

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ»
ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

197376, Санкт-Петербург, ул. проф. Попова, 5.

Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

«З А Ч Т Е Н О»

_____ О.А. Жирнова

“ ” _____ 2021 г.

**ОТЧЁТ
по дисциплине «Базы данных»
Лабораторная работа № 2
«Группировка и агрегирование данных»**

Студент группы 9308

Н.В. Яловега

Санкт Петербург 2021

Цель работы: знакомство с опциями GROUP BY и HAVING, агрегированием данных.

Используемая база данных (БД): AdventureWorks.

Порядок выполнения

Упражнение 1 – использование ключевого слова TOP в команде SELECT

Запрос 1. Из таблицы Sales.SalesPerson выводим значения полей SalesPersonID и Bonus, сортируем по полю Bonus по убыванию.

```
SELECT SalesPersonID, Bonus  
FROM Sales.SalesPerson  
ORDER BY Bonus DESC
```

Результат выполнения запроса представлен на рисунке 1



	SalesPersonID	Bonus
1	279	6700.00
2	290	5650.00
3	285	5150.00
4	280	5000.00
5	282	5000.00
6	275	4100.00
7	287	3900.00
8	281	3550.00
9	283	3500.00
10	277	2500.00
11	276	2000.00
12	286	985.00
13	278	500.00
14	289	75.00
15	268	0.00
16	288	0.00
17	284	0.00

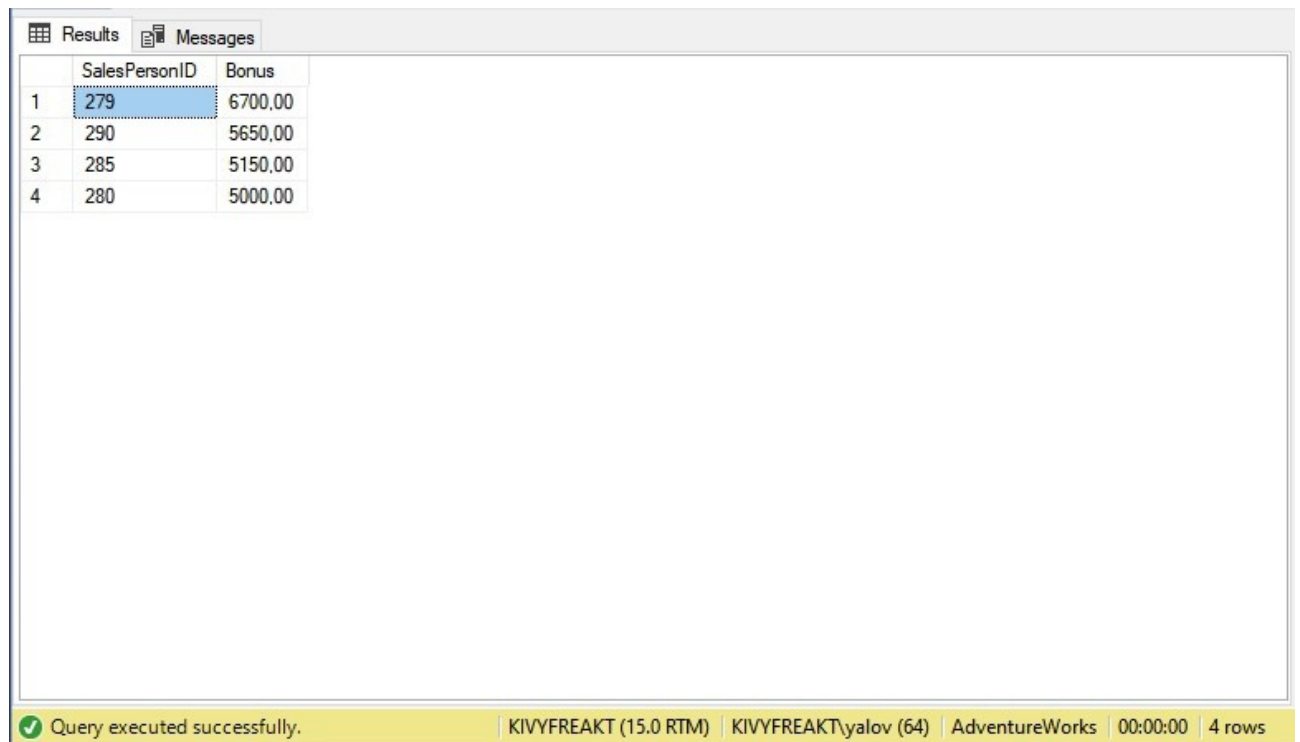
Query executed successfully. | KIVYFREACT (15.0 RTM) | KIVYFREACT\yalov (64) | AdventureWorks | 00:00:00 | 17 rows

Рисунок 1

Запрос 2. Ограничим количество возвращаемых записей из предыдущего запроса до 4 самых больших премий для продавцов.

```
SELECT TOP 4 SalesPersonID, Bonus  
FROM Sales.SalesPerson  
ORDER BY Bonus DESC
```

Результат выполнения запроса представлен на рисунке 2



The screenshot shows a SQL Server Enterprise Manager window with a 'Results' tab active. It displays a table with two columns: 'SalesPersonID' and 'Bonus'. The table contains four rows of data, ordered by Bonus in descending order. The first row is highlighted with a blue selection bar. The status bar at the bottom indicates 'Query executed successfully.' and '4 rows'.

	SalesPersonID	Bonus
1	279	6700,00
2	290	5650,00
3	285	5150,00
4	280	5000,00

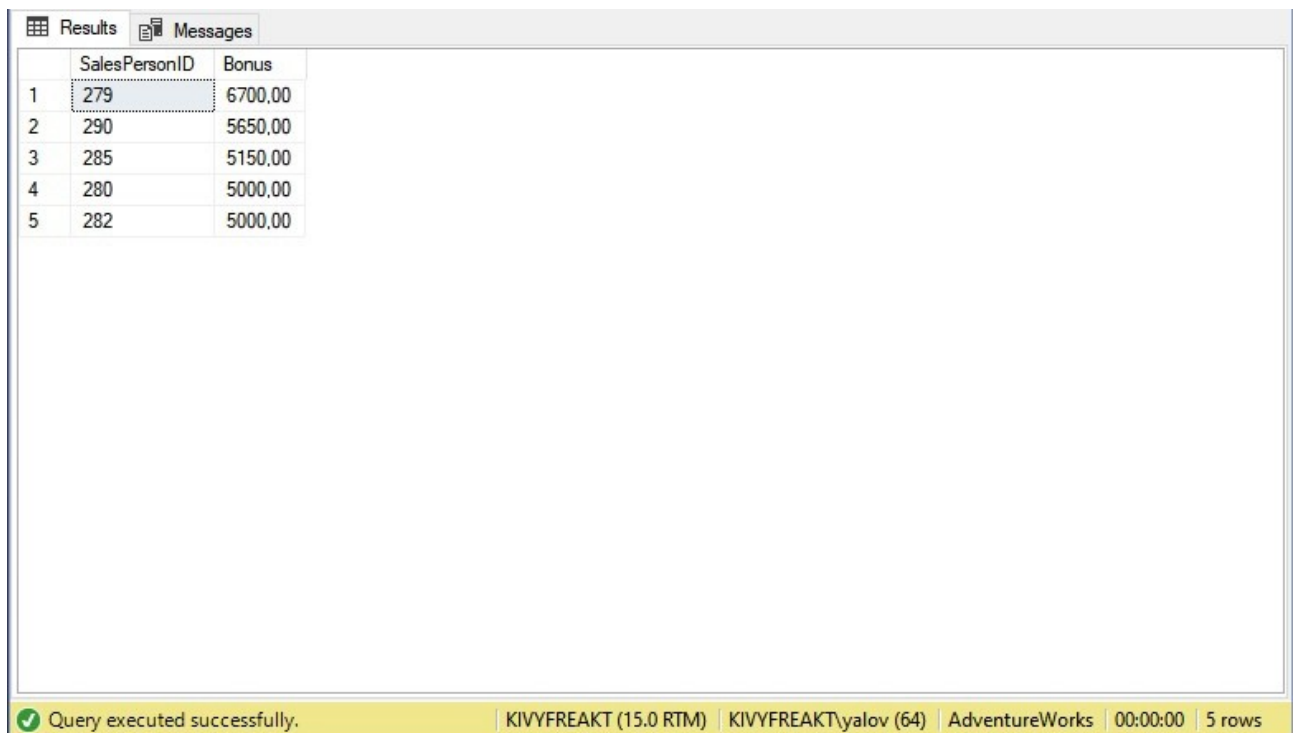
Query executed successfully. | KIVYFREACT (15.0 RTM) | KIVYFREACT\yakov (64) | AdventureWorks | 00:00:00 | 4 rows

Рисунок 2

Запрос 3. Теперь сделаем так, чтобы запрос возвращал строки не только со значениями первых четырех самых больших премий для продавцов, но и данные по тем продавцам, чьи премии имеют то же значение, что и последнее значение, полученное в предыдущем задании.

```
SELECT TOP 4 WITH TIES SalesPersonID, Bonus  
FROM Sales.SalesPerson  
ORDER BY Bonus DESC
```

Результат выполнения запроса приведен на рисунке 3.



	SalesPersonID	Bonus
1	279	6700,00
2	290	5650,00
3	285	5150,00
4	280	5000,00
5	282	5000,00

Рисунок 3

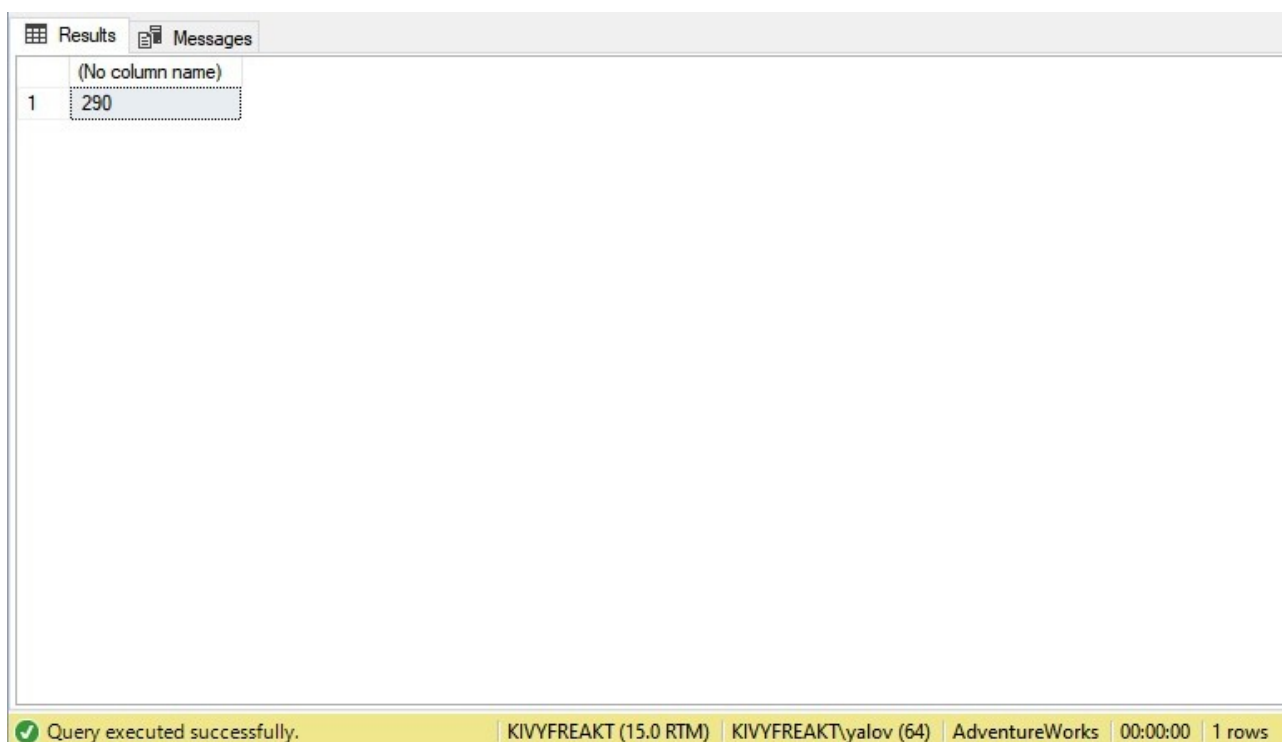
Упражнение 2 – использование агрегатных функций и конструкций GROUP BY И HAVING

Запрос 1.1. Подсчет общего количества строк в таблице Employee схемы HumanResources.

```
SELECT COUNT(*)
```

```
FROM HumanResources.Employee
```

Результат выполнения запроса приведен на рисунке 4.



The screenshot shows the SQL Server Enterprise Manager interface. At the top, there are two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a single row of data. The column header is '(No column name)' and the value is '290'. The row is numbered '1' in the first column. At the bottom of the window, a status bar provides additional information: a green checkmark icon followed by 'Query executed successfully.', the server name 'KIVYFREAKT (15.0 RTM)', the user 'KIVYFREAKT\yvalov (64)', the database 'AdventureWorks', the execution time '00:00:00', and '1 rows'.

	(No column name)
1	290

Query executed successfully. | KIVYFREAKT (15.0 RTM) | KIVYFREAKT\yvalov (64) | AdventureWorks | 00:00:00 | 1 rows

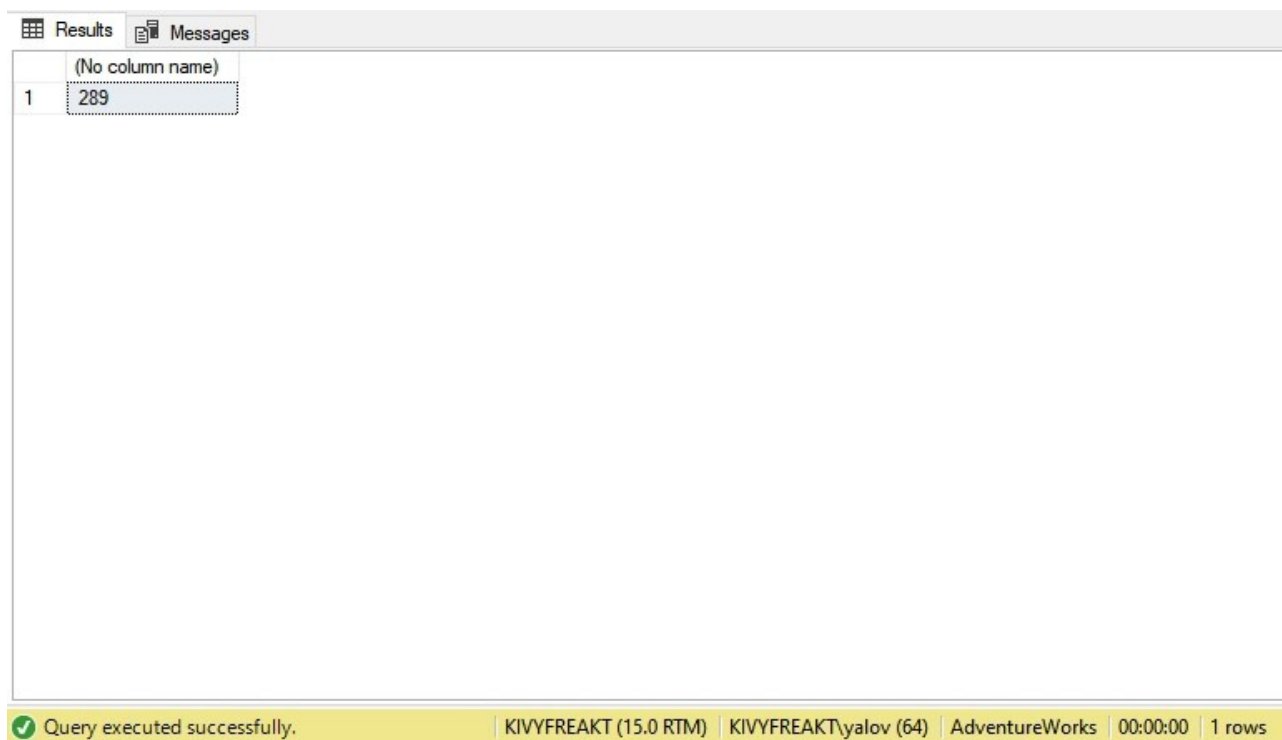
Рисунок 4

Запрос 1.2. Подсчет общего количества сотрудников, имеющих менеджеров (поле ManagerID).

```
SELECT COUNT(ManagerID)
```

```
FROM HumanResources.Employee
```

Результат выполнения запроса приведен на рисунке 5.



The screenshot shows the SQL Server Enterprise Manager interface. At the top, there are two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a single row of data. The column header is '(No column name)' and the value is '289'. The status bar at the bottom indicates that the query was executed successfully, showing the server name 'KIVYFREAKT (15.0 RTM)', the user 'KIVYFREAKT\yakov (64)', the database 'AdventureWorks', the execution time '00:00:00', and the number of rows '1 rows'.

(No column name)
289

Query executed successfully. | KIVYFREAKT (15.0 RTM) | KIVYFREAKT\yakov (64) | AdventureWorks | 00:00:00 | 1 rows

Рисунок 5

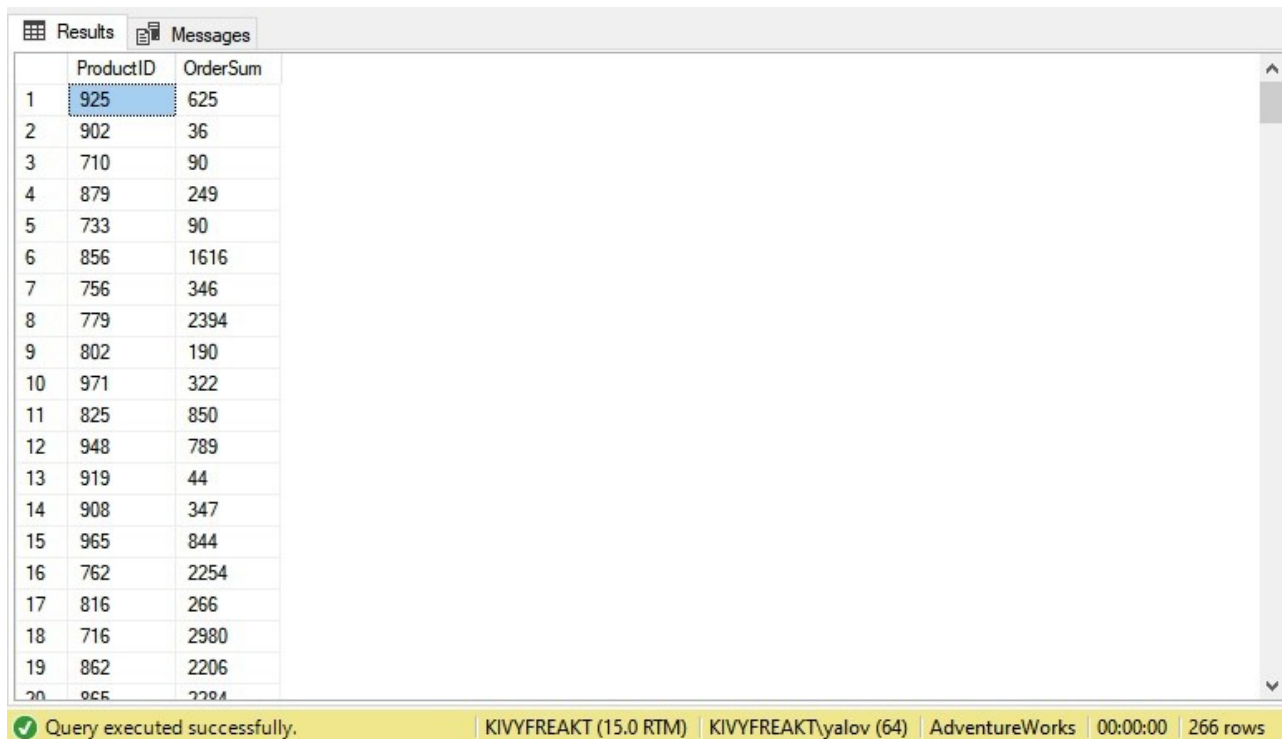
Запрос 2.1. Подсчитаем суммарное количество заказного товара (поле OrderQty) для каждого продукта (поле ProductID).

```
SELECT ProductID, SUM(OrderQty) AS OrderSum
```

```
FROM Sales.SalesOrderDetail
```

```
GROUP BY ProductID
```

Результат выполнения запроса приведен на рисунке 6.



	ProductID	OrderSum
1	925	625
2	902	36
3	710	90
4	879	249
5	733	90
6	856	1616
7	756	346
8	779	2394
9	802	190
10	971	322
11	825	850
12	948	789
13	919	44
14	908	347
15	965	844
16	762	2254
17	816	266
18	716	2980
19	862	2206
20	865	2284

Query executed successfully. | KIVYFREACT (15.0 RTM) | KIVYFREACT\yalov (64) | AdventureWorks | 00:00:00 | 266 rows

Рисунок 6

Запрос 2.2. Отсортируем прошлый результат запроса по суммарному количеству товара.

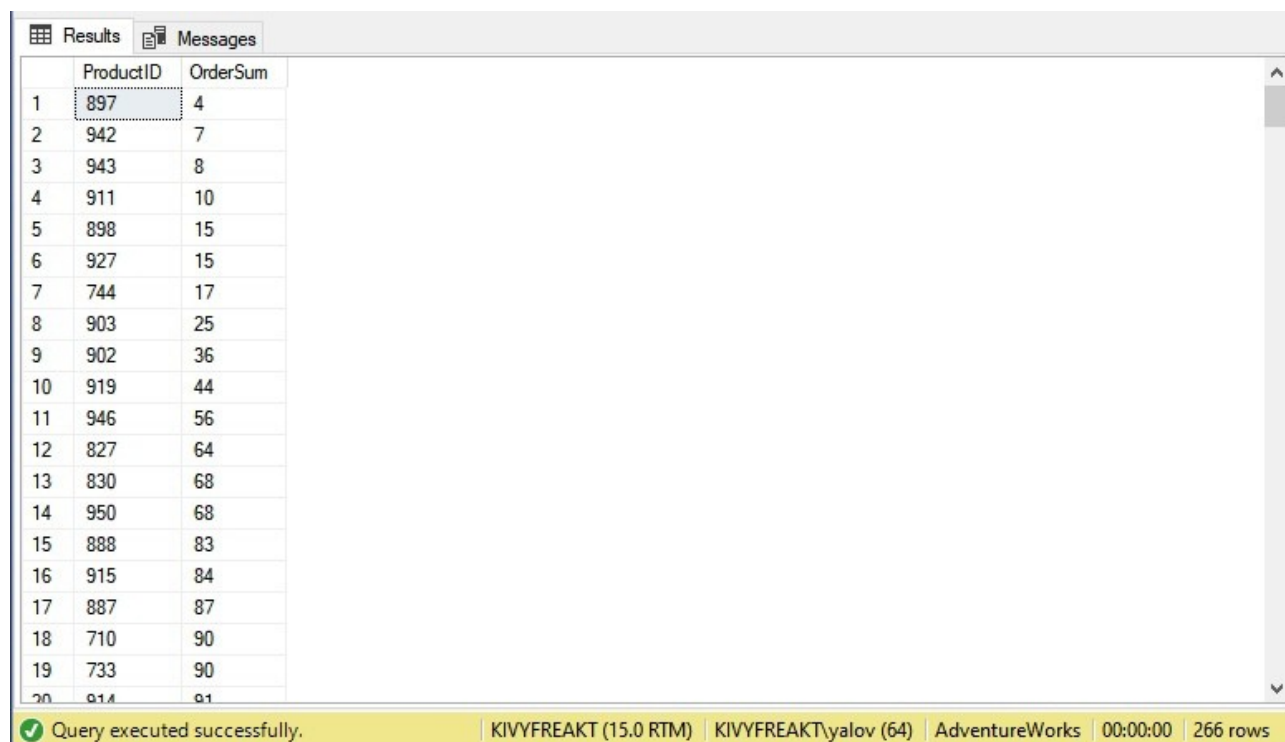
```
SELECT ProductID, SUM(OrderQty) AS OrderSum
```

```
FROM Sales.SalesOrderDetail
```

```
GROUP BY ProductID
```

```
ORDER BY OrderSum
```

Результат выполнения запроса приведен на рисунке 7.



	ProductID	OrderSum
1	897	4
2	942	7
3	943	8
4	911	10
5	898	15
6	927	15
7	744	17
8	903	25
9	902	36
10	919	44
11	946	56
12	827	64
13	830	68
14	950	68
15	888	83
16	915	84
17	887	87
18	710	90
19	733	90
20	914	91

Query executed successfully. KIVYFREAKT (15.0 RTM) KIVYFREAKT\yalov (64) AdventureWorks 00:00:00 266 rows

Рисунок 7

Запрос 2.3. Модифицируем запрос таким образом, чтобы в результирующий набор попадали только те товары, суммарное значение заказов по которым не менее 2000.

```
SELECT ProductID, SUM(OrderQty) AS OrderSum
```

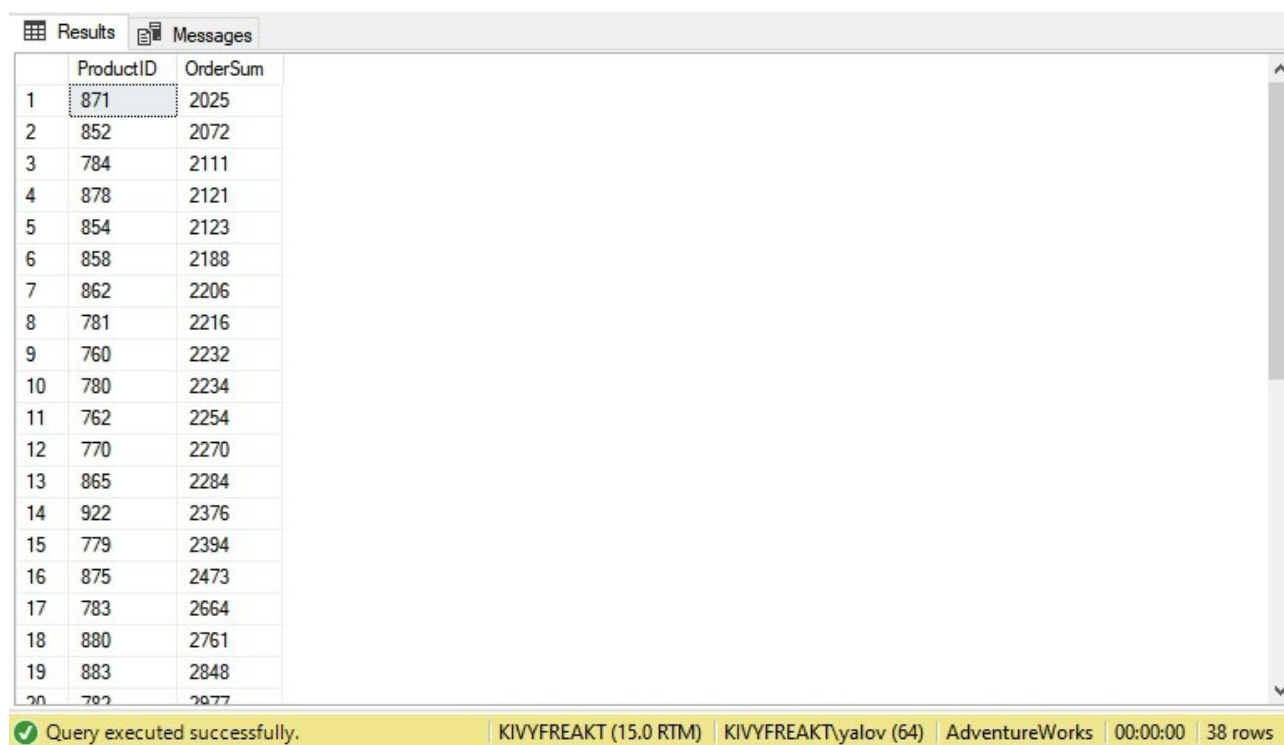
```
FROM Sales.SalesOrderDetail
```

```
GROUP BY ProductID
```

```
HAVING SUM(OrderQty) >= 2000
```

```
ORDER BY OrderSum
```

Результат выполнения запроса приведен на рисунке 8.



	ProductID	OrderSum
1	871	2025
2	852	2072
3	784	2111
4	878	2121
5	854	2123
6	858	2188
7	862	2206
8	781	2216
9	760	2232
10	780	2234
11	762	2254
12	770	2270
13	865	2284
14	922	2376
15	779	2394
16	875	2473
17	783	2664
18	880	2761
19	883	2848
20	782	2877

Query executed successfully. | KIVYFREACT (15.0 RTM) | KIVYFREACT\yalov (64) | AdventureWorks | 00:00:00 | 38 rows

Рисунок 8

Запрос 3.1. Выведем поля ProductID, SpecialOfferID, среднее значение цены за единицу товара (поле UnitPrice) и суммарного значения по полю LineTotal. Выполним группировку.

SELECT ProductID, SpecialOfferID,

AVG(UnitPrice) **AS** AveragePrice,

SUM(LineTotal) **AS** SumLineTotal

FROM Sales.SalesOrderDetail

GROUP BY ProductID, SpecialOfferID

Результат выполнения запроса приведен на рисунке 9.

	ProductID	SpecialOfferID	AveragePrice	SumLineTotal
1	815	1	36,447	22013.988000
2	758	1	874,794	621103.740000
3	955	1	1923,6978	869708.736000
4	925	2	144,8782	1561.786996
5	954	14	1030,9491	197210.270400
6	898	1	200,052	3000.780000
7	998	1	439,98	540097.998000
8	754	2	845,6342	20718.037900
9	941	1	48,594	7143.318000
10	854	2	43,4942	12275.803008
11	884	1	43,3937	84893.876000
12	827	1	165,231	10574.784000
13	797	2	1074,87	56093.635360
14	707	11	15,7455	2971.175850
15	958	13	334,0575	55937.928375
16	937	2	46,9742	2301.735800
17	877	4	3,975	89.437500
18	770	1	488,0852	894057.112800
19	967	1	1856,1687	1071401.058000
20	880	2	21,9842	12252.677884

Query executed successfully. | KIVYFREACT (15.0 RTM) | KIVYFREACT\yalov (64) | AdventureWorks | 00:00:00 | 484 rows

Рисунок 9

Запрос 3.2. Отсортируем полученные данные по полю ProductID по возрастанию, а также присвоим псевдонимы тем элементам списка SELECT которые соответствуют агрегированным значениям.

SELECT ProductID, SpecialOfferID,

AVG(UnitPrice) AS AveragePrice,

SUM(LineTotal) AS SumLineTotal

FROM Sales.SalesOrderDetail

GROUP BY ProductID, SpecialOfferID

ORDER BY ProductID ASC

Результат выполнения запроса приведен на рисунке 10.

	ProductID	SpecialOfferID	AveragePrice	SumLineTotal
1	707	11	15,7455	2971.175850
2	707	8	16,8221	2452.662180
3	707	3	18,9272	2191.058910
4	707	1	31,3436	141271.252000
5	707	2	20,0556	8886.245452
6	708	8	16,8221	2316.403170
7	708	11	15,7455	2997.943200
8	708	3	18,9753	3461.676690
9	708	2	20,0502	11689.730276
10	708	1	30,9648	140403.764500
11	709	2	5,51	723.573200
12	709	3	5,225	853.765000
13	709	1	5,70	4235.100000
14	709	4	4,75	247.950000
15	710	1	5,70	513.000000
16	711	8	16,8221	2679.760530
17	711	2	20,0284	11421.237324
18	711	11	15,7455	3131.779950
19	711	3	18,9977	4384.931245
20	711	1	20,9697	142798.900000

Query executed successfully. | KIVYFREACT (15.0 RTM) | KIVYFREACT\yalov (64) | AdventureWorks | 00:00:00 | 484 rows

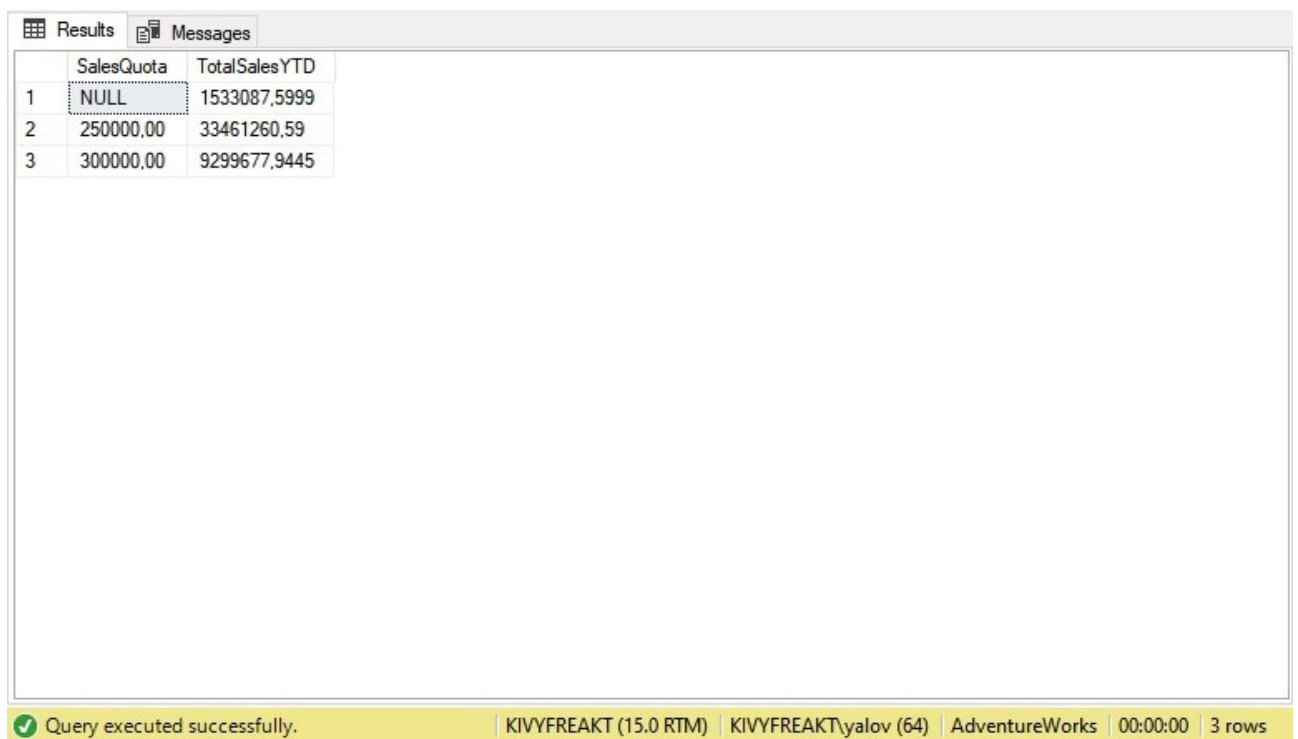
Рисунок 10

Упражнение 3 – использование операторов ROLLUP И CUBE

Запрос 1.1. Выведем поля SalesQuota и суммарное значение по полю SalesYTD. Выполним группировку. Также зададим псевдоним TotalSalesYTD для суммы.

```
SELECT SalesQuota,  
SUM(SalesYTD) AS TotalSalesYTD  
FROM Sales.SalesPerson  
GROUP BY SalesQuota
```

Результат выполнения запроса приведен на рисунке 11.



The screenshot shows a SQL Server interface with a 'Results' tab. It displays a table with two columns: 'SalesQuota' and 'TotalSalesYTD'. There are three rows of data. The first row has a NULL value for SalesQuota. The second and third rows have numerical values for SalesQuota. The status bar at the bottom indicates the query was executed successfully, showing the server name 'KIVYFREACT (15.0 RTM)', the user 'KIVYFREACT\ygalov (64)', the database 'AdventureWorks', the execution time '00:00:00', and the number of rows '3 rows'.

	SalesQuota	TotalSalesYTD
1	NULL	1533087,5999
2	250000,00	33461260,59
3	300000,00	9299677,9445

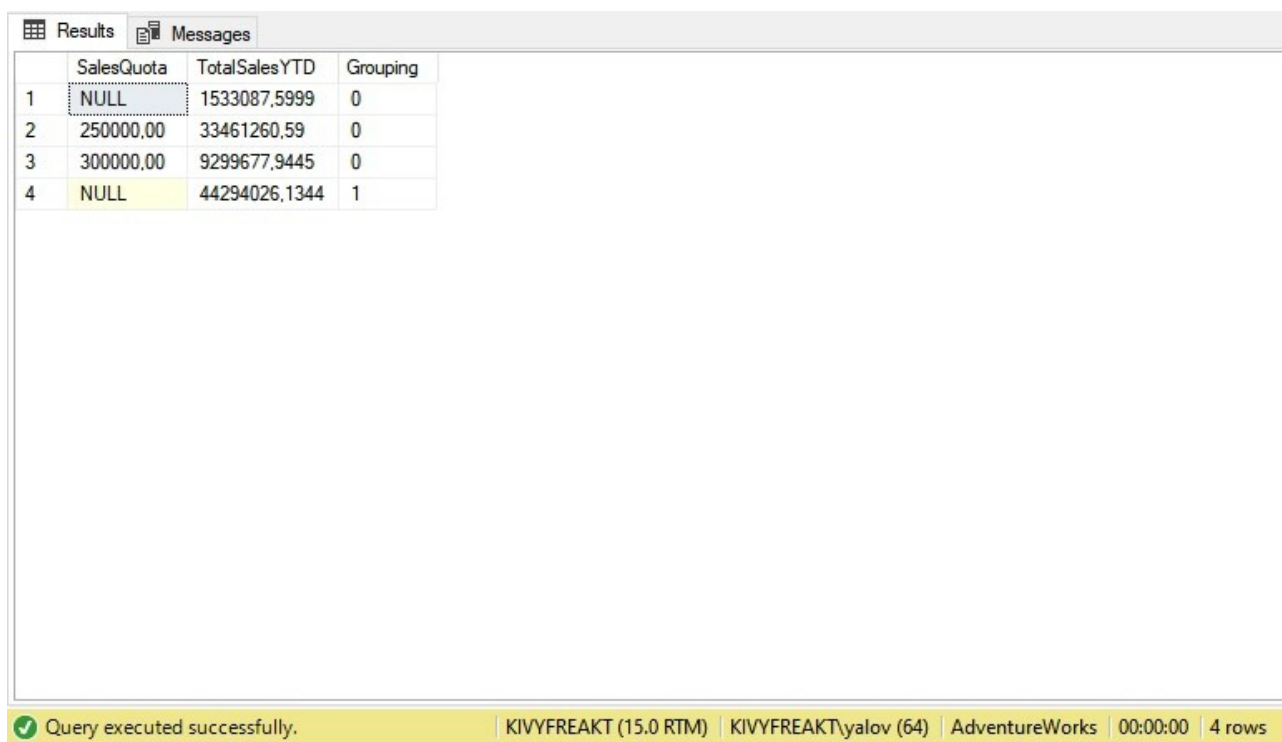
Query executed successfully. | KIVYFREACT (15.0 RTM) | KIVYFREACT\ygalov (64) | AdventureWorks | 00:00:00 | 3 rows

Рисунок 11

Запрос 1.2. Предыдущий запрос был изменен так, чтобы получить сводный результат по полученной выборке, дополнительно применена функция GROUPING.

```
SELECT SalesQuota,  
SUM(SalesYTD) AS TotalSalesYTD, GROUPING(SalesQuota) AS 'Grouping'  
FROM Sales.SalesPerson  
GROUP BY SalesQuota WITH ROLLUP
```

Результат выполнения запроса приведен на рисунке 12.



	SalesQuota	TotalSalesYTD	Grouping
1	NULL	1533087,5999	0
2	250000,00	33461260,59	0
3	300000,00	9299677,9445	0
4	NULL	44294026,1344	1

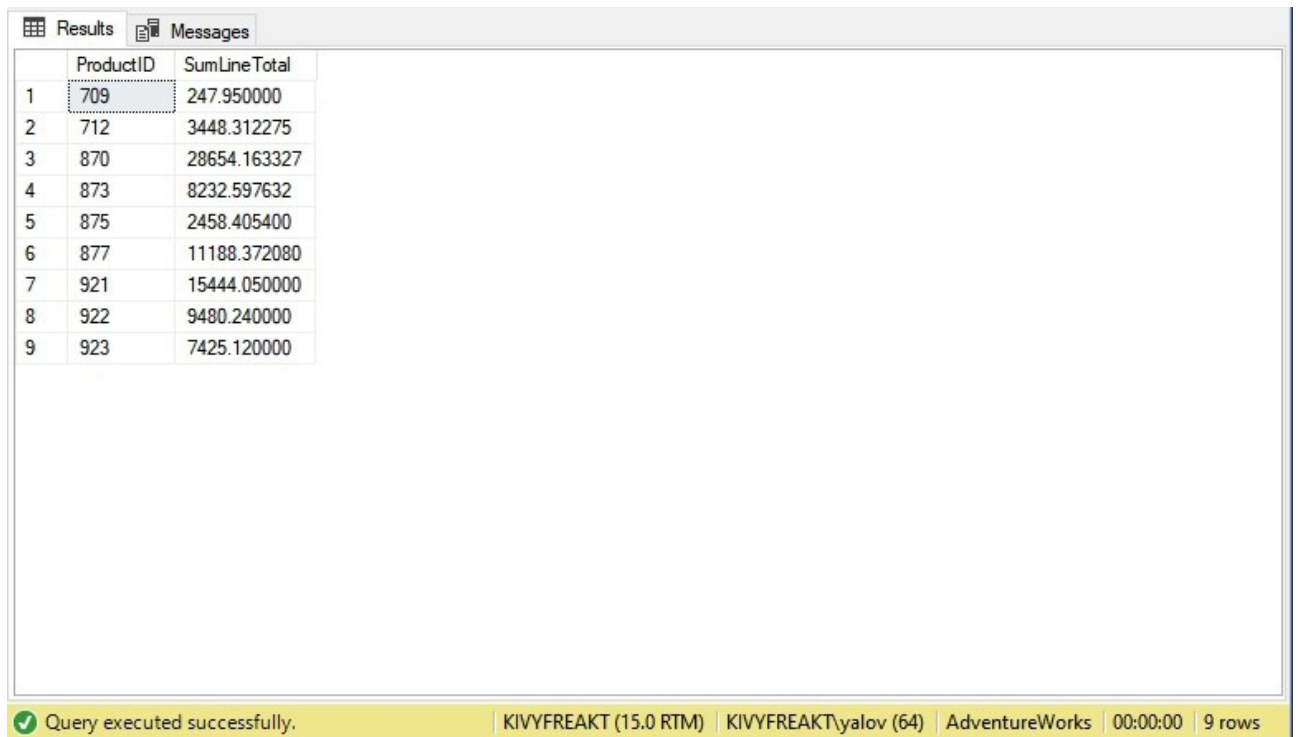
Рисунок 12

Запрос 1.3. Результат совпал со скриншотом из методических указаний, следовательно, задание выполнено верно. Смысл значений NULL - «поле, не содержащее никакого значения».

Запрос 2.1. Выведем поля ProductID и сумма по полю LineTotal. Выведены только те значения, для которых UnitPrice < \$5.00. Выполнена сортировка и группировка произведена по полю ProductID.

```
SELECT ProductID,  
SUM(LineTotal) AS SumLineTotal  
FROM Sales.SalesOrderDetail  
WHERE UnitPrice<5.00  
GROUP BY ProductID  
ORDER BY ProductID ASC
```

Результат выполнения запроса приведен на рисунке 13.



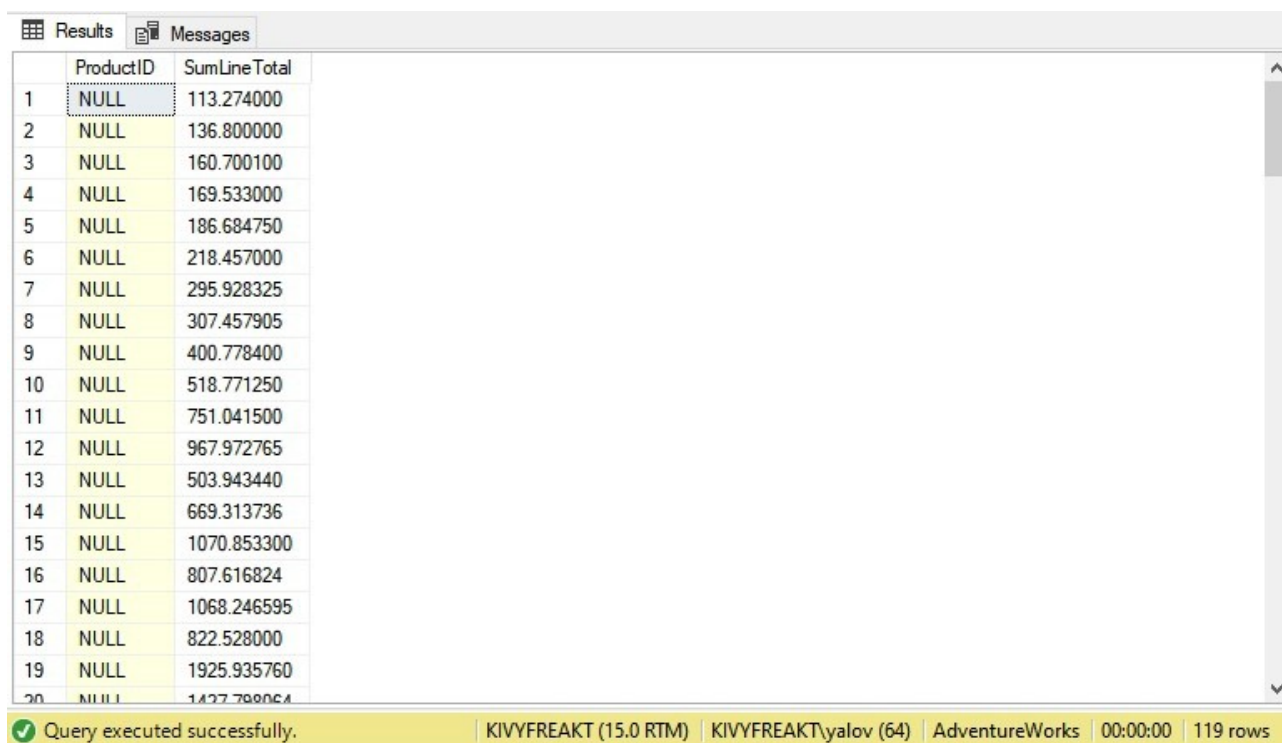
	ProductID	SumLineTotal
1	709	247.950000
2	712	3448.312275
3	870	28654.163327
4	873	8232.597632
5	875	2458.405400
6	877	11188.372080
7	921	15444.050000
8	922	9480.240000
9	923	7425.120000

Рисунок 13

Запрос 2.2. Предыдущий запрос модифицирован путем добавления оператора CUBE, а в группировку добавлено поле OrderQty.

```
SELECT ProductID,  
SUM(LineTotal) AS SumLineTotal  
FROM Sales.SalesOrderDetail  
WHERE UnitPrice<5.00  
GROUP BY CUBE (ProductID, OrderQty)  
ORDER BY ProductID ASC
```

Результат выполнения запроса приведен на рисунке 14.



	ProductID	SumLineTotal
1	NULL	113.274000
2	NULL	136.800000
3	NULL	160.700100
4	NULL	169.533000
5	NULL	186.684750
6	NULL	218.457000
7	NULL	295.928325
8	NULL	307.457905
9	NULL	400.778400
10	NULL	518.771250
11	NULL	751.041500
12	NULL	967.972765
13	NULL	503.943440
14	NULL	669.313736
15	NULL	1070.853300
16	NULL	807.616824
17	NULL	1068.246595
18	NULL	822.528000
19	NULL	1925.935760
20	NULL	1427.788064

Рисунок 14

Выводы

Ознакомился с опциями GROUP BY и HAVING (например, для формирования нескольких групп), а также с агрегированием данных. Использовал ключевые слова TOP и предложение WITH TIES для возвращения части отсортированных значений из результирующего набора данных. Для создания сводных результатов использовал операторы ROLLUP и CUBE, также применял функцию GROUPING.

Список использованных источников

1. Горячев А. В., Новакова Н. Е. Распределенные базы данных. Мет. указания к лаб. работам., СПб. Изд-во СПбГЭТУ «ЛЭТИ», 2008
2. Горячев А.В, Новакова Н.Е. Особенности разработки и администрирования приложений баз данных: учеб. пособие. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2016. 68 с.
3. Дейт К. Введение в системы баз данных. : Пер. с англ. – 6-е изд. - К.:Диалектика, 1998.