

# Дисциплина: Базы данных



*«Data Manipulation Language: Select»*

Преподаватель:  
Лут А.В.

# Data Manipulation Language (DML)

Операторы выборки (*SELECT*), вставки (*INSERT*), обновления (*UPDATE*) и удаления (*DELETE*) относятся к операторам манипулирования данными. Эту группу называют также *DATA MANIPULATION LANGUAGE (DML)*.

## Оператор SELECT

Оператор *SELECT* довольно сложен, но основные его возможности описываются следующей конструкцией:

***SELECT <список выражений выборки>***

***[FROM <список источников данных>]***

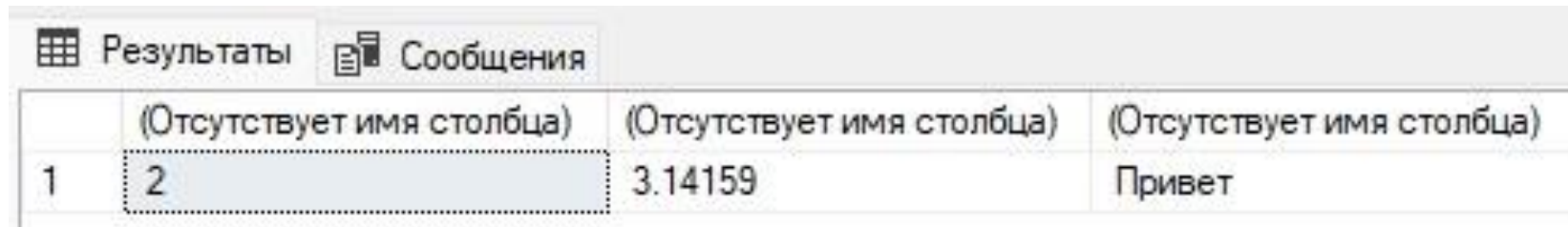
***[WHERE <условие выборки>]***

***[ORDER BY <список выражений, по которым выполняется сортировка>] [ASC|DESC]***

Результатом работы оператора SELECT является таблица со столбцами, перечисленными в списке полей выборки.

Выражение из списка выражений выборки – произвольное выражение языка Transact SQL:

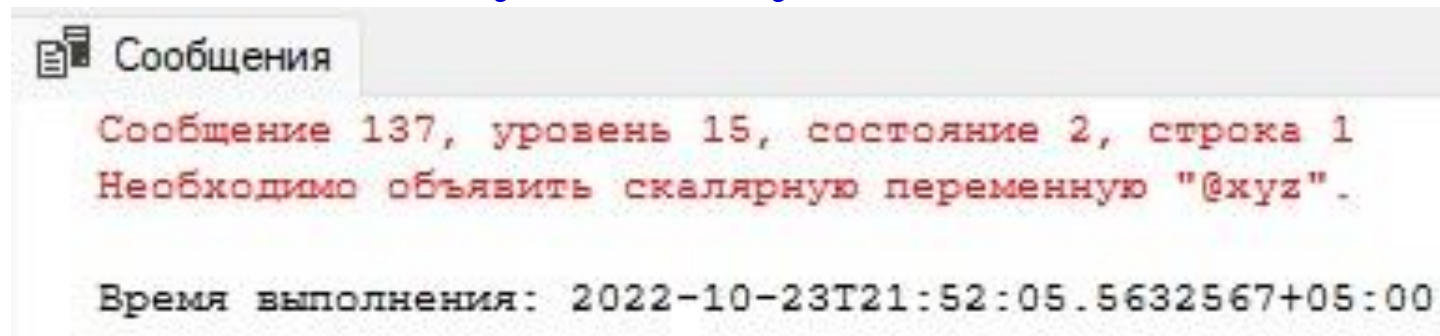
1) константа любого типа, например: **SELECT 2,3.14159,'Привет'**



Результаты Сообщения			
	(Отсутствует имя столбца)	(Отсутствует имя столбца)	(Отсутствует имя столбца)
1	2	3.14159	Привет

Рисунок 1 – Результат константного запроса

2) переменная: **SELECT @xyz, @MyVar**



Почему  
ошибка?

Рисунок 2 – Ошибка после запроса переменных

3) **\$IDENTITY** — значение столбца, обладающего свойством IDENTITY:

`select $IDENTITY from Passgr  
order by $IDENTITY`

Результаты		Сообщения	
	Passgr_ID		
1	3		
2	4		
3	5		
4	7		
5	9		
6	11		
7	15		
8	19		
9	20		
10	21		
11	25		
12	27		
13	28		
14	29		
15	30		
16	32		
17	33		

Рисунок 3 —  
Запрос \$IDENTITY

4) **\$ROWGUID** — значение столбца типа uniqueidentifier

`select * from Passgr`

Результаты		Сообщения			
	Passgr_ID	Flight_ID	Seat_ID	FIO	Passp
1	3	1	1	Сидоров Сидор Сидорович	66 00 432326
2	4	3	10	Аслямов Ильдар Флоридович	23 00 777777
3	5	7	30	Варлей Наталья Михайловна	76 00 888888
4	7	11	10	Шумахер Михаэль Васильевич	765432
5	9	12	10	Пушкин Александр Сергеевич	66 88 54321
6	11	13	1	Петряев Василий Батькович	75 00 234566
7	15	23	46432	Иванов Иван Иванович	54 98 654321
8	19	20	46431	Горбачев Михаил Сергеевич	23 54 456789
9	20	3	10	Петров Петр Петрович	75 00 765234
10	21	5	46432	Федоров Федор Федорович	78 00 543987
11	25	5	52107	Михайлов Михаил Михайлович	76 00 543751
12	27	1	46428	Пётр Иванович Неуважай-Корыто	55 00 999999
13	28	23	52062	Джонсон	66 00 555555
14	29	24	43418	Шестаков Александр	44 345876
15	30	24	52102	Медников Пётр Иванович	66 876123
16	32	26	43418	Шестаков Александр	66 876123
17	33	25	52104	Шестаков Александр	66 876123

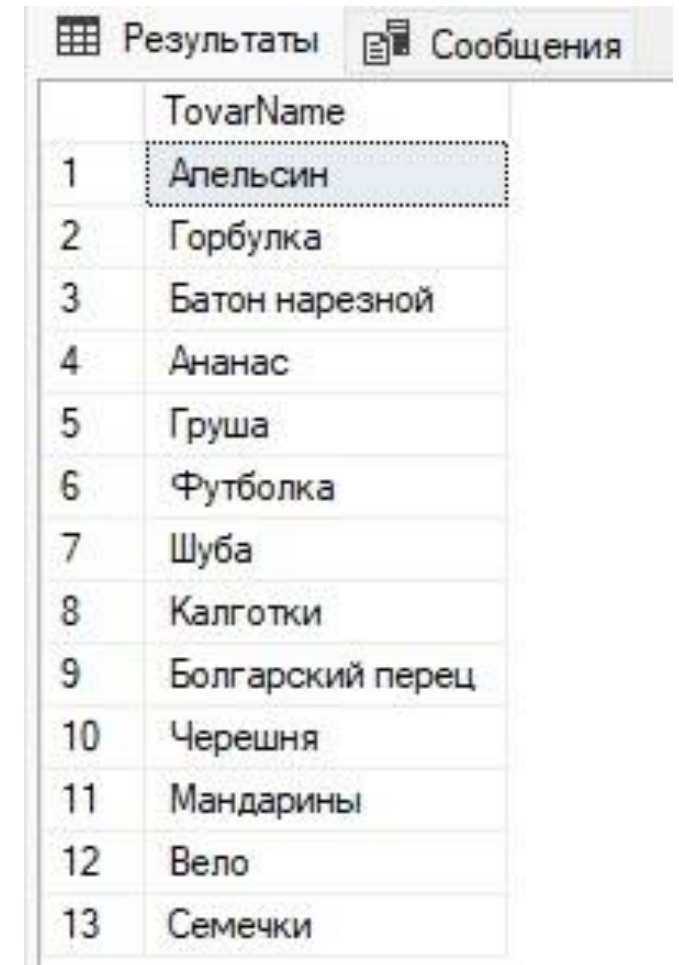
Рисунок 4 — Все данные в таблице Passgr

5) имя поля из таблицы базы данных. Синтаксис обращения к полю таблицы имеет в общем случае вид

[<имя базы данных>.] [<имя схемы>.] [<имя таблицы>.]<имя поля>

```
SELECT BD_Warehouse.dbo.Tovar.TovarName  
FROM BD_Warehouse.dbo.Tovar  
WHERE IsTovar=1
```

Таким образом, можно выполнять запрос находясь в другой базе данных.



The screenshot shows a window titled 'Результаты' (Results) with a sub-tab 'Сообщения' (Messages). It displays a table with two columns: an index and 'TovarName'. The first row is highlighted with a dotted border.

	TovarName
1	Апельсин
2	Горбулка
3	Батон нарезной
4	Ананас
5	Груша
6	Футболка
7	Шуба
8	Калготки
9	Болгарский перец
10	Черешня
11	Мандарины
12	Вело
13	Семечки

Рисунок 5 – Запрос на поле таблицы



Данные извлекаются из источников данных, в качестве которых могут выступать:

1) таблица базы данных, например: **SELECT A.x, B.y from A,B**

**SELECT** Seat.Seat\_ID, Passgr\_ID

**FROM** Passgr, Seat

2) view (представление) – это виртуальная таблица, основанная на наборе результатов оператора SQL.

**CREATE VIEW** NewCity **AS**

**SELECT** City\_ID, CityName

**FROM** City

**WHERE** City\_ID>3

**SELECT \***

**FROM** NewCity

**DROP VIEW** NewCity



	City_ID	CityName
1	4	Магадан
2	5	Екатеринбург
3	209	Чебоксары
4	359	Мелитополь
5	431	Миасс
6	440	Абакан
7	455	Санкт-Петербург
8	572	Полетаево

Рисунок 6 –  
Запрос на view

3) функция, возвращающая таблицу (будет дальше)

`select *`

`from [dbo].[TicketSales] ('20000101', '20240101')`

4) оператор *SELECT*, например:

`select A.FIO, A.SeatName`

`from (select FIO, SeatName`

`from Passgr P, Seat S`

`where P.Seat_ID=S.Seat_ID) as A`

5) соединение (имеется в виду операция соединения реляционной алгебры) двух и более таблиц (через join).

Результаты			Сообщения		
	FIO	SeatName			
1	Сидоров Сидор Сидорович	1А			
2	Аслямов Ильдар Флоридович	18А			
3	Варлей Наталья Михайловна	20Б			
4	Шумахер Михаэль Васильевич	18А			
5	Пушкин Александр Сергеевич	18А			
6	Петряев Василий Батькович	1А			
7	Иванов Иван Иванович	3Б			
8	Горбачев Михаил Сергеевич	1А			
9	Петров Петр Петрович	18А			
10	Федоров Федор Федорович	3Б			
11	Михайлов Михаил Михайлович	25Г			
12	Пётр Иванович Неуважай-Корыто	23А			
13	Джонсон	16А			
14	Шестаков Александр	1А			
15	Медников Пётр Иванович	18А			
16	Шестаков Александр	1А			
17	Шестаков Александр	20Б			

Рисунок 7 – Select  
как источник данных

## Синтаксис соединения (join):

<источник 1>

[<тип соединения>] JOIN <источник 2>

ON <условие соединения>

Пример:

```
select FIO, SeatName, PlaneTypeName
```

```
from Passgr P
```

```
[INNER] JOIN Seat S ON P.Seat_ID=S.Seat_ID
```

```
[INNER] JOIN PlaneType PT ON PT.PlaneType_ID=S.PlaneType_ID
```

```
select FIO, SeatName, PlaneTypeName -- тоже самое
```

```
from Passgr P, Seat S, PlaneType PT
```

```
WHERE PT.PlaneType_ID=S.PlaneType_ID
```

```
AND P.Seat_ID=S.Seat_ID
```

	FIO	SeatName	PlaneTypeName
1	Сидоров Сидор Сидорович	1А	ТУ-134
2	Аслямов Ильдар Флоридович	18А	ТУ-134
3	Варлей Наталья Михайловна	20Б	ТУ-134
4	Шумахер Михаэль Васильевич	18А	ТУ-134
5	Пушкин Александр Сергеевич	18А	ТУ-134
6	Петряев Василий Батькович	1А	ТУ-134
7	Иванов Иван Иванович	3Б	Боинг-747
8	Горбачев Михаил Сергеевич	1А	Боинг-747
9	Петров Петр Петрович	18А	ТУ-134
10	Федоров Федор Федорович	3Б	Боинг-747
11	Михайлов Михаил Михайлович	25Г	Боинг-747
12	Пётр Иванович Неуважай-Корыто	23А	АН-2
13	Джонсон	16А	А-310
14	Шестаков Александр	1А	А-310
15	Медников Пётр Иванович	18А	А-310
16	Шестаков Александр	1А	А-310
17	Шестаков Александр	20Б	А-310

Рисунок 8 – join



# Типы соединений

A [inner] join B on ...

A left [outer] join B on ...

A right [outer] join B on ...

A full [outer] join B on ... (как LEFT и RIGHT)

A cross join B

Например,

```
select FIO, P.Seat_ID, S.Seat_ID, SeatName
```

```
from Passgr P
```

```
RIGHT [OUTER] JOIN Seat S ON P.Seat_ID=S.Seat_ID
```

Результаты				
	FIO	Seat_ID	Seat_ID	SeatName
1	Сидоров Сидор Сидорович	1	1	1А
2	Петряев Василий Батькович	1	1	1А
3	Аслямов Ильдар Флоридович	10	10	18А
4	Шумахер Михаэль Васильевич	10	10	18А
5	Пушкин Александр Сергеевич	10	10	18А
6	Петров Петр Петрович	10	10	18А
7	Варлей Наталья Михайловна	30	30	20Б
8	Шестаков Александр	43418	43418	1А
9	Шестаков Александр	43418	43418	1А
10	NULL	NULL	44168	3Г
11	NULL	NULL	46426	2А
12	NULL	NULL	46427	1Б
13	Пётр Иванович Неуважай-Корыто	46428	46428	23А
14	NULL	NULL	46429	23Б
15	NULL	NULL	46430	23В
16	Горбачев Михаил Сергеевич	46431	46431	1А
17	Иванов Иван Иванович	46432	46432	3Б
18	Федоров Федор Федорович	46432	46432	3Б
19	NULL	NULL	46448	2Б

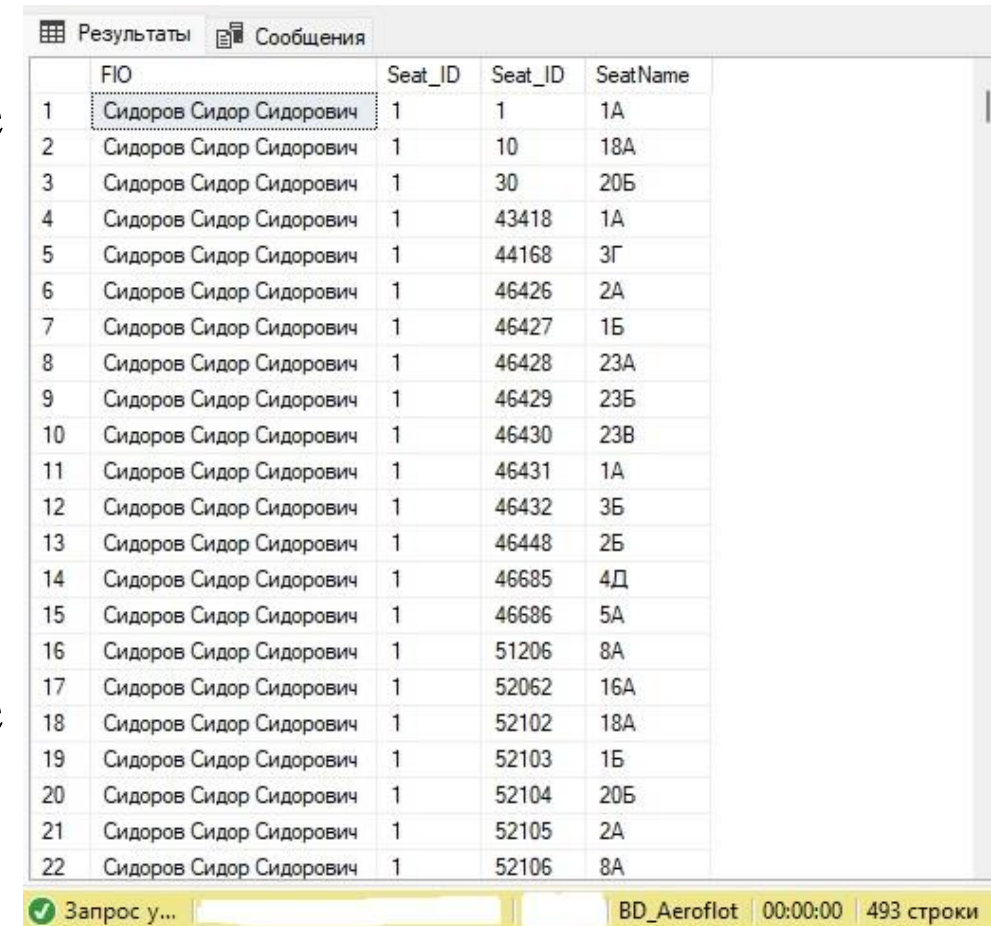
Рисунок 9 – right join

Результатом CROSS JOIN является декартово произведение левого и правого операндов операции соединения.

```
select FIO, P.Seat_ID, S.Seat_ID, SeatName
from Passgr P
CROSS JOIN Seat S
```

До версии 2008 г. записывалось так:

```
select FIO, P.Seat_ID, S.Seat_ID, SeatName
from Passgr P, Seat S
where P.Seat_ID*=S.Seat_ID
```



	FIO	Seat_ID	Seat_ID	SeatName
1	Сидоров Сидор Сидорович	1	1	1A
2	Сидоров Сидор Сидорович	1	10	18A
3	Сидоров Сидор Сидорович	1	30	20Б
4	Сидоров Сидор Сидорович	1	43418	1A
5	Сидоров Сидор Сидорович	1	44168	3Г
6	Сидоров Сидор Сидорович	1	46426	2A
7	Сидоров Сидор Сидорович	1	46427	1Б
8	Сидоров Сидор Сидорович	1	46428	23A
9	Сидоров Сидор Сидорович	1	46429	23Б
10	Сидоров Сидор Сидорович	1	46430	23В
11	Сидоров Сидор Сидорович	1	46431	1A
12	Сидоров Сидор Сидорович	1	46432	3Б
13	Сидоров Сидор Сидорович	1	46448	2Б
14	Сидоров Сидор Сидорович	1	46685	4Д
15	Сидоров Сидор Сидорович	1	46686	5A
16	Сидоров Сидор Сидорович	1	51206	8A
17	Сидоров Сидор Сидорович	1	52062	16A
18	Сидоров Сидор Сидорович	1	52102	18A
19	Сидоров Сидор Сидорович	1	52103	1Б
20	Сидоров Сидор Сидорович	1	52104	20Б
21	Сидоров Сидор Сидорович	1	52105	2A
22	Сидоров Сидор Сидорович	1	52106	8A

Запрос у... BD\_Aeroflot 00:00:00 493 строки

Рисунок 10 – cross join

# WHERE

Условие выборки – логическое выражение, которому должны удовлетворять записи выборки.

Логическое выражение может включать любые выражения, сравниваемые с помощью обычных, рассмотренных ранее, операций сравнения, логические операции *AND*, *OR*, *NOT*.

Пример:

```
select Passgr_ID  
FROM Passgr P
```

	Passgr_ID	FIO
1	3	Сидоров Сидор Сидорович
2	4	Аслямов Ильдар Флоридович
3	5	Варлей Наталья Михайловна
4	7	Шумахер Михаэль Васильевич
5	9	Пушкин Александр Сергеевич
6	11	Петряев Василий Батькович
7	15	Иванов Иван Иванович
8	19	Горбачев Михаил Сергеевич
9	20	Петров Петр Петрович
10	21	Федоров Федор Федорович
11	25	Михайлов Михаил Михайлович
12	27	Пётр Иванович Неуважай-Корыто
13	28	Джонсон
14	29	Шестаков Александр
15	30	Медников Пётр Иванович
16	32	Шестаков Александр
17	33	Шестаков Александр

Рисунок 11 –  
Весь набор

```
select Passgr_ID, FIO  
FROM Passgr P  
WHERE P.Passgr_ID >= 25  
OR P.FIO = 'Аслямов Ильдар Флоридович'
```

	Passgr_ID	FIO
1	4	Аслямов Ильдар Флоридович
2	25	Михайлов Михаил Михайлович
3	27	Пётр Иванович Неуважай-Корыто
4	28	Джонсон
5	29	Шестаков Александр
6	30	Медников Пётр Иванович
7	32	Шестаков Александр
8	33	Шестаков Александр

Рисунок 12 –  
Указанный набор

## Предикат LIKE

<выражение> [NOT] LIKE <шаблон> [ESCAPE <символ>]

Возвращает истину, если выражение удовлетворяет шаблону. В шаблон могут входить обычные символы, которые обозначают сами себя, а также символы – заменители:

1) % – любая строка, например, @x LIKE '%eee%' означает поиск буквосочетания 'eee' в переменной @x.

2) \_ (подчеркивание) – любой одиночный символ, например, WHERE [Фамилия] LIKE '\_a%' будет отыскивать фамилии со второй буквой 'a'.

3) [ ] – любой одиночный символ из указанного диапазона, например, [a-f] или [asxz]. Также, [^] – любой одиночный символ, не принадлежащий указанному диапазону, например, [^a-f] или [^asxz].



## Примеры like

select CityName  
FROM City C  
WHERE CityName like 'М%'

Результаты		Сообщения
CityName		
1	Москва	
2	Магадан	
3	Мелитополь	
4	Миасс	

select CityName  
FROM City C  
WHERE CityName not like '\_a%'

Результаты		Сообщения
CityName		
1	Москва	
2	Челябинск	
3	Екатеринбург	
4	Чебоксары	
5	Мелитополь	
6	Миасс	
7	Абакан	
8	Полетаево	

select CityName  
FROM City C  
WHERE CityName like '[a-e]%'  
ИЛИ WHERE CityName not like '[^a-e]%'

Результаты		Сообщения
CityName		
1	Васюки	
2	Екатеринбург	
3	Абакан	

Результаты		Сообщения
CityName		
1	Москва	
2	Васюки	
3	Челябинск	
4	Магадан	
5	Екатеринбург	
6	Чебоксары	
7	Мелитополь	
8	Миасс	
9	Абакан	
10	Санкт-Петербург	
11	Полетаево	

Рисунок 13 – Исходная  
таблица и примеры like

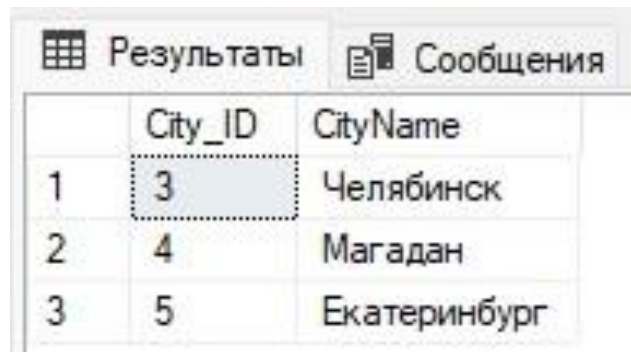
ДЗ: Что выдает запрос?  
select \* from Student  
where StudName  
like '\_e%!e%' escape '!'



## Предикат BETWEEN (между)

Например, **@x between 1 and 12**. Результат будет иметь значение *true*, если  $@x \geq 1$  и  $@x \leq 12$

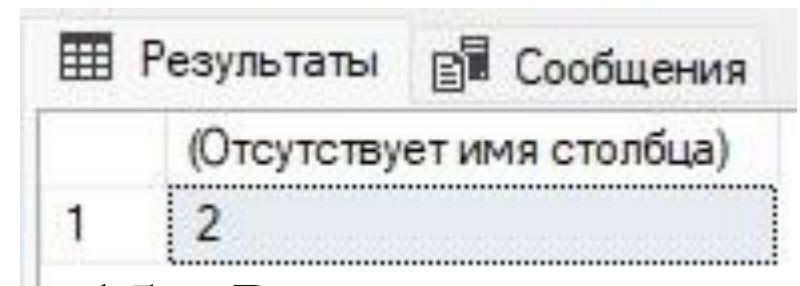
```
select City_ID, CityName
FROM City C
WHERE City_ID between 3 and 10
```



	City_ID	CityName
1	3	Челябинск
2	4	Магадан
3	5	Екатеринбург

Рисунок 14 – Поиск «между»

```
declare @q int, @x int
set @q=1
if @q+1 between 0 and 5-@q
    set @x=2
else
    set @x=4+@q
select @x
```



(Отсутствует имя столбца)
2

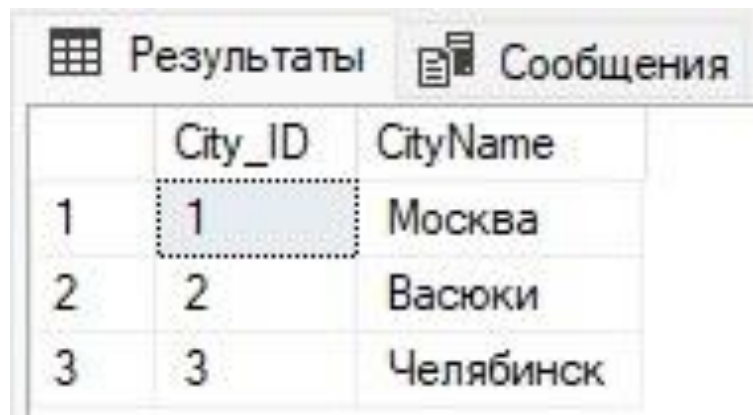
Рисунок 15 – Результат вычисления

## Предикат IN

означает принадлежность множеству значений и имеет 2 формы.

Во первых, может быть указано фиксированное множество значений, например:

```
select City_ID, CityName  
FROM City C  
WHERE City_ID in(1,2,3)
```

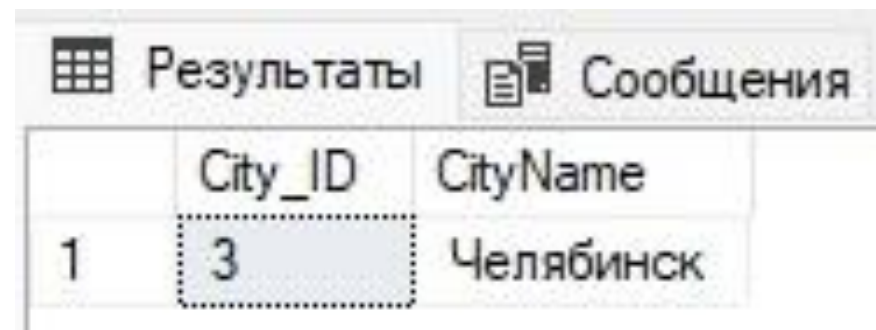


	City_ID	CityName
1	1	Москва
2	2	Васюки
3	3	Челябинск

Рисунок 16 – IN в значениях

Во втором случае множество задается оператором *SELECT*:

```
select c.City_ID, c.CityName  
FROM City C  
WHERE City_ID in (select c1.City_ID  
from City c1  
where c1.CityName='Челябинск')
```



	City_ID	CityName
1	3	Челябинск

Рисунок 17 – IN в select

## Предикат ALL

<скалярное выражение><операция сравнения>  
**ALL**(<подзапрос>)

Пример:  $x \geq \text{ALL}(\text{select } y \text{ from MyTable})$ .

Результат – истина, если  $x$  больше или равно всех значений  $y$ , возвращаемых оператором *SELECT*.

```
select City_ID, CityName
FROM City C
WHERE City_ID > ALL(select City_ID
from City
where CityName='Челябинск')
```

Результаты			Сообщения		
	City_ID	CityName			
1	4	Магадан			
2	5	Екатеринбург			
3	209	Чебоксары			
4	359	Мелитополь			
5	431	Миасс			
6	440	Абакан			
7	455	Санкт-Петербург			
8	572	Полетаево			

Рисунок 18 – ALL

## Предикат ANY (SOME)

<скалярное выражение> <операция сравнения> {ANY | SOME}  
(<подзапрос>)

Пример:  $x \leq \text{ANY}(\text{select } y \text{ from MyTable})$ .

Результат – истина, если  $x$  меньше или равно чем хотя бы одно значение  $y$ , возвращаемое оператором *SELECT*.

```
select c.City_ID, c.CityName
FROM City C
WHERE City_ID >= SOME(select c1.City_ID
    from City c1
    where c1.CityName = 'Челябинск')
and City_ID <= ANY(select c1.City_ID
    from City c1
    where c1.CityName = 'Мелитополь')
```

Результаты			Сообщения		
	City_ID	CityName			
1	3	Челябинск			
2	4	Магадан			
3	5	Екатеринбург			
4	209	Чебоксары			
5	359	Мелитополь			

Рисунок 19 – SOME

# Квантор существования (*EXISTS*)

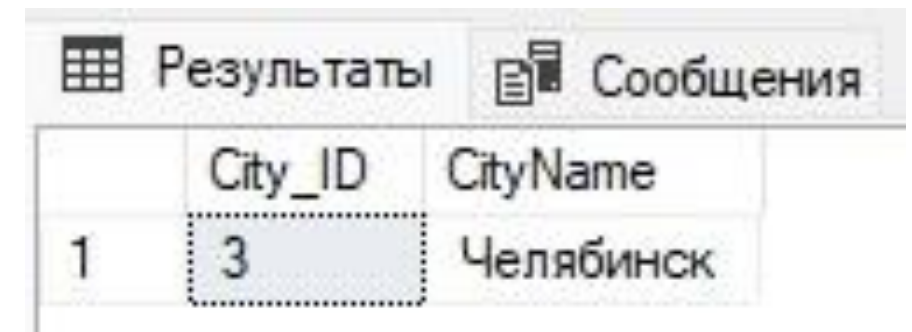
Синтаксис:

**[NOT] EXISTS(<подзапрос>)**

Пример: ... **WHERE EXISTS (SELECT \* FROM TT).**

Результат — истина, если оператор *SELECT* возвращает непустое множество записей.

```
select c.City_ID, c.CityName
FROM City C
WHERE exists (select c1.City_ID
from City c1
where c1.CityName='Челябинск'
and c.City_ID=c1.City_ID) -- строка обязательна
```



Результаты		Сообщения
	City_ID	CityName
1	3	Челябинск

Рисунок 20 – EXISTS



# Квантор общности (*NOT EXISTS*)

Синтаксис:

**NOT EXISTS(<подзапрос>)**

Квантор общности — не существует такого объекта, для которого не выполнено условие.

```
select c.City_ID, c.CityName
FROM City C
WHERE not exists (select c1.City_ID
                  from City c1
                  where c1.CityName='Челябинск'
                  and c.City_ID=c1.City_ID) -- строка обязательна
```

Результаты			Сообщения		
	City_ID	CityName			
1	1	Москва			
2	2	Васюки			
3	4	Магадан			
4	5	Екатеринбург			
5	209	Чебоксары			
6	359	Мелитополь			
7	431	Миасс			
8	440	Абакан			
9	455	Санкт-Петербург			
10	572	Полетаево			

Рисунок 21 – NOT EXISTS

## Фраза ORDER BY

Предписывает порядок, в котором должны следовать записи результирующего множества. Синтаксис:

**[ ORDER BY { выражение [ ASC | DESC ] } [ ,...n] ]**

Например:

**...ORDER BY A.t, B.x DESC**

что означает, что выборка упорядочивается по *A.t* в порядке возрастания (*ASC*), а в пределах подмножеств записей с одинаковыми значениями *A.t* — по *B.t* в порядке убывания (*DESC*). В список выражений упорядочения могут входить псевдонимы полей

# Пример фразы ORDER BY

```
select c.City_ID, c.CityName  
FROM City C  
order by CityName asc
```

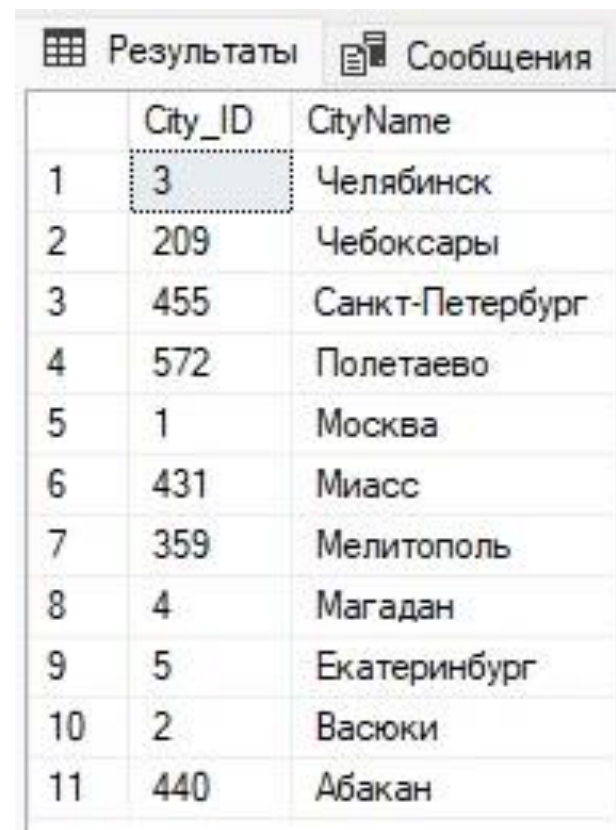


The screenshot shows a database query result with two tabs: 'Результаты' (Results) and 'Сообщения' (Messages). The 'Результаты' tab is active, displaying a table with three columns: 'City\_ID', 'CityName', and an unlabeled column with a numeric index from 1 to 11. The data is sorted by 'CityName' in ascending order. The first row, with 'City\_ID' 440 and 'CityName' 'Абакан', is highlighted with a dashed border.

	City_ID	CityName
1	440	Абакан
2	2	Васюки
3	5	Екатеринбург
4	4	Магадан
5	359	Мелитополь
6	431	Миасс
7	1	Москва
8	572	Полетаево
9	455	Санкт-Петербург
10	209	Чебоксары
11	3	Челябинск

Рисунок 22 – Прямой порядок

```
select c.City_ID, c.CityName  
FROM City C  
order by CityName desc, City_ID asc
```



The screenshot shows a database query result with two tabs: 'Результаты' (Results) and 'Сообщения' (Messages). The 'Результаты' tab is active, displaying a table with three columns: 'City\_ID', 'CityName', and an unlabeled column with a numeric index from 1 to 11. The data is sorted by 'CityName' in descending order, and then by 'City\_ID' in ascending order for cities with the same name. The first row, with 'City\_ID' 3 and 'CityName' 'Челябинск', is highlighted with a dashed border.

	City_ID	CityName
1	3	Челябинск
2	209	Чебоксары
3	455	Санкт-Петербург
4	572	Полетаево
5	1	Москва
6	431	Миасс
7	359	Мелитополь
8	4	Магадан
9	5	Екатеринбург
10	2	Васюки
11	440	Абакан

Рисунок 23 – Набор порядков

# SELECT INTO

Оператор *SELECT* может помещать результат выборки в новую таблицу.  
Например:

```
SELECT T1.x, T2.y into NewTable  
FROM T1, T2  
WHERE T1.z=T2.z
```

Если таблица NewTable уже существует, то операция будет отвергнута.

Пример:

```
select g.GruppName, s.StudName into ttt  
from student s, Grupp g  
where g.Grupp_ID=s.grupp_ID  
order by GruppName, StudName
```

## Ограничение объёма выборки

В операторе `SELECT` может присутствовать фраза, ограничивающая объем выборки:

**`SELECT [ ALL | DISTINCT ] [ TOP n [ PERCENT ] ]...`**

Умолчанием является *ALL*. В этом случае оператор *SELECT* возвращает все записи, удовлетворяющие условию во фразе *WHERE*.

- *DISTINCT* означает, что все возвращаемые записи должны быть различны. Дубликаты будут исключены из результирующего множества.
- *TOP n* – в результат будут включены только *n* первых записей.
- *TOP n PERCENT* – в результат будут включены только *n* процентов числа записей, удовлетворяющих условию выборки.



**Distinct.** Список пассажиров (FIO), когда либо летавших в Санкт-Петербург на самолете А-310. (буква А – кириллица).  
Примечание: предполагается, что FIO пассажиров уникальны.  
Упорядочить по алфавиту.

```
select distinct FIO[, Dat]
from Passgr pg, Flight f, City c, PlaneType pt
where pg.Flight_ID=f.Flight_ID
and f.PlaneType_ID=pt.PlaneType_ID
and f.CityTo_ID=c.City_ID
and c.CityName='Санкт-Петербург'
and pt.PlaneTypeName='А-310'
order by FIO
```

Результаты			Сообщения		
	FIO	Dat			
1	Медников Пётр Иванович	2020-10-01 23:00:00			
2	Шестаков Александр	2025-10-21 12:34:00			
3	Шестаков Александр	2020-10-01 23:00:00			

Результаты			Сообщения		
	FIO				
1	Медников Пётр Иванович				
2	Шестаков Александр				

Рисунок 24 –  
Distinct

```
select Top 50 percent c.City_ID, c.CityName
FROM City C
order by City_ID
```

```
select Top 7 c.City_ID, c.CityName
FROM City C
order by City_ID
```

Результаты   Сообщения

	CityName
1	Москва
2	Васюки
3	Челябинск
4	Магадан
5	Екатеринбург
6	Чебоксары
7	Мелитополь
8	Миасс
9	Абакан
10	Санкт-Петербург
11	Полетаево

```
select Top 7 c.City_ID, c.CityName
FROM City C
order by City_ID desc
```

Результаты   Сообщения

	City_ID	CityName
1	1	Москва
2	2	Васюки
3	3	Челябинск
4	4	Магадан
5	5	Екатеринбург
6	209	Чебоксары

Результаты   Сообщения

	City_ID	CityName
1	1	Москва
2	2	Васюки
3	3	Челябинск
4	4	Магадан
5	5	Екатеринбург
6	209	Чебоксары
7	359	Мелитополь

Результаты   Сообщения

	City_ID	CityName
1	572	Полетаево
2	455	Санкт-Петербург
3	440	Абакан
4	431	Миасс
5	359	Мелитополь
6	209	Чебоксары
7	5	Екатеринбург

Рисунок 25 – Исходная  
таблица и выборки

Вместе с TOP может быть использован параметр **WITH TIES**, который позволяет дополнить строки, получаемые применением TOP строками, имеющими такие же значение выражений, упомянутых во фразе ORDER BY. Например, пусть требуется получить список из 2х товаров, имеющих наименьшие цены. Без применения WITH TIES это запрос имеет вид:

```
select top 6 TovarName, Price  
from Tovar  
where 1=IsTovar and price is not null  
order by Price
```

Результаты			Сообщения		
	TovarName	Price			
1	Болгарский перец	20,00			
2	Черешня	22,00			
3	Груша	23,00			
4	Горбулка	30,00			
5	Ананас	45,80			
6	Футболка	50,00			

Рисунок 26 –  
Сравнение  
результатов

```
select top 6 with ties TovarName, Price  
from Tovar  
where 1=IsTovar and price is not null  
order by Price
```

Результаты			Сообщения		
	TovarName	Price			
1	Болгарский перец	20,00			
2	Черешня	22,00			
3	Груша	23,00			
4	Горбулка	30,00			
5	Ананас	45,80			
6	Футболка	50,00			
7	Калготки	50,00			

даёт результат, соответствующий тому факту, что более чем один товар имеет одинаковую цену.

## Агрегатные запросы

Оператор *SELECT* может подсчитывать итоги. Для этих целей используются агрегатные функции и группировка. Имеются следующие агрегатные функции:

*SUM*(выражение) – суммирование;

*AVG*(выражение) – вычисление среднего значения;

*COUNT*(выражение) – подсчет числа записей;

*MIN*(выражение) – нахождение минимального значения;

*MAX*(выражение) – нахождение максимального значения.

Фраза

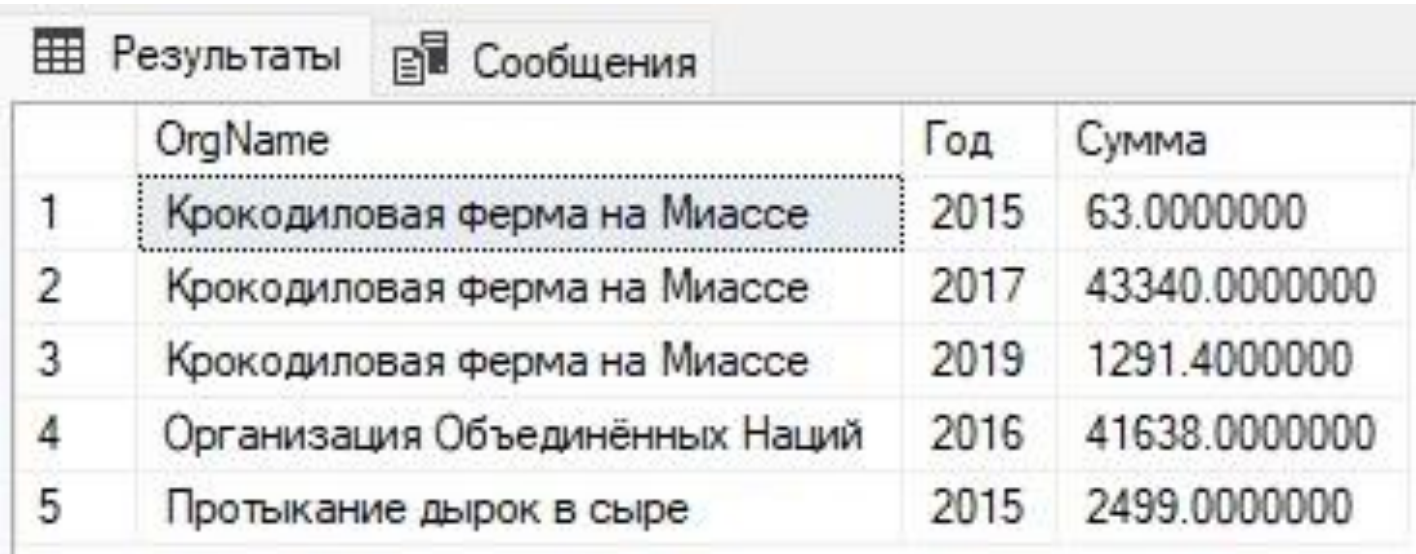
**GROUP BY <список выражений группировки>**

указывает выражения, по которым выполняется группировка.

use Warehouse

-- На какую сумму организации приобрели товары по годам

```
select OrgName, year(n.Dat) as [Год], sum(Amount*Price) as [Сумма]
from Org o, Nakl n, SostNakl sn
where o.Org_ID=n.Org_ID
and n.Nakl_ID=sn.Nakl_ID
and n.InOut='-'
group by OrgName, year(Dat)
order by OrgName, year(Dat)
```



The screenshot shows a database query results window with two tabs: 'Результаты' (Results) and 'Сообщения' (Messages). The 'Результаты' tab is active, displaying a table with 4 columns: 'OrgName', 'Год' (Year), and 'Сумма' (Sum). The table contains 5 rows of data. The first row is highlighted with a dashed border.

	OrgName	Год	Сумма
1	Крокодиловая ферма на Миассе	2015	63.0000000
2	Крокодиловая ферма на Миассе	2017	43340.0000000
3	Крокодиловая ферма на Миассе	2019	1291.4000000
4	Организация Объединённых Наций	2016	41638.0000000
5	Протыкание дырок в сыре	2015	2499.0000000

Рисунок 27 – Товары по годам

Все поля, возвращаемые оператором *SELECT* с группировкой, должны либо находиться под агрегатной функцией, либо входить во фразу *GROUP BY*.

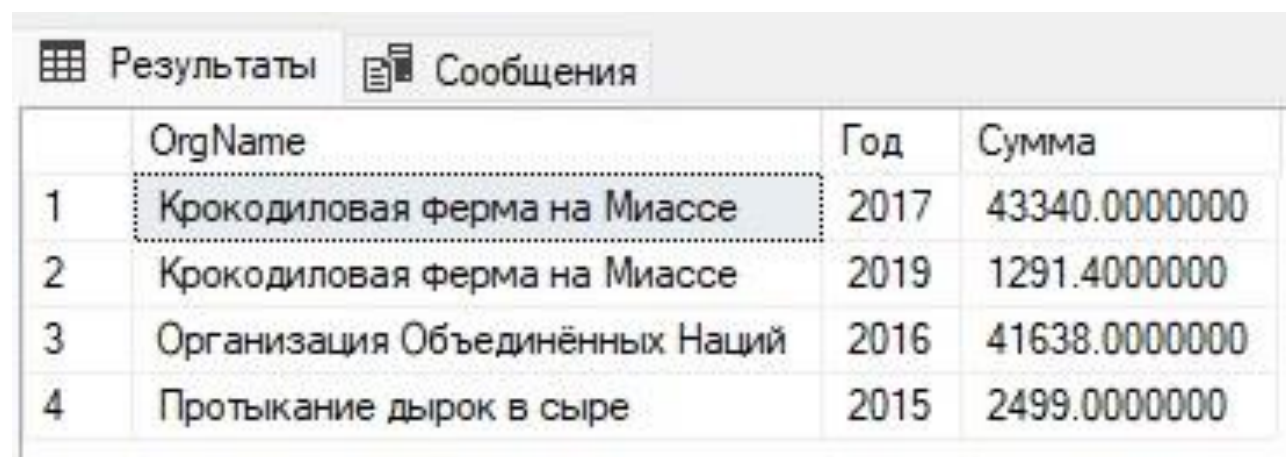


# HAVING

Агрегатный запрос может включать фразу *HAVING*, которая играет для результата группировки ту же роль, что и фраза *WHERE* для отдельных записей, которые являются исходными данными для выполнения группировки.

Модифицируем последний запрос следующим образом: искать только суммы превышающие 1000 рублей.

```
select OrgName, year(n.Dat) as [Год], sum(Amount*Price) as [Сумма]
from Org o, Nakl n, SostNakl sn
where o.Org_ID=n.Org_ID
      and n.Nakl_ID=sn.Nakl_ID
      and n.InOut='-'
group by OrgName, year(Dat)
having sum(Amount*Price)>1000
order by OrgName, year(Dat)
```



Результаты		Сообщения	
	OrgName	Год	Сумма
1	Крокодиловая ферма на Миассе	2017	43340.0000000
2	Крокодиловая ферма на Миассе	2019	1291.4000000
3	Организация Объединённых Наций	2016	41638.0000000
4	Протыкание дырок в сыре	2015	2499.0000000

Рисунок 28 – Товары по годам  
превышающие 1000 руб.

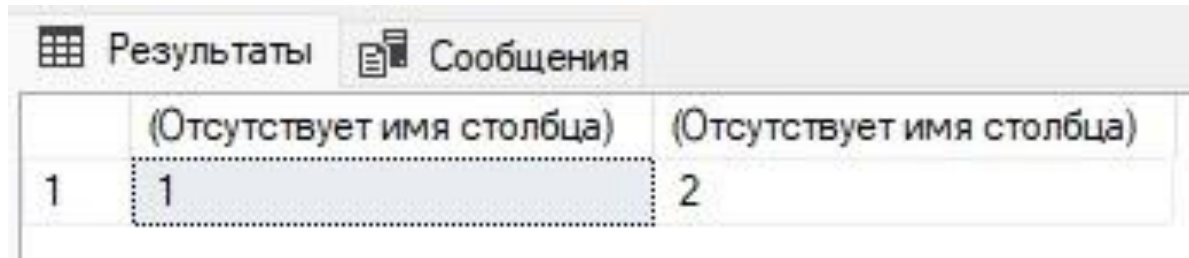
# Операторы UNION, EXCEPT, INTERSECT

Оператор **UNION** может объединять результаты двух операторов *SELECT*, например:

**SELECT** 1, 2

**UNION**

**SELECT** 1, 2



Результаты		Сообщения	
	(Отсутствует имя столбца)	(Отсутствует имя столбца)	
1	1	2	

Рисунок 29 – Union

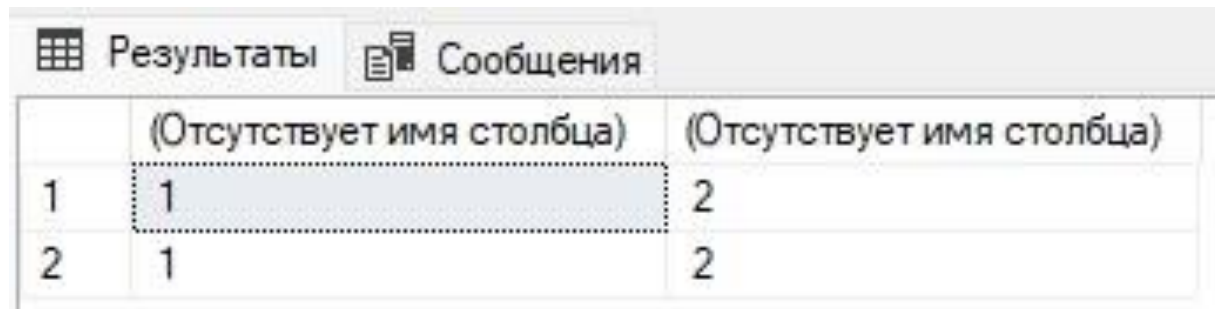
имеет результат, содержащий две строки.

Если *ALL* присутствует, то в результат будут включены все записи, в противном случае дубликаты будут исключены.

**SELECT** 1, 2

**UNION ALL**

**SELECT** 1, 2



Результаты		Сообщения	
	(Отсутствует имя столбца)	(Отсутствует имя столбца)	
1	1	2	
2	1	2	

Рисунок 30 – Union all

## EXCEPT и INTERSECT – вычитание и пересечение

<оператор SELECT> EXCEPT <оператор SELECT>

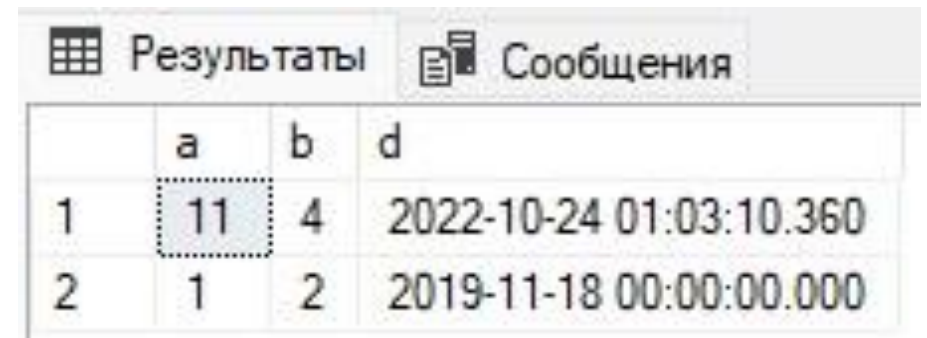
<оператор SELECT> INTERSECT <оператор SELECT>

Для все трёх операторов поля с одинаковыми номерами необязательно должны быть одного типа, но необходимо, чтобы в операторах *SELECT* могли быть неявно приведены к типу одного из них, например:

SELECT 3+8 AS a, 4 AS b, *getdate()* AS d

UNION

SELECT 1, '2', '20191118'



	a	b	d
1	11	4	2022-10-24 01:03:10.360
2	1	2	2019-11-18 00:00:00.000

Рисунок 31 – Неявное приведение типов

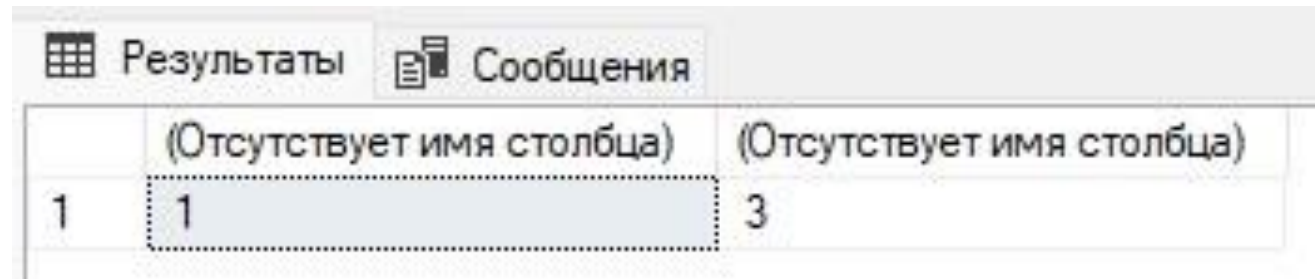
Поля результата будут иметь те же имена, что и в первом из операторов *SELECT*.

Для операторов EXCEPT и INTERSECT дубликатные записи в результат не помещаются.

Примеры.

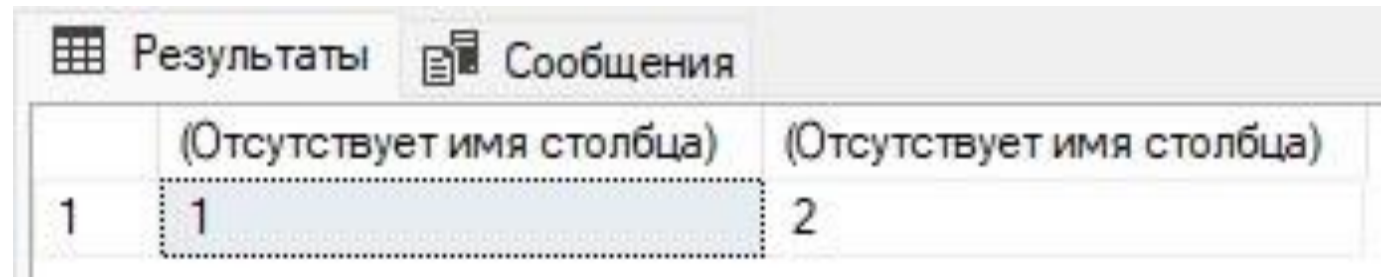
(SELECT 1, 3  
UNION  
SELECT 1, 2)  
EXCEPT  
SELECT 1, 2

(SELECT 1, 3  
UNION  
SELECT 1, 2)  
INTERSECT  
SELECT 1, 2



	(Отсутствует имя столбца)	(Отсутствует имя столбца)
1	1	3

Рисунок 32 – Вычитание



	(Отсутствует имя столбца)	(Отсутствует имя столбца)
1	1	2

Рисунок 32 – Пересечение

# Примеры операторов *SELECT*

1) Товары и история цен до сегодняшнего дня.

```
SELECT Товар.Tovar_ID, Товар.TovarName,  
DateStart, PriceList.Price, DateEnd  
FROM Товар left join PriceList  
on Товар.Tovar_ID=PriceList.Tovar_ID  
where IsТовар=1  
--and DateStart<=getdate()  
ORDER BY ТovarName
```

Результаты		Сообщения			
	Tovar_ID	TovarName	DateStart	Price	DateEnd
1	35	Ананас	2010-11-08 00:00:00.000	45,80	2016-12-11 00:00:00.000
2	10	Апельсин	2014-02-02 00:00:00.000	12,00	2016-07-30 00:00:00.000
3	10	Апельсин	2008-09-01 00:00:00.000	60,00	2009-02-25 00:00:00.000
4	10	Апельсин	2016-07-31 00:00:00.000	77,00	NULL
5	28	Батон нарезной	2011-04-28 00:00:00.000	30,00	2015-05-31 00:00:00.000
6	28	Батон нарезной	2015-06-01 00:00:00.000	120,00	2015-06-30 00:00:00.000
7	28	Батон нарезной	2019-01-01 00:00:00.000	20,00	2019-01-31 00:00:00.000
8	28	Батон нарезной	2019-01-15 00:00:00.000	25,00	2019-02-10 00:00:00.000
9	42	Болгарский перец	2010-06-27 00:00:00.000	77,00	2015-07-30 00:00:00.000
10	69	Вело	NULL	NULL	NULL
11	27	Горбулка	2011-09-12 00:00:00.000	43,00	2013-01-01 00:00:00.000
12	36	Груша	2010-11-08 00:00:00.000	55,90	2015-12-11 00:00:00.000
13	39	Калготки	2010-11-08 00:00:00.000	50,00	2015-12-11 00:00:00.000
14	51	Мандарины	NULL	NULL	NULL
15	70	Семечки	NULL	NULL	NULL
16	37	Футболка	2010-06-27 00:00:00.000	30,00	2015-07-30 00:00:00.000
17	43	Черешня	2008-10-26 00:00:00.000	80,00	2016-02-28 00:00:00.000
18	43	Черешня	2020-06-02 00:00:00.000	200,00	2020-06-30 00:00:00.000
19	43	Черешня	2020-06-20 00:00:00.000	250,00	2020-07-30 00:00:00.000
20	38	Шуба	2010-06-27 00:00:00.000	100,00	2016-07-30 00:00:00.000

Рисунок 34 – Товары и цены

*Примечание:* используется левое внешнее соединение таблиц *Tovar* и *PriceList*, с тем, чтобы в результат вошли все товары, в том числе не имеющие цены.



2) другое решение задачи 1

```
SELECT t.Tovar_ID, t.TovarName, p.Price, p.DateStart, DateEnd
FROM Tovar t left join PriceList p
on t.Tovar_ID=p.Tovar_ID
where t.IsTovar=1
and DateStart<=getdate()
and (DateEnd is null or DateEnd>=getdate())
ORDER BY TovarName
```

Результаты		Сообщения			
	Tovar_ID	TovarName	Price	DateStart	DateEnd
1	10	Апельсин	77,00	2016-07-31 00:00:00.000	NULL

Рисунок 35 – Товары и цены

В чем отличие?



3) оператор *SELECT* используется в списке извлекаемых полей

```

SELECT Товар.Товар_ID, Товар.ТоварName,
    (SELECT top 1 Price
     FROM PriceList
     WHERE Товар.Товар_ID=PriceList.Товар_ID
      and DateStart<=getdate()
      and (DateEnd is null
           or DateEnd>=getdate()))
ORDER BY DateStart desc
) as Price
FROM Товар
where IsТовар=1 ORDER BY ТоварName

```

Результаты		Сообщения	
	Товар_ID	ТоварName	Price
1	35	Ананас	NULL
2	10	Апельсин	77,00
3	28	Батон нарезной	NULL
4	42	Болгарский перец	NULL
5	69	Вело	NULL
6	27	Горбулка	NULL
7	36	Груша	NULL
8	39	Калготки	NULL
9	51	Мандарины	NULL
10	70	Семечки	NULL
11	37	Футболка	NULL
12	43	Черешня	NULL
13	38	Шуба	NULL

Рисунок 36 – Select  
как поле выборки

4) список организаций, закупавших «Апельсин»

**SELECT distinct** Org.\*

**FROM** Org, Nakl, SostNakl, Tovar

**WHERE** Nakl.Org\_ID=Org.Org\_ID

and Nakl.Nakl\_ID=SostNakl.Nakl\_ID

and SostNakl.Tovar\_ID=Tovar.Tovar\_ID

and Nakl.InOut='-'

and Tovar.TovarName='Апельсин'

**ORDER BY** OrgName

Результаты				
Сообщения				
	Org_ID	OrgName	Address	Phone
1	7	Крокодиловая ферма на Миассе	Зоопарк за углом в Лондоне на Бейкер-street	NoPhone
2	8	Протыкание дырок в сыре	Лондон-сити	222222

Рисунок 37 – Покупатели апельсинов

5) другое решение задачи 4 с квантором существования

**SELECT** Org.\*

**FROM** Org

**WHERE** exists

(**SELECT** SostNakl.\*

**FROM** Nakl, SostNakl, Товар

**WHERE** Nakl.Org\_ID=Org.Org\_ID

and Nakl.Nakl\_ID=SostNakl.Nakl\_ID

and SostNakl.Товар\_ID=Товар.Товар\_ID

and Товар.ТоварName='Апельсин'

and Nakl.InOut='-'

)

**ORDER BY** OrgName

Результаты				
	Org_ID	OrgName	Address	Phone
1	7	Крокодиловая ферма на Миассе	Зоопарк за углом в Лондоне на Бейкер-street	NoPhone
2	8	Протыкание дырок в сыре	Лондон-сити	222222

Рисунок 38 – Покупатели апельсинов

б) пример иллюстрирует соединение таблицы с самой собой. Получить список пар организаций, имеющих одинаковый адрес.

```
SELECT t1.OrgName AS Org1, t2.OrgName AS Org2
FROM Org t1, Org t2
WHERE t1.Address=t2.Address
and t1.Org_ID<t2.Org_ID
```

	Org1	Org2
1	Организация Объединённых Наций	Бюро по приватизации жилья
2	Страусиный питомник	Черные ассоциации восточной океании
3	Страусиный питомник	Новая организация
4	Черные ассоциации восточной океании	Новая организация
5	Страусиный питомник	Уральский Социально-Экономический Инстит
6	Черные ассоциации восточной океании	Уральский Социально-Экономический Инстит
7	Новая организация	Уральский Социально-Экономический Инстит
8	Организация Объединённых Наций	ЮУрГУ
9	Бюро по приватизации жилья	ЮУрГУ

Рисунок 39 – Одинаковые адреса

условие t1.Org\_ID<t2.Org\_ID добавлено, чтобы в результирующем множестве записей не появлялись пары вида:

(A, A), (A, B), (B, A).

7) оператор *SELECT* во фразе *FROM*. Получить список накладных на поступление товаров от организаций, расположенных на улице 'Свободы' с 1 января 2001 г. по 30 апреля 2022 г.

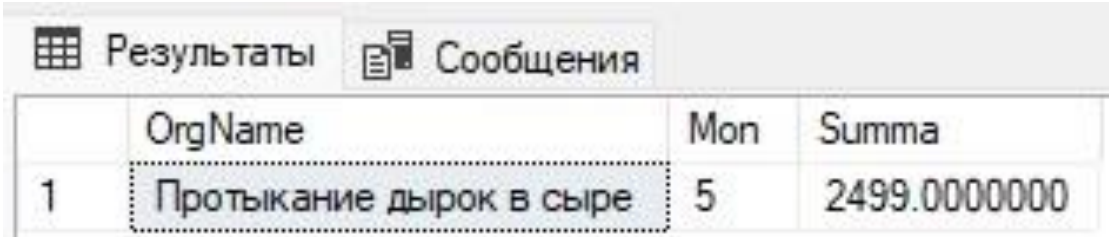
```
SELECT Nakl.*, o.OrgName, Address
FROM Nakl, (SELECT * FROM Org) AS o
WHERE Nakl.Org_ID=o.Org_ID
      and Address like '%Свободы%'
      and Nakl.InOut='+'
      and (Dat BETWEEN '20010101' and '20220430')
ORDER BY o.OrgName, Nakl.Dat
```

Результаты		Сообщения						
	Nakl_ID	Dat	Numb	Org_ID	InOut	SumNakl	OrgName	Address
1	52	2015-05-15 00:00:00.000	224	22	+	20792,80	Новая организация	Свободы 155-А

Рисунок 40 – Накладные

8) запрос с группировкой. Получить суммы закупок организаций по месяцам 2009 г. для организаций, для которых сумма закупок превысила 1000 р. Результатом должна явиться таблица с полями Организация, Месяц, Сумма закупок. Обратите внимание, что во фразах *GROUP BY, HAVING, ORDER BY* использование псевдонимов полей не допускается.

```
SELECT OrgName, month(Nakl.Dat) AS Mon,  
       SUM(SostNakl.Amount*SostNakl.Price) AS Summa  
FROM Org, Nakl, SostNakl  
WHERE Org.Org_ID=Nakl.Org_ID  
       and Nakl.Nakl_ID=SostNakl.Nakl_ID  
       and Nakl.InOut='-' and year(Nakl.Dat)=2015  
GROUP BY OrgName, month(Nakl.Dat)  
HAVING SUM(SostNakl.Amount*SostNakl.Price)>1000  
ORDER BY OrgName, month(Nakl.Dat)
```



Результаты		Сообщения	
	OrgName	Mon	Summa
1	Протыкание дырок в сыре	5	2499.00000000

Рисунок 41 – Запрос  
с группировкой



9) оператор select может появиться и во фразе WHERE. Получить данные о последней накладной для каждой организации.

```
select o.OrgName, n1.*  
from Org o, Nakl n1  
where o.Org_ID=n1.Org_ID  
and n1.Dat = (select max(Dat)  
from Nakl n2  
where n2.Org_ID=o.Org_ID)
```

Результаты		Сообщения					
	OrgName	Nakl_ID	Dat	Numb	Org_ID	InOut	SumNakl
1	Новая организация	52	2015-05-15 00:00:00.000	224	22	+	20792,80
2	Microsoft	66	2018-10-21 00:00:00.000	4333	14	+	0,00
3	Протыкание дырок в сыре	65	2019-10-21 00:00:00.000	3334	8	+	0,00
4	Крокодиловая ферма на Миассе	50	2019-09-30 00:00:00.000	4321	7	-	1291,40
5	Крокодиловая ферма на Миассе	64	2019-09-30 00:00:00.000	4321	7	-	0,00
6	Бюро по приватизации жилья	51	2011-05-13 00:00:00.000	22	2	+	857,44
7	Организация Объединённых Наций	59	2016-02-15 00:00:00.000	300	1	-	41638,00

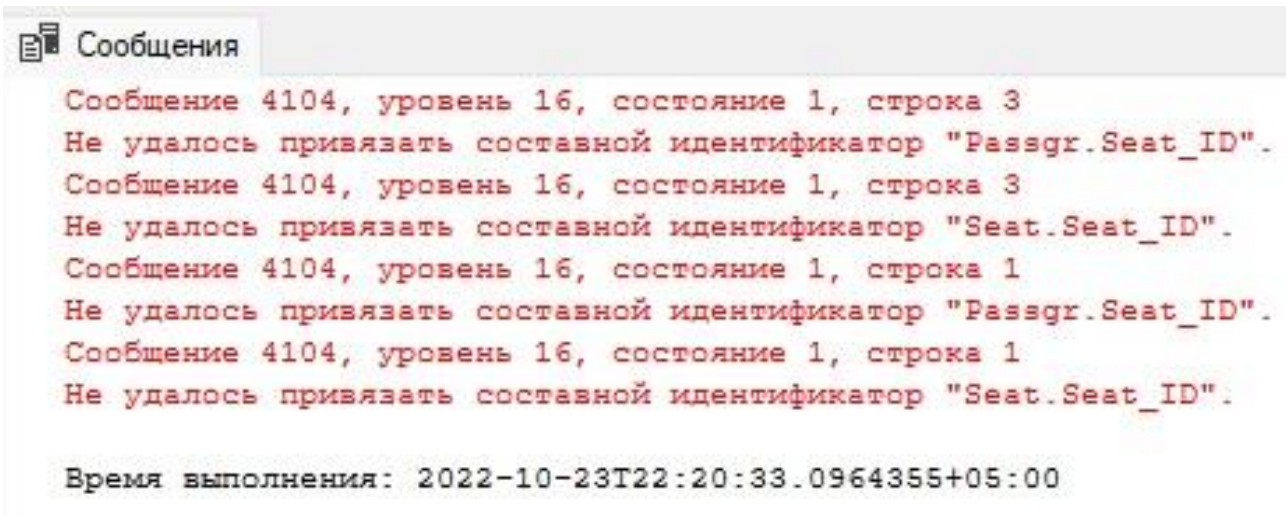
Рисунок 42 – Последние накладные

## Замечания

1. Объявив другое название таблицы, следует указывать только его.

```
SELECT FIO, Passgr.Seat_ID,  
       Seat.Seat_ID, SeatName  
FROM Passgr p, Seat s  
Where Passgr.Seat_ID=Seat.Seat_ID
```

```
SELECT FIO, p.Seat_ID,  
       s.Seat_ID, SeatName  
FROM Passgr p, Seat s  
Where p.Seat_ID=s.Seat_ID
```



Результаты				
Сообщения				
	FIO	Seat_ID	Seat_ID	SeatName
1	Сидоров Сидор Сидорович	1	1	1A
2	Аслямов Ильдар Флоридович	10	10	18A
3	Варлей Наталья Михайловна	30	30	20Б
4	Шумахер Михаэль Васильевич	10	10	18A
5	Пушкин Александр Сергеевич	10	10	18A
6	Петряев Василий Батькович	1	1	1A
7	Иванов Иван Иванович	46432	46432	3Б
8	Горбачев Михаил Сергеевич	46431	46431	1A
9	Петров Петр Петрович	10	10	18A
10	Федоров Федор Федорович	46432	46432	3Б
11	Михайлов Михаил Михайлович	52107	52107	25Г
12	Пётр Иванович Неуважай-Корыто	46428	46428	23A
13	Джонсон	52062	52062	16A
14	Шестаков Александр	43418	43418	1A
15	Медников Пётр Иванович	52102	52102	18A
16	Шестаков Александр	43418	43418	1A
17	Шестаков Александр	52104	52104	20Б

Рисунок 43 – Неправильное указание источника и правильное

## 2. Поля таблицы находятся только после указания их в строке FROM

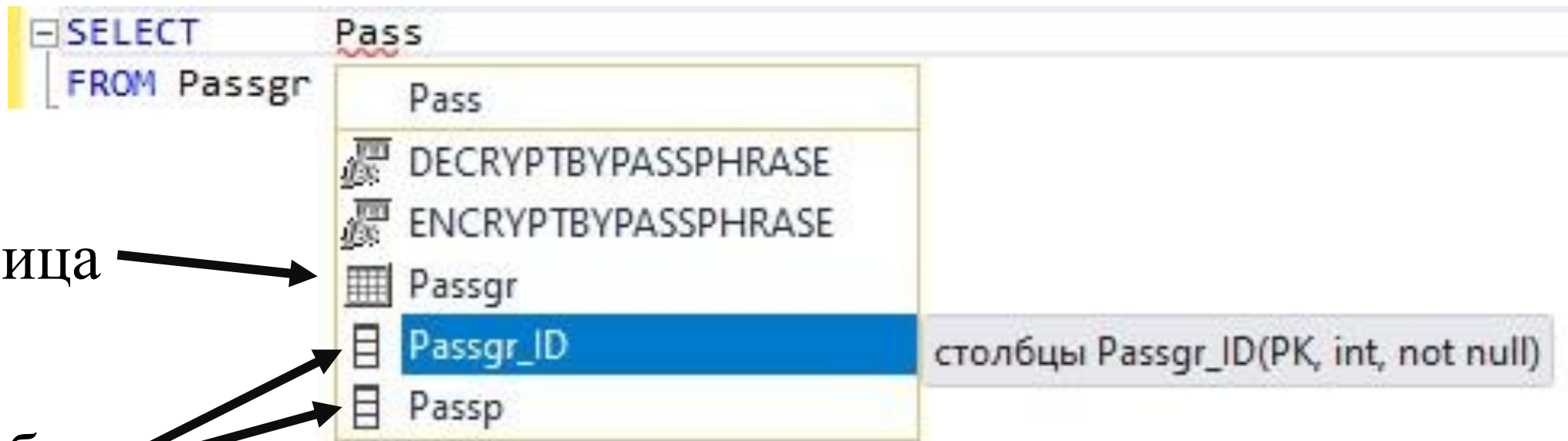
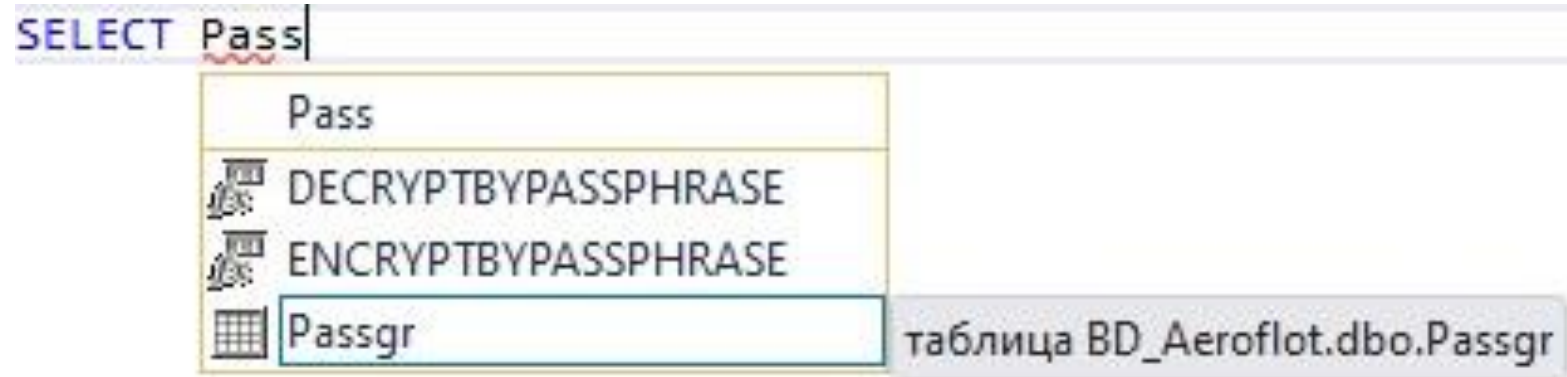


Таблица →

Столбцы →

Рисунок 44 – Поля и таблицы в MS SQL



3. Упорядочивание полей без строки ORDER BY идет по порядку  
внесения записей в таблицы

```
SELECT FIO, Passgr_ID
FROM Passgr

SELECT Passgr_ID, Seat.Seat_ID
FROM Passgr, Seat

SELECT Seat.Seat_ID, Passgr_ID
FROM Passgr, Seat
```

	FIO	Passgr_ID
1	Сидоров Сидор Сидорович	3
2	Аслямов Ильдар Флоридович	4
3	Варлей Наталья Михайловна	5
4	Шумахер Михаэль Васильевич	7
5	Пушкин Александр Сергеевич	9
6	Петряев Василий Батькович	11
7	Иванов Иван Иванович	15
8	Горбачев Михаил Сергеевич	19
9	Петров Петр Петрович	20
10	Федоров Федор Федорович	21
11	Михайлов Михаил Михайлович	25
12	Пётр Иванович Неуважай-Корыто	27
13	Джонсон	28
14	Шестаков Александр	29
15	Медников Пётр Иванович	30
16	Шестаков Александр	32
17	Шестаков Александр	33

Т.к. Seat был заполнен ранее и на него ссылался Passgr требуя наличия

Рисунок 45 – Порядок без указания

	Passgr_ID	Seat_ID
1	3	1
2	3	10
3	3	30
4	3	43418
5	3	44168
6	3	46426
7	3	46427
8	3	46428
9	3	46429
10	3	46430
11	3	46431
12	3	46432
13	3	46448
14	3	46685
15	3	46686
16	3	51206
17	3	52062
18	3	52102
19	3	52103
20	3	52104
21	3	52105
22	3	52106
23	3	52107
24	3	54057
25	3	54058
26	3	54059
27	3	54063
28	3	54091
29	3	54092
30	27	1
31	27	10

	Seat_ID	Passgr_ID
1	1	3
2	10	3
3	30	3
4	43418	3
5	44168	3
6	46426	3
7	46427	3
8	46428	3
9	46429	3
10	46430	3
11	46431	3
12	46432	3
13	46448	3
14	46685	3
15	46686	3
16	51206	3
17	52062	3
18	52102	3
19	52103	3
20	52104	3
21	52105	3
22	52106	3
23	52107	3
24	54057	3
25	54058	3
26	54059	3
27	54063	3
28	54091	3
29	54092	3
30	1	27
31	10	27
32	30	27

4. Способы переименовать поля выборки (столбцы), а также как их указать в упорядочивании.

```
SELECT FIO as ФИО, p.Seat_ID Место1,  
       s.Seat_ID Wsv, SeatName [Место]  
FROM Passgr p, Seat s  
Where p.Seat_ID=s.Seat_ID  
order by ФИО, p.Seat_ID, Wsv, [Место]
```

Результаты				
	ФИО	Место1	Wsv	Место
1	Аслямов Ильдар Флоридович	10	10	18А
2	Варлей Наталья Михайловна	30	30	20Б
3	Горбачев Михаил Сергеевич	46431	46431	1А
4	Джонсон	52062	52062	16А
5	Иванов Иван Иванович	46432	46432	3Б
6	Медников Пётр Иванович	52102	52102	18А
7	Михайлов Михаил Михайлович	52107	52107	25Г
8	Пётр Иванович Неуважай-Корыто	46428	46428	23А
9	Петров Петр Петрович	10	10	18А
10	Петряев Василий Батькович	1	1	1А
11	Пушкин Александр Сергеевич	10	10	18А
12	Сидоров Сидор Сидорович	1	1	1А
13	Федоров Федор Федорович	46432	46432	3Б
14	Шестаков Александр	43418	43418	1А
15	Шестаков Александр	43418	43418	1А
16	Шестаков Александр	52104	52104	20Б
17	Шумахер Михаэль Васильевич	10	10	18А

Рисунок 46 – Результат запроса