SQL Analiza

infoShareAcademy.com

SQL Analiza

Agenda

1. **Funkcje okna**
   - **ROW_NUMBER(), LAG(), LEAD(), FIRST_VALUE(), LAST_VALUE()**
   - **Funkcje okna - agregaty**
2. Wskaźniki analityczne
   - YoY, MoM
   - Funkcje statystyczne
3. Podzapytania

SQL Analiza

Agenda

1. Funkcje okna
   - ROW_NUMBER(), LAG(), LEAD(), FIRST_VALUE(), LAST_VALUE()
   - Funkcje okna - agregaty
2. **Wskaźniki analityczne**
   - **YoY, MoM**
   - **Funkcje statystyczne**
3. Podzapytania

SQL Analiza

Agenda

1. Funkcje okna
   - ROW_NUMBER(), LAG(), LEAD(), FIRST_VALUE(), LAST_VALUE()
   - Funkcje okna - agregaty
2. Wskaźniki analityczne
   - YoY, MoM
   - Funkcje statystyczne
3. **Podzapytania**

SQL Analiza

Czym są funkcje okna?

infoShare ACADEMY

Window Functions

Aggregate

- AVG()
- MAX()
- MIN()
- SUM()
- COUNT()

Ranking

- ROW_NUMBER
- RANK()
- DENSE_RANK()
- PERCENT_RANK()
- NTILE()

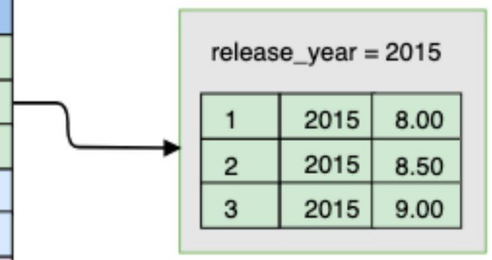
Value

- ROW_NUMBER
- RANK()
- DENSE_RANK()
- PERCENT_RANK()
- NTILE()

SQL Analiza

Czym jest okno (window) w SQL?

id	release_year	rating
1	2015	8.00
2	2015	8.50
3	2015	9.00
4	2016	8.20
5	2016	8.40
6	2017	7.00

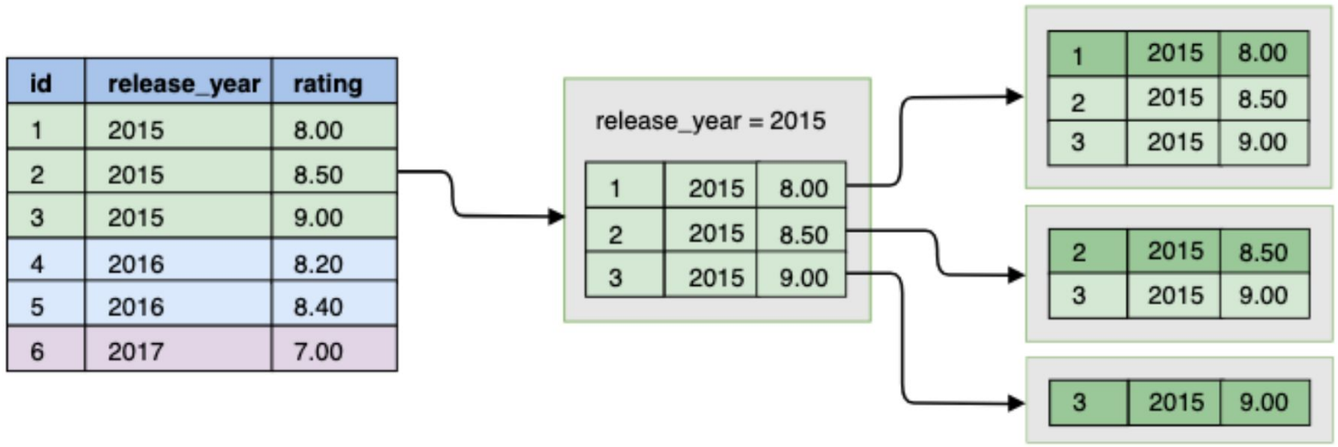


release_year = 2015

1	2015	8.00
2	2015	8.50
3	2015	9.00

SQL Analiza

Czym jest ramka (window frame) w SQL?



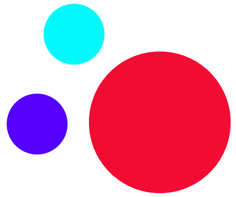
id	release_year	rating
1	2015	8.00
2	2015	8.50
3	2015	9.00
4	2016	8.20
5	2016	8.40
6	2017	7.00

1	2015	8.00
2	2015	8.50
3	2015	9.00

1	2015	8.00
2	2015	8.50
3	2015	9.00

2	2015	8.50
3	2015	9.00

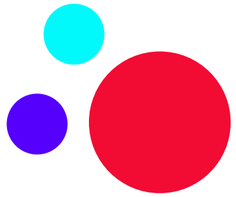
3	2015	9.00



SQL Analiza

Składnia funkcji okna

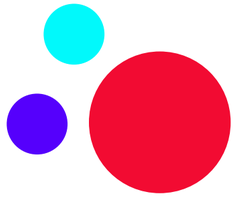
```
FunkcjaOkna(argumenty) OVER (
            [ PARTITION BY klauzula_partycjonowania]
            [ ORDER BY klauzula_sortowania [ ASC | DESC ]
            [ NULLS { FIRST | LAST } ]
            [ { ROWS | RANGE | GROUPS } BETWEEN początek_ramki AND koniec_ramki ]
      )
```



SQL Analiza

Klauzula partycjonowania PARTITION BY

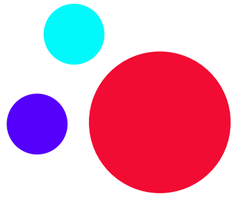
```
SELECT
    category,
    sales,
    SUM(sales) OVER (PARTITION BY category) AS total_category_sales
FROM
    sales_table;
```



SQL Analiza

Klauzula partycjonowania ORDER BY

```
SELECT
    value,
    SUM(value) OVER (ORDER BY date_column) AS cumulative_sum
FROM
    your_table;
```



SQL Analiza

Klauzula ramki

```
SELECT
    column1,
    column2,
    SUM(column3) OVER (PARTITION BY partition_column ORDER BY order_column
    ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS sum_with_frame
FROM
    your_table;
```

SQL Analiza

Utworzenie tabeli tools_sales

```
-- Tworzenie tabeli
CREATE TABLE tools_sales (
    sale_id SERIAL PRIMARY KEY,
    tool_id INT,
    tool_group VARCHAR(50),
    sale_date DATE,
    quantity_sold INT,
    sale_amount DECIMAL(10, 2)
);
-- Wstawianie danych
INSERT INTO tools_sales (tool_id, tool_group,
sale_date, quantity_sold, sale_amount) VALUES
    (1, 'Power Tools', '2023-01-01', 10, 500.00),
    (2, 'Hand Tools', '2023-01-01', 5, 300.00),
    (1, 'Power Tools', '2023-01-02', 8, 400.00),
    (3, 'Safety Gear', '2023-01-02', 12, 600.00),
    (2, 'Hand Tools', '2023-01-03', 7, 350.00),
    (1, 'Power Tools', '2023-01-03', 15, 750.00);
```

SQL Analiza

ROW_NUMBER()

```
SELECT
    sale_id,
    tool_id,
    tool_group,
    sale_date,
    quantity_sold,
    sale_amount,
    ROW_NUMBER() OVER (PARTITION BY tool_group ORDER BY
sale_date) AS row_num
FROM
    tools_sales
ORDER BY
    tool_group, sale_date;
```

sale_id	tool_id	tool_group	sale_date	quantity_sold	sale_amount	row_num
2	2	Hand Tools	2023-01-01	5	300	1
5	2	Hand Tools	2023-01-03	7	350	2
1	1	Power Tools	2023-01-01	10	500	1
3	1	Power Tools	2023-01-02	8	400	2
6	1	Power Tools	2023-01-03	15	750	3
4	3	Safety Gear	2023-01-02	12	600	1

Zadanie 3.1

ROW_NUMBER() (instrukcja)

Dla tabeli tools_sales napisz kwerendę przy użyciu funkcji okienkowej ROW_NUMBER(), która numeruje wiersze w ramach grup narzędzi (tool_group), malejąco według kwoty sprzedaży (sale_amount).

SQL Analiza

Funkcje rangi

- RANK()
- DENSE_RANK()
- PERCENT_RANK()

SQL Analiza

RANK()

```
SELECT
  sale_id,
  tool_id,
  tool_group,
  sale_date,
  quantity_sold,
  sale_amount,
  RANK() OVER (PARTITION BY tool_group ORDER BY
quantity_sold DESC) AS rank
FROM
  tools_sales;
```

sale_id	tool_id	tool_group	sale_date	quantity_sold	sale_amount	rank
5	2	Hand Tools	2023-01-03	7	350	1
2	2	Hand Tools	2023-01-01	5	300	2
6	1	Power Tools	2023-01-03	15	750	1
1	1	Power Tools	2023-01-01	10	500	2
3	1	Power Tools	2023-01-02	8	400	3
4	3	Safety Gear	2023-01-02	12	600	1

SQL Analiza

DENSE_RANK()

```
SELECT
    sale_id,
    tool_id,
    tool_group,
    sale_date,
    quantity_sold,
    sale_amount,
    DENSE_RANK() OVER (PARTITION BY tool_group ORDER BY
quantity_sold DESC) AS dense_rank
FROM
    tools_sales;
```

sale_id	tool_id	tool_group	sale_date	quantity_sold	sale_amount	dense_rank
5	2	Hand Tools	2023-01-03	7	350	1
2	2	Hand Tools	2023-01-01	5	300	2
6	1	Power Tools	2023-01-03	15	750	1
1	1	Power Tools	2023-01-01	10	500	2
3	1	Power Tools	2023-01-02	8	400	3
4	3	Safety Gear	2023-01-02	12	600	1

SQL Analiza

PERCENT_RANK()

```
SELECT
    sale_id,
    tool_id,
    tool_group,
    sale_date,
    quantity_sold,
    sale_amount,
   PERCENT_RANK() OVER (PARTITION BY tool_group ORDER BY
quantity_sold DESC) AS percent_rank
FROM
    tools_sales;
```

sale_id	tool_id	tool_group	sale_date	quantity_sold	sale_amount	percent_rank
5	2	Hand Tools	2023-01-03	7	350	0
2	2	Hand Tools	2023-01-01	5	300	1
6	1	Power Tools	2023-01-03	15	750	0
1	1	Power Tools	2023-01-01	10	500	0.5
3	1	Power Tools	2023-01-02	8	400	1
4	3	Safety Gear	2023-01-02	12	600	0

Zadanie 3.2

RANK() (instrukcja)

Dla tabeli tools_sales przypisz rangi dla rekordów w tabeli w oparciu o rosnącą datę sprzedaży (sale_date). Potraktuj rekordy jako jedną grupę.

SQL Analiza

CUME_DIST()

```
SELECT
    tool_id,
    tool_group,
    sale_amount,
   CUME_DIST() OVER (ORDER BY sale_amount) AS
cumulative_distribution
FROM
    tools_sales;
```

123 tool_id	ABC tool_group	123 sale_amount	123 cumulative_distribution
2	Hand Tools	300	0.1666666667
2	Hand Tools	350	0.3333333333
1	Power Tools	400	0.5
1	Power Tools	500	0.6666666667
3	Safety Gear	600	0.8333333333
1	Power Tools	750	1

SQL Analiza

NTILE()

```
SELECT
    tool_id,
    tool_group,
    sale_amount,
    NTILE(4) OVER (ORDER BY sale_amount) AS quartile
FROM
    tools_sales;
```

123 tool_id	ABC tool_group	123 sale_amount	123 quartile
2	Hand Tools	300	1
2	Hand Tools	350	1
1	Power Tools	400	2
1	Power Tools	500	2
3	Safety Gear	600	3
1	Power Tools	750	4

Zadanie 3.3

NTILE() (instrukcja)

Dla tabeli tools_sales podziel rekordy na trzy grupy (kwantyle) w obrębie każdej grupy narzędziowej (tool_group) uszeregowanej według rosnącej kwoty sprzedaży.

SQL Analiza

LAG(), LEAD()

LEAD(expression, offset, default)

LAG(expression, offset, default)

wyrażenie

liczba wierszy wstecz/do przodu

wartość domyślna, gdy brak poprzedniego/ następnego wiersza

SQL Analiza

LAG()

```
SELECT
    sale_id,
    tool_id,
    sale_date,
    quantity_sold,
    LAG(quantity_sold, 1, 0) OVER (ORDER BY sale_date) AS
prev_quantity
FROM
    tools_sales
ORDER BY
    sale_date;
```

sale_id	tool_id	sale_date	quantity_sold	prev_quantity
1	1	2023-01-01	10	0
2	2	2023-01-01	5	10
3	1	2023-01-02	8	5
4	3	2023-01-02	12	8
5	2	2023-01-03	7	12
6	1	2023-01-03	15	7

SQL Analiza

LEAD()

```
SELECT
    sale_id,
    tool_id,
    sale_date,
    quantity_sold,
    LEAD(quantity_sold, 1, 0) OVER (ORDER BY sale_date) AS next_quantity
FROM
    tools_sales
ORDER BY
    sale_date;
```

sale_id	tool_id	sale_date	quantity_sold	prev_quantity
1	1	2023-01-01	10	5
2	2	2023-01-01	5	8
3	1	2023-01-02	8	12
4	3	2023-01-02	12	7
5	2	2023-01-03	7	15
6	1	2023-01-03	15	0

Zadanie 3.4

LAG() / LEAD() (instrukcja)

Dla tabeli tools_sales napisz zapytanie, które zawierać będzie informacje o:

- grupie narzędzi,
- dacie sprzedaży,
- ilości sprzedanych sztuk,
- ilości sprzedanych sztuk dla poprzedniej i następnej daty sprzedaży w obrębie tej samej grupy narzędzi.

SQL Analiza

FIRST_VALUE(), LAST_VALUE(), NTH_VALUE()

FIRST_VALUE(column)

LAST_VALUE(column)

NTH_VALUE(column, n), gdzie n to określona pozycja

SQL Analiza

FIRST_VALUE()

```
SELECT
    tool_group,
    sale_date,
    quantity_sold,
    FIRST_VALUE(quantity_sold) OVER (PARTITION BY tool_group
ORDER BY sale_date) AS first_quantity
FROM
    tools_sales;
```

tool_group	sale_date	quantity_sold	first_quantity
Hand Tools	2023-01-01	5	5
Hand Tools	2023-01-03	7	5
Power Tools	2023-01-01	10	10
Power Tools	2023-01-02	8	10
Power Tools	2023-01-03	15	10
Safety Gear	2023-01-02	12	12

SQL Analiza

LAST_VALUE()

```
SELECT
    tool_group,
    sale_date,
    quantity_sold,
    LAST_VALUE(quantity_sold) OVER (PARTITION BY tool_group
ORDER BY sale_date) AS first_quantity
FROM
    tools_sales;
```

tool_group	sale_date	quantity_sold	first_quantity
Hand Tools	2023-01-01	5	5
Hand Tools	2023-01-03	7	7
Power Tools	2023-01-01	10	10
Power Tools	2023-01-02	8	8
Power Tools	2023-01-03	15	15
Safety Gear	2023-01-02	12	12

infoShare ACADEMY

SQL Analiza

NTH_VALUE()

```
SELECT
    tool_group,
    sale_date,
    quantity_sold,
    NTH_VALUE(quantity_sold, 3) OVER (PARTITION BY tool_group ORDER BY sale_date) AS first_quantity
FROM
    tools_sales;
```

tool_group	sale_date	quantity_sold	first_quantity
Hand Tools	2023-01-01	5	[NULL]
Hand Tools	2023-01-03	7	[NULL]
Power Tools	2023-01-01	10	[NULL]
Power Tools	2023-01-02	8	[NULL]
Power Tools	2023-01-03	15	15
Safety Gear	2023-01-02	12	[NULL]

Zadanie 3.5

FIRST/LAST/NTH_VALUE (instrukcja)

Dla tabeli all_seasons o koszykarzach NBA znajdź pierwszy i ostatni sezon dla każdego koszykarza.

SQL Analiza

Agregaty avg(), sum(), count(), min(), max()

AVG(sale_amount) OVER (PARTITION BY tool_group ORDER BY sale_date)

SUM(quantity_sold) OVER (PARTITION BY tool_group ORDER BY sale_date)

COUNT(*) OVER (PARTITION BY tool_group ORDER BY sale_date)

MIN(sale_amount) OVER (PARTITION BY tool_group ORDER BY sale_date)

MAX(quantity_sold) OVER (PARTITION BY tool_group ORDER BY sale_date)

SQL Analiza

AVG()

```
SELECT
    tool_group,
    sale_date,
    quantity_sold,
    AVG(quantity_sold) OVER (PARTITION BY tool_group ORDER BY sale_date)
AS avg_quantity
FROM
    tools_sales;
```

tool_group	sale_date	quantity_sold	avg_quantity
Hand Tools	2023-01-01	5	5
Hand Tools	2023-01-03	7	6
Power Tools	2023-01-01	10	10
Power Tools	2023-01-02	8	9
Power Tools	2023-01-03	15	11
Safety Gear	2023-01-02	12	12

SQL Analiza

SUM()

```
SELECT
    tool_group,
    sale_date,
    quantity_sold,
    SUM(quantity_sold) OVER (PARTITION BY tool_group ORDER BY sale_date)
AS total_quantity
FROM
    tools_sales;
```

tool_group	sale_date	quantity_sold	total_quantity
Hand Tools	2023-01-01	5	5
Hand Tools	2023-01-03	7	12
Power Tools	2023-01-01	10	10
Power Tools	2023-01-02	8	18
Power Tools	2023-01-03	15	33
Safety Gear	2023-01-02	12	12

SQL Analiza

COUNT()

```
SELECT
    tool_group,
    sale_date,
    COUNT(*) OVER (PARTITION BY tool_group) AS days_with_sales
FROM
    tools_sales;
```

tool_group	sale_date	days_with_sales
Hand Tools	2023-01-01	2
Hand Tools	2023-01-03	2
Power Tools	2023-01-01	3
Power Tools	2023-01-02	3
Power Tools	2023-01-03	3
Safety Gear	2023-01-02	1

SQL Analiza

MIN() MAX()

```
SELECT
    tool_group,
    sale_date,
    quantity_sold,
    MIN(quantity_sold) OVER (PARTITION BY tool_group) AS min_quantity_sold,
    MAX(quantity_sold) OVER (PARTITION BY tool_group) AS max_quantity_sold
FROM
    tools_sales;
```

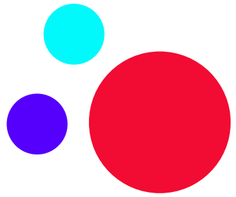
tool_group	sale_date	quantity_sold	min_quantity_sold	max_quantity_sold
Hand Tools	2023-01-01	5	5	7
Hand Tools	2023-01-03	7	5	7
Power Tools	2023-01-01	10	8	15
Power Tools	2023-01-02	8	8	15
Power Tools	2023-01-03	15	8	15
Safety Gear	2023-01-02	12	12	12

Zadanie 3.6

Agregaty (instrukcja)

Dla zbioru danych tools_sales znajdź:

- sumę i średnią wartość sprzedaży dla każdej grupy narzędzi, posortowanych według daty sprzedaży,
- maksymalną wartość sprzedaży dla każdej grupy narzędzi, posortowanych według daty sprzedaży,
- najwcześniejszą datę sprzedaży dla każdej grupy narzędzi.



SQL Analiza

Funkcje okna - podsumowanie

1. ROW_NUMBER(), RANK(), DENSE_RANK()

2. FIRST_VALUE(), LAST_VALUE(), NTH_VALUE()

3. LAG(), LEAD()

4. CUME_DIST()

5. NTILE()

6. AVG(), SUM(), COUNT(), MIN(), MAX()

Zadanie 3.7

Podsumowanie (instrukcja)

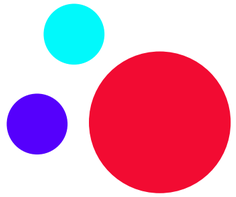
Analiza Sprzedaży Narzędzi - masz do dyspozycji tabelę tools_sales z informacjami o sprzedaży narzędzi w sklepie. Twoim zadaniem jest przygotowanie analizy danych przy użyciu funkcji okienkowych.

a. **Numeracja Zamówień:** Dodaj kolumnę order_number, w której przypiszesz unikalne numery porządkowe dla każdego zamówienia, sortując je według daty sprzedaży.

b. **Rangi Grup Narzędziowych:** Nadaj rangi dla każdej grupy narzędziowej, zaczynając od jedynki, sortując według ilości sprzedanych narzędzi malejąco.

c. **Kumulatywny Rozkład Sprzedaży:** Utwórz kolumnę cumulative_sales, która będzie zawierać kumulatywną sumę sprzedaży dla każdego narzędzia, uporządkowanego według daty sprzedaży.

d. **Analiza Dynamiki Sprzedaży:** Stwórz kolumny sales_change i percentage_change, które pokażą zmiany w ilości sprzedanych narzędzi i procentową zmianę w stosunku do poprzedniego zamówienia.



SQL Analiza

Funkcje statystyczne

- **YoY**
- **MoM**

SQL Analiza

YoY

- Year over Year - procentowa zmiana w stosunku do poprzedniego roku

$$\text{YoY \%} = \frac{\text{aktualny rok - poprzedni rok}}{\text{poprzedni rok}}$$

```
SELECT
    Year,
    Sales,
    LAG(Sales, 1) OVER (ORDER BY Year) AS SalesLastYear,
    ((Sales - LAG(Sales, 1) OVER (ORDER BY Year)) / LAG(Sales, 1)
OVER (ORDER BY Year)) * 100 AS YoYChange
FROM
    SalesData;
```

123 year	123 sales	123 saleslastyear	123 yoychange
2,019	120,000	[NULL]	[NULL]
2,020	150,000	120,000	25
2,021	180,000	150,000	20
2,022	200,000	180,000	11.1111111111

SQL Analiza

MoM

- Month over Month - procentowa zmiana w stosunku do poprzedniego miesiące

$$\text{MoM \%} = \frac{\text{aktualny miesiąc - poprzedni miesiąc}}{\text{poprzedni miesiąc}}$$

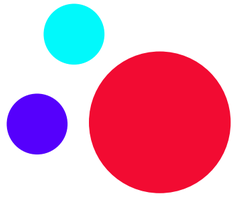
```
SELECT
    Month,
    Sales,
    LAG(Sales, 1) OVER (ORDER BY Month) AS SalesLastMonth,
    ((Sales - LAG(Sales, 1) OVER (ORDER BY Month)) / LAG(Sales, 1)
OVER (ORDER BY Month)) * 100 AS MoMChange
FROM
    SalesData;
```

month	sales	saleslastmonth	momchange
2023-01-01	120,000	[NULL]	[NULL]
2023-02-01	130,000	120,000	8.3333333333
2023-03-01	150,000	130,000	15.3846153846
2023-04-01	140,000	150,000	-6.6666666667
2023-05-01	160,000	140,000	14.2857142857

Zadanie 3.8

YoY / MoM (instrukcja)

Dla tabeli all_seasons o koszykarzach NBA znajdź średnią liczbę zdobytych punktów w sezonie dla każdego koszykarza oraz oblicz procentową zmianę średnich punktów rok do roku (YoY).



SQL Analiza

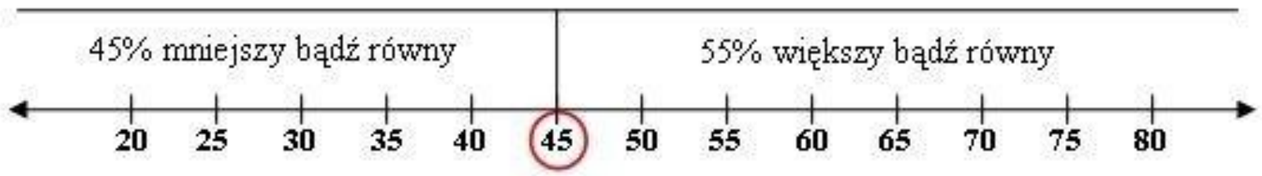
Funkcje statystyczne

- Mode()
- Percenitle_disc()
- Percentile_cont()

SQL Analiza

Kwantyle / percentyle

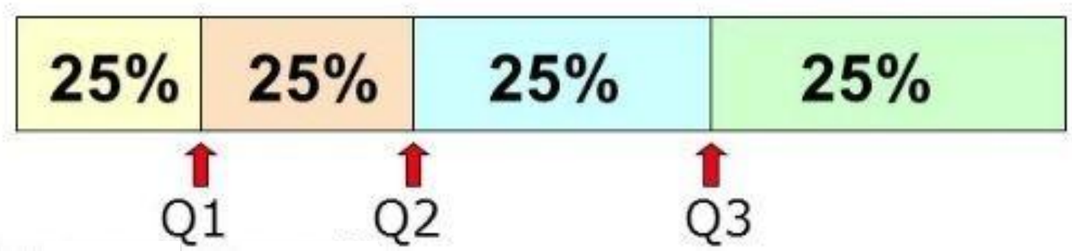
- **Kwantyl** rzędu p to wartość liczbowa ze zbioru, od której p-ta część zbioru jest mniejsza.
- **Percentyle** - kwantyle rzędów x/100 - podział rosnąco posortowanego zbioru na 100 części.



SQL Analiza

Kwantyle

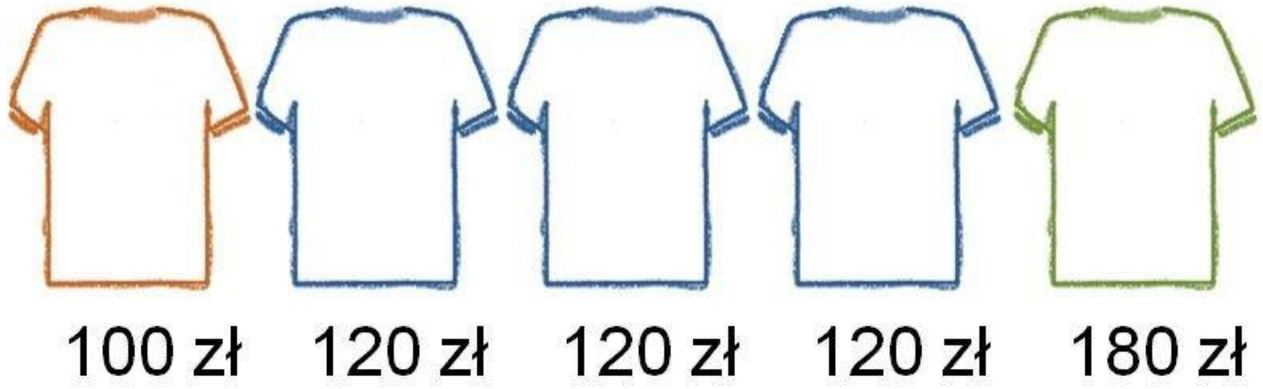
- Kwantyle - kwantyle rzędów x/4 - podział rosnąco posortowanego zbioru na 4 części.
- Kwartyl Q2 to mediana, wartość środkowa.



SQL Analiza

Moda

- Moda, dominanta - wartość najczęstsza.



SQL Analiza

MODE()

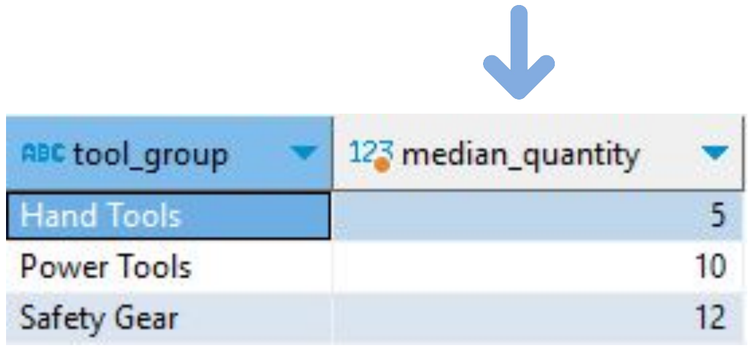
SELECT tool_group, **MODE() WITHIN GROUP** (ORDER BY quantity_sold) AS mode_quantity
FROM tools_sales
GROUP BY tool_group;

tool_group	mode_quantity
Hand Tools	5
Power Tools	8
Safety Gear	12

SQL Analiza

PERCENTILE_DISC()

SELECT tool_group, **PERCENTILE_DISC(0.5) WITHIN GROUP** (ORDER BY quantity_sold) AS median_quantity
FROM tools_sales
GROUP BY tool_group;



tool_group	median_quantity
Hand Tools	5
Power Tools	10
Safety Gear	12

SQL Analiza

PERCENTILE_CONT()

SELECT tool_group, **PERCENTILE_CONT(0.5) WITHIN GROUP** (ORDER BY quantity_sold) AS median_quantity
FROM tools_sales
GROUP BY tool_group;

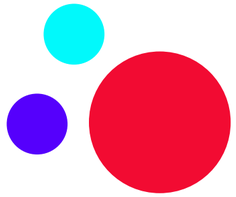
tool_group	median_quantity
Hand Tools	6
Power Tools	10
Safety Gear	12

Zadanie 3.9

Funkcje statystyczne (instrukcja)

Dla zbioru tools_sales oblicz ile wynosi:

- najczęściej występująca data sprzedaży dla każdej grupy narzędzi,
- mediana kwoty sprzedaży dla każdej grupy narzędzi (użyj zarówno funkcji percentile_disc jak i percentile_cont i porównaj wyniki).



SQL Analiza

Funkcje statystyczne - c.d.

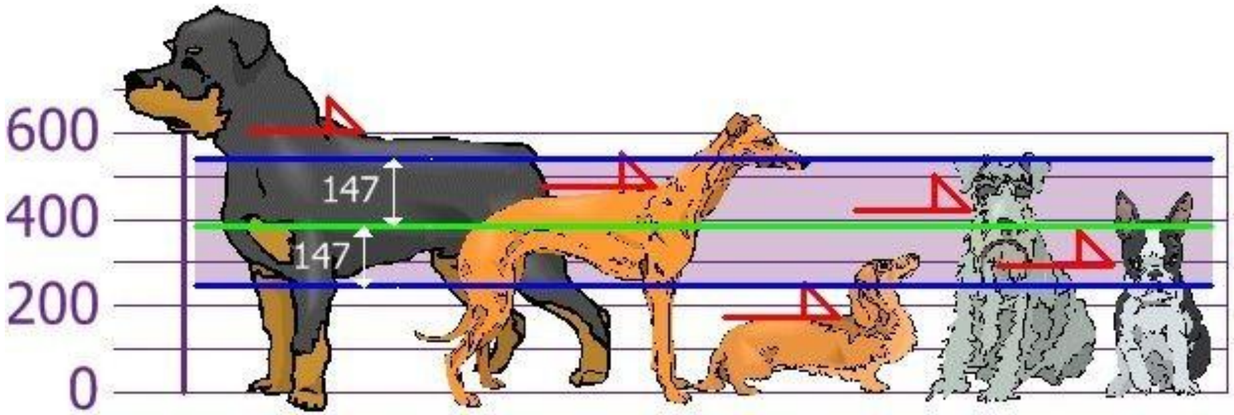
- Wariancja
- Odchylenie standardowe
- Korelacja

SQL Analiza

Wariancja i odchylenie standardowe

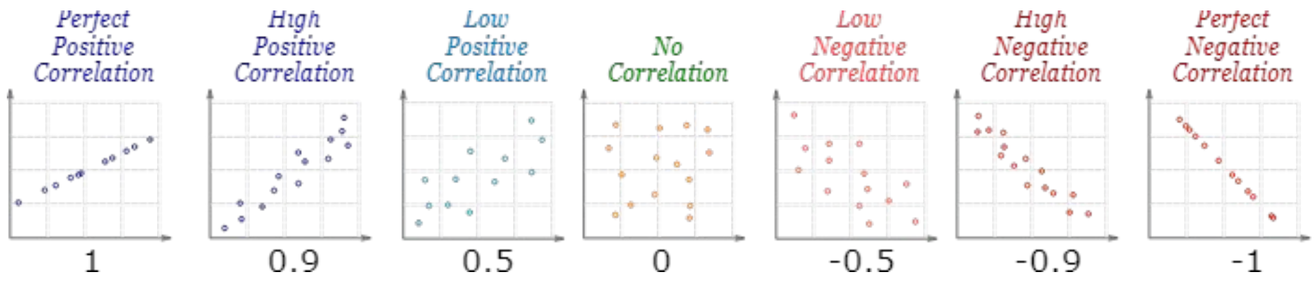
Wariancja (VAR) i odchylenie standardowe (STDDEV) to typowe miary rozrzutu danych. Można je policzyć dla całej populacji jak i dla próbki.

STDDEV to po prostu pierwiastek kwadratowy z wariancji. Odchylenie std jest wyrażone w oryginalnych jednostkach, natomiast VAR w jednostkach do kwadratu.



SQL Analiza

Korelacja



infoShareAcademy.com

SQL Analiza

Kowariancja

- Liczba określająca odchylenie elementów od sytuacji idealnej, w której występuje zależność liniowa.
- W przypadku braku korelacji liniowej, kowariancja przyjmuje wartość 0.
- Zależność pomiędzy kowariancją a korelacją:

$$\mathbf{cov(X,Y) = corr(X,Y)\delta_x\delta_y}$$

gdzie:

corr(X,Y) - współczynnik korelacji liniowej pomiędzy zmiennymi X i Y

$\mathbf{\delta_x}$ - odchylenie standardowe zmiennej X

$\mathbf{\delta_y}$ - odchylenie standardowe zmiennej Y

SQL Analiza

Funkcje statystyczne agregujące

- corr(dependent, independent)
- covar_pop() / covar_samp()
- stddev_pop() / stddev_samp() / stddev()
- var_pop() / var_samp() / variance()
- * regr_intercept()
- * regr_slope()

SQL Analiza

STDDEV()

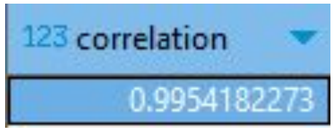
```
SELECT
    tool_group,
    STDDEV(quantity_sold) AS std_dev_quantity
FROM
    tools_sales
GROUP BY
    tool_group;
```

tool_group	std_dev_quantity
Hand Tools	1.4142135624
Safety Gear	[NULL]
Power Tools	3.6055512755

SQL Analiza

CORR()

```
SELECT
    CORR(quantity_sold, sale_amount) AS correlation
FROM
    tools_sales;
```



Zadanie 3.10

Funkcje statystyczne (instrukcja)

Dla tabeli tools_sales oblicz wariancję ilości sprzedanych sztuk narzędzi.

SQL Analiza

Funkcje statystyczne - c.d.

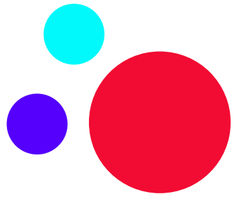
- YoY, MoM
- Mode()
- Percenitle_disc()
- Percentile_cont()
- Wariancja
- Odchylenie standardowe
- Korelacja

Zadanie 3.11

Podsumowanie funkcji statystycznych (instrukcja)

Przy użyciu tabeli all_seasons, która zawiera dane o koszykarzach NBA, wykonaj następujące analizy:

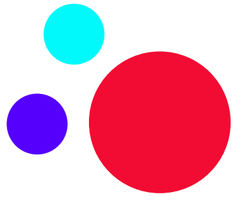
- Oblicz średnią liczbę punktów (pts) dla każdego sezonu.
- Oblicz medianę wzrostu (player_height) dla wszystkich graczy.
- Oblicz korelację między liczbą punktów (pts) a liczbą asyst (ast).
- Znajdź najczęściej występującą rundę draftu (draft_round).



SQL Analiza

Podzapytania

- Podzapytanie pozwala na wykorzystanie w ramach języka DQL (SELECT) kolejnego zapytania SELECT.
- Na ogół podzapytanie można wykonać jako poznany już JOIN.
- W większości przypadków wydajność JOIN jak i podzapytania są zbliżone.
- Czasem jednak jedna forma pozwala skrócić kod lub jest szybsza do napisania.



SQL Analiza

Podzapytania - występowanie

- SELECT
- FROM
- WHERE
- ORDER BY
- LIMIT
- HAVING

SQL Analiza

Podzapytania w klauzuli SELECT

1. Pobieranie jednej wartości

SELECT column1, **(SELECT MAX(column2) FROM table2)** AS max_value
FROM table1;

2. Pobieranie więcej niż jednej kolumny

SELECT column1, column2 ,
(SELECT column3, column4 FROM table2 WHERE condition) AS subquery_result
FROM table1;

SQL Analiza

Podzapytania w klauzuli FROM

1. Podzapytanie jako źródło danych

SELECT column1, column2

FROM **(SELECT * FROM table1 WHERE condition)** AS subquery;

2. Łączenie wielu źródeł danych

SELECT column1, column2

FROM table1

JOIN **(SELECT * FROM table2 WHERE condition)** AS subquery

ON table1.columnX = subquery.columnY;

3. Podzapytania zwracające więcej niż jedną kolumnę

SELECT column1, column2

FROM **(SELECT columnX, columnY FROM table2 WHERE condition)** AS subquery;

SQL Analiza

Podzapytania w klauzuli WHERE

1. Porównanie z pojedynczym wynikiem

```
SELECT column1, column2
FROM table1
WHERE columnX = (SELECT columnY FROM table2 WHERE condition);
```

2. Porównanie z wynikami podzapytania

```
SELECT column1, column2
FROM table1
WHERE columnX IN (SELECT columnY FROM table2 WHERE condition);
```

3. Użycie podzapytania w warunku logicznym

```
SELECT column1, column2
FROM table1
WHERE EXISTS (SELECT 1 FROM table2 WHERE condition);
```

Zadanie 3.12

Instrukcja

Dla tabeli all_seasons:

- Wybierz wszystkich koszykarzy z sezonu 2020/2021, którzy zdobyli więcej punktów niż średnia liczba punktów zdobytych przez wszystkich koszykarzy w sezonie 2019/2020.

SQL Analiza

Podzapytania w klauzuli ORDER BY

1. Sortowanie wyników zapytania za pomocą podzapytania

SELECT column1, column2
FROM table
ORDER BY **(SELECT some_column FROM another_table WHERE condition) DESC;**

2. Sortowanie po wynikach innego zapytania

SELECT column1, column2

FROM table1

ORDER BY **(SELECT AVG(some_column) FROM table2) DESC;**

3. Sortowanie warunkowe z podzapytaniem

SELECT column1, column2

FROM table

ORDER BY **CASE WHEN (SELECT COUNT(*) FROM another_table WHERE condition) > 0 THEN column1 ELSE column2 END** DESC;

SQL Analiza

Podzapytania w klauzuli LIMIT

1. Dynamiczne ustawianie LIMIT z podzapytaniem

```
SELECT column1, column2
FROM table
LIMIT (SELECT COUNT(*) FROM another_table WHERE condition);
```

SQL Analiza

Podzapytania w klauzuli HAVING

1. Grupowanie po kolumnie i filtrowanie wyników z użyciem HAVING

```
SELECT category, AVG(price) AS average_price
FROM products
GROUP BY category
HAVING AVG(price) > (SELECT AVG(price) FROM products);
```

2. Podzapytanie w HAVING z Warunkiem IN

```
SELECT category, COUNT(*) AS product_count
FROM products
GROUP BY category
HAVING COUNT(*) IN (SELECT MAX(product_count) FROM
products_by_category);
```

Zadanie 3.13

Instrukcja

Dla tabeli all_seasons o koszykarzach NBA wybierz drużyny, które mają średnią liczbę punktów większą niż globalna średnia.

SQL Analiza

Podzapytania podsumowanie

1. **Unikaj nadmiernego użycia podzapytań.**
2. Staraj się, aby podzapytania były czytelne.
3. Zwracaj uwagę na wydajność.
4. Unikaj zbyt dużych zagnieżdżeń.
5. Testuj i analizuj wyniki.
6. Zwracaj uwagę na indeksy.

SQL Analiza

Podzapytania podsumowanie

1. Unikaj nadmiernego użycia podzapytań.
2. **Staraj się, aby podzapytania były czytelne.**
3. Zwracaj uwagę na wydajność.
4. Unikaj zbyt dużych zagnieżdżeń.
5. Testuj i analizuj wyniki.
6. Zwracaj uwagę na indeksy.

SQL Analiza

Podzapytania podsumowanie

1. Unikaj nadmiernego użycia podzapytań.
2. Staraj się, aby podzapytania były czytelne.
3. **Zwracaj uwagę na wydajność.**
4. Unikaj zbyt dużych zagnieżdżeń.
5. Testuj i analizuj wyniki.
6. Zwracaj uwagę na indeksy.

SQL Analiza

Podzapytania podsumowanie

1. Unikaj nadmiernego użycia podzapytań.
2. Staraj się, aby podzapytania były czytelne.
3. Zwracaj uwagę na wydajność.
4. **Unikaj zbyt dużych zagnieżdżeń.**
5. Testuj i analizuj wyniki.
6. Zwracaj uwagę na indeksy.

SQL Analiza

Podzapytania podsumowanie

1. Unikaj nadmiernego użycia podzapytań.
2. Staraj się, aby podzapytania były czytelne.
3. Zwracaj uwagę na wydajność.
4. Unikaj zbyt dużych zagnieżdżeń.
5. **Testuj i analizuj wyniki.**
6. Zwracaj uwagę na indeksy.

SQL Analiza

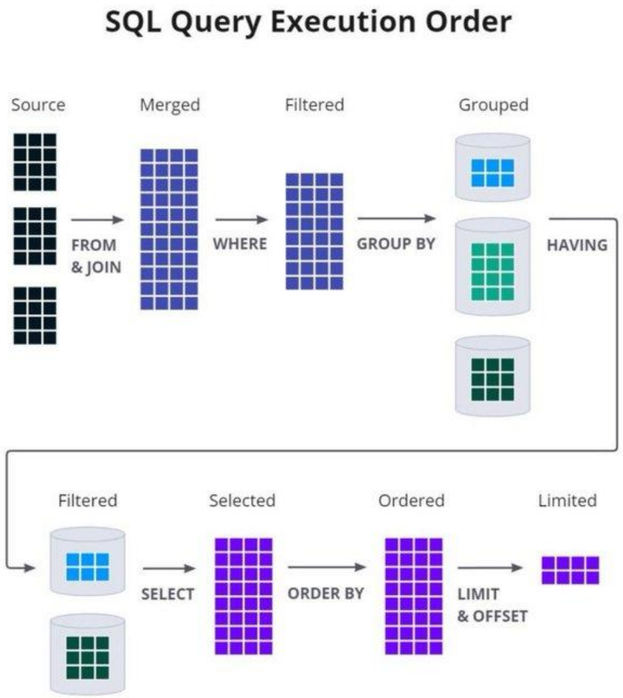
Podzapytania podsumowanie

1. Unikaj nadmiernego użycia podzapytań.
2. Staraj się, aby podzapytania były czytelne.
3. Zwracaj uwagę na wydajność.
4. Unikaj zbyt dużych zagnieżdżeń.
5. Testuj i analizuj wyniki.
6. **Zwracaj uwagę na indeksy.**

SQL Analiza

Podzapytania podsumowanie

- SELECT
- FROM
- WHERE
- ORDER BY
- LIMIT
- HAVING



Zadanie 3.14

Podsumowujące (instrukcja)

Dla tabeli all_seasons o koszykarzach NBA oblicz:

- Jaka drużyna ma średnią liczbę punktów mniejszą niż drużyny z największą liczbą zwycięstw.
- Ile wynosi różnica pomiędzy średnią liczbą punktów a średnią liczbą zbiórek dla każdego gracza.

SQL Analiza

Podsumowanie

infoShareAcademy.com