe wydarzenie do bazy dla użytkownika.

```
CREATE PROCEDURE nowe_wydarzenie( id_wydarzenia int, nazwa varchar(50), opis varchar(200),
data_rozpoczecia date, data_zakonczenia date, id_lokalu int)
LANGUAGE SQL
AS $$
  insert into wydarzenia values (id_wydarzenia, nazwa, opis, data_rozpoczecia, data_zakonczenia,
id_lokalu);
$$;

CALL nowe_wydarzenie( 6, 'zlot wedkarzy', 'zlot fanow wedkowania i ryb', '2024-10-10', '2024-10-12',
3);
```

- Zmiana ceny biletu

```
CREATE PROCEDURE zmien_cene_biletu(bilet int, koszt NUMERIC(10, 2))
LANGUAGE SQL
AS $$
        UPDATE bilety SET cena = koszt WHERE id_biletu = bilet;
$$;

CALL zmien_cene_biletu(1, 1000.00)
```

Funkcja

- Oblicza przychód dla każdego wydarzenia

```
CREATE OR REPLACE FUNCTION oblicz_przychody()
RETURNS TABLE (id_wydarzenia INT, przychod NUMERIC(10, 2))
LANGUAGE SQL
AS $$
   SELECT wydarzenia.id_wydarzenia, SUM(kupione_bilety.ilosc * bilety.cena) AS przychod
   FROM wydarzenia
   JOIN bilety ON wydarzenia.id_wydarzenia = bilety.id_wydarzenia
   JOIN kupione_bilety ON bilety.id_biletu = kupione_bilety.id_biletu
   GROUP BY wydarzenia.id_wydarzenia;
$$;

SELECT * FROM oblicz_przychody();
```

- pokazuje jakie na jakie wydarzenia ktoś idzie

```
CREATE OR REPLACE FUNCTION pobierz_wydarzenia_dla_uczestnika(id_uczestnika INT)
RETURNS TABLE (id_wydarzenia INT, nazwa_wydarzenia VARCHAR, data_rozpoczecia DATE,
data_zakonczenia DATE)
LANGUAGE SQL
AS $$
   SELECT w.id_wydarzenia, w.nazwa AS nazwa_wydarzenia, w.data_rozpoczecia, w.data_zakonczenia
   FROM wydarzenia w
   JOIN bilety b ON w.id_wydarzenia = b.id_wydarzenia
   JOIN kupione_bilety kb ON b.id_biletu = kb.id_biletu
   WHERE kb.id_uczestnika = id_uczestnika
   GROUP BY w.id_wydarzenia;
```

```
$$;

SELECT * FROM pobierz_wydarzenia_dla_uczestnika(1);
```

Widok

- Widok przedstawia wszystkie zakupione bilety

```
CREATE VIEW widok_wydarzenia_uczestnicy AS
SELECT w.id_wydarzenia, w.nazwa AS nazwa_wydarzenia, w.opis AS opis_wydarzenia,
w.data_rozpoczecia, w.data_zakonczenia, u.imie AS imie_uczestnika, u.nazwisko AS
nazwisko_uczestnika, b.nazwa AS nazwa_biletu, b.cena AS cena_biletu, kb.ilosc AS ilosc_biletow
FROM wydarzenia w
JOIN bilety b ON w.id_wydarzenia = b.id_wydarzenia
JOIN kupione_bilety kb ON b.id_biletu = kb.id_biletu
JOIN uczestnicy u ON kb.id_uczestnika = u.id_uczestnika;


SELECT * FROM widok_wydarzenia_uczestnicy;
```

Wyzwalacz

- Usuwa bilety oraz inne powiązania gdy chcemy zarchiwizować wydarzenie

```
CREATE OR REPLACE FUNCTION usun_wydarzenie()
RETURNS TRIGGER AS $$
BEGIN
   DELETE FROM kupione_bilety WHERE id_biletu IN (SELECT id_biletu FROM bilety WHERE
id_wydarzenia = OLD.id_wydarzenia);
   DELETE FROM bilety WHERE id_wydarzenia = OLD.id_wydarzenia;
   DELETE FROM personel_wydarzenia WHERE id_wydarzenia = OLD.id_wydarzenia;
   RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER usun_wydarzenie_trigger
BEFORE DELETE ON wydarzenia
FOR EACH ROW
EXECUTE FUNCTION usun_wydarzenie();

DELETE FROM wydarzenia WHERE id_wydarzenia = 1;
```

- Nie pozwala na dodanie wydarzenia jeśli lokal jest już zajęty w danym okresie.

```
CREATE OR REPLACE FUNCTION sprawdz_lokal_przed_wstawieniem()
RETURNS TRIGGER AS $$
BEGIN
   IF EXISTS (
      SELECT 1
      FROM wydarzenia
      WHERE id_lokalu = NEW.id_lokalu
      AND NOT (
         NEW.data_zakonczenia <= data_rozpoczecia OR
```

```
        NEW.data_rozpoczecia >= data_zakonczenia
      )
  ) THEN
    RAISE EXCEPTION 'Lokal jest już zajęty przez inne wydarzenie w wybranym okresie trwania.';
  ELSE
    RETURN NEW;
  END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER przed_wstawieniem_nowego_wydarzenia
BEFORE INSERT ON wydarzenia
FOR EACH ROW
EXECUTE FUNCTION sprawdz_lokal_przed_wstawieniem();


INSERT INTO wydarzenia (id_wydarzenia, nazwa, opis, data_rozpoczecia, data_zakonczenia, id_lokalu)
VALUES (6, 'Nowe wydarzenie', 'Opis nowego wydarzenia', '2024-07-01', '2024-07-20', 1);

Having: ( lokale które mają zaplanowane więcej niż jedno wydarzenie)
SELECT nazwa, COUNT(*) AS liczba_wydarzen
FROM lokale
JOIN wydarzenia ON lokale.id_lokalu = wydarzenia.id_lokalu
GROUP BY nazwa
HAVING COUNT(*) > 1;


3 połączone tabele:

SELECT w.id_wydarzenia, w.nazwa AS nazwa_wydarzenia, w.opis AS opis_wydarzenia,
w.data_rozpoczecia, w.data_zakonczenia, u.imie AS imie_uczestnika, u.nazwisko AS
nazwisko_uczestnika, b.nazwa AS nazwa_biletu, b.cena AS cena_biletu, kb.ilosc AS ilosc_biletow
FROM wydarzenia w
JOIN bilety b ON w.id_wydarzenia = b.id_wydarzenia
JOIN kupione_bilety kb ON b.id_biletu = kb.id_biletu
JOIN uczestnicy u ON kb.id_uczestnika = u.id_uczestnika;

Podzapytanie:
SELECT *
FROM uczestnicy
WHERE id_uczestnika IN (
  SELECT id_uczestnika
  FROM kupione_bilety
  GROUP BY id_uczestnika
  HAVING COUNT(*) = 1
);
```