

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра Систем автоматизированного проектирования**

**ОТЧЕТ**  
**по лабораторной работе №2**  
**по дисциплине «Базы данных»**  
**Тема: Выполнение выборки из таблицы**

Студенты гр. 2308

Попов Н.А.

Бебия Р.А.

Чиков А.А.

Преподаватель

Горяинов С.В.

Санкт-Петербург

2024

## Цель работы

Знакомство с опциями GROUP BY и HAVING, а также агрегированием данных

## Используемая база данных

AdventureWorks

## Выполнение работы

### Упражнение 1 – использование ключевого слова TOP в команде SELECT

Запрос 1: Вывести значения полей и отсортировать запрос по Bonus по убыванию

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

Результат выполнения запроса показан 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

Рисунок 1 – строк результата выполнения запроса

Запрос 2: Модификация запроса, чтобы возвращались 4 самых больших бонуса

**SELECT TOP(4)** SalesPersonID, Bonus

**From** Sales.SalesPerson **ORDER BY** Bonus **DESC**;

Результат выполнения показан на рисунке 2.

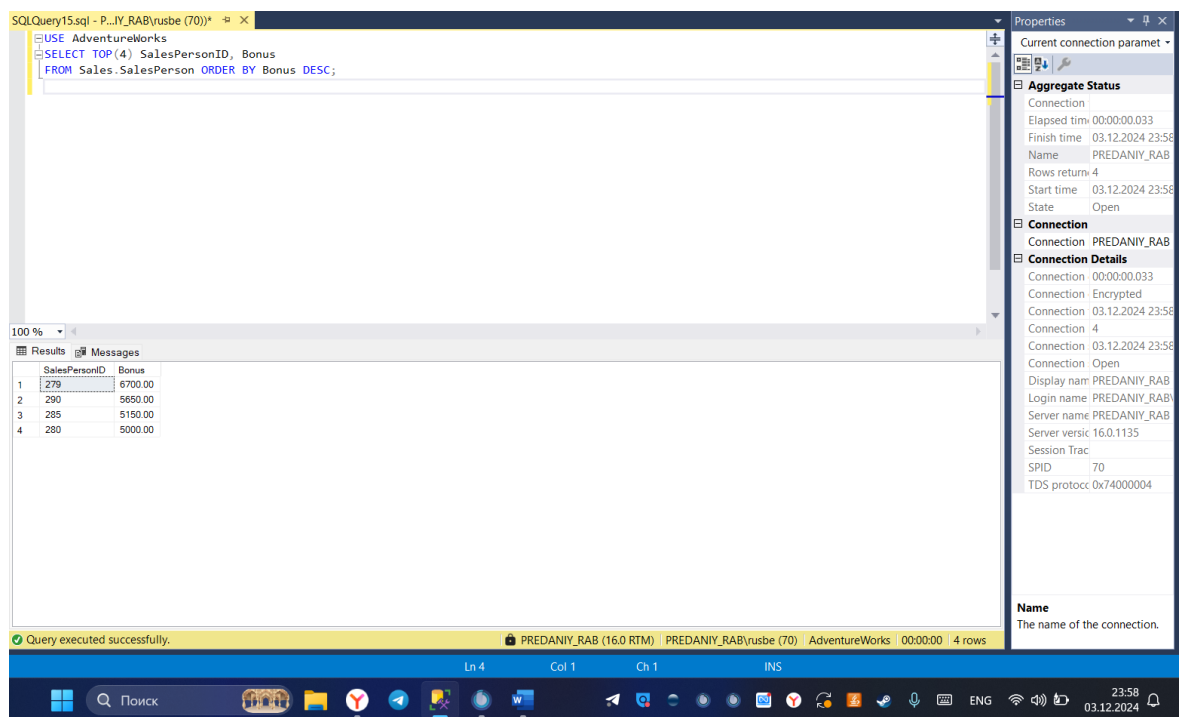


Рисунок 2 – Первые 4 строк результата выполнения запроса

Запрос 3: **SELECT** с условием **WITH TIES**

**SELECT TOP (4) WITH TIES** SalesPersonID, Bonus

**FROM** Sales.SalesPersonID **ORDER BY** Bonus **DESC**;

Результат выполнения запроса представлен на рисунке 3

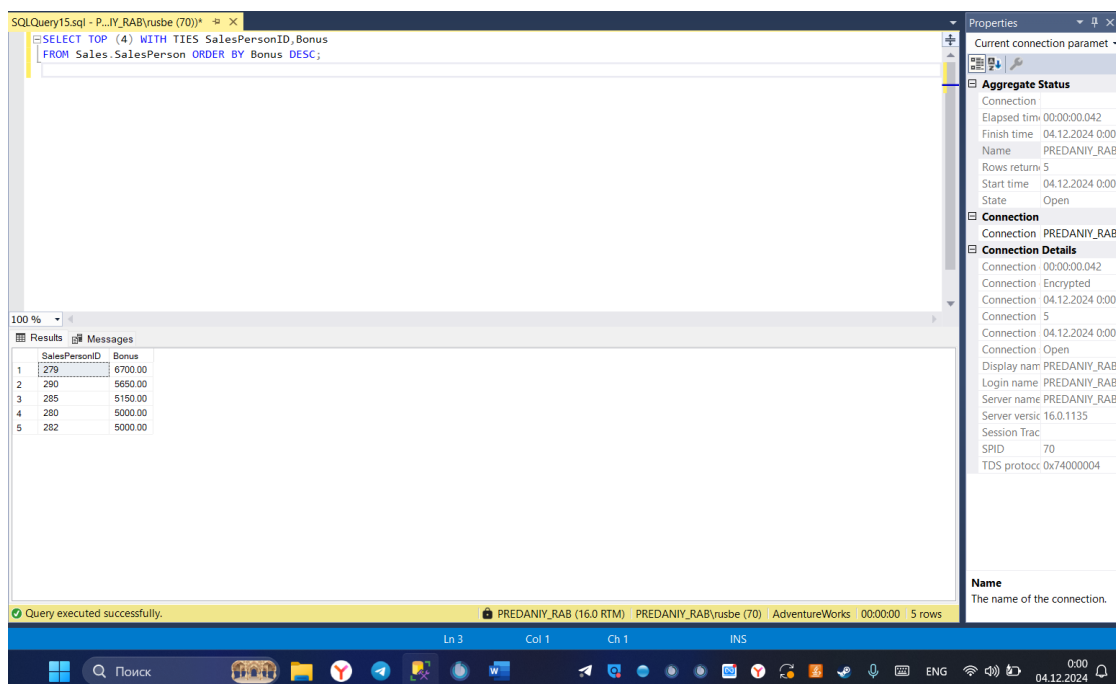


Рисунок 3 – Первые 5 строк результата выполнения запроса

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

Запрос 1: Подсчитать общее количество строк в таблице Employee

**SELECT COUNT(\*) AS TotalEmployees**

**FROM HumanResources.Employee;**

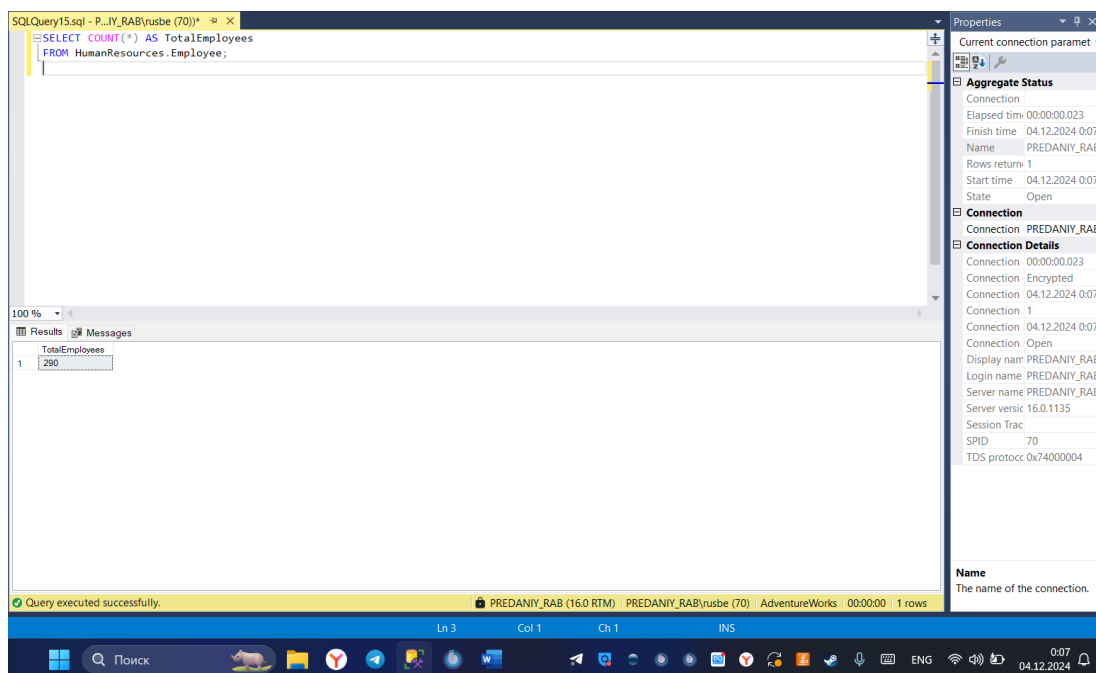


Рисунок 4 – Результат выполнения запроса

Запрос 2: Подсчитать общее количество сотрудников, имеющих менеджеров

```
SELECT COUNT (*) AS TotalEmployees
```

```
FROM HumanResources.Employee
```

```
WHERE ManagerID IS NOT NULL
```

Результат выполнения запроса представлен на рисунке

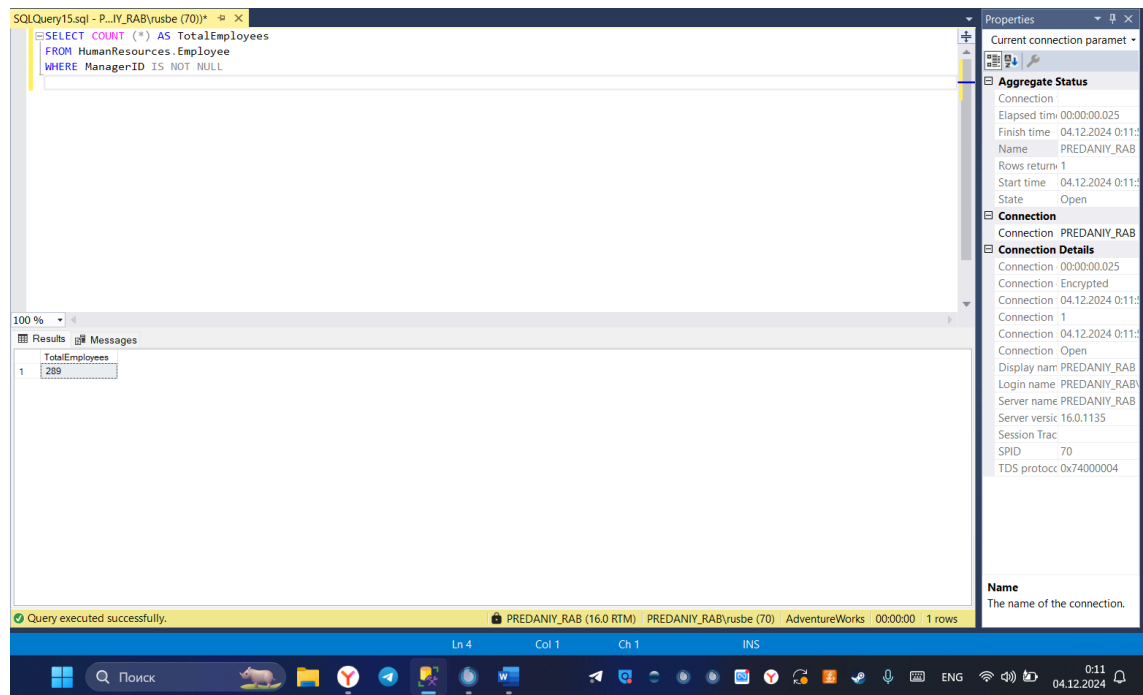


Рисунок 5 – Результат выполнения запроса с WHERE

Запрос 3: Подсчитать поле OrderQty для каждого продукта (ProductID)

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

```
FROM Sales.SalesOrderDetail
```

```
GROUP BY ProductID
```

Результат выполнения запроса представлен на рисунке

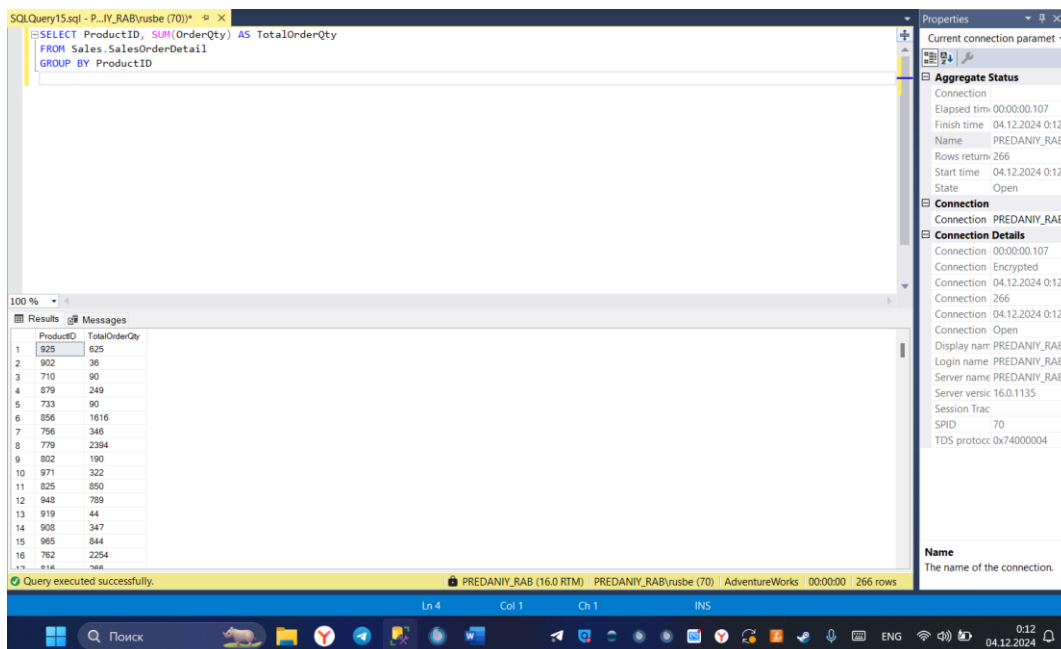


Рисунок 6 – Результат выполнения запроса с SUM

Запрос 4: Отсортировать результат запроса по суммарному количеству

**SELECT** ProductID, SUM(OrderQty) **AS** TotalOrderQty

**FROM** Sales.SalesOrderDetail

**GROUP BY** ProductID

**ORDER BY** TotalOrderQty **DESC**

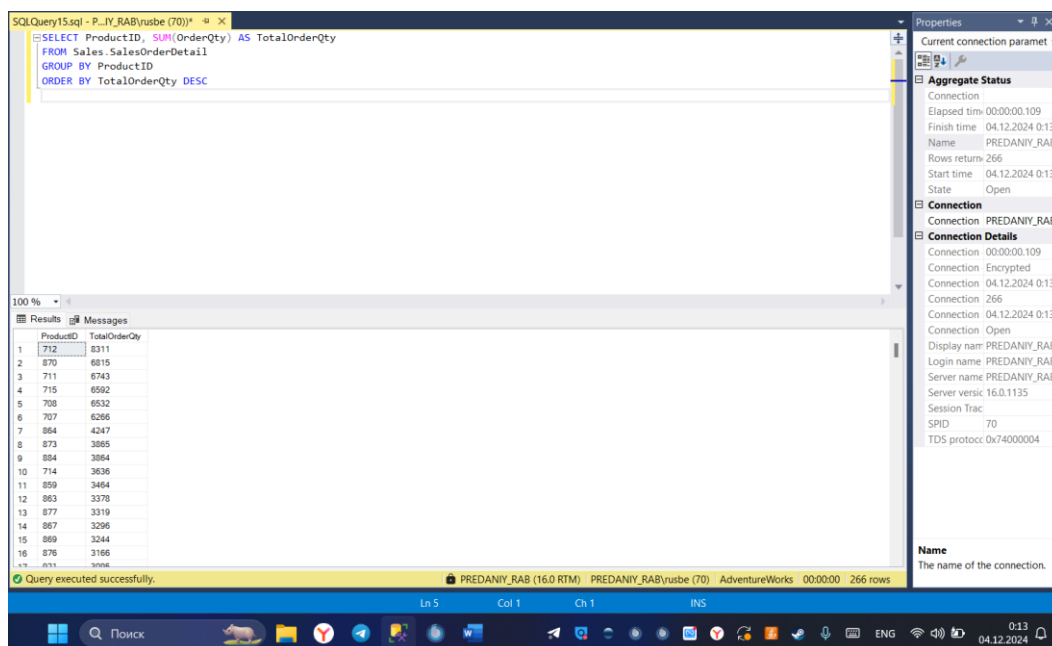


Рисунок 7 – Результат выполнения запроса по суммарному количеству

Анализ: Запрос корректно выводит суммированные значения, в порядке убывания

Запрос 5: Отсортировать результат запроса по суммарному количеству с модификацией ( $\geq 2000$ )

```
SELECT ProductID, SUM (OrderQty) AS TotalOrderQty
FROM Sales.SalesOrderDetail
GROUP BY ProductID
HAVING SUM (OrderQty)  $\geq$  2000
ORDER BY TotalOrderQty DESC
```

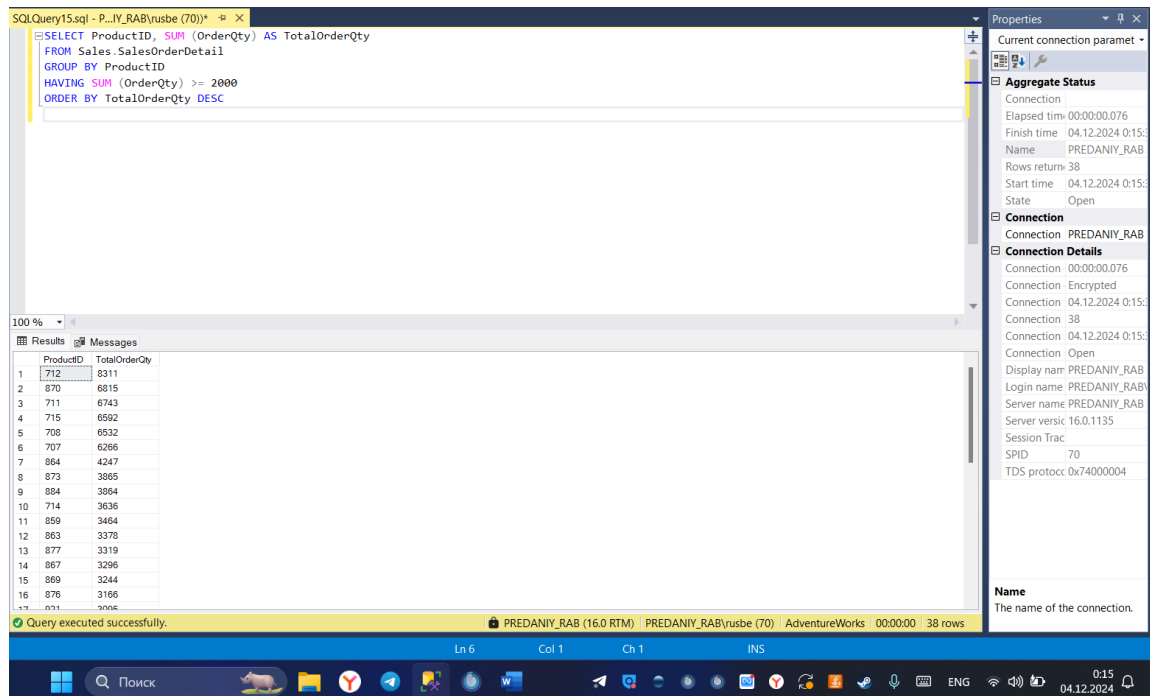


Рисунок 8 –Выполнение запроса по суммарному количеству с условием

Анализ: Запрос корректно выводит значения, удовлетворяя условие

Запрос 6: Написать запрос к таблице Sales.SalesOrderDetail, выполнить группировку

```
SELECT ProductID, SpecialOfferID, AVG(UnitPrice) AS AvgUnitPrice, SUM
(LineTotal) AS TotallineAmount
FROM Sales.SalesOrderDetail
```

## GROUP BY ProductID, SpecialOfferID

SQLQuery15.sql - P...IY\_RAB\rusbe (70)\*

```
SELECT ProductID, SpecialOfferID, AVG(UnitPrice) AS AvgUnitPrice, SUM(LineTotal) AS TotallineAmount
FROM Sales.SalesOrderDetail
GROUP BY ProductID, SpecialOfferID
```

ProductID	SpecialOfferID	AvgUnitPrice	TotallineAmount
815	1	36.447	22013.988000
758	1	874.794	621103.740000
955	1	1923.6978	869708.736000
925	2	144.8782	1561.786996
954	14	1030.9491	197210.270400
896	1	200.052	3000.780000
995	1	439.98	540097.998000
754	2	845.6342	20718.037900
941	1	48.594	7143.318000
954	2	43.4942	12275.803008
884	1	43.3937	84893.876000
827	1	165.231	10574.784000
797	2	1074.87	56093.636360
707	11	15.7455	2971.175850
958	13	334.0575	55937.928375
937	2	46.9742	2301.735800
877	4	3.076	89.427600

Query executed successfully. PREDANIY\_RAB (16.0 RTM) PREDANIY\_RAB\rusbe (70) AdventureWorks 00:00:00 484 rows

Рисунок 9 –Выполнение запроса с использованием GROUP BY для нескольких групп

Запрос 7: Написать запрос к таблице Sales.SalesOrderDetail с сортировкой по ProductID по возрастанию, дать псевдонимы, выполнить группировку

SELECT ProductID, SpecialOfferID, AVG(UnitPrice) AS AvgUnitPrice,

SUM (LineTotal) AS TotallineAmount

FROM Sales.SalesOrderDetail

GROUP BY ProductID, SpecialOfferID

ORDER BY ProductID



SQLQuery15.sql - P...RY\_RAB\rusbe (70)\*

```

SELECT ProductID, SpecialOfferID, AVG(UnitPrice) AS AvgUnitPrice,
SUM (LineTotal) AS TotalLineAmount
FROM Sales.SalesOrderDetail
GROUP BY ProductID, SpecialOfferID
ORDER BY ProductID

```

100 %

Results Messages

	ProductID	SpecialOfferID	AvgUnitPrice	TotalLineAmount
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	8896.245452
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	853.785000
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

Query executed successfully. PREDANIY\_RAB (16.0 RTM) PREDANIY\_RAB\rusbe (70) AdventureWorks 00:00:00 484 rows

Ln 6 Col 1 Ch 1 INS

Поиск

ENG 0:17 04.12.2024

Properties

Current connection param

Aggregate Status

Connection

Elapsed time 00:00:00.180

Finish time 04.12.2024 0:17:00

Name PREDANIY\_RAB

Rows returned 484

Start time 04.12.2024 0:17:00

State Open

Connection

Connection PREDANIY\_RAB

Connection Details

Connection 00:00:00.180

Connection Encrypted

Connection 04.12.2024 0:17:00

Connection 484

Connection 04.12.2024 0:17:00

Connection Open

Display name PREDANIY\_RAB

Login name PREDANIY\_RAB

Server name PREDANIY\_RAB

Server version 16.0.1135

Session Trac

SPID 70

TDS protocol 0x74000004

Name

The name of the connection.

Рисунок 10 –Выполнение запроса с использованием GROUP BY для нескольких групп с сортировкой и группировкой

### Упражнение 3 – использование операторов ROLLUP и CUBE

Запрос 1:Написать запрос к таблице Sales.SalesOrderDetail,выполнить группировку ,дать псевдоним

**SELECT** SalesQuota, **SUM** (SalesYTD) **AS** TotalSalesYTD

**FROM** Sales.SalesPerson

**GROUP BY** SalesQuota

Результат выполнения запроса представлен на рисунке

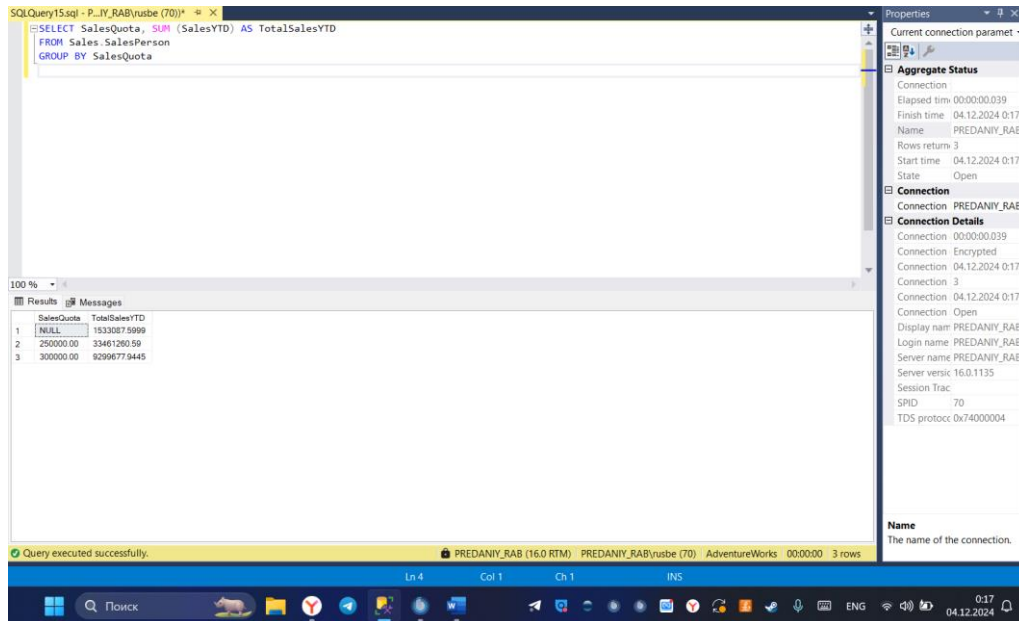


Рисунок 11 –Выполнение запроса с использованием GROUP BY для нескольких групп с суммарным значением и группировкой

Запрос 2: Написать запрос к таблице Sales.SalesOrderDetail, выполнить группировку, дать псевдоним и применить GROUPING

**SELECT** SalesQuota, **SUM** (SalesYTD) 'TotalSalesYTD' , **GROUPING**(SalesQuota) **AS** 'Grouping'

**FROM** Sales.SalesPerson

**GROUP BY** SalesQuota **WITH ROLLUP**

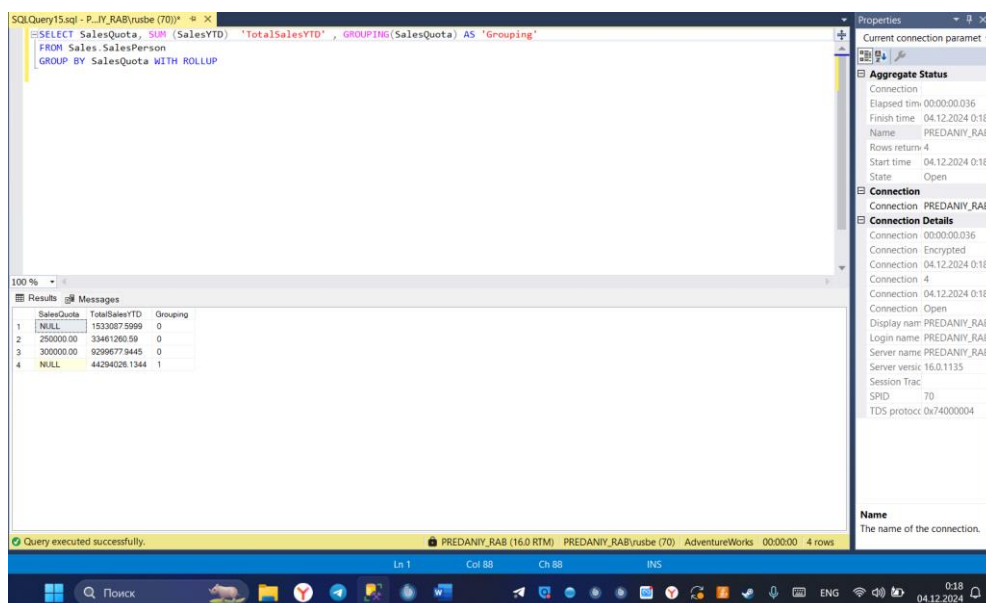


Рисунок 12 –Выполнение запроса с использованием ROLLUP

NULL в поле SalesQuota указывает на то, что это сводная строка, обобщающая данные по всем SalesQuota. NULL значения позволяют легче интерпретировать итоги, отделяя их от обычных записей, что полезно для анализа данных.

Запрос 3: Написать запрос к таблице Sales.SalesOrderDetail, выполнить группировку, сортировку и ограничение < 5.00

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

ProductID	OrderQty	TotalLineAmount
NULL	NULL	86579.210714
NULL	1	61159.530000
NULL	2	833.124000
NULL	3	1486.154000
NULL	4	2059.752000
NULL	5	1873.860000
NULL	6	1941.300000
NULL	7	1860.264000
NULL	8	1480.832000
NULL	9	822.528000
NULL	10	1400.760000
NULL	11	503.943440
NULL	12	1427.780064
NULL	13	669.313736
NULL	14	807.616824
NULL	15	518.771250

Рисунок 13 –Выполнение запроса с использованием WHERE, GROUP BY CUBE, ORDER BY

Запрос 4: Модифицировать прошлый запрос, добавив оператор CUBE и поле OrderQty

**SELECT** ProductID, OrderQty **SUM**(LineTotal) **AS** TotalLineAmount

**FROM** Sales.SalesOrderDetail

**WHERE** UnitPrice < 5.00

**GROUP BY** OrderQty, **CUBE** (ProductID)

**ORDER BY** ProductID **DESC**;

The screenshot displays the SQL Server Enterprise Manager interface. The main window shows a query executed successfully, returning 109 rows. The query is as follows:

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

The Results pane shows a table with three columns: ProductID, OrderQty, and TotalLineAmount. The data is grouped by OrderQty and ProductID using the CUBE operator. The Properties pane on the right shows the connection details for the current connection, including the connection name, display name, login name, server name, and session ID.

ProductID	OrderQty	TotalLineAmount
923	1	7425.120000
922	1	9480.240000
921	1	15444.050000
877	1	186.030000
877	2	477.500000
877	3	501.360000
877	4	1087.560000
877	5	1001.700000
877	6	1116.180000
877	7	1001.700000
877	8	915.840000
877	9	472.230000
877	10	763.200000
877	11	347.946060
877	12	867.605780
877	13	411.208980
875	14	640.540000

Рисунок 14 –Выполнение запроса с использованием CUBE

## Упражнение 4 – использование предложений COMPUTE и COMPUTE BY в команде SELECT для создания отчетов

Запрос 1: Написать запрос к таблице Sales.SalesOrderDetail,указать 5 полей, отсортировать по двум заданным полям

Результат выполнения запроса представлен на рисунке

```
SELECT SalesPersonID, CustomerID, OrderDate, SubTotal, TotalDue
FROM Sales.SalesOrderHeader
ORDER BY SalesPersonID, OrderDate;
SELECT
    SUM(SubTotal) AS TotalSubTotal,
    SUM(TotalDue) AS TotalTotalDue
FROM Sales.SalesOrderHeader;
```

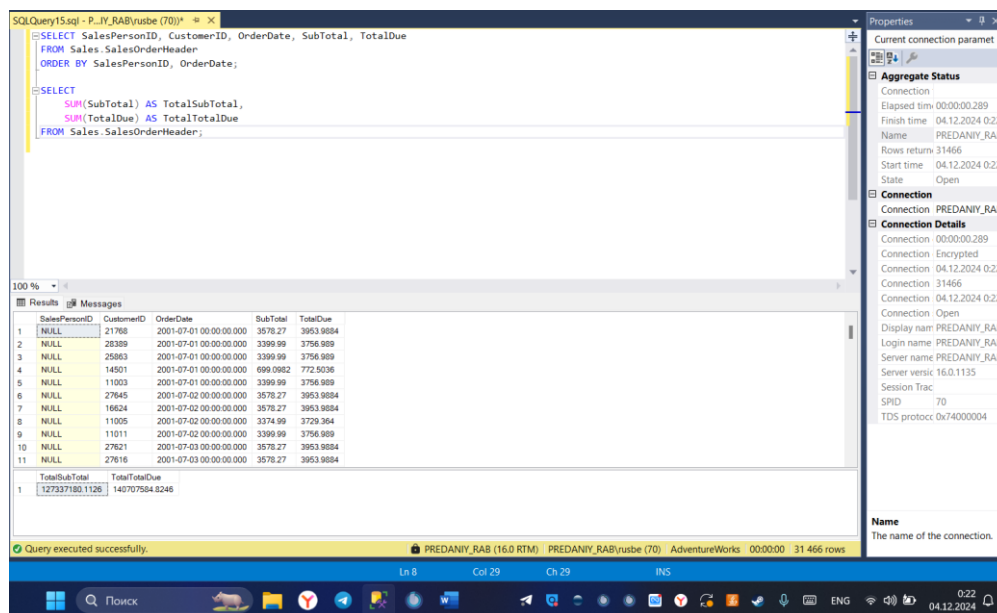


Рисунок 15 –Выполнение запроса с использованием Sum вместо Compute

## **Вывод**

В ходе лабораторной работы были изучены TOP, WITH TIES, GROUP BY, HAVING, CUBE, ROLLUP используемые для анализа и агрегирования данных из базы данных. В процессе выполнения заданий были проведены выборки из таблиц базы данных Adventure Works с применением различных ключевых слов, таких как WHERE, AS, NULL, FROM, ORDER BY. Это позволило закрепить методы фильтрации и упорядочивания данных. Эти навыки и знания обеспечат более эффективное извлечение и анализ данных из базы данных в будущих проектах.