**Лабораторная работа № 6. Группировка данных**

**КУРНОСЕНКО СОФЬЯ**

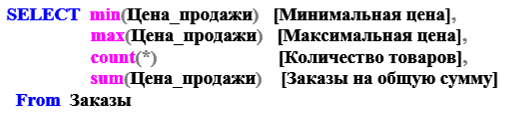
Основное назначение **группировки** с помощью секции GROUP BY – разбиение множества строк, сформированных секциями FROM и WHERE, на группы в соответствии со значениями в заданных столбцах, а также выполнение вычислений над группами строк с помощью наиболее часто используемых функций: **AVG** (вычисление среднего значения), **COUNT** (вычисление количества строк), **MAX** (вычисление максимального значения), **MIN** (вычисление минимального значения), **SUM** (вычисление суммы значений).

При использовании секции **GROUP BY** в SELECT-списке допускается указывать **только** те столбцы, по которым осуществляется группировка.

1.На основе таблицы **AUDITORIUM** разработать SELECT-запрос, вычисляющий максимальную, минимальную и среднюю вместимость аудиторий, суммарную вместимость всех аудиторий и общее количество аудиторий.

***→ Теория (методичка)***

Определить минимальную и максимальную цены продажи товаров, количество заказанных товаров и общую сумму заказов:



Пусть товары при продаже продаются заказчикам по разным ценам.

Определить наименования и максимальные цены товаров при продаже, количество заказанных товаров для тех товаров, количество которых на складе больше 5:

**SELECT Наименование\_товара,**

**max(Цена\_продажи) [Максимальная цена],**

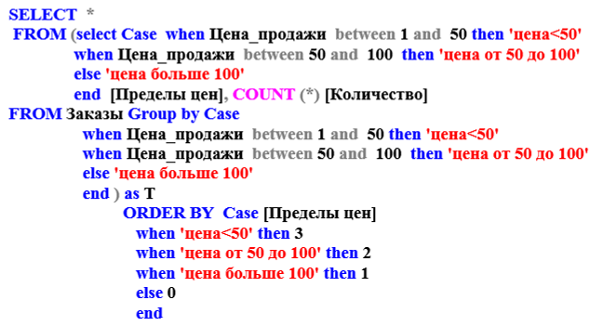
**count(\*) [Количество заказанных товаров]**

**From Заказы Inner Join Товары**

**On Заказы.Наименование\_товара = Товары.Наименование**

**And Товары.Количество >5 Group by Наименование\_товара**

Определить пределы изменения цен и соответствующее количество товаров при продаже:



***→ Выполнение заданий***

select max(AUDITORIUM\_CAPACITY) as [Максимальная вместимость ауд.],

min(AUDITORIUM\_CAPACITY) as [Минимальная вместимость ауд.],

avg(AUDITORIUM\_CAPACITY) as [Средняя вместимость ауд.],

sum(AUDITORIUM\_CAPACITY) as [Суммарная вместимость всех ауд.],

count(\*) as [Общее кол-во ауд.]

from AUDITORIUM;



2.На основе таблиц **AUDITORIUM** и **AUDITORIUM\_TYPE** разработать запрос, вычисляющий для каждого типа аудиторий максимальную, минимальную, среднюю вместимость аудиторий, суммарную вместимость всех аудиторий и общее количество аудиторий данного типа.

Результирующий набор должен содержать столбец с наименованием типа аудиторий (столбец **AUDITORIUM\_TYPE.AUDITORIUM\_TYPENAME**) и столбцы с вычисленными величинами. Использовать внутреннее соединение таблиц, секцию GROUP BY и агрегатные функции.

***→ Выполнение заданий***

select AUDITORIUM\_TYPE.AUDITORIUM\_TYPENAME as [Тип ауд.],

max(AUDITORIUM.AUDITORIUM\_CAPACITY) as [Максимальная вместимость],

min(AUDITORIUM.AUDITORIUM\_CAPACITY) as [Минимальная вместимость],

avg(AUDITORIUM.AUDITORIUM\_CAPACITY) as [Средняя вместимость],

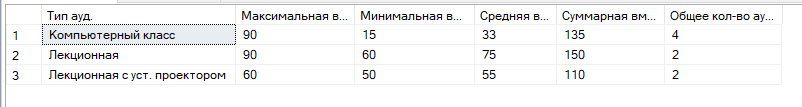
sum(AUDITORIUM.AUDITORIUM\_CAPACITY) as [Суммарная вместимость всех ауд.],

count(\*) as [Общее кол-во ауд. данного типа]

from AUDITORIUM\_TYPE inner join AUDITORIUM

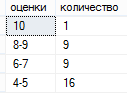
on AUDITORIUM\_TYPE.AUDITORIUM\_TYPE = AUDITORIUM.AUDITORIUM\_TYPE

group by AUDITORIUM\_TYPE.AUDITORIUM\_TYPENAME;



3.Разработать запрос на основе таблицы **PROGRESS**, который содержит количество экзаменационных оценок в заданном интервале. При этом учесть, что сортировка строк должна осуществляться в порядке, обратном величине оценки; сумма значений в столбце **количество** должна быть равна количеству строк в таблице **PROGRESS**.

Использовать подзапрос в секции FROM, в подзапросе применить GROUP BY, сортировку осуществить во внешнем запросе. В секции GROUP BY, в SELECT-списке подзапроса и в ORDER BY внешнего запроса применить CASE.



***→ Выполнение заданий***

select \*

from (select

case

when PROGRESS.NOTE = 10 then '10'

when PROGRESS.NOTE between 8 and 9 then '8-9'

when PROGRESS.NOTE between 6 and 7 then '6-7'

when PROGRESS.NOTE between 4 and 5 then '4-5'

else 'не сдано'

-- обязательно даем столбцу название в следующей строке

end [Оценки],

count(\*) as [Количество]

from PROGRESS

group by case

when PROGRESS.NOTE = 10 then '10'

when PROGRESS.NOTE between 8 and 9 then '8-9'

when PROGRESS.NOTE between 6 and 7 then '6-7'

when PROGRESS.NOTE between 4 and 5 then '4-5'

else 'не сдано'

end

) as T

-- сортировка по значению строки в столбце [Оценки]

order by case [Оценки]

when '10' then 1

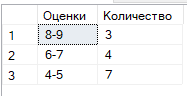
when '8-9' then 2

when '6-7' then 3

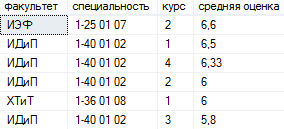
when '4-5' then 4

else 5

end;



4. Разработать SELECT-запрос на основе таблиц **FACULTY**, **GROUPS**, **STUDENT** и **PROGRESS**, который содержит среднюю экзаменационную оценку для каждого курса каждой специальности. Строки отсортировать в порядке убывания средней оценки.



При этом следует учесть, что средняя оценка должна рассчитываться с точностью до двух знаков после запятой. Использовать внутреннее соединение таблиц, агрегатную функцию AVG и встроенные функции CAST и ROUND.

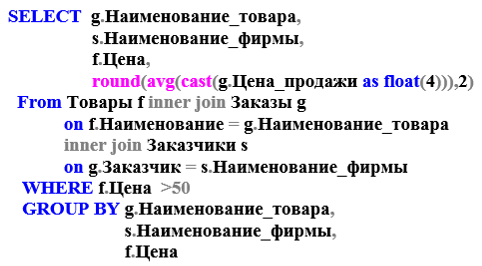
Переписать SELECT-запрос, разработанный в задании 4 так, чтобы в расчете среднего значения оценок использовались оценки только по дисциплинам с кодами **БД** и **ОАиП**. Использовать WHERE.

***→ Теория (методичка)***

Пусть требуется определить наименования товаров, наименования фирм, заказавших эти товары, исходные цены и средние значения цен данных товаров при продаже. При этом исходная цена должна быть больше 50.

Встроенная функция **CAST** используется в запросе для приведения значения цены к вещественному числу.

Функция **ROUND** обеспечивает расчет значений с точностью до двух знаков после запятой:



***→ Выполнение заданий***

-- 4

select FACULTY.FACULTY\_NAME,

GROUPS.PROFESSION,

CASE

WHEN GROUPS.YEAR\_FIRST = 2013 then 1

WHEN GROUPS.YEAR\_FIRST = 2012 then 2

WHEN GROUPS.YEAR\_FIRST = 2011 then 3

WHEN GROUPS.YEAR\_FIRST = 2010 then 4

END [Курс],

round(avg (cast(PROGRESS.NOTE as float(4))), 2) as [Средняя оценка]

from FACULTY inner join GROUPS

on FACULTY.FACULTY = GROUPS.FACULTY

inner join STUDENT

on GROUPS.IDGROUP = STUDENT.IDGROUP

inner join PROGRESS

on STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

group by FACULTY.FACULTY\_NAME,

GROUPS.PROFESSION,

GROUPS.YEAR\_FIRST

order by [Средняя оценка] desc

-- 4 в расчете среднего значения оценок используются оценки только по дисциплинам с кодами БД и ОАиП

select FACULTY.FACULTY\_NAME,

GROUPS.PROFESSION,

-- год поступления ↓

CASE

WHEN GROUPS.YEAR\_FIRST = 2013 then 1

WHEN GROUPS.YEAR\_FIRST = 2012 then 2

WHEN GROUPS.YEAR\_FIRST = 2011 then 3

WHEN GROUPS.YEAR\_FIRST = 2010 then 4

END [Курс],

round(avg (cast(PROGRESS.NOTE as float(4))), 2) as [Средняя оценка]

from FACULTY inner join GROUPS

on FACULTY.FACULTY = GROUPS.FACULTY

inner join STUDENT

on GROUPS.IDGROUP = STUDENT.IDGROUP

inner join PROGRESS

on STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

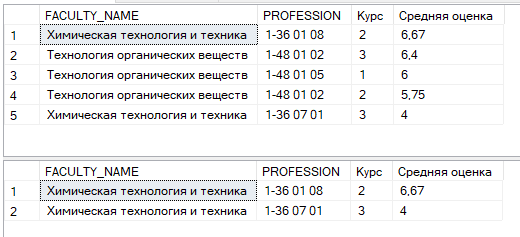
where PROGRESS.SUBJECT in ('БД', 'ОАиП')

group by FACULTY.FACULTY\_NAME,

GROUPS.PROFESSION,

GROUPS.YEAR\_FIRST

order by [Средняя оценка] desc



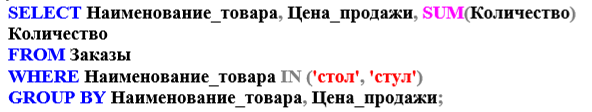
5. На основе таблиц **FACULTY**, **GROUPS**, **STUDENT** и **PROGRESS** разработать SELECT-запрос, в котором выводятся специальность, дисциплины и средние оценки при сдаче экзаменов на факультете ТОВ. Использовать группировку по полям FACULTY, PROFESSION, SUBJECT.

Добавить в запрос конструкцию **ROLLUP** и проанализировать результат.

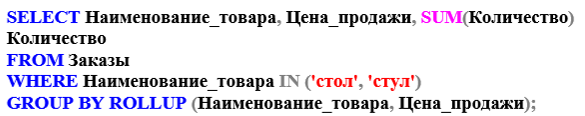
***→ Теория (методичка)***

Конструкции **ROLLUP** и **CUBE** применяются в секции GROUP BY и служат для вычисления значений агрегатных функций для подмножеств строк.

Пусть требуется определить цены продаж и количество продаваемых столов и стульев:



**ROLLUP** возвращает комбинацию групп и итоговых строк, которая опре­делена в порядке, в котором заданы группируемые столбцы:



Конструкция **CUBE** возвращает любую возможную комбинацию групп и итоговых строк.

***→ Выполнение заданий***

-- 5 (без ROLLUP)

select FACULTY.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT,

round(avg (cast(PROGRESS.NOTE as float(4))), 2) as [Средняя оценка]

from FACULTY inner join GROUPS

on FACULTY.FACULTY = GROUPS.FACULTY

inner join STUDENT

on GROUPS.IDGROUP = STUDENT.IDGROUP

inner join PROGRESS

on STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

where FACULTY.FACULTY = 'ТОВ'

group by FACULTY.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT

-- 5 (с ROLLUP)

select FACULTY.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT,

round(avg (cast(PROGRESS.NOTE as float(4))), 2) as [Средняя оценка]

from FACULTY inner join GROUPS

on FACULTY.FACULTY = GROUPS.FACULTY

inner join STUDENT

on GROUPS.IDGROUP = STUDENT.IDGROUP

inner join PROGRESS

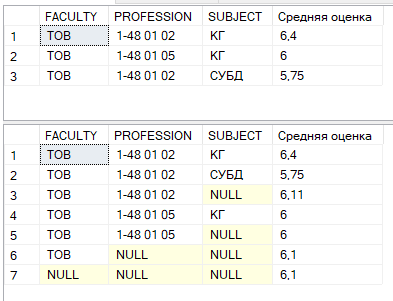
on STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

where FACULTY.FACULTY = 'ТОВ'

group by ROLLUP(FACULTY.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT)



**6.** Выполнить исходный SELECT-запрос п.5 с использованием **CUBE**-группировки. Проанализировать результат.

***→ Выполнение заданий***

select FACULTY.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT,

round(avg (cast(PROGRESS.NOTE as float(4))), 2) as [Средняя оценка]

from FACULTY inner join GROUPS

on FACULTY.FACULTY = GROUPS.FACULTY

inner join STUDENT

on GROUPS.IDGROUP = STUDENT.IDGROUP

inner join PROGRESS

on STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

where FACULTY.FACULTY = 'ТОВ'

group by CUBE(FACULTY.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT)

Суммирование в пределах комбинаций; комбинации (сочетания, точнее):

1) факультет, спец., предмет

2) предмет

3) факультет

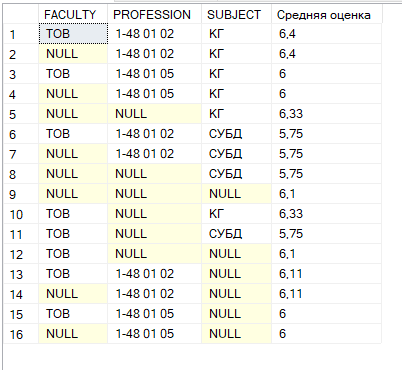
4) спец.

5) спец., предмет

6) факультет, предмет

7) факультет, спец.

+ результирующая суммирующая строка



7. На основе таблиц **GROUPS**, **STUDENT** и **PROGRESS** разработать SELECT-запрос, в котором определяются результаты сдачи экзаменов.

В запросе должны отражаться специальности, дисциплины, средние оценки студентов на факультете ТОВ.

Отдельно разработать запрос, в котором определяются результаты сдачи экзаменов на факультете ХТиТ.

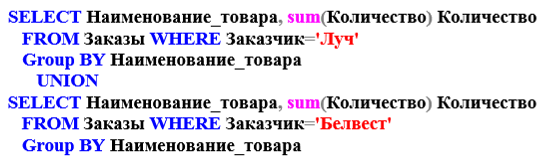
Объединить результаты двух запросов с использованием операторов UNION и UNION ALL. Объяснить результаты.

***→ Теория (методичка)***

Оператор **UNION** выполняет теоретико-множественную операцию объединения, т.е. результатом является множество строк, в котором строки не могут повторяться.

Если требуется механическое объединение строк, можно применить оператор **UNION ALL**.

Получить информацию о заказанных товарах двух фирм и их количестве можно с помощью запроса:



Результатом оператора **INTERSECT** является набор строк, являющийся пересечением двух исходных наборов строк.

Результатом оператора **EXCEPT** является набор строк, являющийся разностью двух исходных наборов строк (т.е. в результирующий набор включаются те строки, которые есть в первом запросе, но отсутствуют во втором).

***→ Выполнение заданий***

-- 7 (union)

select GROUPS.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT,

round(avg (cast(PROGRESS.NOTE as float(4))), 2) as [Средняя оценка]

from GROUPS inner join STUDENT

on GROUPS.IDGROUP = STUDENT.IDGROUP

inner join PROGRESS

on STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

where GROUPS.FACULTY = 'ТОВ'

group by GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT

union

select GROUPS.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT,

round(avg (cast(PROGRESS.NOTE as float(4))), 2) as [Средняя оценка]

from GROUPS inner join STUDENT

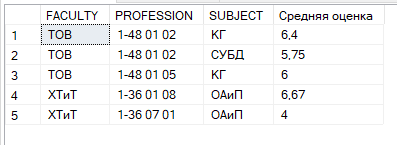
on GROUPS.IDGROUP = STUDENT.IDGROUP

inner join PROGRESS

on STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

where GROUPS.FACULTY = 'ХТиТ'

group by GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT



-- 7 (union all)

select GROUPS.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT,

round(avg (cast(PROGRESS.NOTE as float(4))), 2) as [Средняя оценка]

from GROUPS inner join STUDENT

on GROUPS.IDGROUP = STUDENT.IDGROUP

inner join PROGRESS

on STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

where GROUPS.FACULTY = 'ТОВ'

group by GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT

union all

select GROUPS.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT,

round(avg (cast(PROGRESS.NOTE as float(4))), 2) as [Средняя оценка]

from GROUPS inner join STUDENT

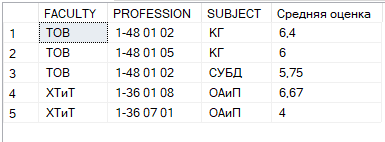
on GROUPS.IDGROUP = STUDENT.IDGROUP

inner join PROGRESS

on STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

where GROUPS.FACULTY = 'ХТиТ'

group by GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT



8.Получить пересечение двух множеств строк, созданных в результате выполнения запросов пункта 7. Объяснить результат.

Использовать оператор INTERSECT.

***→ Выполнение заданий***

select GROUPS.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT,

round(avg (cast(PROGRESS.NOTE as float(4))), 2) as [Средняя оценка]

from GROUPS inner join STUDENT

on GROUPS.IDGROUP = STUDENT.IDGROUP

inner join PROGRESS

on STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

where GROUPS.FACULTY = 'ТОВ'

group by GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT

intersect

select GROUPS.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT,

round(avg (cast(PROGRESS.NOTE as float(4))), 2) as [Средняя оценка]

from GROUPS inner join STUDENT

on GROUPS.IDGROUP = STUDENT.IDGROUP

inner join PROGRESS

on STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

where GROUPS.FACULTY = 'ХТиТ'

group by GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT

В результатах двух запросов нет одинаковых строк, которые могли бы стать результатом пересечения этих запросов. Так что в итоге получаем пустой вывод.



9.Получить разницу между множеством строк, созданных в результате запросов пункта 7.

Объяснить результат.

Использовать оператор EXCEPT.

***→ Выполнение заданий***

select GROUPS.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT,

round(avg (cast(PROGRESS.NOTE as float(4))), 2) as [Средняя оценка]

from GROUPS inner join STUDENT

on GROUPS.IDGROUP = STUDENT.IDGROUP

inner join PROGRESS

on STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

where GROUPS.FACULTY = 'ТОВ'

group by GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT

except

select GROUPS.FACULTY,

GROUPS.PROFESSION,

PROGRESS.SUBJECT,

round(avg (cast(PROGRESS.NOTE as float(4))), 2) as [Средняя оценка]

from GROUPS inner join STUDENT

on GROUPS.IDGROUP = STUDENT.IDGROUP

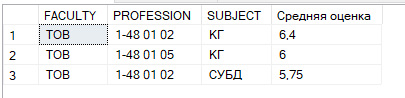
inner join PROGRESS

on STUDENT.IDSTUDENT = PROGRESS.IDSTUDENT

where GROUPS.FACULTY = 'ХТиТ'

group by GROUPS.FACULTY, GROUPS.PROFESSION, PROGRESS.SUBJECT

Так как результатами двух запросов были разные строки, то мы получаем нетронутый результат первого запроса.



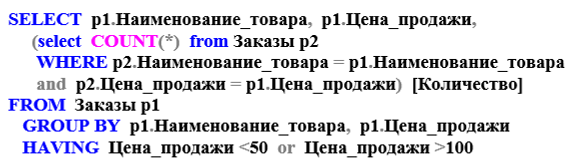
10.На основе таблицы **PROGRESS** определить для каждой дисциплины количество студентов, получивших оценки 8 и 9.

Использовать группировку, секцию HAVING, сортировку.

***→ Теория (методичка)***

Логическое выражение, указанное в секции HAVING вычисляется для каждой строки результирующего набора, сформированного секцией GROUP BY. Как и в случае с секцией WHERE строка отбирается в результирующий набор, если логическое выражение принимает значение «истина».

Наименования заказанных товаров, цена которых меньше 50 или больше 100, и их количество (таблица **Заказы**):



***→ Выполнение заданий***

select p1.IDSTUDENT as [ИД студента],

p1.NOTE as [Оценка],

(select count(\*) from PROGRESS as p2

where p1.IDSTUDENT = p2.IDSTUDENT

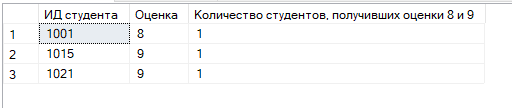
and p1.NOTE = p2.NOTE

) as [Количество студентов, получивших оценки 8 и 9]

from PROGRESS as p1

group by p1.IDSTUDENT, p1.NOTE

having p1.NOTE in (8, 9)



11. Разработать и выполнить аналогичные запросы для базы данных **X\_MyBASE**.

-- 1 таблица "Водители на перевозках", найти макс., мин., сред., общую оплату за перевозки и общее кол-во перевозок

select max([Оплата, USD]) as [Максимальная оплата],

min([Оплата, USD]) as [Минимальная оплата],

avg([Оплата, USD]) as [Средняя оплата],

sum([Оплата, USD]) as [Общая оплата за все перевозки],

count(\*) as [Кол-во перевозок]

from [Водители на перевозках]



-- 2 все таблицы базы данных, для каждого водителя определить макс., мин., сред., суммарн. дальности перевозок и общее кол-во перевозок

select [Водители].Код\_водителя,

[Водители].Фамилия,

[Водители].Имя,

max([Дальность, км]) as [Максимальная дальность],

min([Дальность, км]) as [Минимальная дальность],

avg([Дальность, км]) as [Средняя дальность],

sum([Дальность, км]) as [Общая дальность всех перевозок],

count(\*) as [Кол-во перевозок]

from [Водители] inner join [Водители на перевозках]

on [Водители].Код\_водителя = [Водители на перевозках].Код\_водителя

inner join [Перевозки]

on [Водители на перевозках].Код\_перевозки = [Перевозки].Код\_перевозки

inner join [Маршруты]

on [Перевозки].Код\_маршрута = [Маршруты].Код\_маршрута

group by [Водители].Код\_водителя, [Водители].Фамилия, [Водители].Имя



-- 3 таблица "Маршруты", найти количество диапазонов дальностей маршрутов в заданном диапазоне, при этом осущесвтить сортировку по пределам цен в порядке возрастания

select \*

from (

select case

when Маршруты.[Дальность, км] < 200 then 'меньше 200'

when Маршруты.[Дальность, км] between 200 and 500 then 'от 200 до 500'

else 'больше 500'

end as [Пределы дальностей, км],

count(\*) as [Количество маршрутов]

from Маршруты group by

case

when Маршруты.[Дальность, км] < 200 then 'меньше 200'

when Маршруты.[Дальность, км] between 200 and 500 then 'от 200 до 500'

else 'больше 500'

end

) as T

order by

case [Пределы дальностей, км]

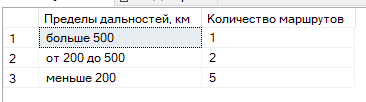
when 'меньше 200' then 3

when 'от 200 до 500' then 2

when 'больше 500' then 1

else 0

end



-- 5 Таблицы "Водители", "Водители на перевозках", "Перевозки" просмотреть общую оплату для каждого водителя за все перевозки, осуществленные после 2021-11-12

select Водители.Фамилия,

Водители.Имя,

Водители.Код\_водителя,

sum([Водители на перевозках].[Оплата, USD]) as [Общая оплата за перевозки после 2021-11-12]

from [Водители на перевозках] inner join Перевозки

on ([Водители на перевозках].Код\_перевозки = Перевозки.Код\_перевозки

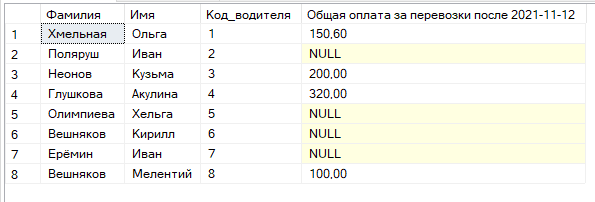
and Перевозки.Дата\_возвращения > '20211112')

right join Водители

on Водители.Код\_водителя = [Водители на перевозках].Код\_водителя

group by Водители.Фамилия, Водители.Имя, Водители.Код\_водителя

Получаем вообще всех водителей, благодаря left join, а для тех из них, кто осуществлял перевозки после определенной даты, высчитываем общую оплату за эти перевозки.



--5 (ROLLUP)

select Водители.Фамилия,

Водители.Имя,

Водители.Код\_водителя,

sum([Водители на перевозках].[Оплата, USD]) as [Общая оплата за перевозки после 2021-11-12]

from [Водители на перевозках] inner join Перевозки

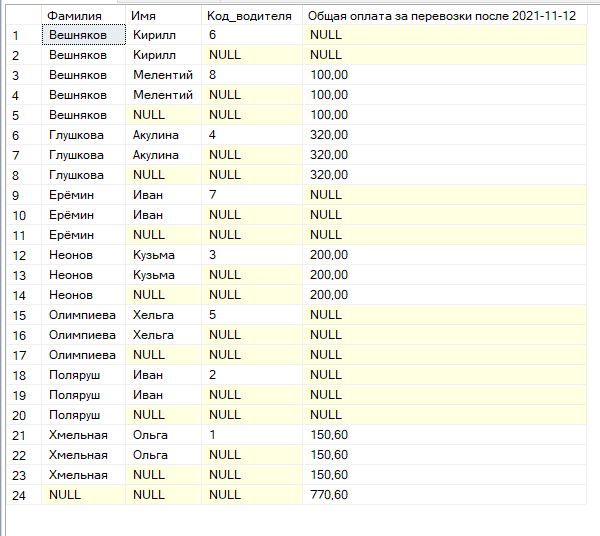
on ([Водители на перевозках].Код\_перевозки = Перевозки.Код\_перевозки

and Перевозки.Дата\_возвращения > '20211112')

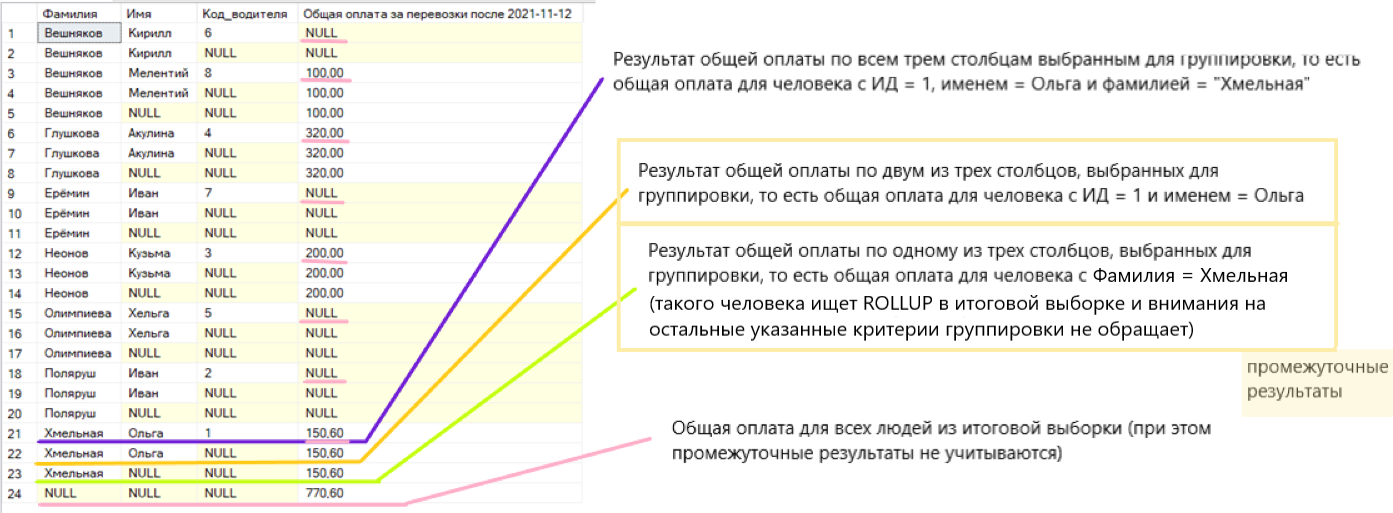
right join Водители

on Водители.Код\_водителя = [Водители на перевозках].Код\_водителя

group by ROLLUP(Водители.Фамилия, Водители.Имя, Водители.Код\_водителя)



В промежуточных результатах как бы исключатся последовательно критерии, и мы суммируем группы, не обращая внимания, сначала на 3-ий критерий, затем на 3-ий и 2-ой, последней строкой на все три (результирующая сумма).



-- 6 (CUBE)

select Водители.Фамилия,

Водители.Имя,

Водители.Код\_водителя,

sum([Водители на перевозках].[Оплата, USD]) as [Общая оплата за перевозки после 2021-11-12]

from [Водители на перевозках] inner join Перевозки

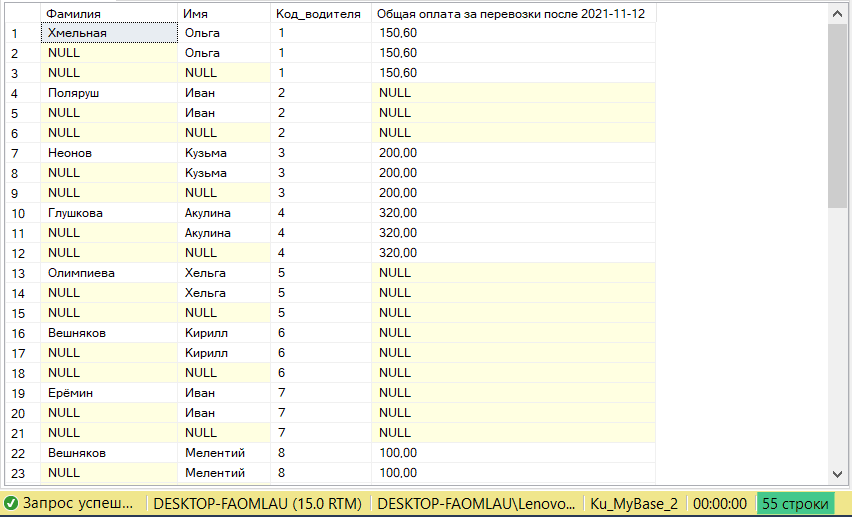
on ([Водители на перевозках].Код\_перевозки = Перевозки.Код\_перевозки

and Перевозки.Дата\_возвращения > '20211112')

right join Водители

on Водители.Код\_водителя = [Водители на перевозках].Код\_водителя

group by CUBE(Водители.Фамилия, Водители.Имя, Водители.Код\_водителя)



12\*. Подсчитать количество студентов в каждой группе, на каждом факультете и всего в университете одним запросом.

-- 12.1

select FACULTY.FACULTY,

GROUPS.IDGROUP,

count (\*) as [Кол-во студентов]

from FACULTY inner join GROUPS

on FACULTY.FACULTY = GROUPS.FACULTY

inner join STUDENT

on GROUPS.IDGROUP = STUDENT.IDGROUP

group by rollup(FACULTY.FACULTY, GROUPS.IDGROUP)

