SQL - Funkcje okna (Window functions)
Lab 1

Imiona i nazwiska: Adam Woźny, Damian Torbus

Celem ćwiczenia jest przygotowanie środowiska pracy, wstępne zapoznanie się z działaniem funkcji okna (window functions) w SQL, analiza wydajności zapytań i porównanie z rozwiązaniami przy wykorzystaniu "tradycyjnych" konstrukcji SQL

Swoje odpowiedzi wpisuj w miejsca oznaczone jako:

> Wyniki:

```
-- ...
```

Ważne/wymagane są komentarze.

Zamieść kod rozwiązania oraz zrzuty ekranu pokazujące wyniki, (dołącz kod rozwiązania w formie tekstowej/ źródłowej)

Zwróć uwagę na formatowanie kodu

Oprogramowanie - co jest potrzebne?

Do wykonania ćwiczenia potrzebne jest następujące oprogramowanie:

- MS SQL Server - wersja 2019, 2022
- PostgreSQL - wersja 15/16/17
- SQLite
- Narzędzia do komunikacji z bazą danych
    - SSMS - Microsoft SQL Managment Studio
    - DtataGrip lub DBeaver
- Przykładowa baza Northwind
    - W wersji dla każdego z wymienionych serwerów

Oprogramowanie dostępne jest na przygotowanej maszynie wirtualnej

Dokumentacja/Literatura

- Kathi Kellenberger, Clayton Groom, Ed Pollack, Expert T-SQL Window Functions in SQL Server 2019, Apres 2019

- Itzik Ben-Gan, T-SQL Window Functions: For Data Analysis and Beyond, Microsoft 2020

- Kilka linków do materiałów które mogą być pomocne - https://learn.microsoft.com/en-us/sql/t-sql/queries/select-over-clause-transact-sql?view=sql-server-ver16
    - https://www.sqlservertutorial.net/sql-server-window-functions/
    - https://www.sqlshack.com/use-window-functions-sql-server/
    - https://www.postgresql.org/docs/current/tutorial-window.html
    - https://www.postgresqltutorial.com/postgresql-window-function/
    - https://www.sqlite.org/windowfunctions.html
    - https://www.sqlitetutorial.net/sqlite-window-functions/
- W razie potrzeby - opis Ikonek używanych w graficznej prezentacji planu zapytania w SSMS jest tutaj:
    - https://docs.microsoft.com/en-us/sql/relational-databases/showplan-logical-and-physical-operators-reference

Przygotowanie

Uruchom SSMS - Skonfiguruj połączenie z bazą Northwind na lokalnym serwerze MS SQL

Uruchom DataGrip (lub Dbeaver)

- Skonfiguruj połączenia z bazą Northwind3
    - na lokalnym serwerze MS SQL
    - na lokalnym serwerze PostgreSQL
    - z lokalną bazą SQLite

Zadanie 1 - obserwacja

Wykonaj i porównaj wyniki następujących poleceń.

```
select avg(unitprice) avgprice
from products p;

select avg(unitprice) over () as avgprice
from products p;

select categoryid, avg(unitprice) avgprice
from products p
group by categoryid

select avg(unitprice) over (partition by categoryid) as avgprice
from products p;
```

Jaka jest są podobieństwa, jakie różnice pomiędzy grupowaniem danych a działaniem funkcji okna?

> Wyniki:

```
-- zwraca jeden rekord wynikowy wraz z średnią ceną produktu, bez żadnych
kryteriów
select avg(unitprice) avgprice
from products p;

-- jako funkcja okna dla każdego rekordu zwraca wartość funkcji avg(unitprice),
-- również bez przyjętego kryterium kategoryzującego, tak więc dla każdego rekordu
zwraca
-- średnią cene wszystkich produktów
select avg(unitprice) over () as avgprice
from products p;

-- jako funkcja agregująca zwraca wartość avg(unitprice), ale z jawnie podanym
-- kryterium grupowania po ID kategorii
-- jako wynik jest podana tabela z wszystkimi ID kategorii i przypisanymi im
średnimi cenami z danej kategorii
select categoryid, avg(unitprice) avgprice
from products p
group by categoryid;

-- funkcja okna z jawnie podanym kryterium kategoryzującym na poziomie ID
kategorii
-- zwraca tyle rekordów ile jest produktów, dla każdeggo produktu zwraca średnią
cene produktów w tej kategorii co on jest
select avg(unitprice) over (partition by categoryid) as avgprice
from products p;

-- Można zauważyć, że funkcje okna wykonują się szybciej od funkcji agregujących
-- Jest to intuicyjne, bo funkcje agregujące poza wyliczeniem wartości, muszą
wykonać również obliczenia umożliwiające agregacje
-- Jednak, z racji na bardzo krótkie czasy wykonania, ciężko stwierdzić to z całą
pewnością
-- oraz określić rząd o jakie funkcje okna są potencjalnie szybsze
```

Zadanie 2 - obserwacja

Wykonaj i porównaj wyniki następujących poleceń.

```
--1)

select p.productid, p.ProductName, p.unitprice,
       (select avg(unitprice) from products) as avgprice
from products p
where productid < 10

--2)
select p.productid, p.ProductName, p.unitprice,
       avg(unitprice) over () as avgprice
```

```
from products p
where productid < 10
```

Jaka jest różnica? Czego dotyczy warunek w każdym z przypadków? Napisz polecenie równoważne

- 1. z wykorzystaniem funkcji okna. Napisz polecenie równoważne
- 2. z wykorzystaniem podzapytania

> Wyniki:

```
--  Różnica (poza śladową wydajnością) polega na tym, że w piewszym zapytaniu w
wyniku dla każdego produktu z ID > 10 jest zwracana
--  średnia cena WSZYSTKICH produktów w tej tabeli, a w drugim dla dla każdego
produktu z ID > 10 jest zwracana produktów z ID > 10

-- 1) z funkcją okna
SELECT * FROM
    (SELECT
        p.productid,
        p.ProductName,
        p.unitprice,
        AVG(p.unitprice) OVER () AS avgprice
    FROM products p
) t
WHERE t.productid < 10;
-- where zawsze wykonuje sie przed nimi wiec trzeba dac takie cos

-- 2) z podzapytaniem
SELECT p.PRODUCTID
    , p.PRODUCTNAME
    , p.UNITPRICE
    , (SELECT AVG(UNITPRICE) FROM PRODUCTS WHERE PRODUCTID < 10)
FROM PRODUCTS p
WHERE p.PRODUCTID < 10;
```

Zadanie 3

Baza: Northwind, tabela: products

Napisz polecenie, które zwraca: id produktu, nazwę produktu, cenę produktu, średnią cenę wszystkich produktów.

Napisz polecenie z wykorzystaniem z wykorzystaniem podzapytania, join'a oraz funkcji okna. Porównaj czasy oraz plany wykonania zapytań.

Przetestuj działanie w różnych SZBD (MS SQL Server, PostgreSql, SQLite)

W SSMS włącz dwie opcje: Include Actual Execution Plan oraz Include Live Query Statistics

 w:700

W DataGrip użyj opcji Explain Plan/Explain Analyze

 w:700

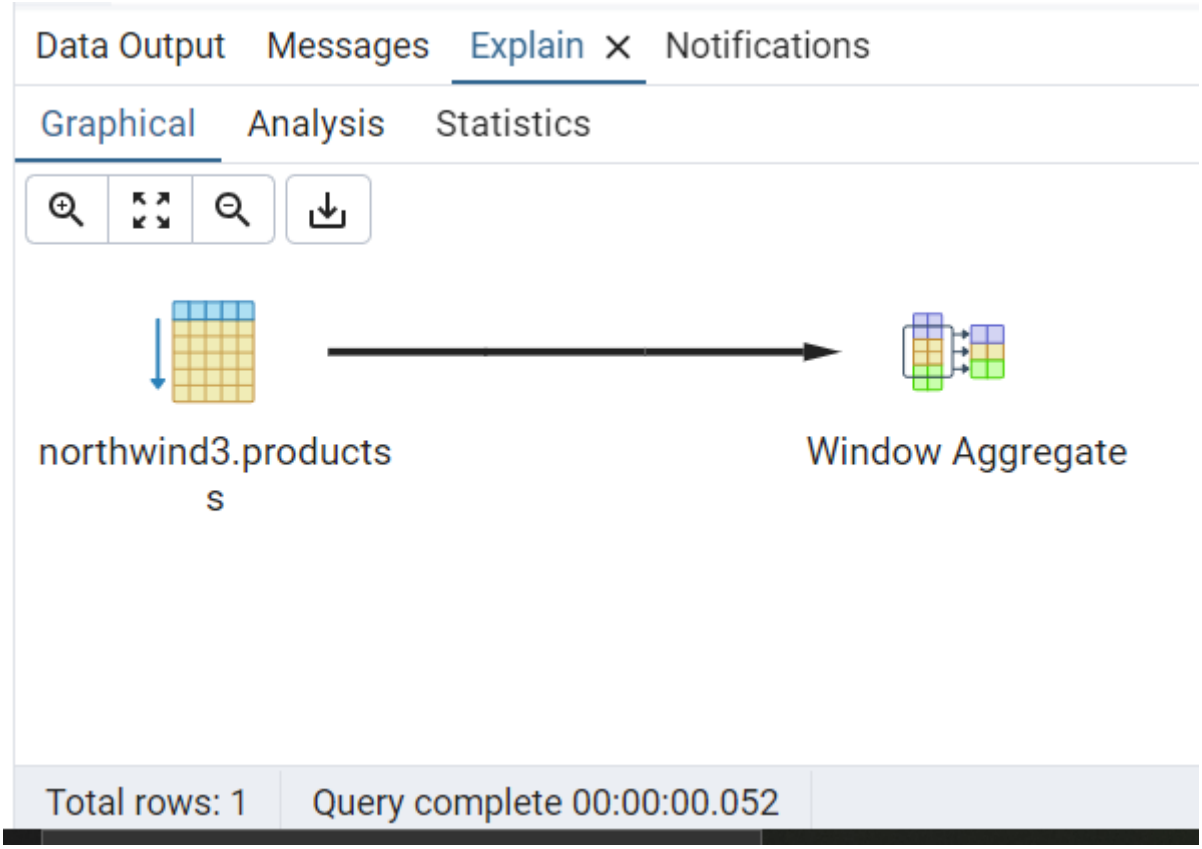
 w:700

> Wyniki:

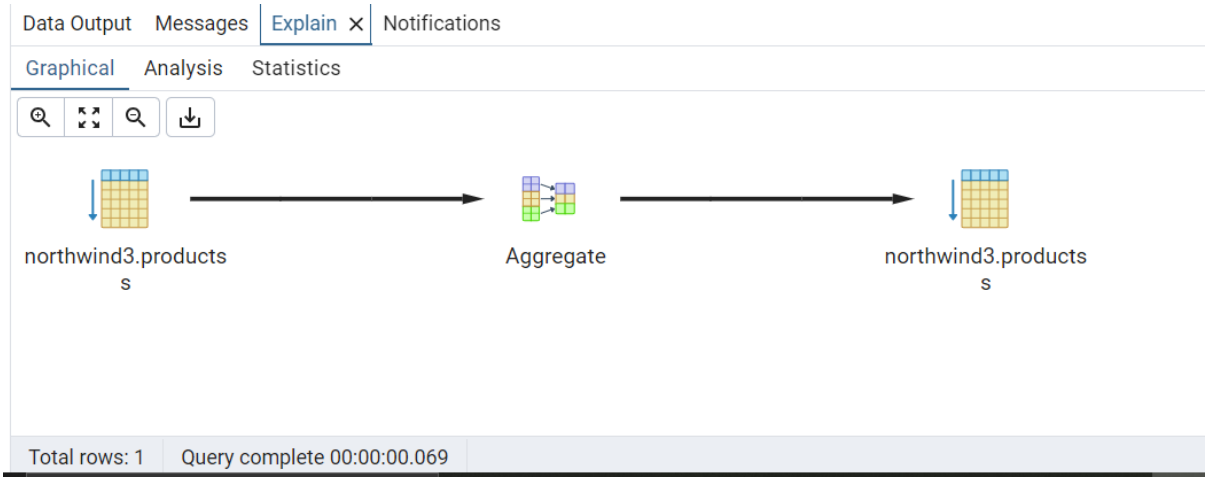
```
-- z funkcją okna:
SELECT PRODUCTID
    , PRODUCTNAME
    , UNITPRICE
    , AVG(UNITPRICE) OVER ()
FROM dbo.PRODUCTS;

-- z podzapytaniem
SELECT PRODUCTID
    , PRODUCTNAME
    , UNITPRICE
    , (SELECT AVG(UNITPRICE) FROM dbo.PRODUCTS)
FROM dbo.PRODUCTS;

-- z joinem
SELECT
    p.PRODUCTID,
    p.PRODUCTNAME,
    p.UNITPRICE,
    avg_prices.avgprice
FROM products p
JOIN (SELECT AVG(UNITPRICE) AS avgprice FROM products) avg_prices
ON 1=1;
```

Wyniki PostrgreSQL

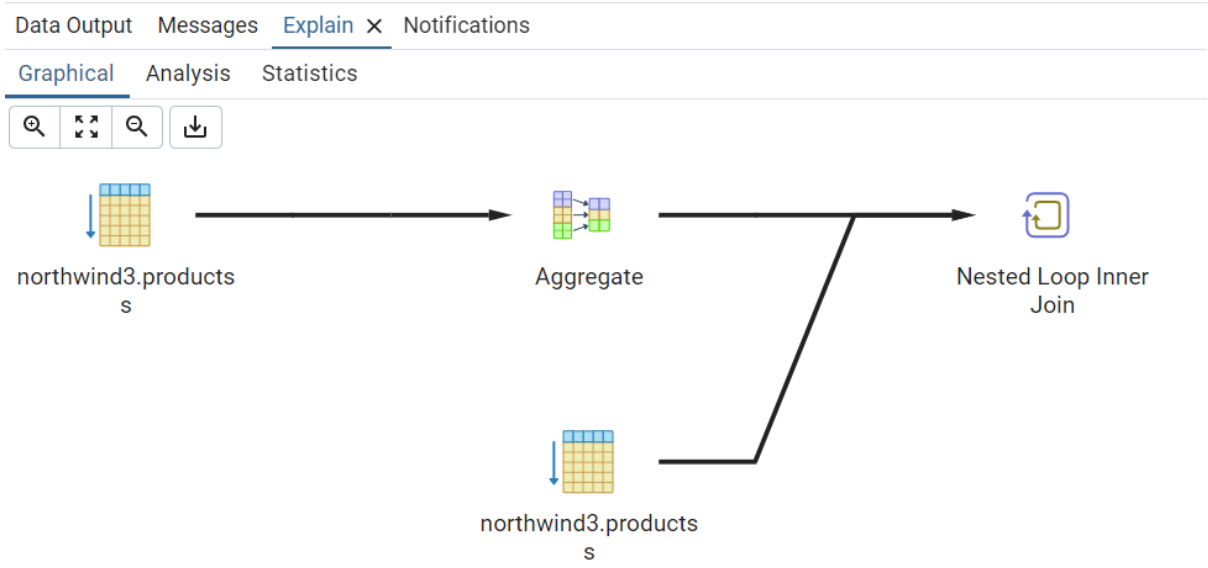
1. z funckją okna
    - Czas wykonania 52ms



2. z podzapytaniem
   - Czas wykonanie 69ms



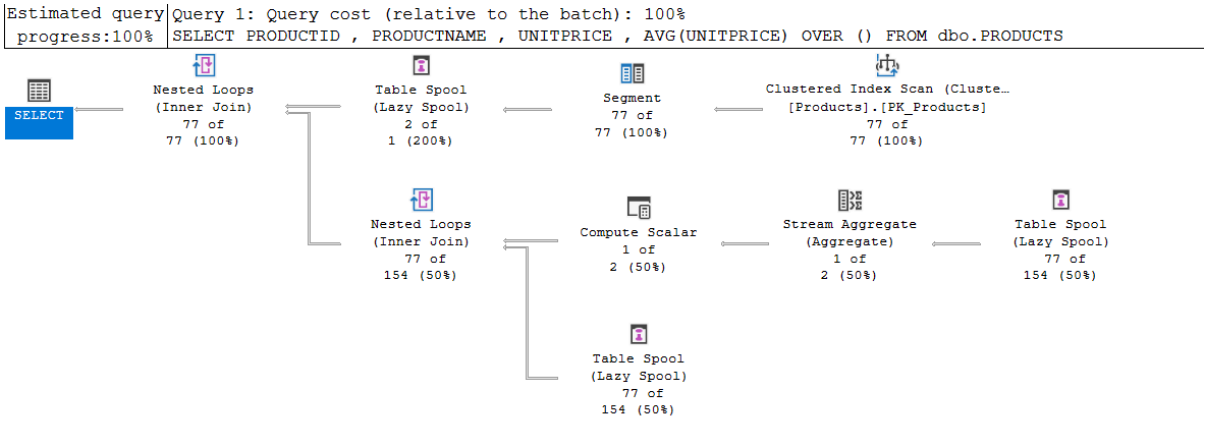
3. z joinem
   - Czas wykonanie 52ms



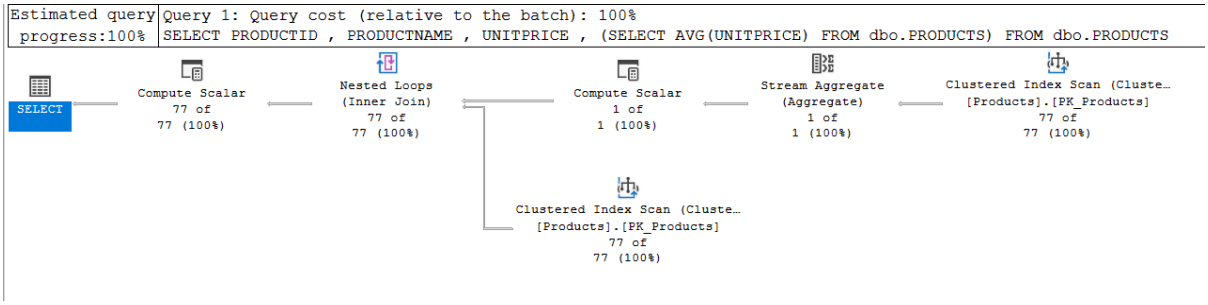
- Total rows: 1 | Query complete 00:00:00.052

Wyniki MsSQL

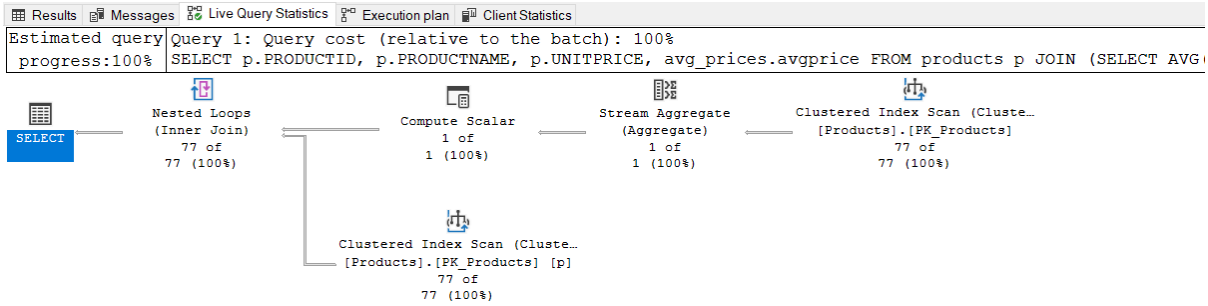
1. z funkcją okna
   - Czas wykonania 76ms



2. z podzapytaniem
   - Czas wykonania 72ms



3. z joinem
   - Czas wykonania 70ms



Wyniki SQLite

1. z funkcją okna
   - Czas wykonania 3ms

```sql
SELECT PRODUCTID
    , PRODUCTNAME
    , UNITPRICE
    , AVG(UNITPRICE) OVER ()
FROM PRODUCTS;
```

	ProductID	ProductName	UnitPrice	AVG(UNITPRICE) OVER ()
1	1	Chai	18	28.8663636363636
2	2	Chang	19	28.8663636363636
3	3	Aniseed Syrup	10	28.8663636363636
4	4	Chef Anton's Cajun Seasoning	22	28.8663636363636
5	5	Chef Anton's Gumbo Mix	21.35	28.8663636363636
6	6	Grandma's Boysenberry Spread	25	28.8663636363636
7	7	Uncle Bob's Organic Dried Pears	30	28.8663636363636

```
Wykonano bez błędów.
Wynik: Zwrócono 77 wierszy w czasie 3ms
W wierszu 1:
SELECT PRODUCTID
            , PRODUCTNAME
            , UNITPRICE
            , AVG(UNITPRICE) OVER ()
FROM PRODUCTS;
```

2. z podzapytaniem
   - Czas wykonania 4ms

```sql
SELECT PRODUCTID
    , PRODUCTNAME
    , UNITPRICE
    , (SELECT AVG(UNITPRICE) FROM PRODUCTS)
FROM PRODUCTS;
```

	ProductID	ProductName	UnitPrice	(SELECT AVG(UNITPRICE) FROM PRODU
1	1	Chai	18	28.8663636363636
2	2	Chang	19	28.8663636363636
3	3	Aniseed Syrup	10	28.8663636363636
4	4	Chef Anton's Cajun Seasoning	22	28.8663636363636
5	5	Chef Anton's Gumbo Mix	21.35	28.8663636363636
6	6	Grandma's Boysenberry Spread	25	28.8663636363636

```
Wykonano bez błędów.
Wynik: Zwrócono 77 wierszy w czasie 4ms
W wierszu 1:
SELECT PRODUCTID
            , PRODUCTNAME
            , UNITPRICE
            , (SELECT AVG(UNITPRICE) FROM PRODUCTS)
FROM PRODUCTS;
```

-
3. z joinem
    - Czas wykonania 3ms

```sql
SELECT
    p.PRODUCTID,
    p.PRODUCTNAME,
    p.UNITPRICE,
    avg_prices.avgprice
FROM products p
JOIN (SELECT AVG(UNITPRICE) AS avgprice FROM products) avg_prices
ON 1=1;
```

	ProductID	ProductName	UnitPrice	avgprice
1	1	Chai	18	28.8663636363636
2	2	Chang	19	28.8663636363636
3	3	Aniseed Syrup	10	28.8663636363636
4	4	Chef Anton's Cajun Seasoning	22	28.8663636363636
5	5	Chef Anton's Gumbo Mix	21.35	28.8663636363636
6	6	Grandma's Boysenberry Spread	25	28.8663636363636
7	7	Uncle Bob's Organic Dried Pears	30	28.8663636363636

```
Wykonano bez błędów.
Wynik: Zwrócono 77 wierszy w czasie 3ms
W wierszu 1:
SELECT
    p.PRODUCTID,
    p.PRODUCTNAME,
    p.UNITPRICE,
    avg_prices.avgprice
FROM products p
JOIN (SELECT AVG(UNITPRICE) AS avgprice FROM products) avg_prices
ON 1=1;
```

Zadanie 4

Baza: Northwind, tabela products

Napisz polecenie, które zwraca: id produktu, nazwę produktu, cenę produktu, średnią cenę produktów w kategorii, do której należy dany produkt. Wyświetl tylko pozycje (produkty) których cena jest większa niż średnia cena.

Napisz polecenie z wykorzystaniem podzapytania, join'a oraz funkcji okna. Porównaj zapytania. Porównaj czasy oraz plany wykonania zapytań.

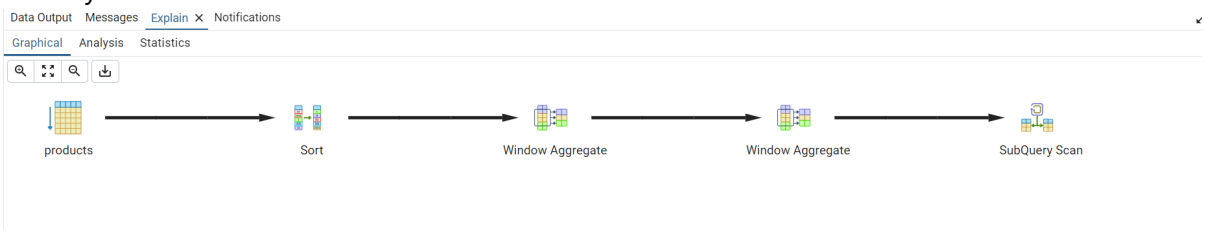
Przetestuj działanie w różnych SZBD (MS SQL Server, PostgreSql, SQLite)

> Wyniki:

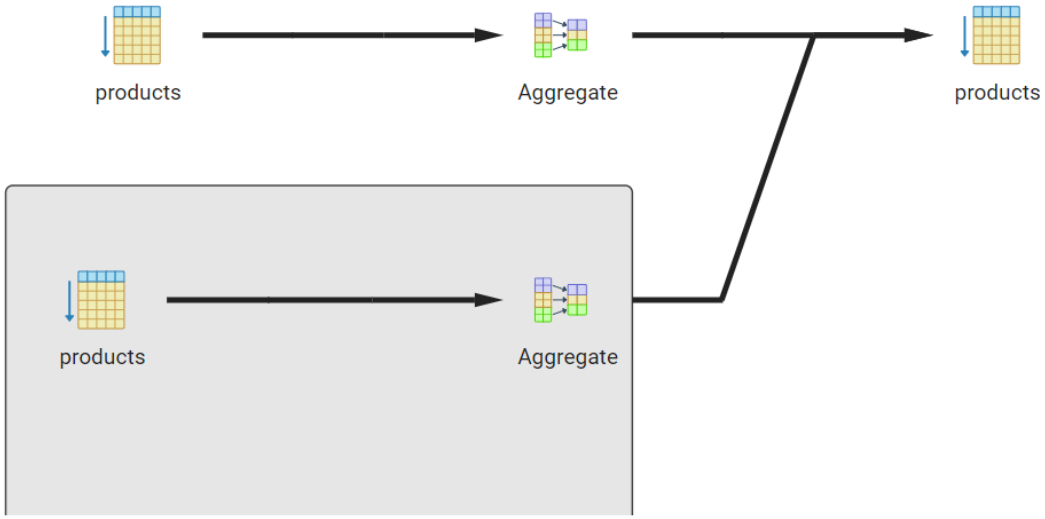
```sql
-- z funkcją oken
WITH PRODUCT_PRICES AS
    (SELECT PRODUCTID
        , PRODUCTNAME
        , UNITPRICE
```

```
        , AVG(UNITPRICE) OVER (PARTITION BY CATEGORYID) AS AVG_PRICE
        , AVG(UNITPRICE) OVER () AS OVERALL_AVG
    FROM dbo.PRODUCTS)
SELECT * FROM PRODUCT_PRICES WHERE UNITPRICE > OVERALL_AVG;

-- z podzapytaniem
SELECT p.PRODUCTID
    , p.PRODUCTNAME
    , p.UNITPRICE
    , (SELECT AVG(UNITPRICE)
        FROM dbo.PRODUCTS q
        WHERE q.CATEGORYID = p.CATEGORYID) AS AVG_PRICE
FROM dbo.PRODUCTS p
WHERE p.UNITPRICE > (SELECT AVG(UNITPRICE)
                     FROM dbo.PRODUCTS);

-- z joinem
SELECT p.PRODUCTID
    , p.PRODUCTNAME
    , p.UNITPRICE
    , q.AVG_PRICE
FROM dbo.PRODUCTS p
INNER JOIN (SELECT CATEGORYID, AVG(UNITPRICE) AS AVG_PRICE
            FROM dbo.PRODUCTS
            GROUP BY CATEGORYID) q
    ON p.CATEGORYID = q.CATEGORYID
WHERE p.UNITPRICE > (SELECT AVG(UNITPRICE)
                     FROM dbo.PRODUCTS);
```

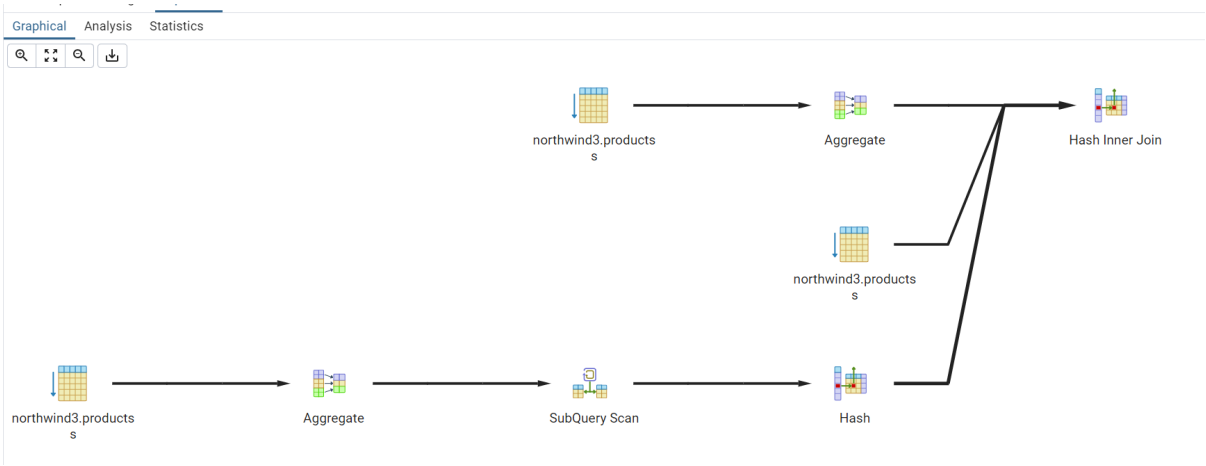
Wyniki PostgreSQL

1. z funkcja okna
    - Czas wykonania 42ms
    - 

2. z podzapytaniem
    - Czas wykonania 61ms

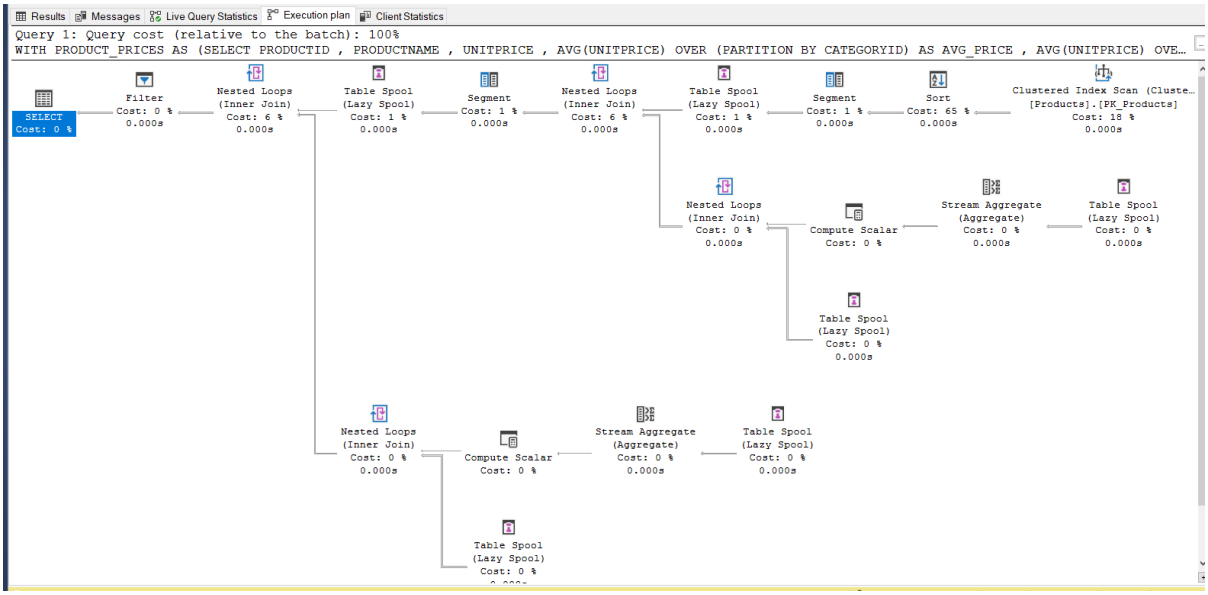


3. z joinem
   - Czas wykonania 95ms

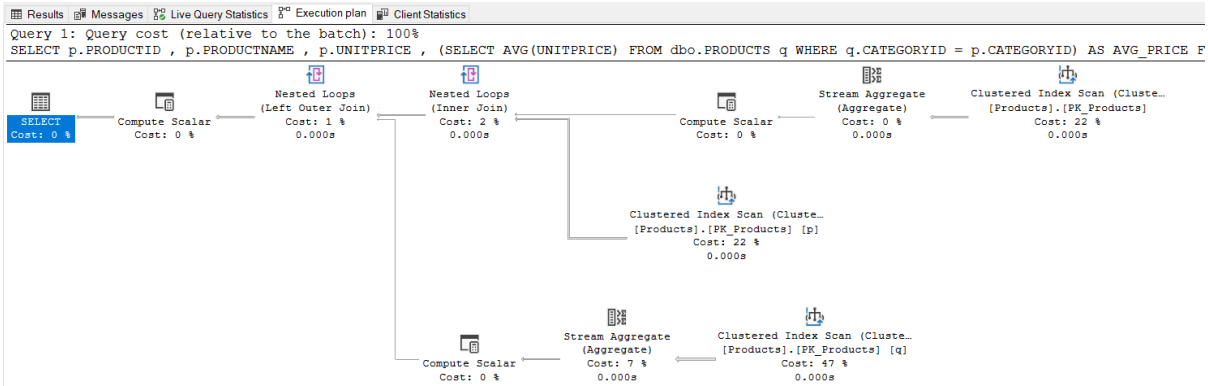


Wyniki MsSQL

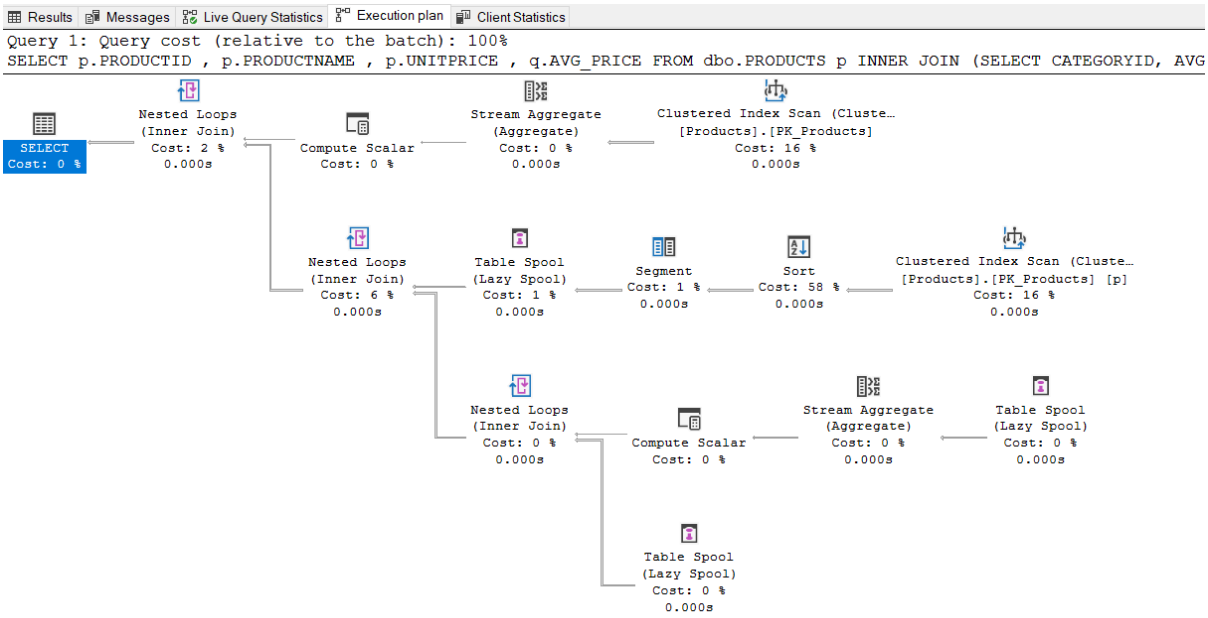
1. z funkcja okna
   - Czas wykonania 90ms



2. z podzapytaniem
   - Czas wykonania 75ms



3. z joinem
   - Czas wykonania 64ms



Wyniki SQLite

1. z funkcja okna
   - Czas wykonania 6ms

```sql
WITH PRODUCT_PRICES AS
    (SELECT PRODUCTID
        , PRODUCTNAME
        , UNITPRICE
        , AVG(UNITPRICE) OVER (PARTITION BY CATEGORYID) AS AVG_PRICE
        , AVG(UNITPRICE) OVER () AS OVERALL_AVG
    FROM PRODUCTS)
SELECT * FROM PRODUCT_PRICES WHERE UNITPRICE > OVERALL_AVG;
```

	PRODUCTID	PRODUCTNAME	UNITPRICE	AVG_PRICE	OVERAI
1	38	Côte de Blaye	263.5	37.9791666666667	28.86636
2	43	Ipoh Coffee	46	37.9791666666667	28.86636
3	8	Northwoods Cranberry Sauce	40	23.0625	28.86636
4	63	Vegie-spread	43.9	23.0625	28.86636
5	20	Sir Rodney's Marmalade	81	25.16	28.86636
6	26	Gumbär Gummibärchen	31.23	25.16	28.86636

```
Wykonano bez błędów.
Wynik: Zwrócono 25 wierszy w czasie 6ms
W wierszu 1:
WITH PRODUCT_PRICES AS
            (SELECT PRODUCTID
                        , PRODUCTNAME
                        , UNITPRICE
                        , AVG(UNITPRICE) OVER (PARTITION BY CATEGORYID) AS AVG_PRICE
                        , AVG(UNITPRICE) OVER () AS OVERALL_AVG
            FROM PRODUCTS)
SELECT * FROM PRODUCT_PRICES WHERE UNITPRICE > OVERALL_AVG;
```

2. z podzapytaniem
    - Czas wykonania 6ms

```sql
SELECT p.PRODUCTID
    , p.PRODUCTNAME
    , p.UNITPRICE
    , (SELECT AVG(UNITPRICE)
        FROM PRODUCTS q
        WHERE q.CATEGORYID = p.CATEGORYID) AS AVG_PRICE
FROM PRODUCTS p
WHERE p.UNITPRICE > (SELECT AVG(UNITPRICE)
                     FROM PRODUCTS);
```

	ProductID	ProductName	UnitPrice	AVG_PRICE
1	7	Uncle Bob's Organic Dried Pears	30	32.37
2	8	Northwoods Cranberry Sauce	40	23.0625
3	9	Mishi Kobe Niku	97	54.0066666666667
4	10	Ikura	31	20.6825
5	12	Queso Manchego La Pastora	38	28.73
6	17	Alice Mutton	39	54.0066666666667
7	18	Carnarvon Tigers	62.5	20.6825

```
Wykonano bez błędów.
Wynik: Zwrócono 25 wierszy w czasie 6ms
W wierszu 1:
SELECT p.PRODUCTID
            , p.PRODUCTNAME
            , p.UNITPRICE
            , (SELECT AVG(UNITPRICE)
                       FROM PRODUCTS q
                       WHERE q.CATEGORYID = p.CATEGORYID) AS AVG_PRICE
FROM PRODUCTS p
WHERE p.UNITPRICE > (SELECT AVG(UNITPRICE)
                                                                FROM PRODUCTS);
```

3. z joinem
    - Czas wykonania 3ms

```sql
SELECT p.PRODUCTID
    , p.PRODUCTNAME
    , p.UNITPRICE
    , q.AVG_PRICE
FROM PRODUCTS p
INNER JOIN (SELECT CATEGORYID, AVG(UNITPRICE) AS AVG_PRICE
            FROM PRODUCTS
            GROUP BY CATEGORYID) q
    ON p.CATEGORYID = q.CATEGORYID
WHERE p.UNITPRICE > (SELECT AVG(UNITPRICE)
                     FROM PRODUCTS);
```

	ProductID	ProductName	UnitPrice	AVG_PRICE
1	38	Côte de Blaye	263.5	37.9791666666667
2	43	Ipoh Coffee	46	37.9791666666667
3	8	Northwoods Cranberry Sauce	40	23.0625
4	63	Vegie-spread	43.9	23.0625
5	20	Sir Rodney's Marmalade	81	25.16
6	26	Gumbär Gummibärchen	31.23	25.16
7	27	Schoggi Schokolade	43.9	25.16

```
Wykonano bez błędów.
Wynik: Zwrócono 25 wierszy w czasie 3ms
W wierszu 1:
SELECT p.PRODUCTID
            , p.PRODUCTNAME
            , p.UNITPRICE
            , q.AVG_PRICE
FROM PRODUCTS p
INNER JOIN (SELECT CATEGORYID, AVG(UNITPRICE) AS AVG_PRICE
                                  FROM PRODUCTS
                                  GROUP BY CATEGORYID) q
           ON p.CATEGORYID = q.CATEGORYID
WHERE p.UNITPRICE > (SELECT AVG(UNITPRICE)
                                                    FROM PRODUCTS);
```

Wnioski

Z uwagi na bardzo niskie czasy wykonania, w pewnych przypadkach ciężko było wysnuć sensowne wnioski

- na różnych silnikach baz danych różne typy zapytań są najszybsze
- w 2/3 przypadkach zapytanie z użyciem joinów jest najbardziej efektywne
- zapytanie z użyciem funkcji okna w każdym wypadku jest najbardziej czytelne i wygodne do napisania
- funkcje okna zawsze wykonują się na danych po klauzurze *WHERE*

---

zadanie	pkt
1	1
2	1
3	1
4	1

razem	4