

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

**ОТЧЕТ**  
**по лабораторной работе №2**  
**по дисциплине «Базы данных»**  
**Тема: «Группировка и агрегирование данных»**

Студенты гр. 1302

Рожественский К.И.  
Серотюкова О.

Преподаватель:

Новакова Н.Е.

Санкт-Петербург

2023

*Цель работы:* знакомство с опциями команд GROUP и BY HAVING, а также агрегированием данных.

*База данных:* AdventureWorks

### Упражнение 1 – использование ключевого слова TOP в запросах

Запрос 1:

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

Результат выполнения запроса:

SalesPersonID	Bonus
279	6700,00
290	5650,00
285	5150,00
280	5000,00
282	5000,00
275	4100,00
287	3900,00
281	3550,00
283	3500,00
277	2500,00
276	2000,00
286	985,00
278	500,00
289	75,00
268	0,00
288	0,00
284	0,00

(17 rows affected)

Completion time: 2023-09-21T12:36:56.7028507+03:00

### Запрос 2. Использование TOP:

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

#### Результаты выполнения запроса:

SalesPersonID	Bonus
279	6700,00
290	5650,00
285	5150,00
280	5000,00

(4 rows affected)

Completion time: 2023-09-21T12:36:56.7028507+03:00

### Запрос 3. Использование TOP WITH TIES:

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

#### Результат выполнения запроса:

SalesPersonID	Bonus
279	6700,00
290	5650,00
285	5150,00
280	5000,00
282	5000,00

(5 rows affected)

Completion time: 2023-09-21T12:36:56.7028507+03:00

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

Запрос 1. Подсчёт всех строк таблицы:

```
SELECT COUNT(*) AS 'Count'  
FROM HumanResources.Employee
```

Результат выполнения запроса:

```
Count  
-----  
290
```

(1 row affected)

Completion time: 2023-09-21T13:00:58.0323335+03:00

Запрос 2. Подсчёт сотрудников, имеющих менеджеров:

```
SELECT COUNT(ManagerID) AS 'Mananger Count'  
FROM HumanResources.Employee
```

Результат выполнения запроса:

```
Mananger Count  
-----  
289
```

Внимание! Значение NULL исключено в агрегатных или других операциях SET.

(1 row affected)

Completion time: 2023-09-21T13:04:46.6716757+03:00

Запрос 3. Подсчёт суммарного количества заказов каждого товара:

```
SELECT ProductID, SUM(OrderQty) AS Sum_Count FROM
Sales.SalesOrderDetail
GROUP BY ProductID
```

Результаты выполнения запроса:

ProductID	Sum_Count
925	625
902	36
710	90
...	
933	858
984	450

(266 rows affected)

Completion time: 2023-09-21T14:21:49.6987885+03:00

В результате выполнения запроса были просуммированы значения поля OrderQty для каждого отдельного значения поля ProductID. Все поля являются столбцами таблицы Sales.SalesOrderDetail.

Запрос 4. Сортировка результата предыдущего запроса:

```
SELECT ProductID, SUM(OrderQty) AS Sum_Count FROM
Sales.SalesOrderDetail
GROUP BY ProductID
ORDER BY Sum_Count
```

Результаты выполнения запроса:

ProductID	Sum_Count
897	4
942	7
943	8
...	
870	6815
712	8311

(266 rows affected)

Completion time: 2023-09-21T14:25:14.3433143+03:00

Запрос 5. Фильтрация количества заказов по каждому товару:

```
SELECT ProductID, SUM(OrderQty) AS Sum_Count FROM
Sales.SalesOrderDetail
GROUP BY ProductID
HAVING SUM(OrderQty) >= 2000
```

Результаты выполнения запроса:

ProductID	Sum_Count
779	2394
762	2254
716	2980
862	2206
865	2284
...	
870	6815
884	3864

(38 rows affected)

Completion time: 2023-09-21T14:28:17.1548642+03:00

В результате выполнения запроса было выведено суммарные количества заказов каждого товара, котырые не меньше 2000.

Запрос 6. Формирование нескольких групп с помощью команды OUP

BY:

```
SELECT ProductID, SpecialOfferID, AVG(UnitPrice)
AS 'Average Price', SUM(LineTotal) AS 'Sum Line'
FROM Sales.SalesOrderDetail
GROUP BY SpecialOfferID, ProductID
ORDER BY ProductID
```

Результаты выполнения запроса:

ProductID	SpecialOfferID	Average Price
707	2	20,0556
8886.245452		
707	11	15,7455
2971.175850		

707	8	16,8221
2452.662180		
707	1	31,3436
141271.252000		
707	3	18,9272
2191.058910		
708	11	15,7455
2997.943200		
708	1	30,9648
140403.764500		
708	8	16,8221
2316.403170		
708	2	20,0502
11689.730276		
708	3	18,9753
3461.676690		
...		
999	2	527,3902
76871.032436		
999	1	428,3185
438795.874000		

(484 rows affected)

Completion time: 2023-09-21T14:55:08.0255906+03:00

### Упражнение 3 – Использование ROLLUP и CUBE

Запрос 1. Использование операторов Grouping и ROLLUP.

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

Результаты выполнения запроса:

SalesQuota	TotalSalesYTD
-----	-----
NULL	1533087,5999
250000,00	33461260,59
300000,00	9299677,9445
NULL	44294026,1344

(4 rows affected)

Completion time: 2023-09-21T15:30:49.8191919+03:00

В результате выполнения запроса кроме просуммированных значений поля SalesYTD для каждого значения поля SalesQuota из таблицы

Sales.SalesPerson была получена дополнительная строка с суммой всех значений. В столбце SalesQuota присутствует два значения NULL, говорящие об отсутствии данных. Первое значение изначально было в обрабатываемой таблицы, а второе появилось, так как для полученного супарного значения по всем группам значения данного поля отсутствует.

## Запрос 2.

Результаты выполнения запроса:

ProductID	Total
-----	
709	247.950000
712	3448.312275
870	28654.163327
873	8232.597632
875	2458.405400
877	11188.372080
921	15444.050000
922	9480.240000
923	7425.120000

(9 rows affected)

Completion time: 2023-09-21T16:05:22.0184402+03:00

Запрос 3. Модификация предыдущего запроса с использованием оператора CUBE.

```
SELECT ProductID, SUM(LineTotal) AS 'Total'
FROM Sales.SalesOrderDetail
WHERE UnitPrice < 5.00
GROUP BY ProductID, OrderQTY WITH CUBE
```

Результаты выполнения запроса:

ProductID	Total
-----	
NULL	86579.210714
923	7425.120000
923	7425.120000

(119 rows affected)

Completion time: 2023-09-21T17:35:25.4454301+03:00



**Вывод:** в процессе выполнения работы нами были изучены и применены такие части команды SELECT, как GROUP BY и HAVING, которые используются для группировки данных и фильтрации групп соответственно. Также были изучены и применены операторы GROUPING, ROLLUP и CUBE, а также различные агрегирующие функции. Были проведены различные запросы и получены выборки из таблиц базы данных AdventureWorks. Некоторые примеры результатов запросов:

1. Подсчёт всех строк таблицы - HumanResources.Employee. В результате получили число 290.
2. При подсчёте сотрудников, имеющих менеджеров получили результат: 289.
3. В результате выполнения запроса 1 из упражнения 3, кроме полученных значений сумм по каждой группе была получена строка, содержащая суммарное значение по всей таблице.

### **Список литературы**

1. Распределенные базы данных: Методические указания к лабораторным работам / Сост.: А. В. Горячев, Н. Е. Новакова. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2008. 32 с
2. SELECT — GROUP BY (Transact-SQL). Microsoft Learn. URL: <https://learn.microsoft.com/ru-ru/sql/t-sql/queries/select-group-by-transact-sql?view=sql-server-ver16>. Дата обращения: 21.09.2023
3. Виртуальный образовательный кластер СПбЭГЭТУ «ЛЭТИ». Курс «Базы данных». URL: <https://vec.etu.ru/moodle/course/view.php?id=14314>  
Дата обращения 21.09.2023.