

DOKUMENTACJA PROJEKTU
ZALICZENIOWEGO NA
PRZEDMIOCIE ADMINISTRACJA
INTERNETOWYMI BAZAMI
DANYCH

WYKONANIE:

Patryk Sobczak

Karolina Lubczyk

1) Opis modelu biznesowego.

Projekt prezentuje internetową Księgarnię. Jej zadaniem jest odpowiednie dystrybuowanie produktami, które oferuje. Głównym źródłem dochodu jest sprzedaż e-booków oraz wysyłanie ich na maile klientów. Księgarnia przewiduje wysyłanie dodatkowych produktów na fizyczny adres klienta.

Dane pobierane od klienta zależą od zamawianego produktu.

W przypadku produktów jakim w bazie są np. puzzle pobierany jest od klienta następujący zestaw: imię, nazwisko, email, ilość produktów, adres wysyłki.

Gdy klient zamawia tylko e-booka od klienta pobierany jest adres email, ilość zamówionych książek oraz imię i nazwisko.

W raportach zbierane są dane, które pozwolą rozwinąć internetową księgarnię: jaki autor jest najczęściej czytany, który gatunek książek przyciąga najwięcej czytelników, które produkty nie sprzedają się w ogóle.

2) Liczba zaprojektowanych i utworzonych tabel oraz relacje pomiędzy tabelami:

-Spis książek:

-ID książki (PK, int, notNULL)

-Tytuł (nvarchar(100), notNULL)

-IDGatunek (FK, non-clastered key[do zrobienia], int, notNULL)

-IDAutora (FK, int, notNULL)

-Cena (decimal (18,2), notNULL)

-Stan magazynowy (int, NULL)

-Spis autorów:

-IDAutora (PK, int, notNULL)

-Imię (varchar (50), notNULL)

-Nazwisko (varchar(50), notNULL)

-Spis Wydawnictw:

-IDWydawnictwa (PK, int, notNULL)

-Nazwa (nvarchar (50), notNULL)

-SpisKlientów:

-Email (PK,nvarchar (100), notNULL, non incr)

-Imię (varchar(50), notNULL)

-Nazwisko (varchar (50), notNULL)

-SpisAdresów :

-IDAdresu (PK, int, notNULL)

-Kod pocztowy (nchar(5), NULL)

- Miasto (varchar(100), null)
- Ulica (nvarchar(50), null)
- Nr domu (nchar(10), null)
- Nr mieszkania (nchar(10), null)
- Spis zamówień (główna tabela, wiele-wiele):
 - ID zamówienia (PK, int, notNULL)
 - Email(FK do klientów = Email, nvarchar (100), not NULL)
 - ID książki (FK do SpisKsiazek = IDksiążki, int, NULL)
 - Id produktu (FK do SpisProduktow = IDproduktu, int, NULL)
 - ID autora (FK do SpisAutorow = IDautora, NULL, int)
 - ID wydawnictwa (FK do SpisWydawnictw = IDwydawnictwa, NULL, int)
 - IDAdresu (FK do SpisAdresów = ID adresu, NULL, int)
 - Ilość_ksiazek (int, NULL)
 - Ilosc_produktow (int, NULL)
 - Suma kosztów (decimal(18.2), notNULL)
 - Walidacja (varchar(3), notNULL)
- Spis gatunków:
 - ID gatunku (PK, int, notNULL)
 - Nazwa gatunku (varchar(50), notNULL)
- Spis Produktów:
 - ID produktu (PK, int, notNULL)
 - Nazwa (nvarchar(50), notNULL)
 - Stan magazynowy (int, NULL)
 - Cena (decimal(18,2), notNULL)
- Karta Klienta:
 - Email (PK, FK, nvarchar(100), notNull, Relacja 1-1 do SpisKlientów = Email)
 - IDKarty (int, notNULL, wartosci pobierane z view.random)

Bazy zostały wypełnione przykładowymi danymi jak na załączonych screenach. W podanych tabelach zostały również zaimplementowane indeksy zgrupowane oraz niezgrupowane.

```

select Tytul, IDAutora, Cena
from KsiegarniaSchemat.Spis_Ksiazek
join KsiegarniaSchemat.Spis_Gatunkow on (IDGatunek = IDGatunku)
where Nazwa_Gatunku like 'fantasy'

```

145 %

Results Messages

	Tytul	IDAutora	Cena
1	Hary Potter	1	25.99
2	Manson	1	1.00
3	Łostatnio Chcica	7	67.00
4	Gladiola Losu	7	45.00
5	Jucha Goroli	7	567.00
6	Ajnfachowy Zait	7	56.00
7	Potka Łognia	7	245.00
8	Zomek Szwabika	7	12.00
9	Frelka Jeźra	7	45.00
10	Zait Blyskonio Sie	7	56.00
11	Hexer: Szpony i Pazury	7	455.00
12	Posledni přání	7	69.69
13	Meč osudu	7	50.35
14	Krev elfu	7	40.68
15	čas pohrdání	7	49.49
16	křest ohněm	7	49.57
17	vlaštovka	7	58.25
18	paní jezera	7	62.52
19	bouřková sezóna	7	36.54
20	Zaklínač drápy a drápy	7	52.25
21	Wiedźmin: Ostatnie ż...	7	35.00
22	Wiedźmin: Miecz prze...	7	35.00
23	Wiedźmin: Krew Elfów	7	35.00
24	Wiedźmin: Czas poga...	7	35.00
25	Wiedźmin: Chrzest og...	7	35.00
26	Wiedźmin: Wieża jas...	7	35.00
27	Wiedźmin: Pani jeziora	7	35.00
28	Wiedźmin: Sezon burz	7	35.00
29	Wiedźmin: Szpony i p...	7	35.00

```
= select IDZamowienia, Koszty, Walidacja  
from KsiegarniaSchemat.Spis_Zamowien  
where Walidacja = '0'
```

145 %

Results Messages

	IDZamowienia	Koszty	Walidacja
1	1	34.99	0
2	2	34.99	0
3	4	189.96	0
4	5	77.97	0
5	6	137.98	0

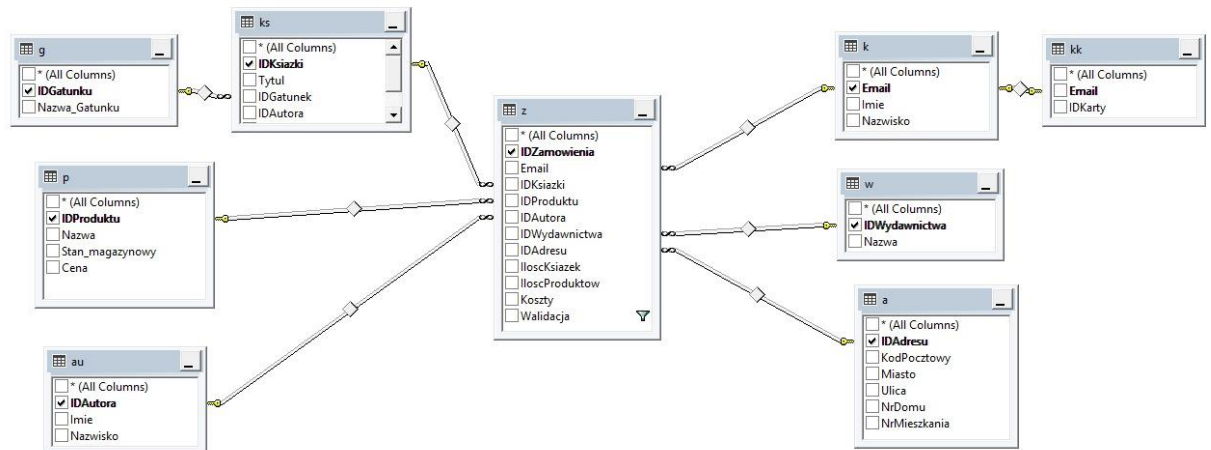
Indeksy zgrupowane są założone na każdej kolumnie, która jest Primary Key.

Przy migracji powinny być wyłączone wszystkie indeksy, oprócz jednego Clustered, który „trzyma” całą bazę. Ułatwi to i przyspieszy wykonywane zadanie. Po odtworzeniu bazy trzeba jednak pamiętać, by je włączyć.

Indeksy niezgrupowane zostały założone na kolumny Spis_Adresów.Miasta, Spis_Autorów.Nazwiska, Spis_Klientów.Nazwiska, Spis_Książek.Tytuł, Spis_Produktów.Nazwa, Spis_Zamówień.Walidacja, ponieważ działają one jak indeks haseł na końcu książki telefonicznej i ułatwiają segregację danych.

3) Zaprojektowane i stworzone widoki:

1)



2) create view Average_Price_Books as

select Tytuł, Cena

from KsiegarniaSchemat.Spis_Ksiazek

where Cena > (select avg(Cena) from KsiegarniaSchemat.Spis_Ksiazek)

Results		
	Tytuł	Cena
1	Jucha Goroli	567.00
2	Potka Łognia	245.00
3	Hexer: Szpony i Pazury	455.00

Spis_Ksiazek (Ksiegar...

☐ * (All Columns)
 ☒ IDKsiazek
 ☒ Tytul
 ☐ IDGatunek
 ☐ IDAutora

Column	Alias	Table	Outp...	Sort Type	Sort Order	Filter	Or...
Tytul		Spis_Ksiazek (Ksiegar...	<input checked="" type="checkbox"/>				
Cena		Spis_Ksiazek (Ksiegar...	<input checked="" type="checkbox"/>			> (SELECT AVG(Cena) AS Expr1 FROM Ksiegar...	
			<input type="checkbox"/>				
			<input type="checkbox"/>				
			<input type="checkbox"/>				
			<input type="checkbox"/>				

```

SELECT Tytul, Cena
FROM Ksiegar...Spis_Ksiazek
WHERE (Cena >
  (SELECT AVG(Cena) AS Expr1
   FROM Ksiegar...Spis_Ksiazek))

```

3) create view Average_Price_Products as
 select Nazwa, Cena
 from Ksiegar...Spis_Produktow
 where Cena > (select avg(Cena) from Ksiegar...Spis_Produktow)

Spis_Produktow (Ksieg...

☐ * (All Columns)
 ☐ IDProduktu
 ☒ Nazwa
 ☐ Stan_magazynowy
 ☒ Cena

Column	Alias	Table	Outp...	Sort Type	Sort Order	Filter	Or...
Nazwa		Spis_Produktow (Ksieg...	<input checked="" type="checkbox"/>				
Cena		Spis_Produktow (Ksieg...	<input checked="" type="checkbox"/>			> (SELECT AVG(Cena) AS Expr1 FROM Ksiegar...	
			<input type="checkbox"/>				
			<input type="checkbox"/>				
			<input type="checkbox"/>				

```

SELECT Nazwa, Cena
FROM Ksiegar...Spis_Produktow
WHERE (Cena >
  (SELECT AVG(Cena) AS Expr1
   FROM Ksiegar...Spis_Produktow))

```

Results		Messages
	Nazwa	Cena
1	Hexer 3 Gryfno Skupina	420.00
2	Gryfne Bity Jaskra	456.00

4) create view Books_View_By_Genre as

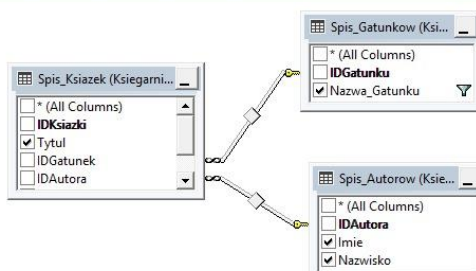
select Spis_Ksiazek.Tytul, Spis_Autorow.Imie, Spis_Autorow.Nazwisko,
Spis_Gatunkow.Nazwa_Gatunku

from KsiegarniaSchemat.Spis_Ksiazek

join KsiegarniaSchemat.Spis_Autorow on (Spis_Ksiazek.IDAutora =
Spis_Autorow.IDAutora)

join KsiegarniaSchemat.Spis_Gatunkow on (Spis_Ksiazek.IDGatunek =
Spis_Gatunkow.IDGatunku)

where Nazwa_Gatunku like 'fantasy'



Column	Alias	Table	Outp...	Sort Type	Sort Order	Filter	Or...	Or...	Or...
Tytul		Spis_Ksiazek (KsiegarniaSchemat)	<input checked="" type="checkbox"/>						
Imie		Spis_Autorow (KsiegarniaSchemat)	<input checked="" type="checkbox"/>						
Nazwisko		Spis_Autorow (KsiegarniaSchemat)	<input checked="" type="checkbox"/>						
Nazwa_Gatunku		Spis_Gatunkow (KsiegarniaSchemat)	<input checked="" type="checkbox"/>			LIKE 'fantasy'			

```

SELECT KsiegarniaSchemat.Spis_Ksiazek.Tytul, KsiegarniaSchemat.Spis_Autorow.Imie, KsiegarniaSchemat.Spis_Autorow.Nazwisko, KsiegarniaSchemat.Spis_Gatunkow.Nazwa_Gatunku
FROM KsiegarniaSchemat.Spis_Ksiazek INNER JOIN
KsiegarniaSchemat.Spis_Autorow ON KsiegarniaSchemat.Spis_Ksiazek.IDAutora = KsiegarniaSchemat.Spis_Autorow.IDAutora INNER JOIN
KsiegarniaSchemat.Spis_Gatunkow ON KsiegarniaSchemat.Spis_Ksiazek.IDGatunek = KsiegarniaSchemat.Spis_Gatunkow.IDGatunku
WHERE (KsiegarniaSchemat.Spis_Gatunkow.Nazwa_Gatunku LIKE 'fantasy')

```


Results		Messages		
	Tytul	Imie	Nazwisko	Nazwa_Gatunku
1	Hary Potter	J.K	Rowling	fantasy
2	Manson	J.K	Rowling	fantasy
3	Łostatnio Chcica	Sandrzej	Apkowski	fantasy
4	Gladiola Losu	Sandrzej	Apkowski	fantasy
5	Jucha Goroli	Sandrzej	Apkowski	fantasy
6	Ajnfachowy Zait	Sandrzej	Apkowski	fantasy
7	Potka Łognia	Sandrzej	Apkowski	fantasy
8	Zomek Szwabika	Sandrzej	Apkowski	fantasy
9	Frelka Jeźra	Sandrzej	Apkowski	fantasy
10	Zait Blyskonio Sie	Sandrzej	Apkowski	fantasy
11	Hexer: Szpony i Pazury	Sandrzej	Apkowski	fantasy
12	Poslední přání	Sandrzej	Apkowski	fantasy
13	Meč osudu	Sandrzej	Apkowski	fantasy
14	Krev elfu	Sandrzej	Apkowski	fantasy
15	čas pohrdání	Sandrzej	Apkowski	fantasy
16	křest ohněm	Sandrzej	Apkowski	fantasy
17	vlaštovka	Sandrzej	Apkowski	fantasy
18	paní jezera	Sandrzej	Apkowski	fantasy
19	bouřková sezóna	Sandrzej	Apkowski	fantasy
20	Zaklínač drápy a drápy	Sandrzej	Apkowski	fantasy

5) create view Books_View_By_Author as

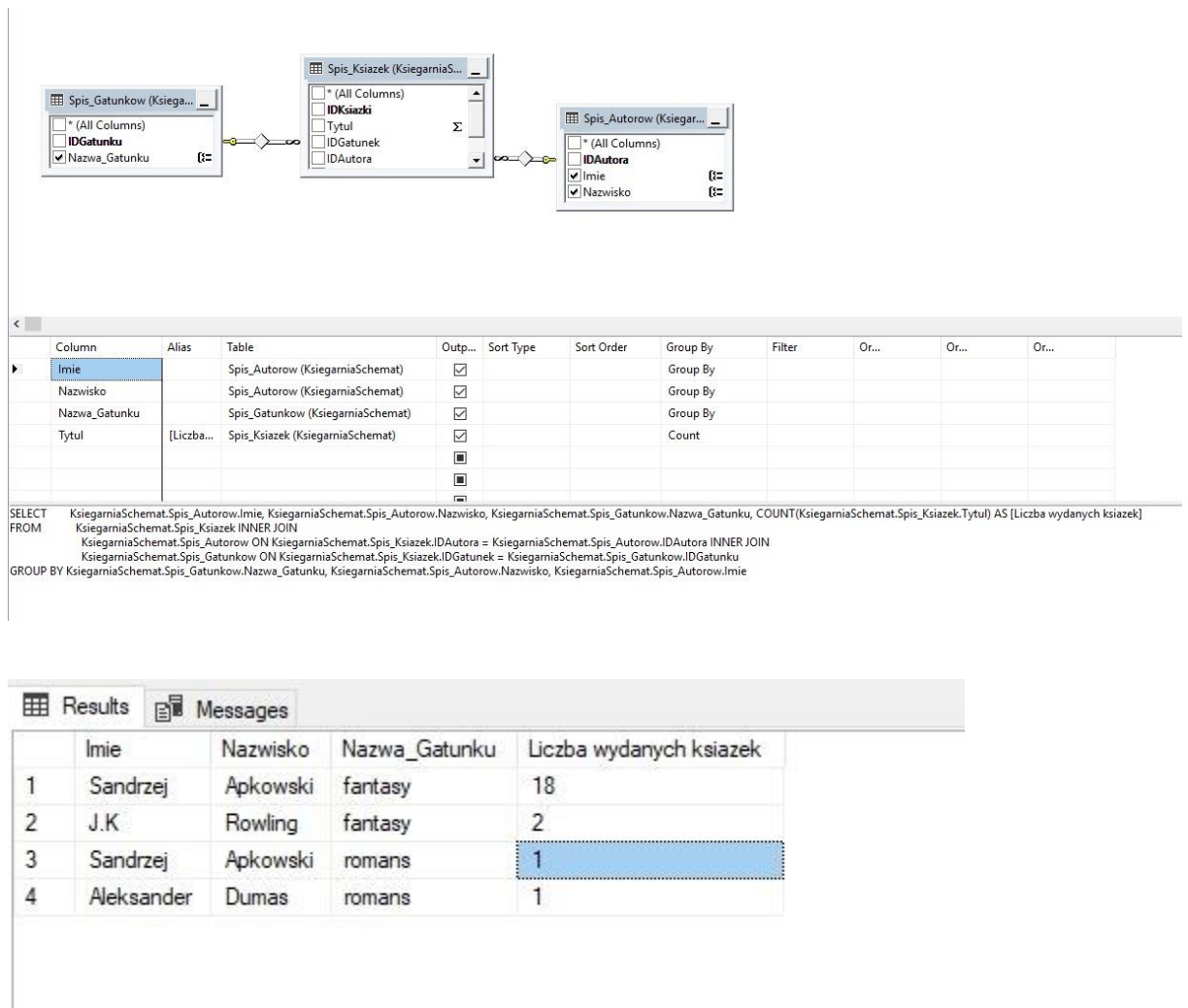
```
select Spis_Autorow.Imie, Spis_Autorow.Nazwisko,
Spis_Gatunkow.Nazwa_Gatunku, count(KsiegarniaSchemat.Spis_Ksiazek.Tytul) as
'Liczba wydanych ksiazek'
```

```
from KsiegarniaSchemat.Spis_Ksiazek
```

```
join KsiegarniaSchemat.Spis_Autorow on (Spis_Ksiazek.IDAutora =
Spis_Autorow.IDAutora)
```

```
join KsiegarniaSchemat.Spis_Gatunkow on (Spis_Ksiazek.IDGatunek =
Spis_Gatunkow.IDGatunku)
```

```
group by Nazwa_Gatunku, Nazwisko, Imie
```



The screenshot displays a SQL query execution in SQL Server Enterprise Manager. At the top, a query plan is visible, showing three tables: **Spis_Gatunkow** (Ksiegar...), **Spis_Ksiazek** (Ksiegar...), and **Spis_Autorow** (Ksiegar...). The tables are connected by inner joins. The **Spis_Ksiazek** table has columns **IDKsiazki**, **Tytul**, **IDGatunek**, and **IDAutora**. The **Spis_Autorow** table has columns **IDAutora**, **Imie**, and **Nazwisko**. The **Spis_Gatunkow** table has columns **IDGatunku** and **Nazwa_Gatunku**.

Below the query plan, the SQL query is shown:

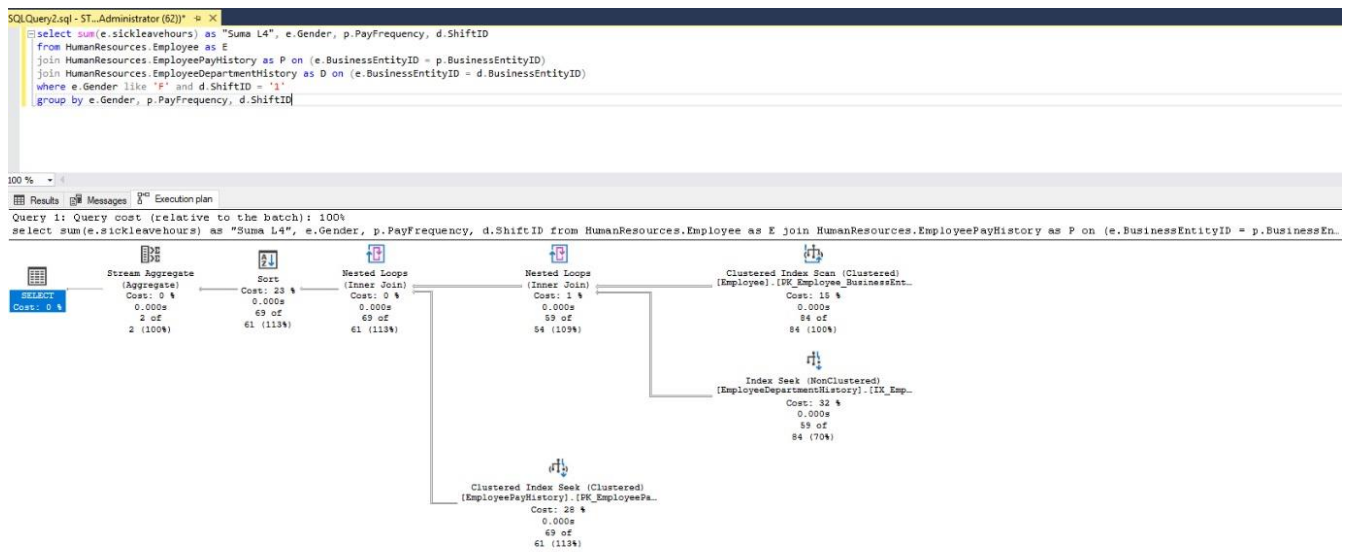
```
SELECT
  Ksiegar...Spis_Autorow.Imie, Ksiegar...Spis_Autorow.Nazwisko, Ksiegar...Spis_Gatunkow.Nazwa_Gatunku, COUNT(Ksiegar...Spis_Ksiazek.Tytul) AS [Liczba wydanych ksiazek]
FROM
  Ksiegar...Spis_Ksiazek INNER JOIN
  Ksiegar...Spis_Autorow ON Ksiegar...Spis_Ksiazek.IDAutora = Ksiegar...Spis_Autorow.IDAutora INNER JOIN
  Ksiegar...Spis_Gatunkow ON Ksiegar...Spis_Ksiazek.IDGatunek = Ksiegar...Spis_Gatunkow.IDGatunku
GROUP BY
  Ksiegar...Spis_Gatunkow.Nazwa_Gatunku, Ksiegar...Spis_Autorow.Nazwisko, Ksiegar...Spis_Autorow.Imie
```

The results of the query are displayed in a table with the following columns: **Imie**, **Nazwisko**, **Nazwa_Gatunku**, and **Liczba wydanych ksiazek**. The results are as follows:

	Imie	Nazwisko	Nazwa_Gatunku	Liczba wydanych ksiazek
1	Sandrzej	Apkowski	fantasy	18
2	J.K	Rowling	fantasy	2
3	Sandrzej	Apkowski	romans	1
4	Aleksander	Dumas	romans	1

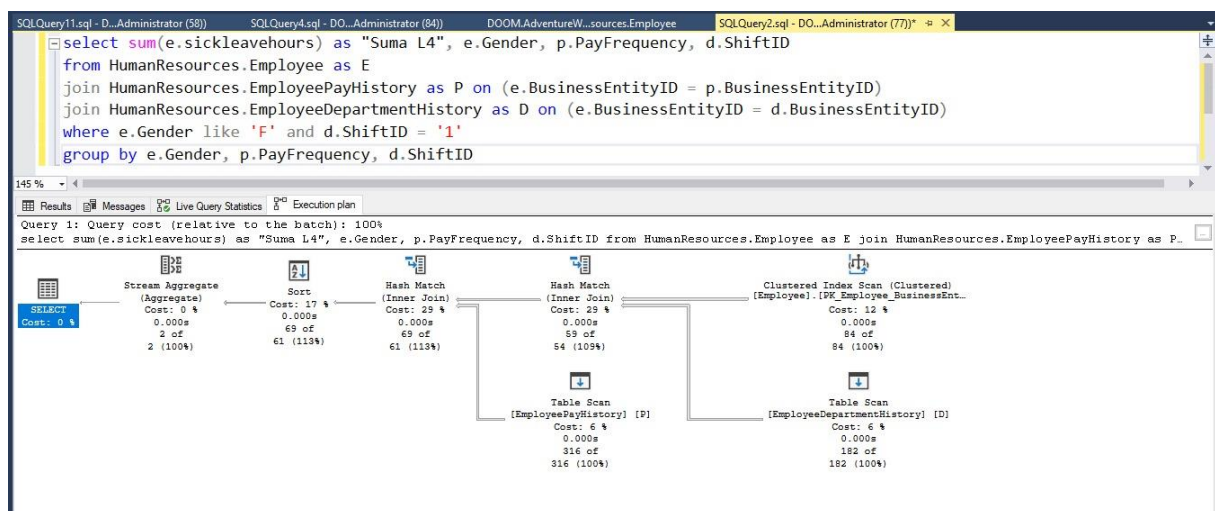
4. Sprawdzenie poprawności zastosowanych indeksów jest niemożliwe, ponieważ tabela posiada za mało rekordów. Została więc do tego celu użyta baza AdventureWorks2019, która posiada dużo więcej danych i różnica pomiędzy bazą z zaimplementowanymi indeksami, a bez będzie dużo bardziej widoczna.

```
select sum(e.sickleavehours) as "Suma L4", e.Gender, p.PayFrequency, d.ShiftID
from HumanResources.Employee as E
join HumanResources.EmployeePayHistory as P on (e.BusinessEntityID =
p.BusinessEntityID)
join HumanResources.EmployeeDepartmentHistory as D on (e.BusinessEntityID =
d.BusinessEntityID)
where e.Gender like 'F' and d.ShiftID = '1'
group by e.Gender, p.PayFrequency, d.ShiftID;
```



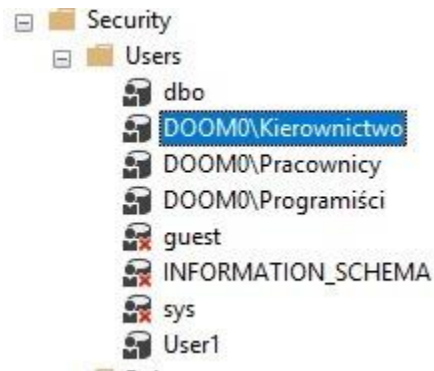
Od prawej strony widać, że używane są indeksy zgrupowane, dzięki czemu wyszukiwanie jest przebiega szybciej. Nested Loops pokazuje prawidłowe użycie indeksów, najpierw z jednego złączenia tabel, potem z drugiego. Operacja sort sortuje elementy. Jeżeli zapytanie zostałoby wykonywane bez uprzednio założonego indeksu Clustered, który porządkuje rekordy w kolumnie jak w książce telefonicznej, to cała tabela musiałaby zostać przeszukana wielokrotnie, co jest bardzo nieopłacalne.

Estymowany koszt operacji dla Sortowania wynosi 23%, co dowodzi, że indeksy zgrupowane wzięły na siebie cały koszt wyszukiwania. Ważne jest też użycie przez silnik NestedLoops (po 1%), które mówią, że dane są posortowane.



Jest użyty tylko jeden indeks zgrupowany, ponieważ usunięcie go jest niemożliwe. Pozbyto się jednak indeksów zgrupowanych dla EmployeeDepartmentHistory i EmployeePayHistory, dlatego na załączonym obrazku widać skanowanie całych tabel z lewej strony - jest to bardzo nieoptymalne. Widoczne na screenie HashMatch dowodzą, że dane nie są posortowane. Dodatkowe sortowanie wynosi 17% (mniej niż w poprzednim przypadku), ponieważ cały koszt operacji został przeniesiony na HashMatch i skanowanie tabel.

5. Użytkownicy oraz ich uprawnienia.

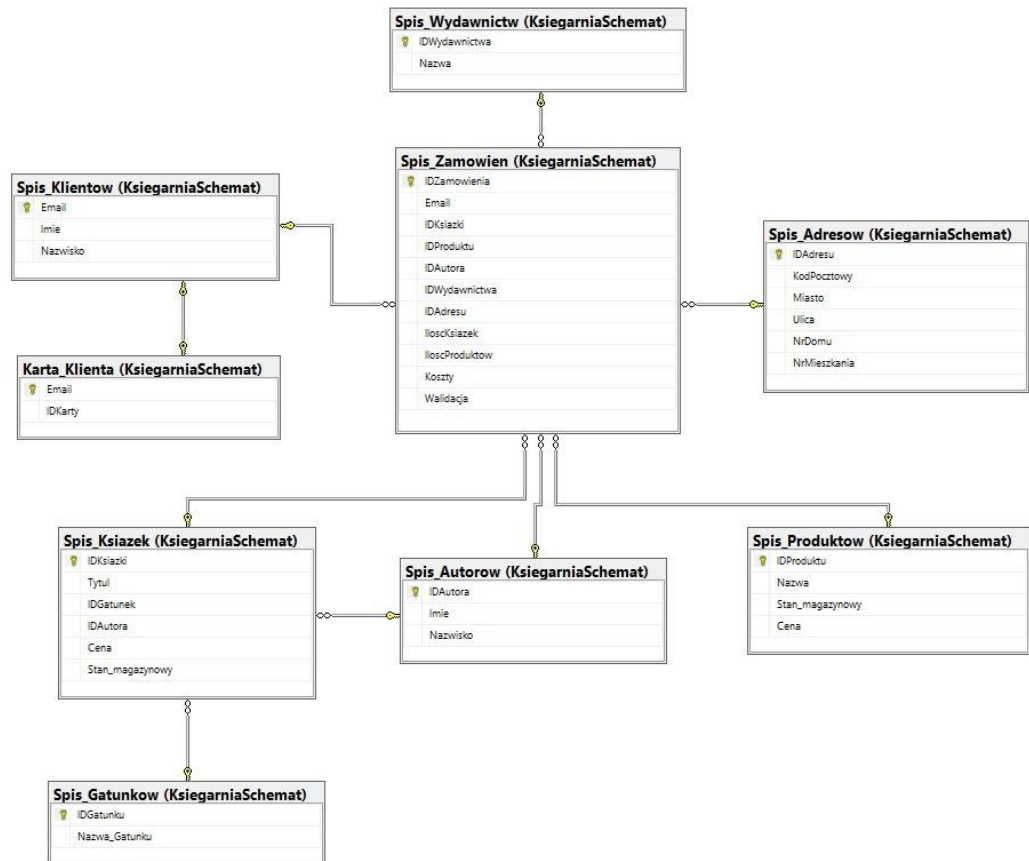


Grupa Pracownicy otrzymała uprawnienia select (datareader i datawriter), które pozwalają na wykorzystywanie procedur oraz modyfikację danych w tabelach.

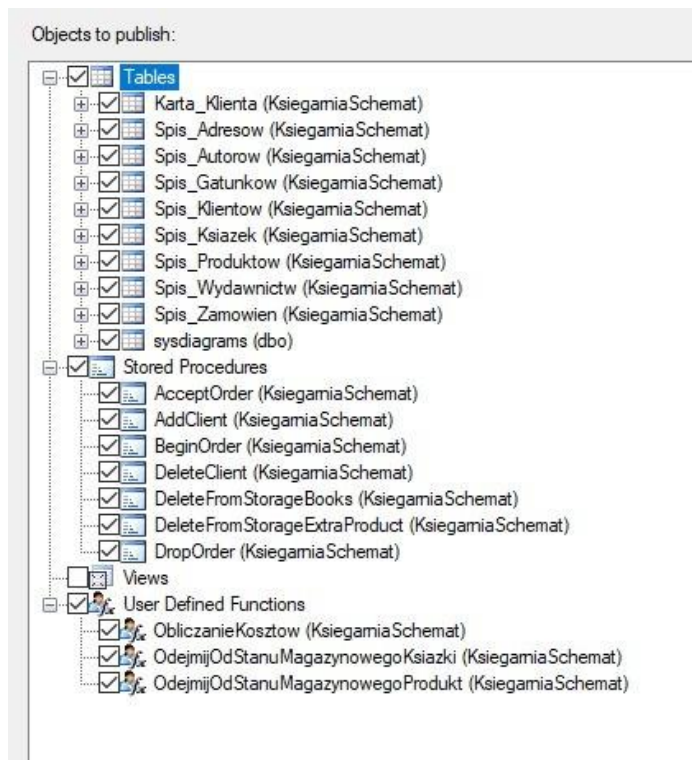
Grupa Informatycy: select, update, delete, write. Posiadają backupoperator, ddladmin, datareader i datawriter, ponieważ mogą zarządzać bazą, jej backupami oraz replikacją.

Kierownictwo ma te same uprawnienia co pracownicy.

6. Diagram całego projektu.



7. Replikowany fragment bazy dla oddziału Księgarni.



W replikowanym fragmencie dodane zostały wszystkie tabele, aby można było obsługiwać zamówienia. Typ wykonanej replikacji to przyrostowa – zapewnia ona aktualizację zmienionych danych na głównym serwerze.

Replikowany fragment bazy otrzymał następujące funkcjonalności:

Dodane procedury i funkcje: AddClient, BeginOrder, AcceptOrder, DeleteClient, DropOrder, DeleteFromStorageBooks, DeleteFromStorageProduct i ObliczanieKosztów, OdejmijOdStanuMagazynowegoProduktu, OdejmijOdStanuMagazynowegoKsiążki.

Dodano grupy użytkowników: Kierownictwo i Pracownicy.

W tym oddziale pracownicy muszą mieć możliwość modyfikacji danych klienta oraz każdego zamówienia. Z powodu możliwości sprzedaży produktów oraz książek muszą mieć również możliwość odejmowania ich ilości ze stanów magazynowych.

Codzienna kopia przyrostowa gwarantuje walidację danych. Dodane są funkcje wspomagające działanie procedur.

8. Użyte procedury oraz funkcje.

Procedury:

- 1) AddBook (Tytuł, IDAutora, IDAutora, Cena, StanMagazynowy) = Dodawanie nowych książek do bazy. Sprawdzenie czy dany tytuł już znajduje się w bazie.

```
ALTER PROCEDURE [KsiegarniaSchemat].[AddBook] (@Tytul nvarchar(100), @IDGatunek int, @IDAutora int, @Cena decimal(18,2), @Stan_magazynowy int) as
```

```
begin
```

```
if(@Tytul in ( select Tytul from KsiegarniaSchemat.Spis_Ksiazek)) print ('Podany tytul juz istnieje')
```

```
else if(@IDGatunek is null) print ('Gatunek nie moze byc pusty')
```

```
else if(@IDAutora is null) print ('Autor nie moze byc pusty')
```

```
else if(@Cena is null) print ('Cena nie moze byc pusta')
```

```
else if(@Stan_magazynowy is null) print ('Stan magazynowy nie moze byc pusty')
```

```
else insert into KsiegarniaSchemat.Spis_Ksiazek(Tytul, IDGatunek, IDAutora, Cena, Stan_magazynowy) values (@Tytul, @IDGatunek, @IDAutora, @Cena, @Stan_magazynowy);
```

```
end;
```

- 2) AddAuthor (Imię, Nazwisko) = Dodawanie nowego autora do bazy. Sprawdzenie czy dany autor znajduje się w bazie.

```
ALTER PROCEDURE[KsiegarniaSchemat].[AddAuthor] (@Imie varchar(50), @Nazwisko
varchar(50)) as
```

```
begin
```

```
if(@Imie is null) print ('Imie nie moze byc puste')
```

```
if(@Nazwisko is null) print ('Nazwisko nie moze byc puste')
```

```
--if(concat ('@Imie', ' ', '@Nazwisko') in (select concat ('Imie',' ','Nazwisko') from
KsiegarniaSchemat.Spis_Autorow )) print ('W bazie juz jest taki autor')
```

```
if(@Imie in (select Imie from KsiegarniaSchemat.Spis_Autorow) AND @Nazwisko in (select
Nazwisko from KsiegarniaSchemat.Spis_Autorow)) print ('jest juz')
```

```
else insert into KsiegarniaSchemat.Spis_Autorow(Imie, Nazwisko) values (@Imie, @Nazwisko);
```

```
end
```

3) AddSupplier(Nazwa) = Dodawanie nowego wydawnictwa do bazy. Sprawdzenie czy dane wydawnictwo znajduje się w bazie.

```
ALTER PROCEDURE [KsiegarniaSchemat].[AddSupplier] (@Nazwa nvarchar(50))
```

```
AS
```

```
BEGIN
```

```
if(@Nazwa in (select Nazwa from KsiegarniaSchemat.Spis_Wydawnictw)) print ('Podane
wydawnictwo znajduje sie w bazie')
```

```
else insert into KsiegarniaSchemat.Spis_Wydawnictw(Nazwa) values (@Nazwa);
```

```
END
```

4) AddClient (Email, Imię, Nazwisko) = Dodawanie nowego klienta do bazy. Sprawdzenie czy dany klient znajduje się w bazie. Dodawanie KartyKlienta z pseudolosową wartością jako IDKarty.

```
ALTER PROCEDURE [KsiegarniaSchemat].[AddClient] (@Email nvarchar(100), @Imie varchar(50),
@Nazwisko varchar(50))
```

```
as
```

```
BEGIN
```

```
declare @IDKarty int;
```

```
DECLARE @Upper INT;
```

```
DECLARE @Lower INT;
```

```
SET @Lower = 10000 ---- The lowest random number
```

```
SET @Upper = 99999 ---- The highest random number
```

```
SELECT @IDKarty = ROUND(((@Upper - @Lower - 1) * rndResult + @Lower), 0)
```

```
from rndView;
```

```

if(@IDKarty in (select IDKarty from KsiegarniaSchemat.Karta_Klienta)) EXEC
KsiegarniaSchemat.AddClient @Email,@Imie,@Nazwisko;

if(@Email in (select email from KsiegarniaSchemat.Spis_Klientow)) print ('Podany mail znajduje sie w
bazie')

if(@Imie in (select Imie from KsiegarniaSchemat.Spis_Klientow) AND

@Nazwisko in (select Nazwisko from KsiegarniaSchemat.Spis_Klientow)) print ('Istnieje juz klient o
tych samych danych ')

else insert into KsiegarniaSchemat.Spis_Klientow(Email, Imie, Nazwisko) values (@Email, @Imie,
@Nazwisko)

insert into KsiegarniaSchemat.Karta_Klienta(Email, IDKarty) values (@Email, @IDKarty);

```

END

5) AddAddress (KodPocztowy, Miasto, Ulica, NrDomu, NrMieszkania) =
Dodawanie nowego adresu do bazy. Sprawdzenie czy dany adres już znajduje się
w bazie.

```

ALTER PROCEDURE [KsiegarniaSchemat].[AddAddress] (@KodPocztowy nchar(5), @Miasto
varchar(100),

@Ulica nvarchar(50),@NrDomu nchar(10), @NrMieszkania nchar(10))

AS

BEGIN

if(@KodPocztowy in (select @KodPocztowy from KsiegarniaSchemat.Spis_Adresow) AND @Miasto
in (select Miasto from KsiegarniaSchemat.Spis_Adresow)

AND @Ulica in (select Ulica from KsiegarniaSchemat.Spis_Adresow)

AND @NrDomu in (select NrDomu from KsiegarniaSchemat.Spis_Adresow)

AND @NrMieszkania in (select NrMieszkania from KsiegarniaSchemat.Spis_Adresow)) print ('Adres
juz jest w bazie')

insert into KsiegarniaSchemat.Spis_Adresow(KodPocztowy, Miasto, Ulica, NrDomu, NrMieszkania)

values (@KodPocztowy, @Miasto, @Ulica, @NrDomu, @NrMieszkania)

END

```

6) AddGenre (NazwaGatunku) = Dodawanie nowego gatunku do bazy. Sprawdzenie
czy dany gatunek znajduje się w bazie.

```

ALTER PROCEDURE[KsiegarniaSchemat].[AddGenre] (@Nazwa_Gatunku varchar(50)) as

begin

if(@Nazwa_Gatunku is null) print ('Gatunek nie moze byc pusty')

else if(@Nazwa_Gatunku in (select Nazwa_Gatunku from KsiegarniaSchemat.Spis_Gatunkow)) print
('Podany gatunek jest juz w bazie')

else insert into KsiegarniaSchemat.Spis_Gatunkow(Nazwa_Gatunku) values (@Nazwa_Gatunku);

end

```


- 7) AddExtraProduct (Nazwa, StanMagazynowy, Cena) = Dodawanie nowego produktu do bazy. Sprawdzenie czy dany produkt znajduje się w bazie.

```
ALTER PROCEDURE [KsiegarniaSchemat].[Add.ExtraProduct](@Nazwa nvarchar(50),  
@Stan_magazynowy int, @Cena decimal(18,2))
```

```
AS
```

```
BEGIN
```

```
if(@Nazwa in (select Nazwa from KsiegarniaSchemat.Spis_Produktow)) print ('Jest juz taki produkt w  
bazie')
```

```
else insert into KsiegarniaSchemat.Spis_Produktow(Nazwa, Stan_magazynowy, Cena)  
values(@Nazwa, @Stan_magazynowy, @Cena);
```

```
END
```

- 8) DodajKartę - nie ma tej procedury, znajduje się w procedurze AddClient.

- 9) BeginOrder (Email, IDksiążki, IDautora, IDwydawnictwa, IDadresu, Ilosc_produktow, Ilość_ksiazek) = Dodawanie nowego niezatwierdzonego zamówienia do bazy. Sprawdzenie czy wszystkie dane są poprawne. Wywołanie Funkcji ObliczanieKosztow(), ale obliczyć koszty zamówienia. Po wykonaniu ustawiamy Validacja = 0.

```
ALTER procedure [KsiegarniaSchemat].[BeginOrder] (@Email nvarchar(100), @IDKsiazki int,  
@IDProduktu int, @IDAutora int,
```

```
@IDWydawnictwa int, @IDAdresu int, @Ilosc_ksiazek int, @Ilosc_produktow int)
```

```
as
```

```
begin
```

```
declare @Koszty decimal (18,2);
```

```
set @Koszty = (select KsiegarniaSchemat.ObliczanieKosztow(@IDProduktu, @IDKsiazki,  
@Ilosc_ksiazek, @Ilosc_produktow));
```

```
if(@Email is null or @Email not in (select Email from KsiegarniaSchemat.Spis_Klientow))  
print('Nieprawidłowy Email klienta');
```

```
else if(@IDKsiazki not in (select IDKsiazki from KsiegarniaSchemat.Spis_Ksiazek))  
print('Nieprawidłowe Id książki');
```

```
else if(@IDProduktu not in (select @IDProduktu from KsiegarniaSchemat.Spis_Produktow)) print  
('Nieprawidłowe ID produktu');
```

```
else if(@IDAutora not in (select @IDAutora from KsiegarniaSchemat.Spis_Autorow)) print('Brak  
autora o takim ID');
```

```
else if(@IDWydawnictwa not in (select IDWydawnictwa from KsiegarniaSchemat.Spis_Wydawnictw))  
print('Nieprawidłowe ID wydawnictwa');
```

```
else if(@IDAdresu is null or @IDAdresu not in (select IDAdresu from  
KsiegarniaSchemat.Spis_Adresow)) print('Nieprawidłowy adres');
```

```
else if (@Ilosc_ksiazek is null or @Ilosc_ksiazek > (select Stan_magazynowy from  
KsiegarniaSchemat.Spis_Ksiazek where @IDKsiazki = IDKsiazki))
```

```
print ('Nieprawidłowa ilość książek');
```

```

else if(@Ilosc_produkow is null or @Ilosc_produkow > (select Stan_magazynowy from
KsiegarniaSchemat.Spis_Produktow where @IDProduktu = IDProduktu))

print ('Nieprawidlowa ilosc produktow');

else if(@Koszty = 0) print ('Nie podano zadnych produktow');

else insert into KsiegarniaSchemat.Spis_Zamowien(Email, IDKsiazki, IDProduktu, IDAutora,
IDWydawnictwa, IDAdresu, IloscKsiazek, IloscProduktow, Koszty, Walidacja)

values (@Email, @IDKsiazki, @IDProduktu, @IDAutora, @IDWydawnictwa, @IDAdresu,
@Ilosc_ksiazek, @Ilosc_produkow,@Koszty, '0');

end;

```

10) AcceptOrder (IDzamówienia) = Jeżeli dane są poprawne ustawia Walidacja = 1.

```

ALTER PROCEDURE [KsiegarniaSchemat].[AcceptOrder] (@IDZamowienia int)

AS

BEGIN

    if (@IDZamowienia is null or @IDZamowienia not in (select IDZamowienia from
KsiegarniaSchemat.Spis_Zamowien)) print ('Brak zamowienia o takim ID')

    else update KsiegarniaSchemat.Spis_Zamowien set Walidacja = '1' where
KsiegarniaSchemat.Spis_Zamowien.IDZamowienia = @IDZamowienia;

END

```

11) DeleteBook (IDksiążki). = Sprawdza czy produkt jest w bazie i usuwa go.

```

ALTER procedure [KsiegarniaSchemat].[DeleteBook](@IDKsiazki int)

as

begin

if (@IDKsiazki not in (select IDKsiazki from KsiegarniaSchemat.Spis_Ksiazek)) print('Podany produkt
nie istnieje w bazie ');

else if (@IDKsiazki is null) print('Nie podano ID produktu');

else delete from KsiegarniaSchemat.Spis_Ksiazek where (KsiegarniaSchemat.Spis_Ksiazek.IDKsiazki
= @IDKsiazki)

end;

```

12) DeleteGenre(IDGatunku) = Sprawdza czy produkt jest w bazie i usuwa go.

```

ALTER procedure [KsiegarniaSchemat].[DeleteGenre] (@IDGatunku int)

as

begin

if (@IDGatunku not in (select IDGatunku from KsiegarniaSchemat.Spis_Gatunkow)) print('Podany
gatunek nie istnieje w bazie ');

else if (@IDGatunku is null) print('Nie podano gatunku produktu');

```

```
else delete from KsiegarniaSchemat.Spis_Gatunkow where  
(KsiegarniaSchemat.Spis_Gatunkow.IDGatunku = @IDGatunku) end;
```

13) DeleteAuthor(IDautora) = Sprawdza czy produkt jest w bazie i usuwa go.

```
ALTER procedure [KsiegarniaSchemat].[DeleteAuthor] (@IDAutora int)
```

```
as
```

```
begin
```

```
if (@IDAutora not in (select IDAutora from KsiegarniaSchemat.Spis_Autorow)) print('Podany autor  
nie istnieje w bazie ');
```

```
else if (@IDAutora is null) print('Nie podano autora produktu');
```

```
else delete from KsiegarniaSchemat.Spis_Autorow where (KsiegarniaSchemat.Spis_Autorow.IDAutora  
= @IDAutora) end;
```

14) DeleteSupplier(IDwydawnictwa) = Sprawdza czy produkt jest w bazie i usuwa go.

```
ALTER procedure [KsiegarniaSchemat].[DeleteSupplier] (@IDWydawnictwa int)
```

```
as
```

```
begin
```

```
if (@IDWydawnictwa not in (select IDWydawnictwa from KsiegarniaSchemat.Spis_Wydawnictw))  
print('Podane wydawnictwo nie istnieje w bazie ');
```

```
else if (@IDWydawnictwa is null) print('Nie podano wydawnictwa');
```

```
else delete from KsiegarniaSchemat.Spis_Wydawnictw where  
(KsiegarniaSchemat.Spis_Wydawnictw.IDWydawnictwa= @IDWydawnictwa) end;
```

15) DeleteClient (Email) = Sprawdza czy klient jest w bazie i usuwa go, wraz z jego KartąKlienta.

```
ALTER procedure [KsiegarniaSchemat].[DeleteClient] (@Email nvarchar(100))
```

```
as
```

```
begin
```

```
if (@Email not in (select Email from KsiegarniaSchemat.Spis_Klientow)) print('Podany klient nie  
istnieje w bazie ');
```

```
else if (@Email is null) print('Nie podano klienta');
```

```
else delete from KsiegarniaSchemat.Karta_Klienta where (KsiegarniaSchemat.Karta_Klienta.Email=  
@Email)
```

```
delete from KsiegarniaSchemat.Spis_Klientow where (KsiegarniaSchemat.Spis_Klientow.Email=  
@Email)
```

```
end;
```

16) DeleteAddress(IDAdresu) = Sprawdza czy produkt jest w bazie i usuwa go.

```
ALTER procedure [KsiegarniaSchemat].[DeleteAddress] (@IDAdresu int)
```

as

begin

if (@IDAdresu not in (select IDAdresu from KsiegarniaSchemat.Spis_Adresow)) print('Podany adres nie istnieje w bazie ');

else if (@IDAdresu is null) print('Nie podano adresu ');

else delete from KsiegarniaSchemat.Spis_Adresow where
(KsiegarniaSchemat.Spis_Adresow.IDAdresu= @IDAdresu) end;

17) DropOrder(IDzamówienia) = Sprawdza czy zamówienie jest w bazie i usuwa je.

ALTER PROCEDURE [KsiegarniaSchemat].[DropOrder](@IDZamowienia int)

AS

BEGIN

if(@IDZamowienia is null) print('Nie ma takiego zamowienia');

else if(@IDZamowienia not in (select IDZamowienia from KsiegarniaSchemat.Spis_Zamowien)) print
('Nie ma takiego zamówienia w bazie');

else delete from KsiegarniaSchemat.Spis_Zamowien where
KsiegarniaSchemat.Spis_Zamowien.IDZamowienia = @IDZamowienia;

END

18) DeleteExtraProduct (IDproduktu) = Sprawdza czy produkt jest w bazie i usuwa go.

ALTER procedure [KsiegarniaSchemat].[DeleteExtraProduct] (@IDProduktu int)

as

begin

if (@IDProduktu not in (select IDProduktu from KsiegarniaSchemat.Spis_Produktow)) print('Podany produkt nie istnieje w bazie ');

else if (@IDProduktu is null) print('Nie podano produktu');

else delete from KsiegarniaSchemat.Spis_Produktow where
(KsiegarniaSchemat.Spis_Produktow.IDProduktu= @IDProduktu) end;

19) UsunKartę (Email) = zaimplementowane w procedurze DeleteClient.

20) AddToStorageBooks (@IDKsiazki, @IleDodajemy) = w środku została wywołana funkcja DodajDoStanuMagazynowegoKsiazki(), sprawdzone zostaje czy posiadamy takie IDKsiazki, aktualizowana jest również wartość w SpisKsiazek.Stan_Magazynowy.

ALTER PROCEDURE [KsiegarniaSchemat].[AddToStorageBooks] (@IDKsiazki int, @IleDodajemy int)

AS

BEGIN

```

declare @Nowallosc int;

set @Nowallosc = (select KsiegarniaSchemat.DodajDoStanuMagazynowegoKsiazki
(@IDKsiazki, @IleDodajemy));

if (@IDKsiazki is null or @IDKsiazki not in
(select IDKsiazki from KsiegarniaSchemat.Spis_Ksiazek))

print ('Brak książki o takim ID');

else if (@IleDodajemy is null) print ('Nieprawidłowa ilość dodawanych książek');

else update KsiegarniaSchemat.Spis_Ksiazek set Stan_magazynowy = @Nowallosc where
IDKsiazki = @IDKsiazki;

END

```

21) AddToStorageExtraProduct (@IDproduktu, @IleDodajemy) = w środku została wywołana funkcja DodajDoStanuMagazynowegoProduktu(), sprawdzone zostaje czy posiadamy takie IDproduktu, aktualizowana jest również wartość w SpisProduktow.Stan_Magazynowy.

```

ALTER PROCEDURE [KsiegarniaSchemat].[AddToStorageExtraProduct] (@IDProduktu int,
@IleDodajemy int)

AS

BEGIN

declare @Nowallosc int;

set @Nowallosc = (select KsiegarniaSchemat.DodajDoStanuMagazynowegoProduktu
(@IDProduktu, @IleDodajemy));

if (@IDProduktu is null or @IDProduktu not in
(select IDProduktu from KsiegarniaSchemat.Spis_Produktow))

print ('Brak produktu o takim ID');

else if (@IleDodajemy is null) print ('Nieprawidłowa ilość dodawanych produktow');

else update KsiegarniaSchemat.Spis_Produktow set Stan_magazynowy = @Nowallosc where
IDProduktu = @IDProduktu;

END

```

22) DeleteFromStorageBooks(@IDKsiazki, @IleOdjac) = w środku została wywołana funkcja OdejmijOdStanuMagazynowegoKsiazki(), sprawdzone zostaje czy posiadamy takie IDKsiazki, aktualizowana jest wartość w SpisKsiazek.Stan_Magazynowy. Jeżeli zwracana z funkcji wartość '-1' pojawia się komunikat "Nieprawidłowa ilość". DeleteFromStorageProduct(@IDproduktu, @IleOdjac) = w środku została wywołana funkcja OdejmijOdStanuMagazynowegoProduktu(), sprawdzone zostaje czy posiadamy konkretne IDproduktu, aktualizowana jest wartość w SpisProduktow.Stan_Magazynowy. Jeżeli zwrócona zostanie z funkcji wartość '-1' to pojawia się komunikat "Nieprawidłowa ilość".

```

ALTER PROCEDURE [KsiegarniaSchemat].[DeleteFromStorageBooks] (@IDksiazki int, @IleOdjac
int)

AS

```

BEGIN

```
declare @Nowallosc int;

set @Nowallosc = (select KsiegarniaSchemat.OdejmijOdStanuMagazynowegoKsiazki
(@IDksiazki,@IleOdjac));

if (@IDksiazki is null or @IDksiazki not in (select IDKsiazki from
KsiegarniaSchemat.Spis_Ksiazek)) print ('Brak ksiazki o takim ID')

else if(@IleOdjac is null) print ('Nieprawidlowa ilosc')

else if (@Nowallosc = '-1') print ('Nieprawidlowa ilosc')

else update KsiegarniaSchemat.Spis_Ksiazek set Stan_magazynowy = @Nowallosc where
IDKsiazki = @IDksiazki;
```

END

23) DeleteFromStorageExtraProduct (@IdProduktu) = w środku została wywołana funkcja OdejmijOdStanuMagazynowegoProdukt(), sprawdzone zostaje czy posiadamy takie IDProduktu, aktualizowana jest wartość w SpisProduktow.Stan_Magazynowy. Jeżeli zwracana z funkcji wartość '-1' pojawia się komunikat "Nieprawidłowa ilość". DeleteFromStorageProduct(@IDproduktu, @IleOdjac) = w środku została wywołana funkcja OdejmijOdStanuMagazynowegoProdukt(), sprawdzone zostaje czy posiadamy konkretne IDproduktu, aktualizowana jest wartość w SpisProduktow.Stan_Magazynowy. Jeżeli zwrócona zostanie z funkcji wartość '-1' to pojawia się komunikat "Nieprawidłowa ilość".

```
ALTER PROCEDURE [KsiegarniaSchemat].[DeleteFromStorageExtraProduct] (@IDProduktu int,
@IleOdjac int)
```

AS

BEGIN

```
declare @Nowallosc int;

set @Nowallosc = (select KsiegarniaSchemat.OdejmijOdStanuMagazynowegoProdukt
(@IDProduktu ,@IleOdjac));

if (@IDProduktu is null or @IDProduktu not in (select IDKsiazki from
KsiegarniaSchemat.Spis_Ksiazek)) print ('Brak ksiazki o takim ID')

else if(@IleOdjac is null) print ('Nieprawidlowa ilosc')

else if (@Nowallosc = '-1') print ('Nieprawidlowa ilosc')

else update KsiegarniaSchemat.Spis_Produktow set Stan_magazynowy = @Nowallosc where
IDProduktu = @IDProduktu;
```

END

Funkcje:

24) DodajDoStanuMagazynowegoKsiazki (@IDKsiazki, @IleDodajemy) = dodaje do obecnego stanu magazynowego konkretną ilość. Połączona z procedurą AddToStorageBooks.

```
ALTER FUNCTION [KsiegarniaSchemat].[DodajDoStanuMagazynowegoKsiazki] (@IDKsiazki int, @IleDodac int)
```

```
RETURNS int
```

```
AS
```

```
BEGIN
```

```
    declare @IloscAktualna int;
```

```
    set @IloscAktualna = (select Stan_magazynowy from
```

```
    KsiegarniaSchemat.Spis_Ksiazek where KsiegarniaSchemat.Spis_Ksiazek.IDKsiazki
```

```
    = @IDKsiazki);
```

```
    declare @NowaIlosc int;
```

```
    set @NowaIlosc = @IloscAktualna + @IleDodac;
```

```
    return @NowaIlosc;
```

```
END
```

25) DodajDoStanuMagazynowegoProduktu (@IDproduktu, @IleDodajemy) =dodaje do obecnego stanu magazynowego konkretną ilość. Połączona z procedurą AddToStorageExtraProduct.

```
ALTER FUNCTION [KsiegarniaSchemat].[DodajDoStanuMagazynowegoProduktu] (@IDProduktu int, @IleDodac int)
```

```
RETURNS int
```

```
AS
```

```
BEGIN
```

```
    declare @IloscAktualna int;
```

```
    set @IloscAktualna = (select Stan_magazynowy from
```

```
    KsiegarniaSchemat.Spis_Produktow where KsiegarniaSchemat.Spis_Produktow.IDProduktu
```

```
    = @IDProduktu);
```

```
    declare @NowaIlosc int;
```

```
    set @NowaIlosc = @IloscAktualna + @IleDodac;
```

```
    return @NowaIlosc;
```

```
END
```

26) OdejmijOdStanuMagazynowegoKsiazki (@IDksiazki, @IleOdejmuje) = odejmuje od obecnego stanu magazynowego konkretną ilość. Połączona z procedurą DeleteFromStorageBooks. Zabezpieczenie, które zwraca jest wartość '-1'. Pojawia się, gdy ilość książek na magazynie jest mniejsza niż ilość do odjęcia.

```
ALTER FUNCTION [KsiegarniaSchemat].[OdejmijOdStanuMagazynowegoKsiazki] (@IDKsiazki int, @IleOdjac int)
```

RETURNS int

AS

BEGIN

```
declare @IloscAktualna int;

set @IloscAktualna = (select Stan_magazynowy from
KsiegarniaSchemat.Spis_Ksiazek where KsiegarniaSchemat.Spis_Ksiazek.IDKsiazki
= @IDKsiazki);

declare @Nowallosc int;

if( @IloscAktualna < @IleOdjac) set @Nowallosc = -1;

else set @Nowallosc = @IloscAktualna - @IleOdjac;

return @Nowallosc;
```

END

27) OdejmijOdStanuMagazynowegoProduktu (@IDproduktu, @IleOdejmuje) = odejmuje od obecnego stanu magazynowego konkretną ilość. Połączona z procedurą DeleteFromStorageProduct. Zabezpieczenie, które zwraca jest wartość '-1'. Pojawia się, gdy ilość produktów na magazynie jest mniejsza niż ilość do odjęcia.

ALTER FUNCTION [KsiegarniaSchemat].[OdejmijOdStanuMagazynowegoProdukt] (@IDProduktu int, @IleOdjac int)

RETURNS int

AS

BEGIN

```
declare @IloscAktualna int;

set @IloscAktualna = (select Stan_magazynowy from
KsiegarniaSchemat.Spis_Produktow where KsiegarniaSchemat.Spis_Produktow.IDProduktu
= @IDProduktu);

declare @Nowallosc int;

if( @IloscAktualna < @IleOdjac) set @Nowallosc = -1;

else set @Nowallosc = @IloscAktualna - @IleOdjac;

return @Nowallosc;
```

END

28) ObliczanieKosztow (@IDproduktu, @IDksiążki, @Ilosc_produktow, @Ilosc_Ksiazek) = oblicza sumę kosztów zamówienia i zostaje ta suma przekazana do procedury.

ALTER FUNCTION [KsiegarniaSchemat].[ObliczanieKosztow] (@IDProduktu int, @IDKsiazki int, @Ilosc_ksiazek int, @Ilosc_produktow int)

returns decimal(18,2)


```

AS
BEGIN

declare @ksiazka_suma decimal(18,2);

declare @produkt_suma decimal(18,2);

SET @ksiazka_suma = 0;

SET @produkt_suma = 0;

if(@IDProduktu is not null) set @produkt_suma= @Ilosc_produktow * (select Cena from
KsiegarniaSchemat.Spis_Produktow where @IDProduktu = IDProduktu);

if(@IDKsiazki is not null) set @ksiazka_suma= @Ilosc_ksiazek * (select Cena from
KsiegarniaSchemat.Spis_Ksiazek where @IDKsiazki = IDKsiazki);

declare @suma decimal (18,2);

set @suma = (@ksiazka_suma+@produkt_suma);

return @suma;

END;

```

9. Strategia tworzenia kopii zapasowych.

Kopia zapasowa pełna jest wykonywana co jeden miesiąc po inwentaryzacji.

Kopia przyrostowa(differential) jest wykonywana codziennie i gwarantuje nam codzienną aktualizację stanu magazynowego.

Automatyzacja repo jest wykonywana codziennie z tego samego powodu, co kopia przyrostowa na oryginalnej bazie.

Kopia przyrostowa gwarantuje nam codzienną aktualizację stanu magazynowego.

10. Wyszukiwanie pełnotekstowe.

Dodano FileGroup o nazwie Księgarnia FSG.

Database Properties - Ksiegarnia

Select a page

General
Files
Filegroups
Options
Change Tracking
Permissions
Extended Properties
Mirroring
Transaction Log Shipping
Query Store

Script

Help

Rows

Name	Files	Read-Only	Default	Autogrow All Files
PRIMARY	1		<input checked="" type="checkbox"/>	<input type="checkbox"/>

Add FilegroupRemove

FILESTREAM

Name	FILESTREAM Files	Read-Only	Default
KsiegarniaFSG	1	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Add FilegroupRemove

MEMORY OPTIMIZED DATA

Name	FILESTREAM Files
------	------------------

Add FilegroupRemove

Connection

Server:
DOOM

Connection:
DOOM0\Administrator

View connection properties

Progress

Ready

OK

Cancel

Dodano plik, który przyjmuje wyszukiwanie pełnotekstowe.

Database Properties - Ksiegarnia

Select a page

General
Files
Filegroups
Options
Change Tracking
Permissions
Extended Properties
Mirroring
Transaction Log Shipping
Query Store

Script

Help

Database name:

Ksiegarnia

Owner:

DOOM0\Administrator

☒ Use full-text indexing

Database files:

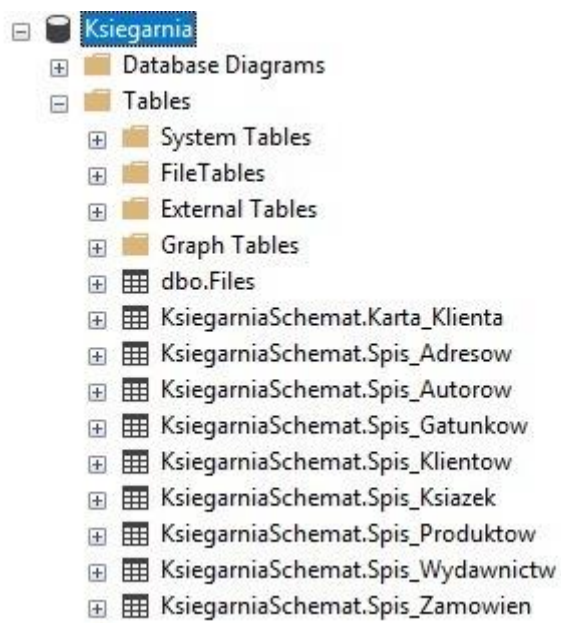
Logical Name	File Type	Filegroup	Size (MB)	Autogrowth / Maxsize	Path
Ksiegarnia	ROWS...	PRIMARY	8	By 64 MB, Unlimited	C:\Program Files\Microsoft SQL Server\MSS...
KsiegarniaFS	FILEST...	KsiegarniaFSG	1	Unlimited	C:\Program Files\Microsoft SQL Server\MSS...
Ksiegarnia_log	LOG	Not Applicable	72	By 64 MB, Limited to 2...	C:\Program Files\Microsoft SQL Server\MSS...

AddRemove

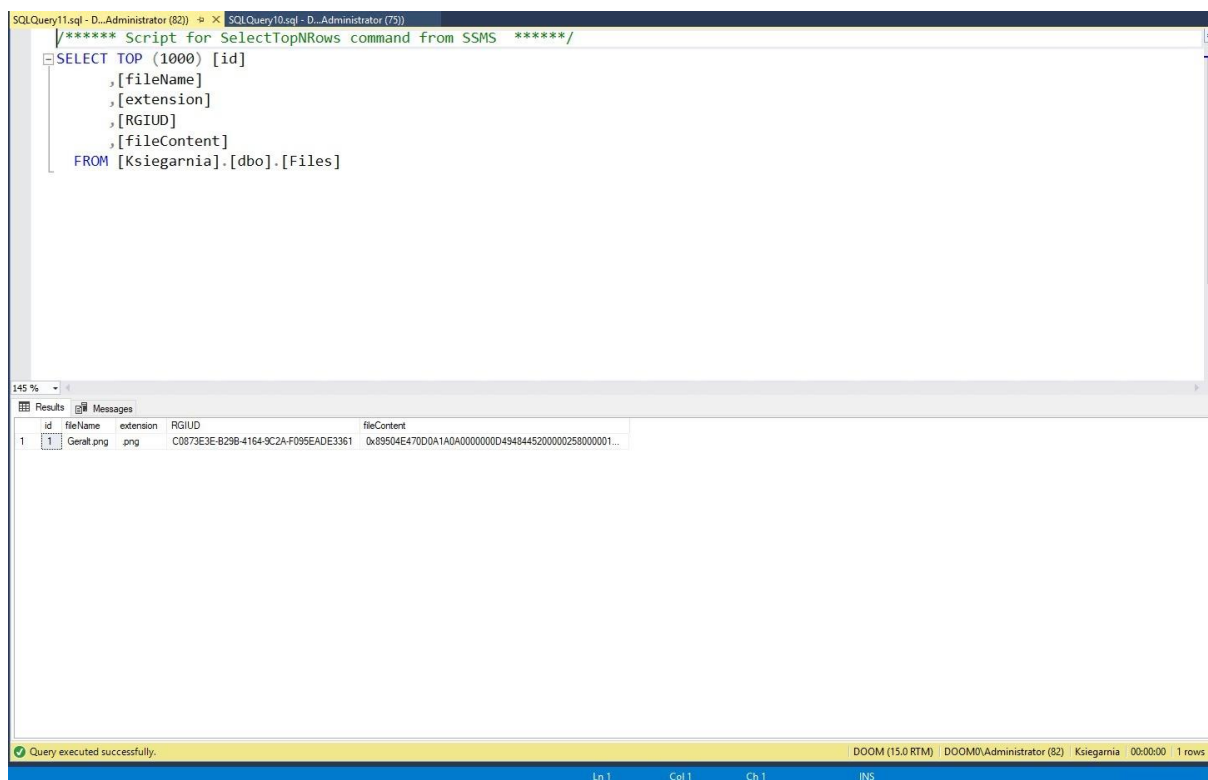
OK

Cancel

Na załączonym screenie widać bazę, która odbiera wyszukiwanie pełnotekstowe.



Wynik wykorzystania funkcjonalności – dodano obrazek do bazy.



Wstawiony do tabeli plik



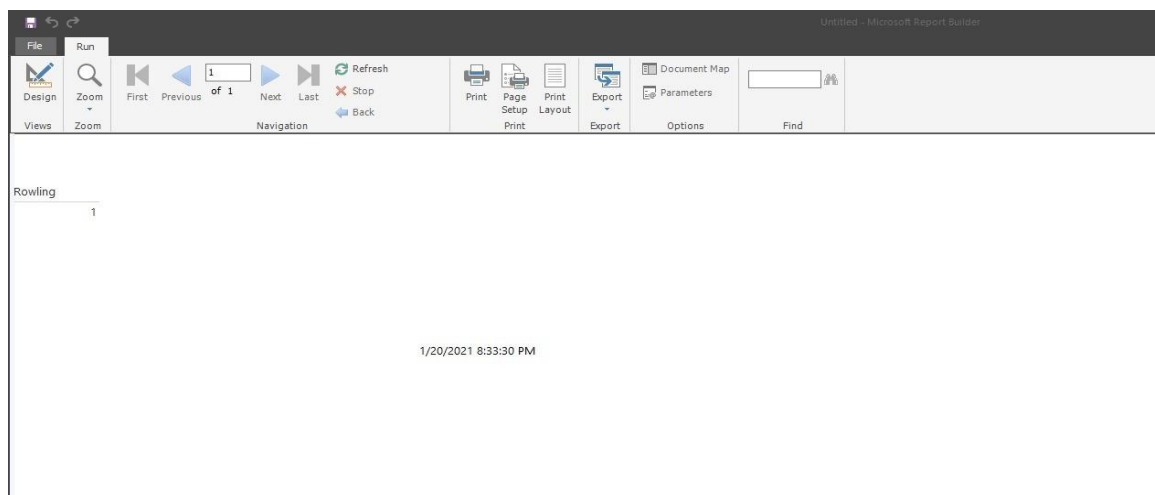
11. Usługi raportowania z wysyłką na maila.

Pierwszy raport. Sprawdzanie popularności zakupionych książek danego autora.

A screenshot of a software interface showing a 'Dataset Properties' dialog box. The dialog has a sidebar on the left with tabs: Query, Fields, Options, Filters, and Parameters. The 'Query' tab is selected. The main area contains the following fields and options:

- Name:** A text box containing 'DataSet1'.
- Use a shared dataset:** An unselected radio button.
- Use a dataset embedded in my report:** A selected radio button.
- Data source:** A dropdown menu showing 'DataSource1' and a 'New...' button.
- Query type:** Three radio buttons: 'Text' (selected), 'Table', and 'Stored Procedure'.
- Query:** A large text area containing a SQL query:

```
select a.Imie as Imie , a.Nazwisko as Nazwisko, count(w.Walidacja) as  
IlosckupionychKsiazek  
from KsiegarniaSchemat.Spis_Autorow as a  
join KsiegarniaSchemat.Spis_Zamowien as w on (a.IDAutora = w.IDAutora)  
where w.Walidacja = '1'  
group by a.Imie,a.Nazwisko;
```
- Buttons:** 'Query Designer...', 'Import...', and 'Refresh Fields'.
- Time out (in seconds):** A spinner box set to '0'.
- Footer:** 'Help', 'OK', and 'Cancel' buttons.



Report ilosc kupionych ksiazek by author - KSIEGARNIA REPORTING SYSTEM

testuj20 [Pokaż historię](#) Do: **mnie** Odebrane Więcej ▾

Generowany automatycznie raport. Temat: ilosc kupionych ksiazek by author



IloscKupionychKsiazekByAuthor.pdf
82.1 KB

Drugi raport. Ilość zamówionych książek danych autorów.

Dataset Properties

Query
Fields
Options
Filters
Parameters

Choose a data source and create a query.

Name: DataSet1

☐ Use a shared dataset.
☒ Use a dataset embedded in my report.

Data source: DataSource1 [New...](#)

Query type: ☒ Text ☐ Table ☐ Stored Procedure

Query:

```
select a.Imie as Imie , a.Nazwisko as Nazwisko, count(w.Walidacja) as  
IloscZamowionychKsiazek  
from KsiegarniaSchemat.Spis_Autorow as a  
join KsiegarniaSchemat.Spis_Zamowien as w on (a.IDAutora = w.IDAutora)  
where w.Walidacja = '0'  
group by a.Imie,a.Nazwisko;
```

[Query Designer...](#) [Import...](#) [Refresh Fields](#)

Time out (in seconds): 0

[Help](#) [OK](#) [Cancel](#)

File Run




Design Zoom First Previous 1 of 1 Next Last Refresh Stop Back

Print Page Setup Print Layout Export

Dumas	Rowling
1	3

1/20/2021 8:36:48 PM

● Raport ilosc zamowionych ksiazek - KSIEGARNIA REPORTING SYSTEM

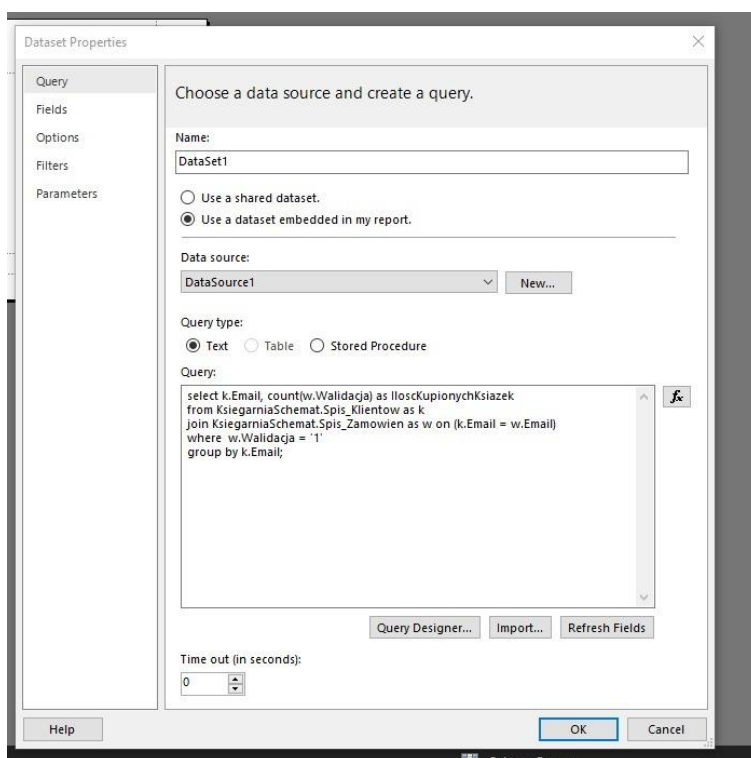
testuj20  Pokaż historię Do: **mnie**  Odebrane  Więcej ▾

Generowany automatycznie raport. Temat: Ilosc zamowionych ksiazek



IloscZamowionychKsiazek.pdf
83.5 KB

Trzeci raport. Sprawdzanie ilości zakupionych książek przez danego klienta.





Dataset Properties

Query
Fields
Options
Filters
Parameters

Choose a data source and create a query.

Name:
DataSet1

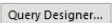


☐ Use a shared dataset.
☒ Use a dataset embedded in my report.


Data source:
DataSource1  



Query type:
☒ Text ☐ Table ☐ Stored Procedure

Query:

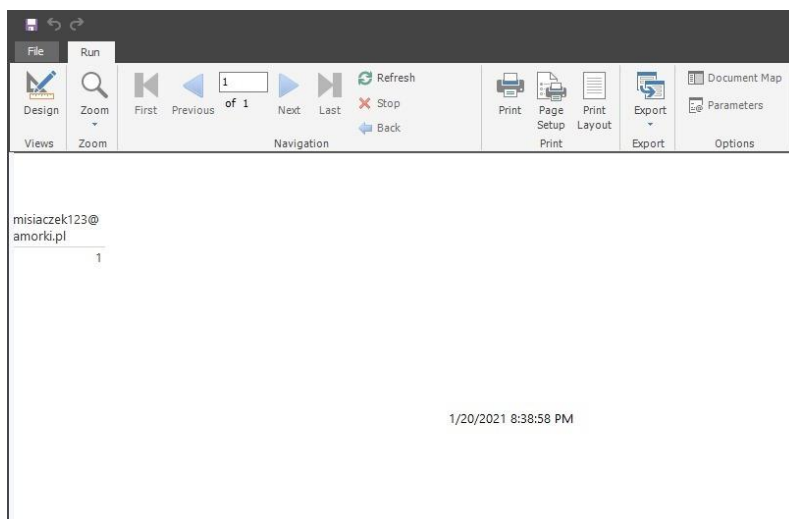
```
select k.Email, count(w.Walidacja) as IloscKupionychKsiazek  
from KsiegarniaSchemat.Spis_Klientow as k  
join KsiegarniaSchemat.Spis_Zamowien as w on (k.Email = w.Email)  
where w.Walidacja = '1'  
group by k.Email;
```

Time out (in seconds):
0 

Help  


Column Groups



Raport ilosc kupionych ksiazek - KSIEGARNIA REPORTING SYSTEM

testuj20 [Pokaż historię](#) Do: **mnie** [Odebrane](#) [Więcej](#)

Generowany automatycznie raport. Temat: Ilość kupionych książek

 IlośćKupionychKsiazek.pdf
83.9 KB

12. Procentowe zaangażowanie w projekt.

Z powodu dobrych stosunków oraz czerpania przyjemności ze wspólnie prowadzonej działalności edukacyjnej podmioty wykonujące zadanie współpracowały niemal na każdym etapie danego projektu (dokumentacja Karolina Lubczyk, główny administrator bazy Patryk Sobczak). Zasiadając przy swoich komputerach, łączyli się przez Hangout i poprzez funkcję udostępniania ekranu wspólnie modyfikowali i dyskutowali nad powierzonymi im wyzwaniami. Wspólnie uważamy, że była to najlepsza możliwa forma współpracy, a żywa i gorączkowa czasami wymiana pomysłów była kluczem do sukcesu. Z powodu przypisania Patryka Sobczaka jako kierownika projektu jego procentowy udział wynosi 51%, natomiast Karoliny Lubczyk 49%.