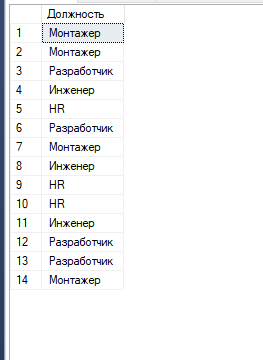
Лабораторная работа №1. Расширенные возможности SELECT.

Задание:

1. Продемонстрируйте работу простого запроса на группировку с отбором записей в исходном наборе до группировки и отбором получившихся групп в результирующем наборе.

До группировки

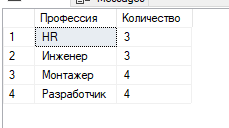
select [career\_objective] AS [Должность] from staff



После группировки

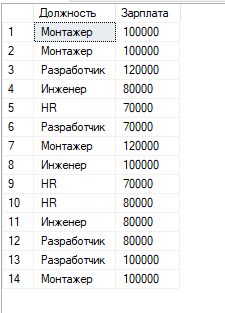
Select [career\_objective] AS Профессия, COUNT ([career\_objective]) AS Количество From staff

Group By [career\_objective]



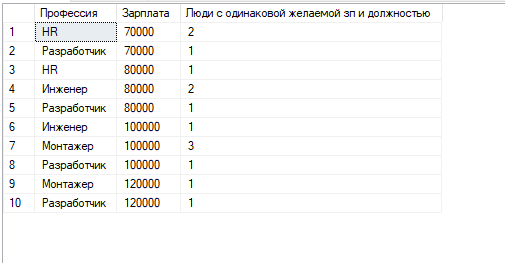
2. Повторите п.1, используя при этом группировку по двум-трём столбцам.

select [career\_objective] AS [Должность], [salary] AS [Зарплата] from staff



Select [career\_objective] AS Профессия,[salary] AS [Зарплата], COUNT ([career\_objective]) AS [Люди с одинаковой желаемой зп и должностью] From staff

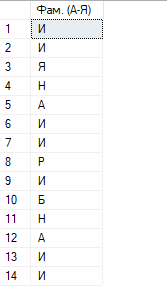
Group By [career\_objective], [salary]



3. Повторите п.1, используя группировку по вычисляемому выражению.

select substring (surnamename, 1, 1)as [Фам. (А-Я)]

from staff



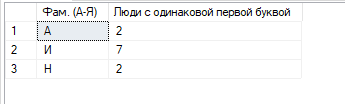
select substring (surnamename, 1, 1)as [Фам. (А-Я)],

count(substring (surnamename, 1, 1)) as [Люди с одинаковой первой буквой]

from staff

group by substring (surnamename, 1, 1)

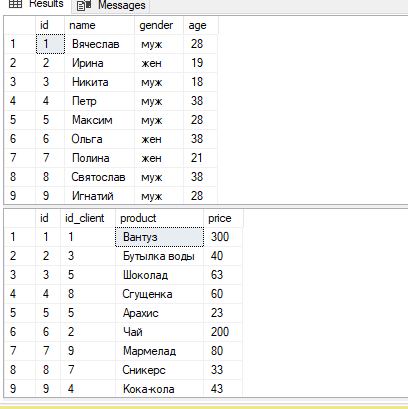
having count(\*)>1



4. Продемонстрируйте работу простого запроса с использованием группировки по результату соединения (join) имеющихся таблиц. Покажите проблему группировки кортежей подчинённой таблицы по неуникальному полю одной из связанных таблиц. Продемонстрируйте более правильный вариант группировки.

select \* from client

select \* from orders

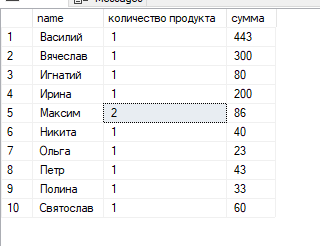


select name, count(product) as [количество продукта], sum(price) as [сумма]

from client

join orders on client.id =orders.id\_client

group by name

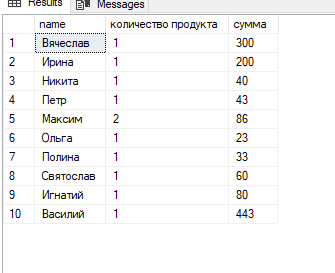


select name, count(product) as [количество продукта], sum(price) as [сумма]

from client

join orders on client.id =orders.id\_client

group by orders.id\_client,name



#5. Модифицируйте запрос из п.4 так, чтобы в нём появились подитоги по иерархии значений в столбцах группировки. Используйте для этого оператор ROLLUP. При этом покажите использование различного количества столбцов в операторе ROLLUP.

Select name , career\_objective, SUM(salary) AS itog

From staff

Group by ROLLUP (name,career\_objective)



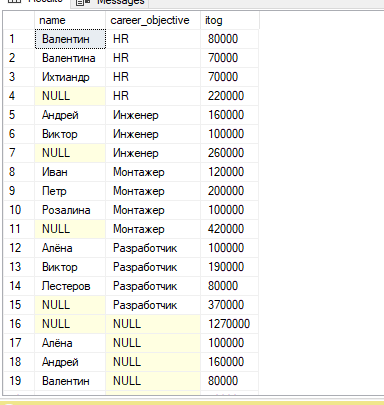
#6. Модифицируйте запрос из п.4 так, чтобы в нём появились подитоги по комбинациям значений в столбцах группировки. Используйте для этого оператор CUBE. При этом покажите использование различного количества столбцов в операторе CUBE.

CUBE — оператор, который формирует результаты для всех возможных перекрестных вычислений.

Select name , career\_objective, SUM(salary) AS itog

From staff

Group by CUBE (name,career\_objective)



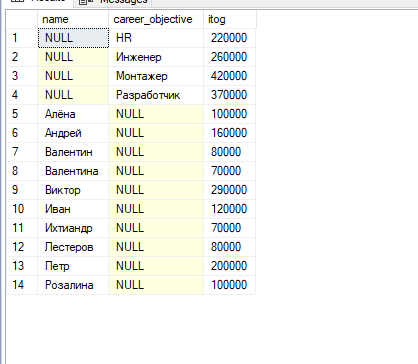
#7. Модифицируйте запрос из п.4. установив при помощи оператора GROUPING SETS произвольный набор конфигураций уровней блокирования.

GROUPING SETS – оператор, который формирует результаты нескольких группировок в один набор данных.

Select name , career\_objective, SUM(salary) AS itog

From staff

Group by GROUPING SETS (name,career\_objective)



#9. При помощи оператора GROUPING отделите в итоговом наборе запроса из п.4 значения NULL, показывающие исключение соответствующего атрибута из группирования от значений NULL, показывающие отсутствующие значения. Для этого замените первые - на строку «ВСЕ», а вторые – на строку «НЕИЗВЕСТНО».

GROUPING – функция, которая возвращает истину, если указанное выражение является статистическим, и ложь, если выражение нестатистическое.

Select name,

ISNULL(CAST(career\_objective as nvarchar(50)),

case when GROUPING(career\_objective)=1 and GROUPING(name)=0

then 'Промежуточный итог' else 'Общий итог' end) as career\_objective,

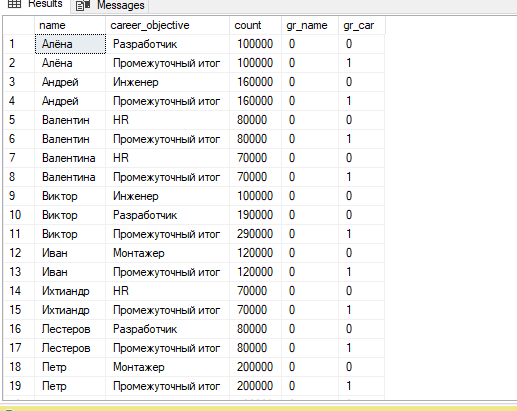
sum(salary) as count,

GROUPING(name) as gr\_name,

grouping(career\_objective) as gr\_car

from staff

group by rollup (name, career\_objective)



#10. Продемонстрируйте работу ранжирующих функций RANK, DENSE\_RANK, ROW\_NUMBER и NTILE. Наглядно покажите разницу между ними.

ROW\_NUMBER – функция нумерации которая возвращает просто номер строки.

RANK – ранжирующая функция, которая возвращает ранг каждой строки. В данном случае, в отличие от row\_number(), идет уже анализ значений и в случае нахождения одинаковых, функция возвращает одинаковый ранг с пропуском следующего.

DENSE\_RANK — ранжирующая функция, которая возвращает ранг каждой строки, но в отличие от rank, в случае нахождения одинаковых значений, возвращает ранг без пропуска следующего.

NTILE – функция, которая делит результирующий набор на группы по определенному столбцу. Количество групп указывается в качестве параметра. В случае если в группах получается не одинаковое количество строк, то в первой группе будет наибольшее количество, например, в нашем случае строк 10 и если мы поделим на три группы, то в первой будет 4 строки, а во второй и третей по 3.

select

surnamename,name,[salary],

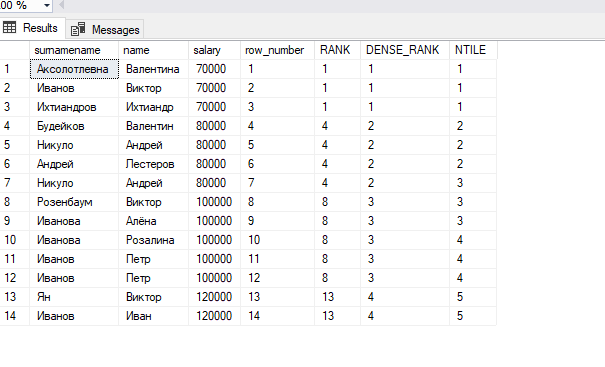
ROW\_NUMBER() OVER (ORDER BY [salary]) as row\_number ,

RANK() OVER (ORDER BY [salary]) as RANK ,

DENSE\_RANK() OVER (ORDER BY [salary]) as DENSE\_RANK,

NTILE(5) OVER (ORDER BY [salary]) as NTILE

from staff



11. Повторите пункт 10, но с применением оконных функций. В качестве критерия выделения окон можно выбрать отдел или должность для таблицы «Сотрудники» или категорию для таблицы «Товары».

select DISTINCT

client.[name] as [Имя заказчика],

COUNT (orders.[id\_client]) OVER (PARTITION BY quantity) AS [Общее кол-во закупок],

SUM(price) OVER (PARTITION BY quantity) AS [Общая цена закупок],

AVG(price) OVER (PARTITION BY quantity) AS [Cредняя цена закупок],

MAX(price) OVER (PARTITION BY quantity) AS [Максимальная цена закупок],

MIN(price) OVER (PARTITION BY quantity) AS [CМинимальная цена закупок]

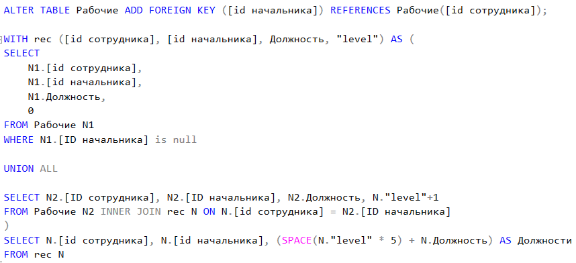
from orders

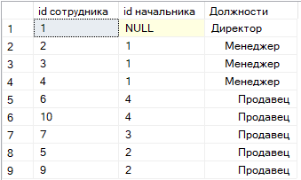
left join client on orders.[id\_client]=client.[id]



12. Добавьте в одну из таблиц Вашей схемы атрибут, который будет являться внешним ключом, указывающим на записи этой же таблицы. Например, в таблицу «Сотрудники»

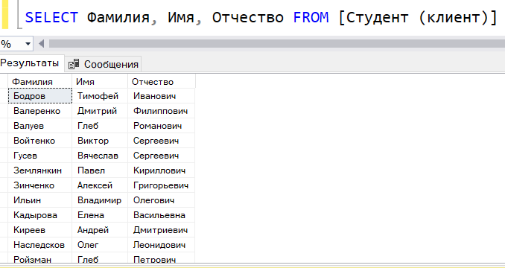
добавьте информацию о руководителе для каждого сотрудника или для таблицы «Товары» - информацию о сопутствующем товаре, который прилагается к данному товару в подарок по рекламной акции. Составьте рекурсивное табличное выражение, в котором наглядно выведите записи вашей таблицы в порядке их иерархии. Предусмотрите визуальное отображение иерархии, например, при помощи отступов различной величины.



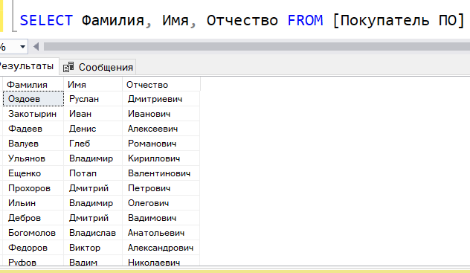


1. **Вставка значений из таблиц покупателей и студентов в единую таблицу, с использованием оператора MERGE**

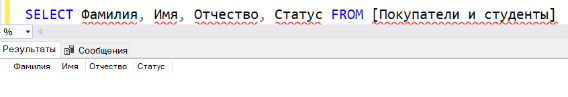
Имеется таблица студентов:



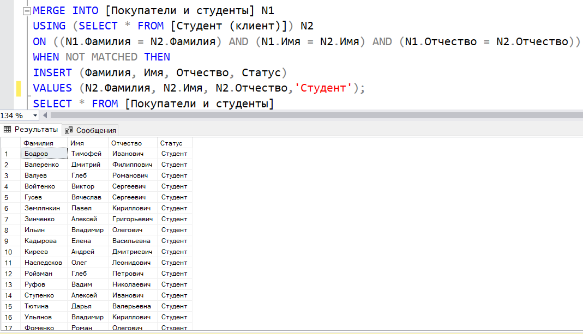
А также таблица покупателей:



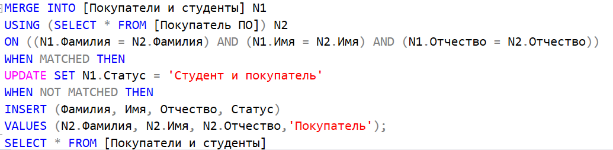
Создадим новую таблицу покупателей и студентов:

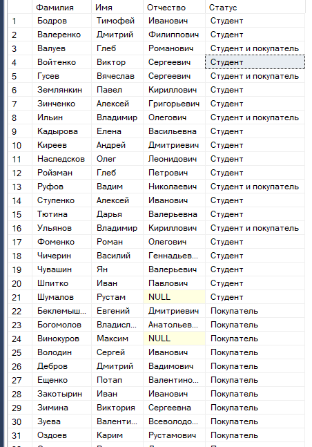


Напишем запрос с оператором MERGE, который вставит значения из таблицы студентов в новую таблицу:



Теперь напишем запрос, который вставит покупателей в новую таблицу:





**Ответы к отчету**

№1. Простая группировка данных. Операторы group by, having.

**GROUP BY** – это оператор для группировки данных по полю, при использовании в запросе агрегатных функций, таких как sum, max, min, count и других.

Оператор **HAVING** определяет, какие группы будут включены в выходной результат, то есть выполняет фильтрацию групп.

№2. Ограничения оператора group by.

Для предложения **GROUP BY**, использующего **ROLLUP**, **CUBE** или **GROUPING SETS**, используется **максимум 32 выражения**. Максимальное количество групп — 4096 (212).

**Группировка по вычисляемым выражениям**

Любой столбец, который используется в выражении SELECT (не считая столбцов, которые хранят результат агрегатных функций), должны быть указаны после оператора GROUP BY.

И если в выражении SELECT производится выборка по одному или нескольким столбцам и также используются агрегатные функции, то необходимо использовать выражение GROUP BY.

Если столбец, по которому производится группировка, содержит значение NULL, то строки со значением NULL составят отдельную группу.

№3. Группировка по вычисляемым выражениям.

№4. Составная группировка по двум и более столбцам. В чём её смысл?

В предложениях GROUP BY можно указывать столько столбцов, сколько необходимо. Таким образом, путем группировки одновременно по нескольким элементам можно создавать группы внутри групп.

№5. Промежуточные подитоги. ROLLUP, CUBE и GROUPING SETS. Принципы работы, отличия, взаимоотношения?

**ROLLUP** – оператор, который формирует промежуточные итоги для каждого указанного элемента и общий итог.

**CUBE** — оператор, который формирует результаты для всех возможных перекрестных вычислений.

**GROUPING SETS** – оператор, который формирует результаты нескольких группировок в один набор данных.

№6. Ранжирующие функции. Их возможности и сферы применения. Указание критериев ранжирования?

Ранжирующие функции — это функции, которые возвращают значение для каждой строки группы в результирующем наборе данных. На практике они могут быть использованы, например, для простой нумерации списка, составления рейтинга или постраничной выборки.

**ROW\_NUMBER** – функция нумерации в Transact-SQL, которая возвращает просто номер строки.

**RANK** – ранжирующая функция, которая возвращает ранг каждой строки. В данном случае, в отличие от row\_number(), идет уже анализ значений и в случае нахождения одинаковых, функция возвращает одинаковый ранг с пропуском следующего. Как было уже сказано выше, здесь также можно использовать partition by для группировки и обязательно нужно указывать столбец сортировки в order by.

**DENSE\_RANK** — ранжирующая функция, которая возвращает ранг каждой строки, но в отличие от rank, в случае нахождения одинаковых значений, возвращает ранг без пропуска следующего.

NTILE – функция Transact-SQL, которая делит результирующий набор на группы по определенному столбцу.  
№7. Оконные функции. Применение совместно с агрегатными или ранжирующими функциями.

Оконные функции - это функции применяемые к набору строк так или иначе связанных с текущей строкой. Наверняка всем известны классические агрегатные функции вроде AVG, SUM, COUNT, используемые при группировке данных. В результате группировки количество строк уменьшается, оконные функции напротив никак не влияют на количество строк в результате их применения, оно остаётся прежним.

Привычные нам агрегатные функции также могут быть использованы в качестве оконных функций, нужно лишь добавить выражение определения "окна". Область применения оконных функций чаще всего связана с аналитическими запросами, анализом данных.

Оконные функции начинаются с оператора OVER и настраиваются с помощью трёх других операторов: PARTITION BY, ORDER BY и ROWS.

№8. Обобщённые табличные выражения. Сферы применения обобщённых табличных выражений.

Common Table Expression (CTE) или обобщенное табличное выражение (OTB) – это временные результирующие наборы (т.е. результаты выполнения SQL запроса), которые не сохраняются в базе данных в виде объектов, но к ним можно обращаться.

* Основной целью OTB является написание рекурсивных запросов, можно сказать для этого они, и были созданы;
* OTB можно использовать также и для замены представлений (VIEW), например, в тех случаях, когда нет необходимости сохранять в базе SQL запрос представления, т.е. его определение;
* Обобщенные табличные выражения повышают читаемость кода путем разделения запроса на логические блоки, и тем самым упрощают работу со сложными запросами;
* Также OTB предназначены и для многократных ссылок на результирующий набор из одной и той же SQL инструкции.

№9. Организация рекурсивных запросов при помощи обобщённых табличных выражений.

Главной особенностью обобщенных табличных выражений является то, что с помощью них можно писать рекурсивные запросы.

Обобщенное табличное выражение определяется с помощью конструкции WITH, и определить его можно как в обычных запросах, так и в функциях, хранимых процедурах, триггерах и представлениях.

Использоваться может, например, для рекурсивного вывода работников, их должностей и их начальников в иерархическом виде.

№10. Оператор слияния наборов MERGE. Возможности и сферы его применения.

**MERGE** – операция в языке T-SQL, при которой происходит обновление, вставка или удаление данных в таблице на основе результатов соединения с данными другой таблицы или SQL запроса. Другими словами, с помощью MERGE можно осуществить слияние двух таблиц, т.е. синхронизировать их.

В операции MERGE происходит объединение по ключевому полю или полям основной таблицы (в которой и будут происходить все изменения) с соответствующими полями другой таблицы или результата запроса. В итоге если условие, по которому происходит объединение, истина (**WHEN MATCHED**), то мы можем выполнить операции обновления или удаления, если условие не истина, т.е. отсутствуют данные (**WHEN NOT MATCHED**), то мы можем выполнить операцию вставки (INSERT добавление данных), также если в основной таблице присутствуют данные, которое отсутствуют в таблице (или результате запроса) источника (**WHEN NOT MATCHED BY SOURCE**), то мы можем выполнить обновление или удаление таких данных.