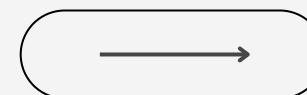


DATA
13/11/2024



BAZA DANYCH

dla internetowego sklepu odzieżowego



Karyna Cherniii
Vladyslav Fedii

CELE I ZAŁOŻENIA BAZY DANYCH.

01 ZARZĄDZANIE KATALOGIEM PRODUKTÓW

02 PRZETWARZANIE ZAMÓWIEŃ

03 ŚLEDZENIE STATUSU ZAMÓWIENIA

04 ZARZĄDZANIE PŁATNOŚCIAMI

05 ZARZĄDZANIE KLIENTAMI I ADRESAMI

06 ANALITYKA I RAPORTOWANIE

Baza danych jest przeznaczona do kompleksowego zarządzania sklepem internetowym, zapewniając płynną interakcję między klientami, administratorami i pracownikami. Pomaga zautomatyzować procesy biznesowe, zminimalizować błędy i poprawić obsługę klienta.

PROBLEMY BIZNESOWE

01

ZARZĄDZANIE DUŻĄ ILOŚCIĄ DANYCH O PRODUKTACH

Kontekst: Sklepy internetowe mają obszerne katalogi z produktami należącymi do różnych kategorii, o różnych rozmiarach i cenach. Ręczne zarządzanie takimi informacjami jest nieefektywne.

02

ZARZĄDZANIE ADRESAMI DOSTAWY

Kontekst: Jeden klient może mieć kilka adresów dostawy (dom, biuro itp.). Musisz przechowywać je wszystkie i wybierać właściwy dla każdego zamówienia.

04

OBSŁUGA ZWROTÓW I WYMIAN

Kontekst: Zwroty i wymiany są częścią pracy każdego sklepu internetowego. Musisz śledzić te transakcje i aktualizować status produktów.

03

ANALIZA SPRZEDAŻY I WYDAJNOŚCI

Kontekst: Aby podejmować decyzje biznesowe, ważne jest, aby zobaczyć, które produkty sprzedają się lepiej, które kategorie są poszukiwane i zrozumieć ogólne wyniki finansowe.

05

ZARZĄDZANIE KLIENTAMI I ICH PREFERENCJAMI

Kontekst: Aby poprawić obsługę klienta i personalizację, ważne jest przechowywanie informacji o klientach, ich historii zamówień i preferencjach.

AKTORZY. GŁÓWNI UŻYTKOWNICY.

Klienci

Interakcja z bazą:

- Rejestrują swoje dane: imię i nazwisko, dane kontaktowe i adresy dostawy.
- Przeglądanie produktów, dostępnych rozmiarów i kategorii.
- Składanie zamówień poprzez wybór produktów i ich rozmiarów.
- Wybór metody płatności i śledzenie statusu zamówienia.

Klienci to użytkownicy końcowi, którzy dokonują zakupów w sklepie internetowym.

Administratorzy

Interakcja z bazą danych:

- Dodawanie i aktualizowanie informacji o produktach (nazwa, cena, kategorie, rozmiary).
- Zarządzanie statusami zamówień (od „zamówione” do „dostarczone”).
- Przetwarzanie zwrotów lub wymian.
- Analizowanie danych dotyczących sprzedaży, przychodów i aktywności klientów.

Administratorzy zarządzają systemem, danymi i zamówieniami.

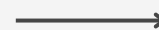
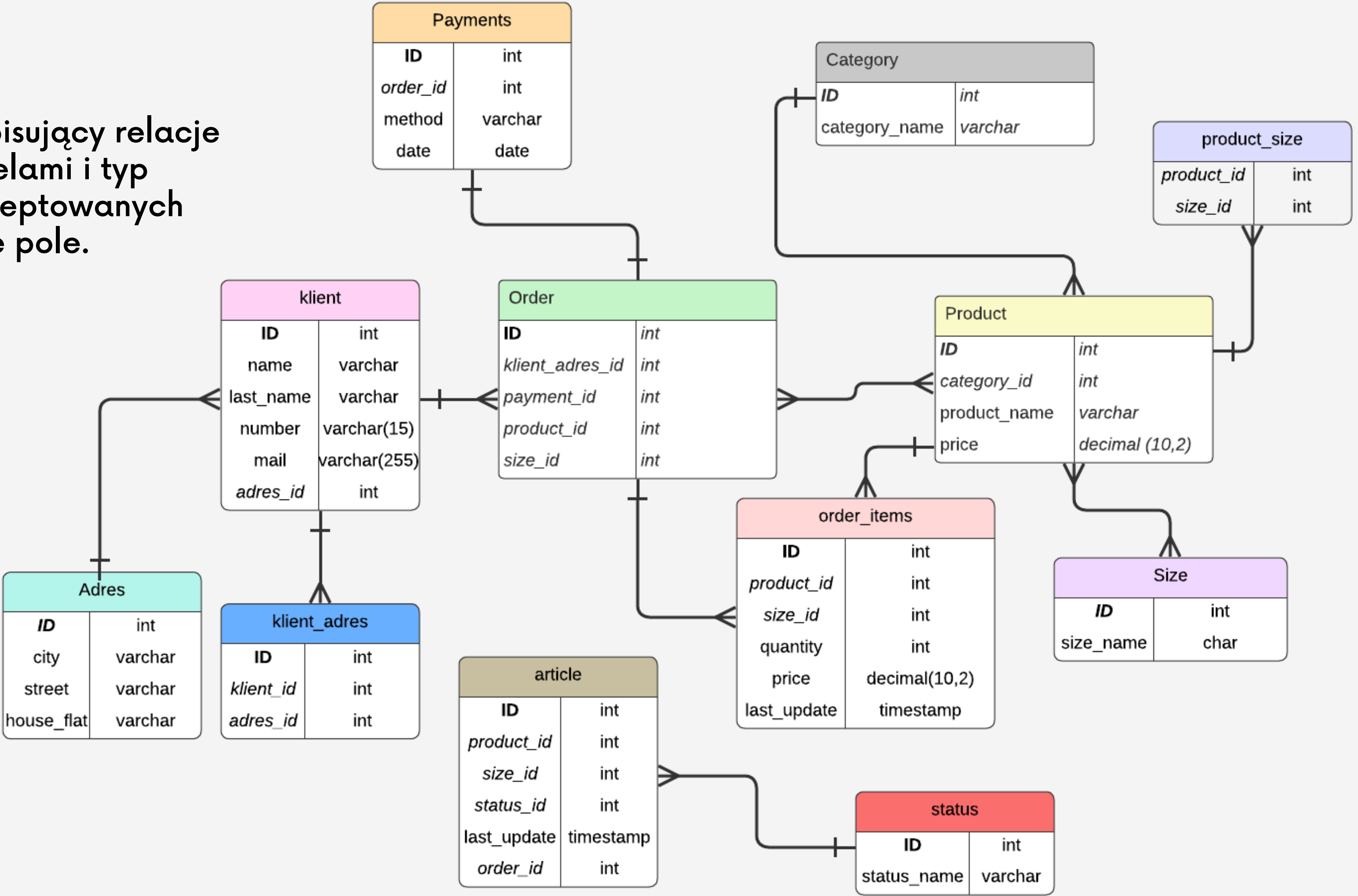


Diagram opisujący relacje między tabelami i typ danych akceptowanych przez każde pole.



Zapytanie SQL:

Pytanie, aby wyświetlić historię zamówień klienta

KLIENCI

- Klienci mogą zobaczyć informacje o swoich zamówieniach, ich statusach i historii zakupów.

Worksheet | Query Builder

```
SELECT
    k.mail AS email,
    p.product_name,
    s.size_name,
    st.status_name AS order_status,
    a.last_update AS status_last_update,
    pay.method AS payment_method,
    pay.payment_date AS payment_date
FROM klient k
JOIN klient_adres ka ON k.id = ka.klient_id
JOIN orders o ON ka.id = o.klient_adres_id
JOIN product p ON o.product_id = p.id
JOIN sizetable s ON o.size_id = s.id
JOIN payments pay ON o.payment_id = pay.id
JOIN article a ON o.id = a.order_id
JOIN status st ON a.status_id = st.id
WHERE k.mail = 'john.doe@mail.com'
ORDER BY pay.payment_date DESC;
```

Wynik:

Script Output | Query Result

SQL | All Rows Fetched: 3 in 0,008 seconds

	EMAIL	PRODUCT_NAME	SIZE_NAME	ORDER_STATUS	STATUS_LAST_UPDATE	PAYMENT_METHOD	PAYMENT_DATE
1	john.doe@mail.com	Waistcoat	S	delivered	11.10.2024 20:05:58,000000000	PayPal	06.01.2024
2	john.doe@mail.com	Jacket	XS	in stock	11.11.2024 18:20:21,000000000	Credit Card	05.01.2024
3	john.doe@mail.com	Jacket	XS	packed	06.11.2024 11:00:00,000000000	Credit Card	01.01.2024

ADMINISTRATOR

- Zarządzanie produktami, zamówieniami i użytkownikami

Administratorzy zarządzają asortymentem produktów, monitorują ich kategorie i statusy.

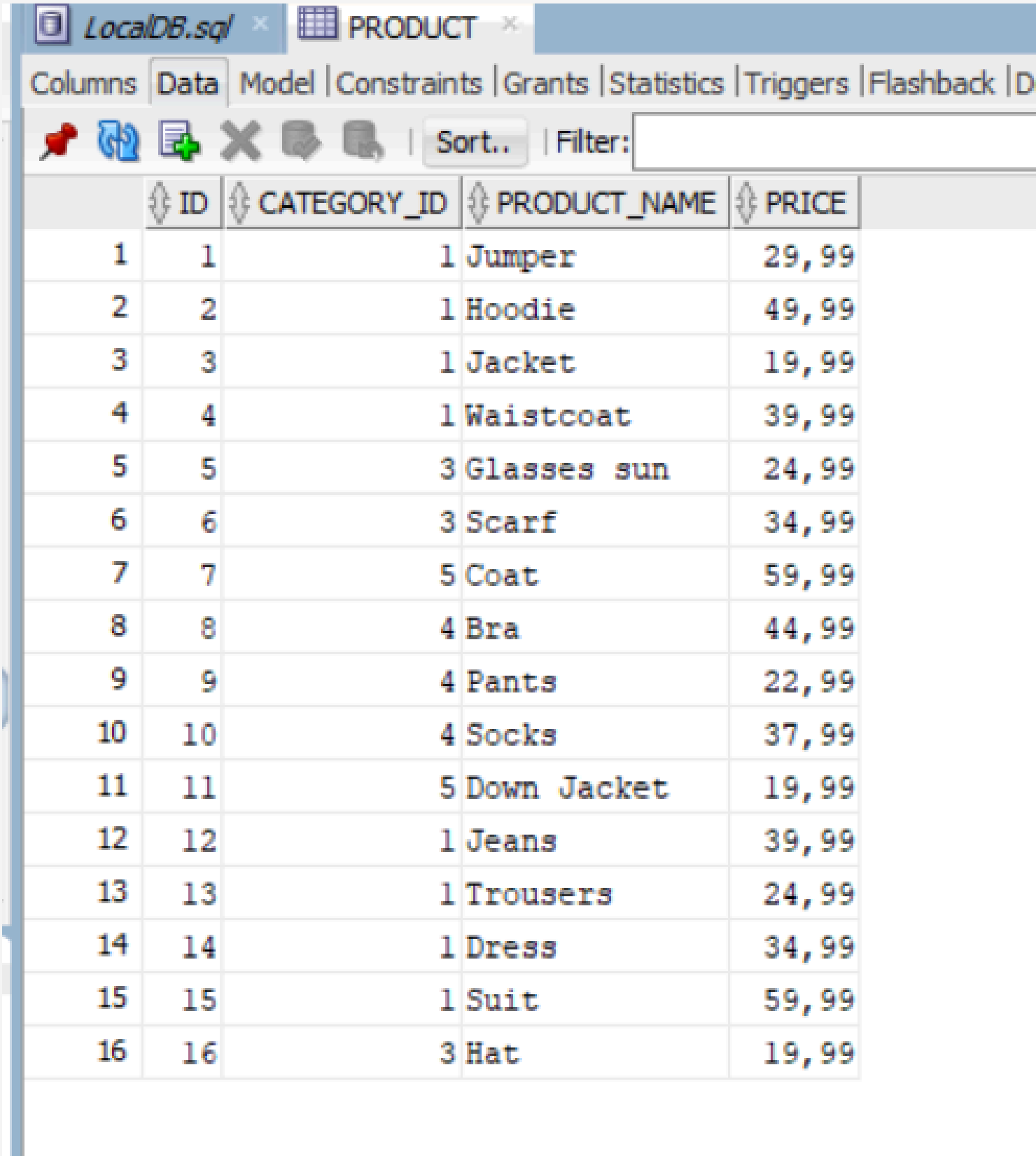
Do katalogu produktów zostaje dodany nowy produkt o określonej kategorii i cenie. Ten produkt jest już dostępny do zamówienia przez klientów.

Zapytanie SQL:

Dodanie nowego produktu do bazy danych

```
INSERT INTO product (id, product_name, category_id, price)
VALUES (16, 'Hat', 3, 19.99);
```

Wynik:



	ID	CATEGORY_ID	PRODUCT_NAME	PRICE
1	1	1	Jumper	29,99
2	2	1	Hoodie	49,99
3	3	1	Jacket	19,99
4	4	1	Waistcoat	39,99
5	5	3	Glasses sun	24,99
6	6	3	Scarf	34,99
7	7	5	Coat	59,99
8	8	4	Bra	44,99
9	9	4	Pants	22,99
10	10	4	Socks	37,99
11	11	5	Down Jacket	19,99
12	12	1	Jeans	39,99
13	13	1	Trousers	24,99
14	14	1	Dress	34,99
15	15	1	Suit	59,99
16	16	3	Hat	19,99

ADMINISTRATOR

Zapytanie SQL:

Dodawanie klienta

Worksheet	Query Builder
	<pre>INSERT INTO klient (ID, first_name, last_name, phone_number, mail, adres_id) VALUES (11, 'Alexander', 'Petrov', '1122334455', 'alex.petrov@mail.com', 13);</pre>

Wynik:

	ID	FIRST_NAME	LAST_NAME	PHONE_NUMBER	MAIL	ADRES_ID
1	1	John	Doe	1234567890	john.doe@mail.com	1
2	2	Jane	Smith	987654321	jane.smith@mail.com	2
3	3	Michael	Johnson	2345678901	michael.johnson@mail.com	3
4	4	Emily	Davis	3456789012	emily.davis@mail.com	4
5	5	Daniel	Brown	4567890123	daniel.brown@mail.com	5
6	6	Sarah	Wilson	5678901234	sarah.wilson@mail.com	6
7	7	David	Taylor	6789012345	david.taylor@mail.com	7
8	8	Jessica	Anderson	7890123456	jessica.anderson@mail.com	8
9	9	Robert	Thomas	8901234567	robert.thomas@mail.com	9
10	10	Laura	Martinez	9012345678	laura.martinez@mail.com	10
11	11	Alexander	Petrov	1122334455	alex.petrov@mail.com	13

Zapytanie SQL:

Aktualizacja statusu zamówienia

Worksheet	Query Builder
	<pre>UPDATE article SET status_id = 3, last_update = CURRENT_TIMESTAMP WHERE order_id = 5; SELECT ar.order_id AS Order_ID, s.status_name AS Status, ar.last_update AS Last_Update FROM article ar JOIN status s ON ar.status_id = s.ID WHERE ar.order_id = 5;</pre>

Wynik:

Query Result x			
SQL All Rows Fetched: 1 in 0,042 seconds			
	ORDER_ID	STATUS	LAST_UPDATE
1	5	packed	11.11.2024 22:30:23,505000000

DZIAŁ FINANSOWO-KSIĘGOWY

Generowanie raportów o przychodach i płatnościach

- Dział finansowy analizuje płatności, generuje raporty dotyczące przychodów i popularności metod płatności.

Dział finansowy otrzymuje informację o tym, która metoda płatności (np. karta kredytowa, PayPal) generuje większy przychód. Pomaga to analizować popularność metod płatności wśród klientów.

Zapytanie SQL:

Generowanie raportu przychodów według metod płatności

```
SELECT
    pay.method AS payment_method,
    SUM(p.price) AS total_revenue
FROM
    orders o
JOIN
    payments pay ON o.payment_id = pay.id
JOIN
    product p ON o.product_id = p.id
GROUP BY
    pay.method
ORDER BY
    total_revenue DESC;
```

Wynik:

Query Result x		
All Rows Fetched: 4 in 0,002 seconds		
	PAYMENT_METHOD	TOTAL_REVENUE
1	PayPal	184,96
2	Credit Card	87,96
3	Cash	79,98
4	Bank Transfer	72,98

DZIAŁ FINANSOWO-KSIĘGOWY

Wynik:

Query Result x		
All Rows Fetched: 4 in 0,009 seconds		
	CATEGORY	TOTAL_REVENUE
1	Clothing	199,94
2	Underwear	105,97
3	Outerwear	59,99
4	Accesories	59,98

Zapytanie SQL:

Raport dochodów według kategorii produktów

```
SELECT
    c.category_name AS category,
    SUM(p.price) AS total_revenue
FROM
    orders o
JOIN
    product p ON o.product_id = p.id
JOIN
    category c ON p.category_id = c.id
GROUP BY
    c.category_name
ORDER BY
    total_revenue DESC;
```

OGÓLNE ZAPYTANIA ANALITYCZNE

Zapytanie SQL:

Raportuj stany zamówień za wybrany okres

```
SELECT
  s.status_name AS Status,
  COUNT(*) AS Order_Count
FROM
  article ar
JOIN
  status s ON ar.status_id = s.ID
WHERE
  ar.last_update BETWEEN TO_DATE('2024-11-01', 'YYYY-MM-DD') AND TO_DATE('2024-12-31', 'YYYY-MM-DD')
GROUP BY
  s.status_name
ORDER BY
  Order_Count DESC;
```

Wynik:

Query Result		
All Rows Fetched: 7 in 0,002 seconds		
	STATUS	ORDER_COUNT
1	in stock	5
2	returned	2
3	packed	2
4	delivered	2
5	sold	2
6	in transit	1
7	ordered	1

Zapytanie SQL:

Analiza najpopularniejszych produktów

```
SELECT
  p.product_name,
  COUNT(o.id) AS order_count
FROM
  orders o
JOIN
  product p ON o.product_id = p.id
GROUP BY
  p.product_name
ORDER BY
  order_count DESC;
```

Wynik:

	PRODUCT_NAME	ORDER_COUNT
1	Jacket	2
2	Waistcoat	2
3	Socks	1
4	Scarf	1
5	Coat	1
6	Bra	1
7	Pants	1
8	Hoodie	1

PRZYKŁAD RAPORTU FINANSOWEGO DO DRUKU

zapytanie do sporządzenia raportu
podsumowującego miesięczne przychody,
zawierającego podział według kategorii produktów i
metod płatności.

Wynik:

SQL All Rows Fetched: 11 in 0,004 seconds					
	MONTH	PRODUCT_CATEGORY	PAYMENT_METHOD	TOTAL_REVENUE	
1	2024-05	Accesories	Bank Transfer	34,99	
2	2024-05	Accesories	Credit Card	24,99	
3	2024-05	Clothing	Credit Card	19,99	
4	2024-03	Outerwear	PayPal	59,99	
5	2024-03	Underwear	Bank Transfer	37,99	
6	2024-02	Clothing	PayPal	39,99	
7	2024-01	Clothing	Cash	79,98	
8	2024-01	Clothing	Credit Card	19,99	
9	2024-01	Clothing	PayPal	39,99	
10	2024-01	Underwear	Credit Card	22,99	
11	2024-01	Underwear	PayPal	44,99	

Zapytanie SQL:

```
rksheet  Query Builder
SELECT
    TO_CHAR(pay.payment_date, 'YYYY-MM') AS month,
    c.category_name AS product_category,
    pay.method AS payment_method,
    SUM(p.price) AS total_revenue
FROM
    orders o
JOIN
    product p ON o.product_id = p.id
JOIN
    category c ON p.category_id = c.id
JOIN
    payments pay ON o.payment_id = pay.id
GROUP BY
    TO_CHAR(pay.payment_date, 'YYYY-MM'),
    c.category_name,
    pay.method
ORDER BY
    month DESC,
    product_category,
    payment_method;
```

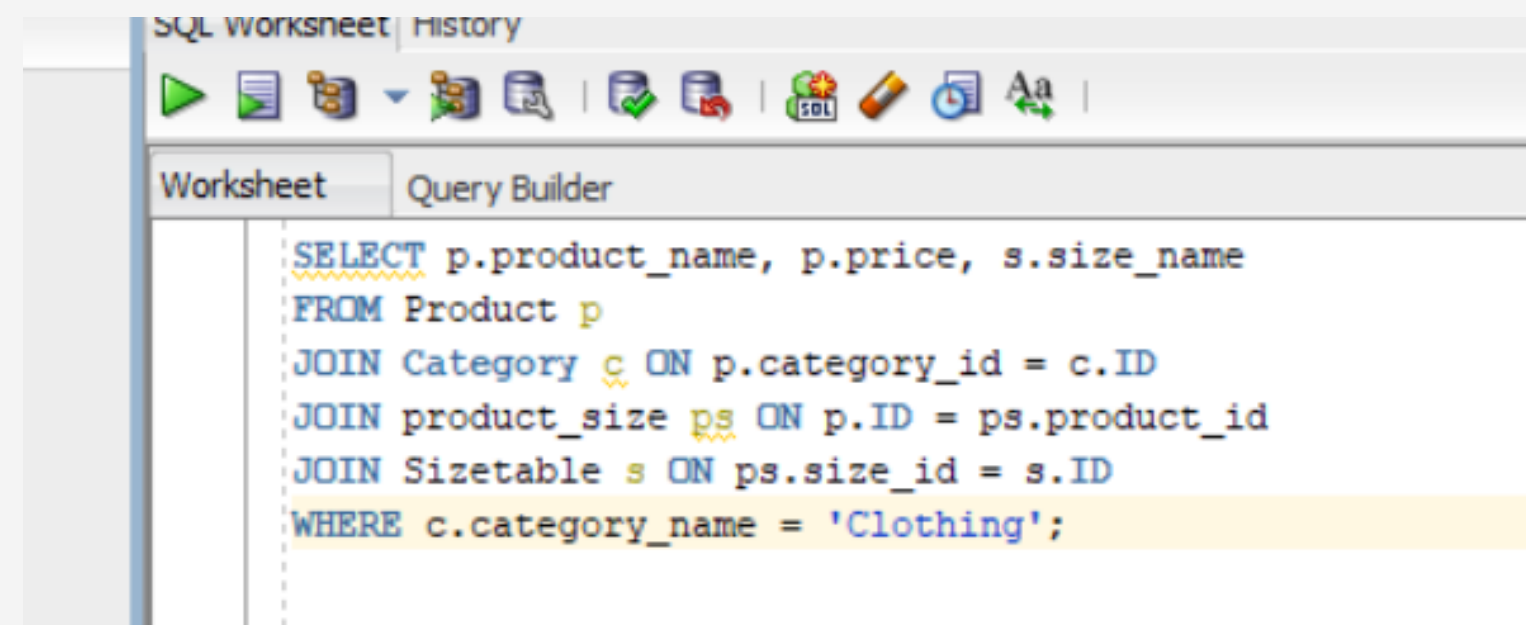
ZARZĄDZANIE DUŻĄ ILOŚCIĄ DANYCH PRODUKTÓW

Nasze rozwiązanie bazodanowe:

- **Katalog produktów, znormalizowany i ustrukturyzowany:**
- Tabela **Product** przechowuje ogólne informacje o produktach.
- Tabele **Category**, **Size** i **product_size** umożliwiają elastyczne zarządzanie atrybutami produktów.
- Łatwo dodawaj nowe produkty, aktualizuj ceny i dostępność rozmiarów.

Zapytanie SQL:

Wyświetl wszystkie produkty w kategorii Odzież wraz z ich cenami i dostępnymi rozmiarami.



```
SELECT p.product_name, p.price, s.size_name
FROM Product p
JOIN Category c ON p.category_id = c.ID
JOIN product_size ps ON p.ID = ps.product_id
JOIN Sizetable s ON ps.size_id = s.ID
WHERE c.category_name = 'Clothing';
```

Wynik:

1	Jumper	29,99	XS
2	Jumper	29,99	S
3	Jumper	29,99	M
4	Hoodie	49,99	S
5	Hoodie	49,99	M
6	Jacket	19,99	XS
7	Jacket	19,99	L
8	Waistcoat	39,99	XS
9	Waistcoat	39,99	S
10	Jeans	39,99	M
11	Jeans	39,99	L
12	Jeans	39,99	XL

ZARZĄDZANIE ADRESAMI DOSTAWY

Nasze rozwiązanie bazodanowe:

- Tabele **Adres** i **klient_adres** umożliwiają klientom zapisywanie wielu adresów.
- Tabela **Order** kojarzy konkretny adres z zamówieniem.

Rezultat: Wygoda dla klientów i redukcja błędów dostawy (błędny adres).

Zapytanie SQL:

Wyświetl listę wszystkich adresów dostawy klienta.

```
Worksheet | Query Builder
SELECT a.city, a.street, a.house_flat
FROM adres a
JOIN klient_adres ka ON a.id = ka.adres_id
WHERE ka.klient_id = 1;
```

Wynik:

Query Result x			
SQL All Rows Fetched: 3 in 0,002 seconds			
	CITY	STREET	HOUSE_FLAT
1	New York	5th Avenue	101A
2	Boston	Beacon St	100A
3	Seattle	Pine St	200B

ANALIZA SPRZEDAŻY I WYDAJNOŚCI

Nasze rozwiązanie bazodanowe:

- Tabele **order_items** i **Payments** pozwalają nam obliczyć:
- Całkowity przychód.
- Pozycje o największym popycie.
- Wydajność poszczególnych kategorii.
- Średni paragon na zamówienie.

Rezultat: Uzasadnione decyzje biznesowe oparte na analizie (np. które produkty powinny być uzupełniane częściej lub które kategorie powinny być przecenione).

Zapytanie SQL:

Znajdź najlepiej sprzedające się produkty.

```
Worksheet | Query Builder
SELECT p.product_name, SUM(oi.quantity) AS total_sold
FROM Product p
JOIN order_items oi ON p.ID = oi.product_id
GROUP BY p.product_name
ORDER BY total_sold DESC
FETCH FIRST 5 ROWS ONLY;
```

Wynik:

SQL All Rows Fetched: 5 in 0,004 seconds		
	PRODUCT_NAME	TOTAL_SOLD
1	Jumper	10
2	Hoodie	5
3	Trousers	4
4	Jacket	3
5	Suit	3

OBSŁUGA ZWROTÓW I WYMIAN

Nasze rozwiązanie bazodanowe:

- Tabela **status** pozwala nam rejestrować zwrot artykułu (np. status to „zwrócony”).
- Aktualizacje w tabelach **article** i **order_items** pozwalają śledzić, które artykuły zostały zwrócone i wymienione.

Rezultat: Przejrzyste zarządzanie zwrotami, minimalizacja strat finansowych i poprawa obsługi klienta.

Zapytanie SQL:

Znajdź wszystkie produkty, które zostały zwrócone przez klientów.

```
SELECT p.product_name, COUNT(a.ID) AS total_returns
FROM Product p
JOIN article a ON p.ID = a.product_id
JOIN status s ON a.status_id = s.ID
WHERE s.status_name = 'returned'
GROUP BY p.product_name
ORDER BY total_returns DESC;
```

Wynik:

Query Result x			
All Rows Fetched: 2 in 0,025 second			
	PRODUCT_NAME	TOTAL_RETURNS	
1	Jeans	1	
2	Coat	1	

ZARZĄDZANIE KLIENTAMI I ICH PREFERENCJAMI

Nasze rozwiązanie bazodanowe:

- Tabela **klient** przechowuje informacje kontaktowe klientów.
- Połączone tabele pozwalają analizować historię zamówień każdego klienta.

Rezultat: Możliwość personalizacji ofert, poprawy marketingu i zwiększenia lojalności klientów.

Zapytanie SQL:

Znajdź wszystkie zamówienia klientów według ich adresów e-mail.

```
Worksheet | Query Builder
SELECT
    k.mail,
    o.id AS order_id,
    p.product_name,
    s.size_name,
    pay.method AS payment_method
FROM klient k
JOIN klient_adres ka ON k.id = ka.klient_id
JOIN orders o ON ka.id = o.klient_adres_id
JOIN product p ON o.product_id = p.id
JOIN sizetable s ON o.size_id = s.id
JOIN payments pay ON o.payment_id = pay.id
ORDER BY k.mail;
```

Wynik:

	MAIL	ORDER_ID	PRODUCT_NAME	SIZE_NAME	PAYMENT_METHOD
1	daniel.brown@mail.com	5	Glasses sun	XL	Credit Card
2	david.taylor@mail.com	7	Jumper	S	Cash
3	emily.davis@mail.com	4	Socks	L	Bank Transfer
4	jane.smith@mail.com	2	Coat	S	PayPal
5	jessica.anderson@mail.com	8	Scarf	M	Bank Transfer
6	john.doe@mail.com	1	Jacket	XS	Credit Card
7	john.doe@mail.com	11	Jacket	XS	Credit Card
8	john.doe@mail.com	12	Waistcoat	S	PayPal
9	laura.martinez@mail.com	10	Waistcoat	XL	PayPal
10	michael.johnson@mail.com	3	Hoodie	M	Cash
11	robert.thomas@mail.com	9	Pants	L	Credit Card
12	sarah.wilson@mail.com	6	Bra	XS	PayPal

DATE
13/11/2024

DZIĘKUJEMY ZA UWAGĘ

Karyna Cherniii kchernii@edu.cdv.pl
Vladyslav Fedii vfedii@edu.cdv.pl

