SQL

structured query language

SQL - структурированный Язык Запросов

- декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.
- язык который дает вам возможность создавать и работать в реляционных базах данных, которые являются наборами связанной информации сохраняемой в таблицах.

SQL является **непроцедурным** языком. Чтобы с его помощью решить задачу, сообщите SQL, **что именно** вам нужно, как если бы вы говорили с джином из лампы Аладдина. И при этом не надо говорить, **каким образом** получить для вас то, что вы хотите. Система управления базами данных (СУБД) сама решит, как лучше всего выполнить ваш запрос.

Интерактивный SQL

- используется для функционирования непосредственно в базе данных чтобы производить вывод для использования его заказчиком. В этой форме SQL, когда вы введете команду, она сейчас же выполнится и вы сможете увидеть вывод (если он вообще получится) - немедленно.

Вложенный SQL

- состоит из команд SQL помещенных внутри программ, которые обычно написаны на некотором другом языке

Состав SQL:

- Язык определения данных (**Data Definition Language**, DDL). Это та часть SQL, которая используется для создания (полного определения) базы данных, изменения ее структуры и удаления базы после того, как она становится ненужной.(crete, alter,drop)
- Язык манипулирования данными (**Data Manipulation Language**, DML). Предназначен для поддержки базы данных. С помощью этого мощного инструмента можно точно указать, что именно нужно сделать с данными, находящимися в базе, – ввести, изменить или выбрать нужные.(select, delete,update, insert)
- Язык управления данными (**Data Control Language**, DCL). Защита базы данных от различных вариантов повреждения. При правильном использовании DCL обеспечивает защиту базы, а степень защищенности зависит от используемой реализации. Если реализация не обеспечивает достаточной защиты, то довести защиту до нужного уровня необходимо при разработке прикладной программы.

ЧТО ТАКОЕ ЗАПРОС?

Запрос

команда которую вы даете вашей программе базы данных, и которая сообщает ей, чтобы она вывела определенную информацию из таблиц в память. Эта информация обычно посылается непосредственно на экран компьютера или терминала которым вы пользуетесь, хотя, в большинстве случаев, ее можно также послать принтеру, сохранить в файле (как объект в памяти компьютера), или представить как вводную информацию для другой команды или процесса.

Запрос – это вопрос, который вы задаете базе данных. Если какие-либо ее данные удовлетворяют условиям вашего запроса, то SQL передает их вам.

КОМАНДЫ SQL

Язык SQL состоит из ограниченного числа команд, специально предназначенных для управления данными. Одни из этих команд служат для определения данных, другие – для их обработки, а остальные – для администрирования данных.

Операторы SQL делятся на:

- операторы определения данных (Data Definition Language, DDL) (CREATE, ALTER, DROP)
- операторы манипуляции данными (Data Manipulation Language, DML) (SELECT, INSERT, UPDATE, DELETE)
- операторы определения доступа к данным (Data Control Language, DCL) (GRANT, REVOKE, DENY)
- операторы управления транзакциями (Transaction Control Language, TCL) (COMMIT, ROLLBACK, SAVEPOINT)

ЗАРЕЗЕРВИРОВАННЫЕ СЛОВА SQL

Кроме команд, специальное значение в SQL имеют и некоторые другие слова. Вместе с командами они зарезервированы для специального использования, поэтому эти слова нельзя применять в качестве имен переменных или любым другим способом, для которого они не предназначены.

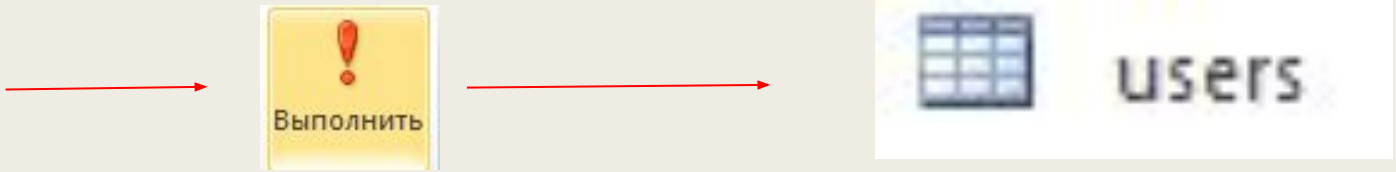
ADD	EXISTS	PRECISION
ALL	EXIT	PRIMARY
ALTER	EXTERNAL	PRINT
AND	FETCH	PROC
ANY	FILE	PROCEDURE
AS	FILLFACTOR	PUBLIC
ASC	FOR	RAISERROR
AUTHORIZATION	FOREIGN	READ
BACKUP	FREETEXT	READTEXT
BEGIN	FREETEXTTABLE	RECONFIGURE
BETWEEN	FROM	REFERENCES
BREAK	FULL	REPLICATION
BROWSE	FUNCTION	RESTORE
BULK	GOTO	RESTRICT
BY	GRANT	RETURN
CASCADE	GROUP	REVERT
CASE	HAVING	REVOKE
CHECK	HOLDLOCK	RIGHT
CHECKPOINT	IDENTITY	ROLLBACK
CLOSE	IDENTITY_INSERT	ROWCOUNT

СОЗДАНИЕ ТАБЛИЦЫ

Создание таблицы CREATE TABLE

CREATE TABLE *название_таблицы*, (*название_столбца1 тип данных* [DEFAULT *выражение*] [ограничение_столбца], *название_столбца2 тип данных* [DEFAULT *выражение*] [*ограничение_столбца*],[*ограничения_таблицы*]).

```
CREATE TABLE users(
id_user int (10) AUTO_INCREMENT,
name varchar(20) NOT NULL,
email varchar(50) NOT NULL,
password varchar(15) NOT NULL,
PRIMARY KEY (id_user)
);
```



Внесение данных INSERT

INSERT INTO *имя_таблицы*

VALUES('*значение_первого_столбца*','*значение_второго_столбца*', ...,'*значение_последнего_столбца*');

```
INSERT INTO users VALUES ('2','sergey', 'sergey@mail.ru', '1111');
```

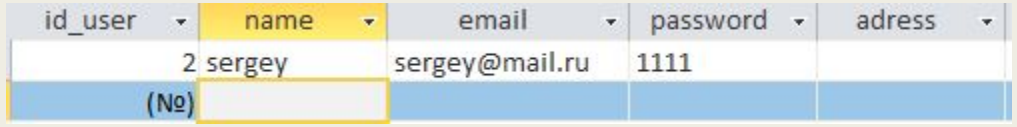
id_user	name	email	password
2	sergey	sergey@mail.ru	1111

Изменение таблицы ALTER TABLE

- **Для добавления столбцов** в таблицу используется оператор *ALTER TABLE - ADD COLUMN*.

ALTER TABLE *имя_таблицы* ADD COLUMN *имя_столбца тип*;

```
ALTER TABLE users ADD COLUMN adress int;
```



id_user	name	email	password	adress
2	sergey	sergey@mail.ru	1111	
(№)				

- Для изменения имени существующего столбца используется оператор *CHANGE*.

ALTER TABLE *имя_таблицы* CHANGE *старое_имя_столбца новое_имя_столбца тип*;

- **изменить только тип столбца**, то мы будем использовать оператор *MODIFY*. Его синтаксис следующий:

ALTER TABLE *имя_таблицы* MODIFY *имя_столбца новый_тип*;

Обновление таблицы UPDATE TABLE

UPDATE *имя_таблицы* SET *имя_столбца=значение_столбца* WHERE *условие*;

id_user	name	email	password	adress
2	sergey	sergey@mail.ru	1111	

```
UPDATE users SET name='sasha'
    WHERE id_user=2;
```

id_user	name	email	password	adress
2	sasha	sergey@mail.ru	1111	

Удаление таблицы и данных DROP/ DELETE

- оператор ***DELETE***, который позволяет **удалять строки из таблицы**.

  DELETE FROM *имя_таблицы* WHERE *условие*;

- **Удалить таблицу**

  DROP TABLE *имя_таблицы;*

ЗАПРОСЫ

выборка

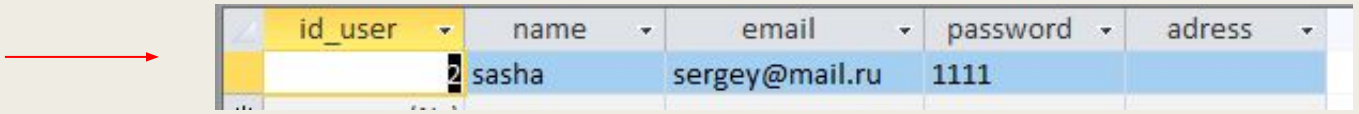
SELECT что_выбрать FROM откуда_выбрать;

Вместо "что_выбрать" мы должны указать либо имя столбца, значения которого хотим увидеть, либо имена нескольких столбцов через запятую, либо символ звездочки (*), означающий выбор всех столбцов таблицы. Вместо "откуда_выбрать" следует указать имя таблицы.

Выборка SELECT

SELECT *имя_столбца* FROM *имя_таблицы*;

```
Select * from users
```



id_user	name	email	password	adress
2	sasha	sergey@mail.ru	1111	

Сортировка ORDER BY

SELECT *имя_столбца* FROM *имя_таблицы* ORDER BY *имя_столбца_сортировки;*

```
SELECT *
FROM users
ORDER BY users.name;
```

id_user	name	email	password	adress
3	masha	maxa@mail.ru	2222	
4	pasha	pasha@mail.ru	3333	
2	sasha	sergey@mail.ru	1111	

По умолчанию сортировка идет по возрастанию, но это можно изменить, добавив ключевое слово ***DESC***

```
SELECT *
FROM users
ORDER BY users.name DESC ;
```

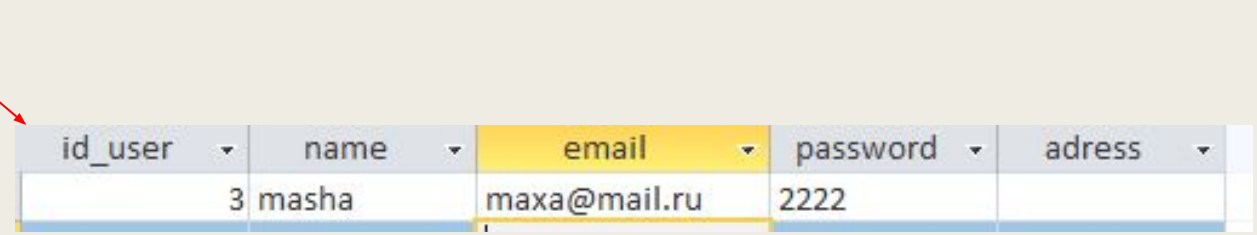
id_user	name	email	password
2	sasha	sergey@mail.ru	1111
4	pasha	pasha@mail.ru	3333
3	masha	maxa@mail.ru	2222

WHERE

Очень часто нам не нужна вся информация из таблицы. Для этого в SQL есть ключевое слово *WHERE*, синтаксис у такого запроса следующий:

SELECT *имя_столбца* FROM *имя_таблицы* WHERE *условие*;

```
SELECT *
FROM users
WHERE users.id_user=3;
```



id_user	name	email	password	adress
3	masha	maxa@mail.ru	2222	

Оператор	Описание	Пример
= (равно)	Отбираются значения равные указанному	SELECT * FROM users WHERE id_user=4;
> (больше)	Отбираются значения больше указанного	SELECT * FROM users WHERE id_user >2;
< (меньше)	Отбираются значения меньше указанного	SELECT * FROM users WHERE id_user <3;
>= (больше или равно)	Отбираются значения большие и равные указанному	SELECT * FROM users WHERE id_user >=2;

!= (не равно)	Отбираются значения не равные указанному	SELECT * FROM users WHERE id_user!=1;
IS NOT NULL	Отбираются строки, имеющие значения в указанном поле	SELECT * FROM users WHERE id_user IS NOT NULL;
IS NULL	Отбираются строки, не имеющие значения в указанном поле	SELECT * FROM users WHERE id_user IS NULL;
BETWEEN (между)	Отбираются значения, находящиеся между указанными	SELECT * FROM users WHERE id_user BETWEEN 1 AND 3;

IN (значение содержится)	Отбираются значения, соответствующие указанным	SELECT * FROM users WHERE id_user IN (1, 4);
NOT IN (значение не содержится)	Отбираются значения, кроме указанных	SELECT * FROM users WHERE id_user NOT IN (1, 4);
LIKE (соответствие)	Отбираются значения, соответствующие образцу	SELECT * FROM users WHERE name LIKE 'вел%';
NOT LIKE (не соответствие)	Отбираются значения, не соответствующие образцу	SELECT * FROM users WHERE name NOT LIKE 'вел%';

Метасимволы оператора LIKE

- Самый распространенный метасимвол - %. Он означает любые символы. Например, если нам надо найти слова, начинающиеся с букв "вел", то мы напишем LIKE 'вел%', а если мы хотим найти слова, которые содержат символы "клуб", то мы напишем LIKE '%клуб%'
- Еще один часто используемый метасимвол - _. В отличие от %, который обозначает несколько или ни одного символа, нижнее подчеркивание обозначает ровно один символ.

ВЛОЖЕННЫЕ ЗАПРОСЫ

Задача: Узнать email пользователя, который сделал заказ.

id_user	name	email	password	adress
2	sasha	sergey@mail.ru	1111	
3	masha	maxa@mail.ru	2222	
4	pasha	pasha@mail.ru	3333	

id_z	id_user	name_z	date_z
1	2	pen	
2	3	pencil	
3	2	paper	

- 1. Выбираем заказ, который нас интересует.

SELECT id_user FROM zakaz WHERE id_z = 1

- 2. Выбираем информацию о имейле интересующего пользователя.

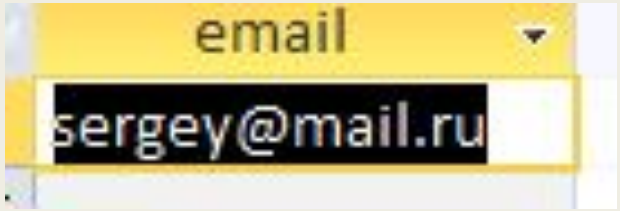
SELECT email FROM users WHERE id_user = *условие*

- 3. Соединяем запросы

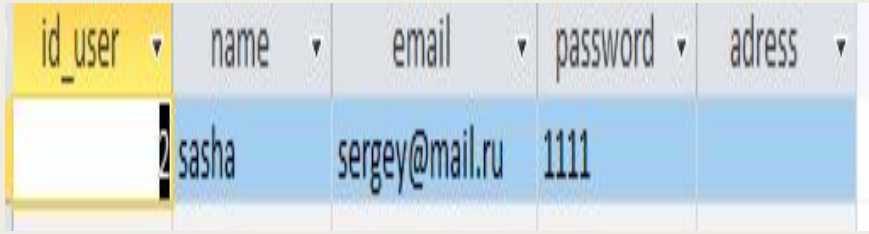
SELECT email FROM users WHERE id_user **IN**

(SELECT id_user FROM zakaz WHERE id_z = 1**)**

```
SELECT users.email FROM users WHERE users.id_user IN
(SELECT zakaz.id_user FROM zakaz WHERE zakaz.id_z = 1)
```




```
SELECT * FROM users WHERE users.id_user IN
(SELECT zakaz.id_user FROM zakaz WHERE zakaz.id_z = 1)
```



ОБЪЕДИНЕНИЕ

Если нам надо получить данные из двух таблиц, то Запросы, которые позволяют это сделать, в SQL называются *Объединениями*.

- *Внутренние объединения*. Такие объединения связывают строки одной таблицы со строками другой таблицы (а может еще и третьей таблицы). Но бывают ситуации, когда необходимо, чтобы в результат были включены строки, не имеющие связанных.
- *Внешние объединения*. позволяющим выводить все строки одной таблицы и имеющиеся связанные с ними строки из другой таблицы.

Декартовое произведение

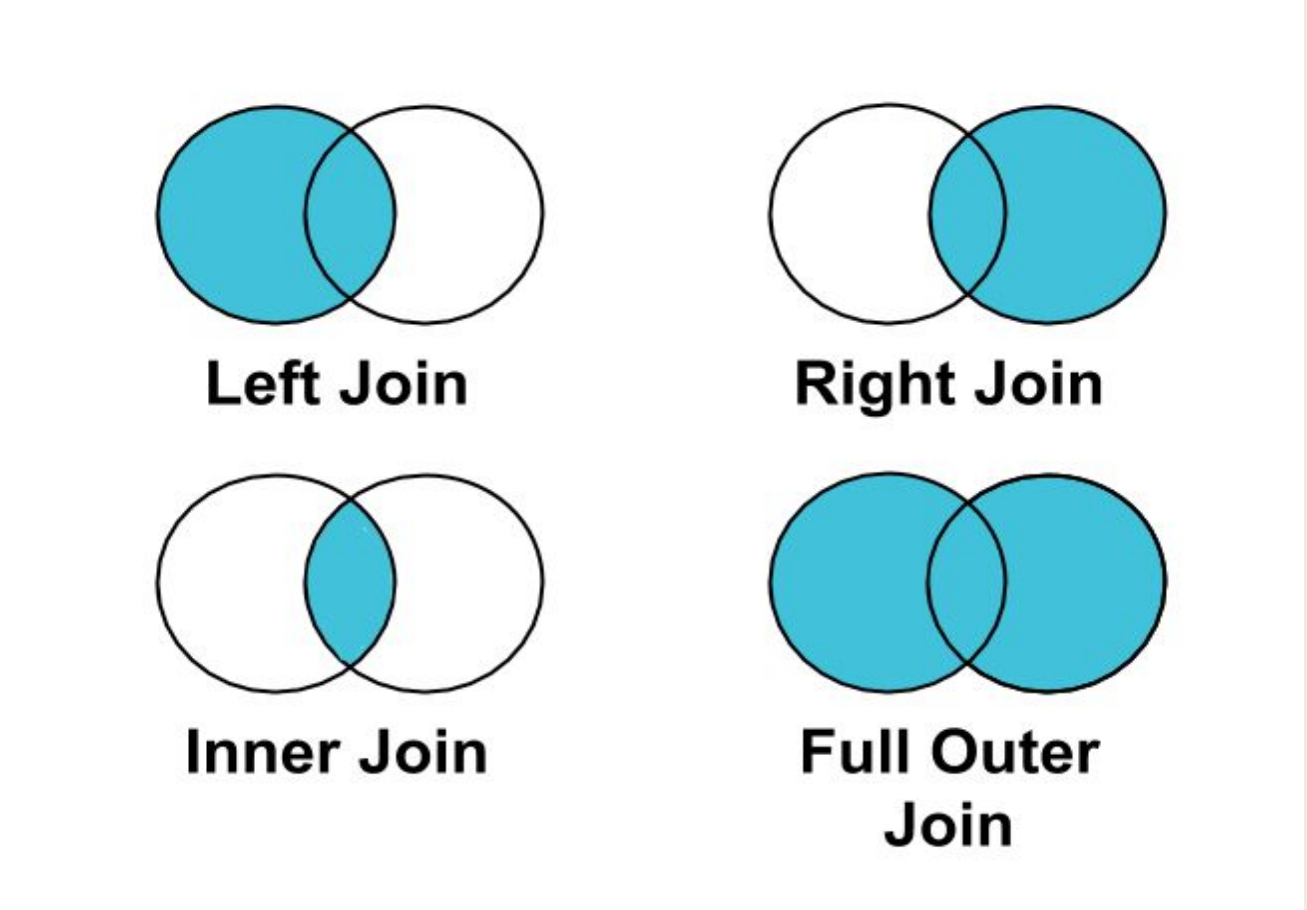
Синтаксис самого простого объединения следующий:

SELECT *имена_столбцов_таблицы_1, имена_столбцов_таблицы_2* FROM *имя_таблицы_1, имя_таблицы_2*;

Такое объединение научно называется ***декартовым произведением****, когда каждой строке первой таблицы ставится в соответствие каждая строка второй таблицы.*

Внешнее объединение
JOIN

SELECT *имя_таблицы_1.имя_столбца, имя_таблицы_2.имя_столбца*
FROM *имя_таблицы_1 ТИП ОБЪЕДИНЕНИЯ имя_таблицы_2*
ON *условие_объединения;*



Inner Join

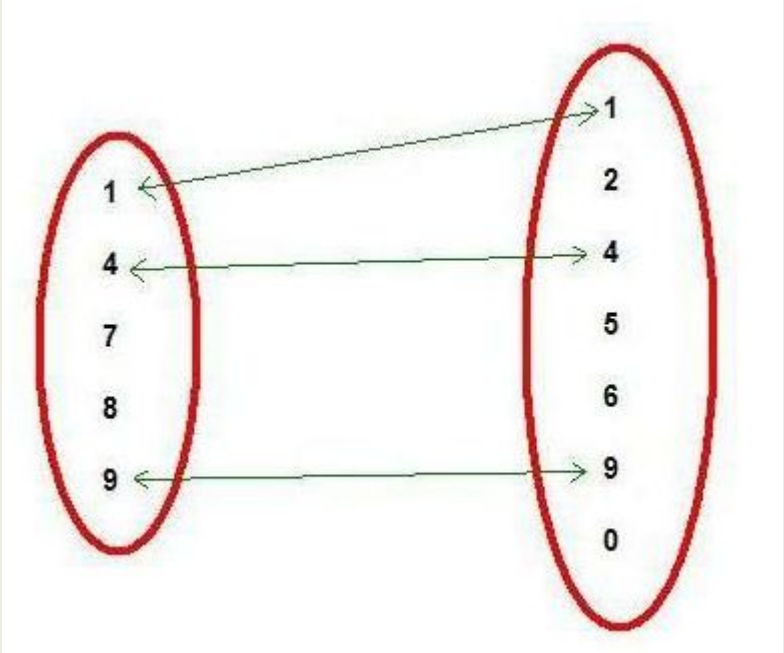
Мы имеем две таблицы.

Они в свою очередь, имеют разное количество записей.

В каждой из таблиц есть поля, которые связаны между собой.

Возвращаемый результат будет в виде набора записей из двух таблиц, где номера связанных между собой полей совпадают.

Проще говоря, запрос вернет только те записи (из таблицы номер два), данные о которых есть в таблице номер один. -



```
SELECT users.name, zakaz.id_z
FROM users INNER JOIN zakaz
ON users.id_user = zakaz.id_user;
```

name	id_z
sasha	1
masha	2
sasha	3

```
SELECT users.name, zakaz.id_z
FROM users LEFT JOIN zakaz
ON users.id_user = zakaz.id_user;
```

name	id_z
pasha	
sasha	3
sasha	1
masha	2

```
SELECT users.name, zakaz.id_z
FROM users RIGHT JOIN zakaz
ON users.id_user = zakaz.id_user;
```

name	id_z
sasha	1
masha	2
sasha	3

LEFT JOIN - из таблицы слева надо взять все строки.
RIGHT JOIN - тогда будут выбираться все строки из правой таблицы и имеющиеся связанные с ними из левой таблицы.

ВСТРОЕННЫЕ ФУНКЦИИ

Функции - это операции, позволяющие манипулировать данными. Можно выделить несколько групп встроенных функций:

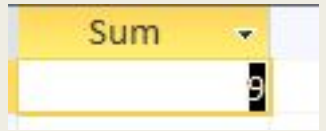
- Строковые функции. Используются для управления текстовыми строками, например, для обрезания или заполнения значений.
- Числовые функции. Используются для выполнения математических операций над числовыми данными. К числовым функциям относятся функции возвращающие абсолютные значения, синусы и косинусы углов, квадратный корень числа и т.д. Используются они только для алгебраических, тригонометрических и геометрических вычислений.
- Итоговые функции. Используются для получения итоговых данных по таблицам, например, когда надо просуммировать какие-либо данные без их выборки.
- Функции даты и времени. Используются для управления значениями даты и времени, например, для возвращения разницы между датами.
- Системные функции. Возвращают служебную информацию СУБД.

Итоговые функции

- AVG() Функция возвращает среднее значение столбца.
- COUNT() Функция возвращает число строк в столбце.
- MAX() Функция возвращает самое большое значение в столбце.
- MIN() Функция возвращает самое маленькое значение в столбце.
- SUM() Функция возвращает сумму значений столбца.

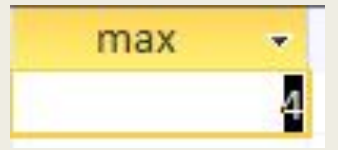
id_user	name	email	password
2	sasha	sergey@mail.ru	1111
3	masha	maxa@mail.ru	2222
4	pasha	pasha@mail.ru	3333

```
SELECT Sum(users.id_user) AS [Sum]
FROM users ;
```



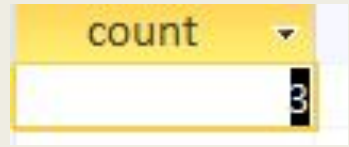
Sum
9

```
SELECT MAX(users.id_user) AS [max]
FROM users ;
```



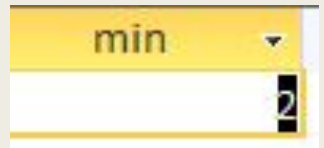
max
4

```
SELECT CUNT(users.id_user) AS [count]
FROM users ;
```



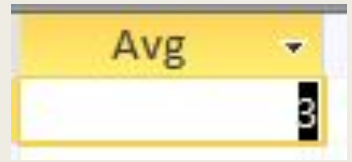
count
3

```
SELECT MIN(users.id_user) AS [min]
FROM users ;
```



min
2

```
SELECT AVG(users.id_user) AS [Avg]
FROM users ;
```



Avg
3

ГРУППИРОВКА И ФИЛЬТРАЦИЯ

Группировка GROUP BY

Оператор *GROUP BY* указывает СУБД сгруппировать данные по столбцу

id_user	name	email	password
2	sasha	sergey@mail.ru	1111
3	masha	maxa@mail.ru	2222
4	pasha	pasha@mail.ru	3333

id_z	id_user	name_z	date_z
1	2	pen	
2	3	pencil	
3	2	paper	

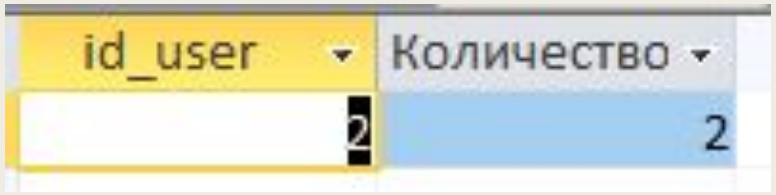
Необходимо узнать сколько заказов у клиентов.

```
SELECT zakaz.id_user,Count(zakaz.id_user) AS [Количество]
FROM zakaz
Group by zakaz.id_user
```

id_user	Количество
2	2
3	1

Фильтрация HAVING

```
SELECT zakaz.id_user,Count(zakaz.id_user) AS [Количество]
FROM zakaz
Group by zakaz.id_user
HAVING  Count(zakaz.id_user) >= 2
```



id_user	Количество
2	2