„ZPR PWr – Zintegrowany Program Rozwoju Politechniki Wrocławskiej”
Hurtownie Danych - laboratorium

Zad 1. Modelowanie danych

1. Zweryfikować model danych w kontekście podanego zbioru reguł i ograniczeń dziedzinowych modyfikując zbiór reguł i ograniczeń (uzupełniając lub poprawiając ich definicję)

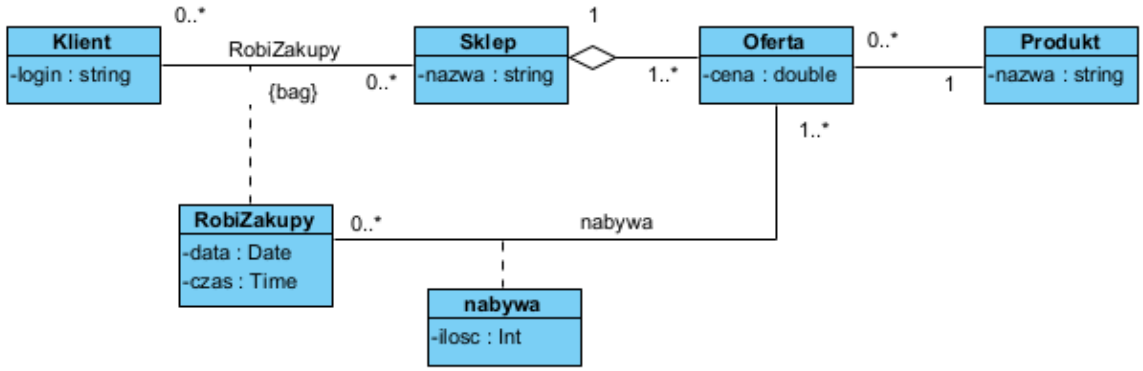
Reguły i ograniczenia dziedzinowe:

Reg/01 – Klient może wielokrotnie robić zakupy w tym samym sklepie.
Reg.02 – W sklepie może robić zakupy dowolny klient.
Reg.03 – Każdy zakup realizowany jest przez klienta w sklepie w określonym dniu i godzinie.
Reg/04 – Sklep musi oferować co najmniej jeden produkt.
Reg/05 – Produkt może być oferowany w wielu sklepach
Reg/06 – Dana oferta w danym sklepie dotyczy jednego produktu.

Ogr/01 – Cena oferty musi być większa od zera.
Ogr/02 –Ilość nabycia musi być większa od zera.
Ogr/03 – Nazwa produktu nie może być pusta
Ogr/04 – Nazwa sklepu nie może być pusta

2. Przedstawić uzupełnioną i poprawioną wersję modelu danych (kompletny diagram klas UML)

Kompletny diagram klas UML



3. Utworzyć logiczny model danych w postaci skryptu w języku DDL SQL (uwzględniając reguły i ograniczenia dziedzinowe), starając się zachować zgodność ze standardem języka SQL (pomijając, o ile to możliwe, natywne konstrukcje implementacji języków SQL)

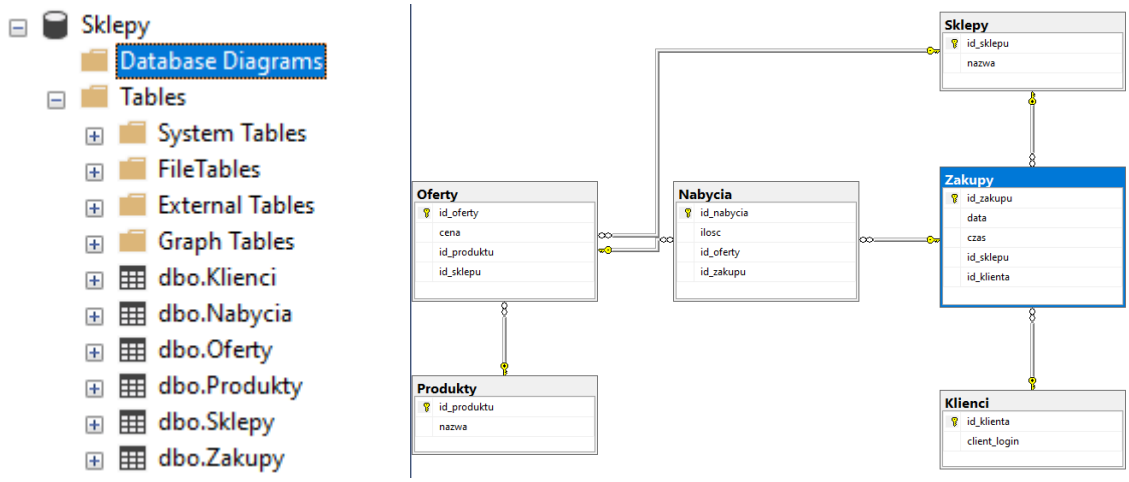
Skrypt DDL SQL

```
CREATE SEQUENCE klient_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE sklep_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE produkt_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE oferta_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE zakup_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE nabycie_seq START WITH 1 INCREMENT BY 1;

CREATE TABLE Klienci(
    id_klienta INT NOT NULL DEFAULT NEXT VALUE FOR klient_seq,
    client_login VARCHAR(255) NOT NULL UNIQUE,
    PRIMARY KEY(id_klienta)
);

CREATE TABLE Sklepy(
    id_sklepu INT NOT NULL DEFAULT NEXT VALUE FOR sklep_seq,
    nazwa VARCHAR(255) NOT NULL,
    PRIMARY KEY(id_sklepu)
);

CREATE TABLE Produkty(
    id_produktu INT NOT NULL DEFAULT NEXT VALUE FOR produkt_seq,
    nazwa VARCHAR(255) NOT NULL,
    PRIMARY KEY(id_produktu)
);

CREATE TABLE Oferty (
    id_oferty INT NOT NULL DEFAULT NEXT VALUE FOR oferta_seq,
    cena DECIMAL(10,2) NOT NULL CHECK (cena > 0),
    id_produktu INT NOT NULL,
    id_sklepu INT NOT NULL,
    PRIMARY KEY(id_oferty),
    FOREIGN KEY (id_produktu) REFERENCES Produkty(id_produktu),
    FOREIGN KEY (id_sklepu) REFERENCES Sklepy(id_sklepu)
);

CREATE TABLE Zakupy(
    id_zakupu INT NOT NULL DEFAULT NEXT VALUE FOR zakup_seq,
    data DATE NOT NULL,
    czas TIME NOT NULL,
    id_sklepu INT NOT NULL,
    id_klienta INT NOT NULL,
    PRIMARY KEY(id_zakupu),
    FOREIGN KEY (id_sklepu) REFERENCES Sklepy(id_sklepu),
    FOREIGN KEY (id_klienta) REFERENCES Klienci(id_klienta)
);

CREATE TABLE Nabycia(
    id_nabycia INT NOT NULL DEFAULT NEXT VALUE FOR nabycie_seq,
    ilosc INT NOT NULL CHECK (ilosc > 0),
    id_oferty INT NOT NULL,
```

```
    id_zakupu INT NOT NULL,
    PRIMARY KEY(id_nabycia),
    FOREIGN KEY (id_oferty) REFERENCES Oferty(id_oferty),
    FOREIGN KEY (id_zakupu) REFERENCES Zakupy(id_zakupu)
);
```

4. Utworzyć bazę danych w systemie MS SQL, która jest fizycznym modelem danych modelowanego wycinka rzeczywistości

FIZYCZNY MODEL DANYCH W MS SQL



5. Wprowadzić klika rekordów do każdej tabeli sprawdzając poprawność implementacji (zarówno poprawne dane, jak i niezgodne z obowiązującymi regułami – komentując i wyjaśniając uzyskane komunikaty z systemu SZBD)
Inserty poprawnych danych

```
INSERT INTO Klienci (client_login) VALUES
('aniakow'),
('marcinpol'),
('zosiawik'),
('tomaszbar'),
('jarekdom'),
('kasiasok');

INSERT INTO Sklepy (nazwa) VALUES
('We Wroclawiu'),
('W Warszawoe'),
('W Krakowie'),
('W Gdansku'),
('W Limanowej'),
('W Poznaniu');

INSERT INTO Produkty (nazwa) VALUES
('Chleb'),
('Mleko'),
('Jabłko'),
('Ser'),
('Woda'),
('Masło'),
('Jajka');

INSERT INTO Oferty (cena, id_produktu, id_sklepu) VALUES
(4.50, 1, 1),  -- Chleb we Wroclawiu
(3.99, 1, 2),  -- Chleb w Warszawie
(3.20, 2, 1),
(5.99, 4, 3),
(2.50, 5, 4),
(6.99, 3, 2),
(8.50, 6, 5),
(12.99, 7, 6);

INSERT INTO Zakupy (data, czas, id_sklepu, id_klienta) VALUES
('2025-03-05', '10:15:00', 1, 1),  -- aniakow we wroclawiu
('2025-03-06', '14:30:00', 2, 2),  -- marcin w warszawie
('2025-03-07', '18:45:00', 3, 3),
('2025-03-08', '09:20:00', 1, 4),
('2025-03-08', '16:10:00', 4, 1),
('2025-03-09', '12:30:00', 5, 5),
('2025-03-10', '17:45:00', 6, 6);

INSERT INTO Nabycia (ilosc, id_oferty, id_zakupu) VALUES
(2, 1, 1),  -- ania 2 chleby w zakupach nr 1
(1, 3, 1),  -- ania 1 mleko tez w zakupach nr 1
(3, 6, 2),
(1, 4, 3),
(4, 1, 4),
(2, 5, 5),
```

```
(1, 7, 6),
(3, 8, 7);
```

Inserty nie poprawnych danych

```
-- cena <= 0
INSERT INTO Oferty (cena, id_produktu, id_sklepu) VALUES (0, 1, 1);
```

```
Msg 547, Level 16, State 0, Line 137
The INSERT statement conflicted with the CHECK constraint "CK__Oferty__cena__5165187F". The conflict occurred in database "Sklepy", table "dbo.Oferty", column 'cena'.
The statement has been terminated.
```

```
-- ilosc <= 0
INSERT INTO Nabycia (ilosc, id_oferty, id_zakupu) VALUES (0, 1, 1);
```

```
Msg 547, Level 16, State 0, Line 140
The INSERT statement conflicted with the CHECK constraint "CK__Nabycia__ilosc__5BE2A6F2". The conflict occurred in database "Sklepy", table "dbo.Nabycia", column 'ilosc'.
The statement has been terminated.
```

```
-- id_produktu 100 nie ma
INSERT INTO Oferty (cena, id_produktu, id_sklepu) VALUES (9.99, 100,
1);
```

```
Msg 547, Level 16, State 0, Line 143
The INSERT statement conflicted with the FOREIGN KEY constraint "FK__Oferty__id_produ__52593CB8". The conflict occurred in database "Sklepy", table "dbo.Produkty", column 'id_produktu'.
The statement has been terminated.
```

```
-- id_klienta 100 nie ma
INSERT INTO Zakupy (data, czas, id_sklepu, id_klienta) VALUES
('2025-03-15', '11:30:00', 1, 100);
```

```
Msg 547, Level 16, State 0, Line 146
The INSERT statement conflicted with the FOREIGN KEY constraint "FK__Zakupy__id_klien__5812160E". The conflict occurred in database "Sklepy", table "dbo.Klienci", column 'id_klienta'.
The statement has been terminated.
```

```
-- id_zakupu 100 nie ma
INSERT INTO Nabycia (ilosc, id_oferty, id_zakupu) VALUES (3, 1,
100);
```

```
Msg 547, Level 16, State 0, Line 149
The INSERT statement conflicted with the FOREIGN KEY constraint "FK__Nabycia__id_zaku__5DCAEF64". The conflict occurred in database "Sklepy", table "dbo.Zakupy", column 'id_zakupu'.
The statement has been terminated.
```

```
--duplikat wartosci
INSERT INTO Klienci(client_login) VALUES ('aniakow');
```

```
Msg 2627, Level 14, State 1, Line 152
Violation of UNIQUE KEY constraint 'UQ__Klienci__1B4D2FBA0510C92A'. Cannot insert duplicate key in object 'dbo.Klienci'. The duplicate key value is (aniakow).
The statement has been terminated.

Completion time: 2025-03-09T23:48:13.0405252+01:00
```

Zad. 2. Podstawy SQL

Proszę zapisać zapytania SQL, które dadzą odpowiedź na poniższe pytania. Proszę zinterpretować wyniki.

Rozwiązania:

1. **Ile jest produktów w bazie? Ile kategorii i podkategorii?**

```
--zadanie 1
SELECT COUNT(*) 'l. produktow'
FROM Production.Product;

SELECT COUNT(*) 'l. kategorii'
FROM Production.ProductCategory;

SELECT COUNT(*) 'l. podkategorii'
FROM Production.ProductSubcategory;
```

Results Messages

	l. produktow
1	504

	l. kategorii
1	4

	l. podkategorii
1	37

2. Wypisz produkty, które nie mają zdefiniowanego koloru.

```
--zadanie 2
SELECT ProductID, Name, Color
FROM Production.Product
WHERE Color IS NULL;
```

Results Messages

	ProductID	Name	Color
1	1	Adjustable Race	NULL
2	2	Bearing Ball	NULL
3	3	BB Ball Bearing	NULL
4	4	Headset Ball Bearings	NULL
5	316	Blade	NULL
6	323	Crown Race	NULL
7	324	Chain Stays	NULL
8	325	Decal 1	NULL
9	326	Decal 2	NULL
10	327	Down Tube	NULL
11	328	Mountain End Caps	NULL

3. Podaj roczną kwotę transakcji (SalesOrderHeader.TotalDue) w poszczególnych latach.

```
--zadanie 3
SELECT YEAR(OrderDate) "Year", SUM(TotalDue) "Sum"
FROM Sales.SalesOrderHeader
GROUP BY YEAR(OrderDate)
ORDER BY 1 DESC;
```

Results | Messages

	Year	Sum
1	2014	22419498,3157
2	2013	48965887,9632
3	2012	37675700,312
4	2011	14155699,525

4. Ilu jest klientów, a ilu sprzedawców w sklepie? Ilu w poszczególnych regionach?

```
-- zadanie 4

SELECT T.Name "Territory", COUNT(*) "Customers"
FROM
    Sales.Customer C JOIN Sales.SalesTerritory T
    ON C.TerritoryID = T.TerritoryID
GROUP BY T.Name
UNION ALL
SELECT 'Total Sum', COUNT(*)
FROM Sales.Customer;
```

Results | Messages

	Territory	Customers
1	Australia	3665
2	Canada	1791
3	Central	132
4	France	1884
5	Germany	1852
6	Northeast	113
7	Northwest	3520
8	Southeast	176
9	Southwest	4696
10	United Kingdom	1991
11	Total Sum	19820

```
SELECT T.Name "Territory", COUNT(*)
```

```
FROM Sales.SalesPerson C LEFT JOIN Sales.SalesTerritory T
ON C.TerritoryID = T.TerritoryID
GROUP BY T.Name
UNION ALL
SELECT 'Total sum', COUNT(*)
FROM Sales.SalesPerson;
```

Results Messages

	Territory	Sales
1	NULL	3
2	Australia	1
3	Canada	2
4	Central	1
5	France	1
6	Germany	1
7	Northeast	1
8	Northwest	3
9	Southeast	1
10	Southwest	2
11	United Kingdom	1
12	Total sum	17

5. Ile było wykonanych transakcji w poszczególnych latach?

```
-- zadanie 5
SELECT YEAR(OrderDate) AS 'Year', COUNT(*) AS 'Transactions'
FROM Sales.SalesOrderHeader
GROUP BY YEAR(OrderDate);
```

Results Messages

	Year	Transactions
1	2013	14182
2	2014	11761
3	2011	1607
4	2012	3915

6. Podaj produkty, które nie zostały kupione przez żadnego klienta. Zestawienie pogrupuj według kategorii i podkategorii.

```
-- zadanie 6
SELECT C.Name, SC.Name, P.Name
FROM
    Production.Product P
    LEFT JOIN Sales.SalesOrderDetail SOD ON P.ProductID =
SOD.ProductID
    LEFT JOIN Production.ProductSubcategory SC ON
P.ProductSubcategoryID = SC.ProductSubcategoryID
    LEFT JOIN Production.ProductCategory C ON SC.ProductCategoryID =
C.ProductCategoryID
WHERE SOD.SalesOrderID IS NULL
```

```
GROUP BY C.Name, SC.Name, P.Name;
```

Results | Messages

	Name	Name	Name
208	NULL	NULL	Touring End Caps
209	NULL	NULL	Touring Rim
210	Acc...	Lights	Headlights - Dual-Beam
211	Acc...	Lights	Headlights - Weatherp...
212	Acc...	Lights	Taillights - Battery-Po...
213	Acc...	Pan...	Touring-Panniers, Large
214	Acc...	Pu...	Mountain Pump
215	Clot...	Sho...	Men's Sports Shorts, XL
216	Com...	Bott...	ML Bottom Bracket
217	Com...	Forks	ML Fork
218	Com...	Han...	ML Road Handlebars
219	Com...	Mo...	HL Mountain Frame - ...
220	Com...	Mo...	HL Mountain Frame - ...
221	Com...	Roa...	HL Road Frame - Blac...
222	Com...	Roa...	HL Road Frame - Blac...

7. Oblicz minimalną i maksymalną kwotę rabatu udzielonego na produkty w poszczególnych podkategoriach.

```
-- zadanie 7
SELECT SC.Name,
    MIN(SOD.UnitPriceDiscount * P.ListPrice) "Min",
    MAX(SOD.UnitPriceDiscount * P.ListPrice) "Max"
FROM
    Sales.SalesOrderDetail SOD
    JOIN Production.Product P ON SOD.ProductID = P.ProductID
    RIGHT JOIN Production.ProductSubcategory SC ON
P.ProductSubcategoryID = SC.ProductSubcategoryID
WHERE
    SOD.UnitPriceDiscount != 0
GROUP BY SC.Name;
```

100 %

Results | Messages

	Name	Min	Max
1	Bib-Shorts	1,7998	4,4995
2	Bike Racks	2,40	12,00
3	Bottles and Cages	0,0998	0,499
4	Brakes	2,13	2,13
5	Caps	0,1798	0,899
6	Chains	0,4048	0,4048
7	Cleaners	0,159	0,795
8	Cranksets	8,0998	8,0998
9	Derailleurs	1,8298	1,8298
10	Forks	4,5898	4,5898
11	Gloves	0,4898	5,6985
12	Handlebars	0,8908	2,227
13	Headsets	2,0458	5,1145
14	Helmets	0,6998	5,2485
15	Hydration Packs	1,0998	2,7495

8. Podaj produkty, których cena jest wyższa od średniej ceny produktów w sklepie.

```
-- zadanie 8
SELECT P.Name, P.ListPrice
FROM Production.Product P
WHERE P.ListPrice >
    (SELECT AVG(ListPrice) FROM Production.Product)
ORDER BY 2;
```

	Name	ListPrice
123	Road-250 Black, 58	2443,35
124	Mountain-100 Black, 38	3374,99
125	Mountain-100 Black, 42	3374,99
126	Mountain-100 Black, 44	3374,99
127	Mountain-100 Black, 48	3374,99
128	Mountain-100 Silver, 38	3399,99
129	Mountain-100 Silver, 42	3399,99
130	Mountain-100 Silver, 44	3399,99
131	Mountain-100 Silver, 48	3399,99
132	Road-150 Red, 62	3578,27
133	Road-150 Red, 44	3578,27
134	Road-150 Red, 48	3578,27
135	Road-150 Red, 52	3578,27
136	Road-150 Red, 56	3578,27

9. Ile średnio produktów w każdej kategorii sprzedaje się w poszczególnych miesiącach?

```
-- zadanie 9
SELECT C.Name "NAME" ,
        MONTH(SOH.OrderDate) "MONTH",
        COUNT(SOD.ProductID) / COUNT(DISTINCT SOH.SalesOrderID) "AVG"
FROM
    Sales.SalesOrderDetail SOD
    JOIN Production.Product P ON SOD.ProductID = P.ProductID
    JOIN Production.ProductSubcategory SC ON P.ProductSubcategoryID =
SC.ProductSubcategoryID
    JOIN Production.ProductCategory C ON SC.ProductCategoryID =
C.ProductCategoryID
    JOIN Sales.SalesOrderHeader SOH ON SOH.SalesOrderID =
SOD.SalesOrderID
GROUP BY C.Name, MONTH(SOH.OrderDate)
```

Results | Messages

	Month	Name	avg
1	1	Accessories	2
2	1	Bikes	4
3	1	Clothing	4
4	1	Components	10
5	2	Accessories	2
6	2	Bikes	2
7	2	Clothing	4
8	2	Components	17
9	3	Accessories	3
10	3	Bikes	5
11	3	Clothing	9
12	3	Components	16
13	4	Accessories	2
14	4	Bikes	2
15	4	Clothing	4

10. Ile średnio czasu klient czeka na dostawę zamówionych produktów? Przygotuj zestawienie w zależności od kodu regionu (SalesTerritory.CountryRegionCode).

```
-- zadanie 10
SELECT
    ST.CountryRegionCode,
    AVG(DATEDIFF(DAY, SOH.OrderDate, SOH.ShipDate)) AS AverageDays
FROM
    Sales.SalesOrderHeader SOH
    JOIN Sales.SalesTerritory ST ON SOH.TerritoryID = ST.TerritoryID
GROUP BY ST.CountryRegionCode;
```

Results | Messages

	CountryRegionCode	AverageDays
1	AU	7
2	CA	7
3	DE	7
4	FR	7
5	GB	7
6	US	7

Wnioski

Dodanie dodatkowej klasy (`oferta`) jest naturalne w przypadku relacji wiele do wiele. Dodatkowo przechowywanie ceny w ofercie, która jest powiązana z dokładnie jednym produktem, zamiast w samym produkcie, umożliwia dostosowanie ceny produktu dla konkretnego sklepu. Podejście wydaje się logiczniejsze dla systemu bazodanowego dla sieci sklepów, porównując z alternatywnym – gdzie cena jest arbitralnie ustawiona dla produktu. Nawiązując do agregacji, taka oferta nie musi być ściśle związana z jednym sklepem – może zostać przeniesiona do innego sklepu.

Użycie `{bag}` oznacza, że klient może robić zakupy w tym samym sklepie wielokrotnie, a każda transakcja jest osobnym wpisem w systemie. Użycie `{set}` wymusiłoby unikalność każdego elementu, a więc zakupy są unikalne, dla danych parametrów asocjacji. Zważywszy na to, użycie `{bag}` wydaje się być naturalnym wyborem.

Do zakończenia zadania pojawiła się potrzeba transformacji diagramu klas do diagramu ERD, było to związane do pojawieniem się kluczy obcych w tabelach. Dodano również pewne atrybuty do tabel, żeby lepiej zobrazować sens systemu bazy danych i umożliwić jego implementację.

Baza danych `AdventureWorks` jest obszerną bazą danych, zawierającą informację na temat produktów konkretnych transakcji zawieranych pomiędzy klientami a sprzedawcami. Na podstawie wyników powyżej rozwiązanych zadań, możemy stwierdzić, że dane nie są wszędzie uzupełnione, występują produkty bez koloru, kategorii czy np. nie wszyscy sprzedawcy mają przydzielony obszar. Sprzedaż sklepu znacznie wzrosła w latach 2012, 2013. W roku 2014 jest mniejsza, jednak należy zwrócić uwagę, że dane są jedynie do czerwca 2014 roku.

```
SELECT MAX(MONTH(OrderDate))
FROM Sales.SalesOrderHeader
WHERE YEAR(OrderDate) = 2014;
```

00 %

Results | Messages

	(No column name)
1	6

Ciekawe jest jednak to, że już w czerwcu liczba transakcji była większa niż w roku 2012, mimo to łączna kwota transakcji pozostaje zdecydowanie niższa. Sugeruje, że klienci dokonywali więcej transakcji na mniejsze kwoty. Analizując wyniki z zadanie 9 nasuwa się wniosek, że średnia sprzedaż we wszystkich kategoriach, osiąga szczyt w miesiącach letnich, kategoria Components konsekwentnie generuje najwyższą liczbę transakcji w każdym miesiącu. Tym bardziej,

zważywszy na sezonowość zakupów, niska łączna kwota transakcji w roku 2014 nie wydaje się być zaskakująca.
Inne zależności, które dało się wyczytać z wyników to fakt, że sprzedaż jest bardzo zróżnicowana geograficznie, niektóre regiony obsługują kilkadziesiąt razy więcej klientów. W tym samym momencie rozłożenie geograficzne sprzedawców wydaje się być umiarkowane, zachowując zdecydowanie mniejsze wahania niż sprzedaż. Podobnie nie zróżnicowany wydaje się być czas wysyłki, który konsekwentnie w każdym rejonie wynosił tydzień.