

**Простые  
запросы**

# Оператор SELECT

- Оператор **SELECT** используется для получения данных из одной или нескольких таблиц
- Самый простой вариант:
- **SELECT** \* **FROM** products;
- Вытаскивает все строки и все столбцы
- Можно указать набор столбцов вручную:
- **SELECT** name, price **FROM** products;

# Оператор SELECT

- В **SELECT** указывать не обязательно имена столбцов, там могут быть любые выражения
- **SELECT** name, lastName, CONCAT(lastName, ' ', name) **AS** fullName **FROM** people;
- **CONCAT** – стандартная функция, конкатенирующая переданные значения в одну строку
- Оператор **AS** задает **алиас (псевдоним/alias)** – в общем, дает полученному столбцу имя
- Также алиас можно использовать если мы хотим переименовать поля в результате запроса

# Необязательность AS а алиасах

- Обычно алиасы пишут с AS
- `SELECT CONCAT(lastName, ' ', name) AS fullName  
FROM people;`
- Но вообще, это слово не обязательно, его можно просто пропускать, результат будет тот же:
- `SELECT CONCAT(lastName, ' ', name) fullName  
FROM people;`

# ORDER BY

- Результат запроса можно отсортировать
- Отсортировать по имени по возрастанию
- `SELECT * FROM products  
ORDER BY name;`
- Аналогично:
- `SELECT * FROM products  
ORDER BY name ASC;`
- Отсортировать по цене по убывания
- `SELECT * FROM products  
ORDER BY price DESC;`

# ORDER BY

- Сортировать можно тоже не только по полям, но и по выражениям
- `SELECT * FROM people  
ORDER BY CONCAT(lastName, ' ', name);`
- Можно сортировать по нескольким столбцам, в любых направлениях
- `SELECT * FROM products  
ORDER BY name ASC, price DESC;`
- `# сначала отсортировали по возрастанию имени,  
# а потом по убыванию цены`

# WHERE

- Можно фильтровать строки
- `SELECT * FROM products`  
`WHERE price > 100;`
- Можно использовать вместе `WHERE` и `ORDER BY` (да и другие части тоже)
- У каждой части есть свое место в синтаксисе, менять порядок нельзя
- `ORDER BY` должен быть после `WHERE`
- `SELECT * FROM products`  
`WHERE price > 100`  
`ORDER BY price;`

# Операторы сравнения в SQL

- Чтобы проверить на равенство используется =, а не ==
- Чтобы проверить, что значения не равны между собой, используется <> , а не !=
- `SELECT * FROM products`  
`WHERE price = 30 AND name <> 'Intel Core i5';`
- Остальные операторы привычные:  
< > <= >=
- В MySQL также можно использовать !=, но это не стандарт



# Логические связи

- Логическое И (конъюнкция) – AND  
Логическое ИЛИ (дизъюнкция) – OR  
Логическое отрицание – NOT
- Примеры:
- `SELECT * FROM products`  
`WHERE price < 30 OR price > 100;`
- `SELECT * FROM products`  
`WHERE NOT (price >= 30 AND price <= 100);`
- Вместе с NOT обычно используются скобки, чтобы показать, к чему применяется отрицание

# Логические связи

- Логическое И – **AND**
  - Результат истинный только если все подусловия истинны
  - **SELECT \* FROM** products  
**WHERE** price  $\geq$  30 **AND** price  $\leq$  100;
- Логическое ИЛИ – **OR**
  - Результат истинный, если хотя бы одно из подусловий истинно
  - **SELECT \* FROM** products  
**WHERE** price  $<$  30 **OR** price  $>$  100;

# Сравнение с NULL

- **Очень важно:** операторы = и <> не работают с NULL
- Вместо этого используются специальные конструкции IS NULL и IS NOT NULL
- Примеры:
- `SELECT * FROM products  
WHERE name IS NOT NULL;`
- `SELECT * FROM products  
WHERE name IS NULL;`

# Вычисление по короткой схеме

- **Важно:** в SQL логические связки всегда вычисляются полностью, нет вычисления по «короткой схеме»
- Причем порядок вычисления подусловий не определен
- Пример:
- `SELECT * FROM table1  
WHERE col1 <> 0 AND col2 / col1 > 2;`
- Может упасть из-за ошибки деления на 0
- Так не упадет
- `SELECT * FROM table1  
WHERE col1 <> 0 AND col2 > 2 * col1;`

# Оператор LIKE

- Для строк кроме точного равенства есть оператор **LIKE**, позволяющий проверить соответствие строки некоторому шаблону
- В шаблоне можно использовать специальные символы:
  - % означает любое количество любых символов (может быть и 0 символов)
  - \_ означает 1 символ
- **SELECT \* FROM** products  
**WHERE** name **LIKE** 'Intel%';  
# получили товары с именами, начинающимися с Intel
- Кстати, поиск строк регистронезависимый, причем не только в **LIKE**, но и в = и <>

# Оператор LIKE

- `SELECT * FROM products`  
`WHERE name LIKE '_____';`  
# получили товары с именами из 5 символов
- Что если в строке должны встречаться \_ и/или %?
- Можно экранировать при помощи \
- `SELECT * FROM products`  
`WHERE name LIKE '%\_%';`  
# получили товары, в именах которых есть \_

# Оператор BETWEEN

- Часто требуется проверить, что число входит в некоторый диапазон
- `SELECT * FROM products`  
`WHERE price >= 100 AND price <= 1000;`
- Для этого уже есть специальный оператор `BETWEEN .. AND`, упрощающий эти действия
- `SELECT * FROM products`  
`WHERE price BETWEEN 100 AND 1000;`

# Операторы IN и NOT IN

- Часто требуется проверить, что значение поля имеет одно из некоторых значений
- Например, хотим получить заявки, статус которых «Новая» или «В процессе»
- `SELECT * FROM requests WHERE status = 1 OR status = 2;`
- # обычно хранят коды статусов, а не строки
- Но есть специальный оператор `IN`, позволяющий более кратко это записать



# Операторы IN и NOT IN

- Оператор **IN**:
- **SELECT \* FROM** requests  
**WHERE** status **IN** (1, 2);
- Есть также версия **NOT IN**, которая проверяет, что значение не является одним из перечисленных
- **SELECT \* FROM** requests  
**WHERE** status **NOT IN** (1, 2);

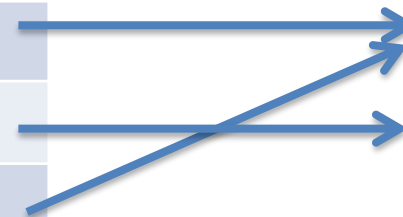
# Задача «Простые запросы»

- БД **world**, таблица **city**
1. Вывести все строки, но только столбцы имя и дистрикт
  2. Выведите только города из России, отсортируйте по имени
  3. Выведите города из Испании, Португалии и Греции, отсортируйте по имени по убыванию
  4. Вывести города, у которых население лежит в диапазоне от 300000 до 500000
  5. Вывести города, начинающиеся с буквы А
  6. Вывести города, содержащие букву А
  7. Вывести города из России с населением не менее 1 млн
  8. Вывести города из Испании, начинающиеся с буквы А, и города из Греции с численностью до 200000 человек

# Группировка, агрегатные функции

# GROUP BY

- В **SELECT** есть возможность группировки строк по некоторым значениям
- В одну группу попадают строки, где эти значения совпадают
- **SELECT** clientName, COUNT(\*) **AS** ordersCount  
**FROM** orders  
**GROUP BY** clientName  
**ORDER BY** ordersCount **DESC**;
- Например, здесь идет группировка по имени клиента. Заказы одного клиента попадают в одну группу



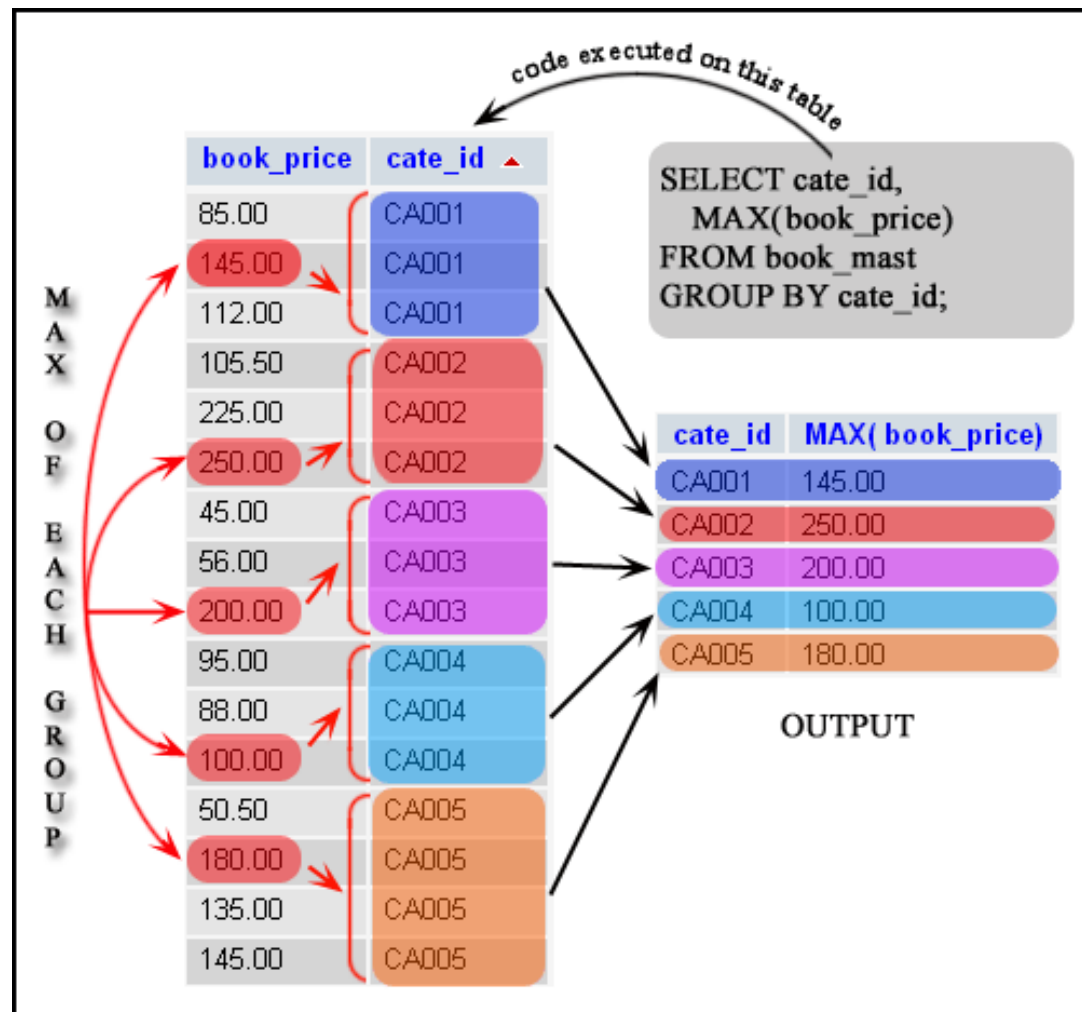
The diagram illustrates the process of grouping data. On the left, a table with columns 'id' and 'clientName' contains three rows: (1, Ivan), (2, Petr), and (3, Ivan). On the right, a table with columns 'clientName' and 'ordersCount' shows the result of grouping by client name: Ivan has 2 orders and Petr has 1 order. Blue arrows show the mapping: two arrows from the 'Ivan' rows in the first table point to the 'Ivan' row in the second table, and one arrow from the 'Petr' row in the first table points to the 'Petr' row in the second table.

id	clientName
1	Ivan
2	Petr
3	Ivan

clientName	ordersCount
Ivan	2
Petr	1

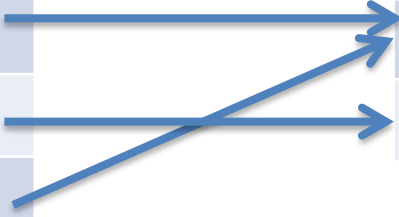
# GROUP BY

- `SELECT cate_id, MAX(book_price)`  
`FROM books`  
`GROUP BY cate_id;`
- Есть таблица с книгами.  
У нее есть столбцы с ценой (**book\_price**) и категорией (**cate\_id**)
- Мы хотим получить категории и максимальную цену по этой категории



# GROUP BY

- Группировка часто используется вместе с **агрегатными функциями** – это функции, которые из набора значений выдают одно значение
- Примеры: **MAX, MIN, COUNT, AVG, SUM**
- `SELECT` clientName, COUNT(\*) `AS` ordersCount  
`FROM` orders  
`GROUP BY` clientName  
`ORDER BY` ordersCount `DESC`;



id	clientName
1	Ivan
2	Petr
3	Ivan

clientName	ordersCount
Ivan	2
Petr	1

# Агрегатные функции

- **MAX** – максимум
- **MIN** – минимум
- **COUNT** – количество
- **AVG** – среднее арифметическое
- **SUM** – сумма

# Разные варианты COUNT

- COUNT имеет 3 полезных варианта:
  - COUNT(\*) - выдает число строк
  - COUNT(имяСтолбца) – выдает число не NULL значений в этом столбце
  - COUNT(DISTINCT имяСтолбца) – выдает число уникальных не NULL значений в этом столбце



# GROUP BY

- При использовании **GROUP BY** по стандарту в выражении **SELECT** можно выводить только:
  - Столбцы, которые перечислены в **GROUP BY**
  - Агрегатные функции от других столбцов
- В MySQL разрешено использовать в **SELECT** и столбцы, которые не участвуют в **GROUP BY**, но это не стандарт
- Тогда берется произвольное значение из группы

# GROUP BY по нескольким столбцам

- GROUP BY можно делать по нескольким столбцам
- Например, есть таблица **city** в базе **world**
- В ней есть поля **District** и **CountryCode**
- У некоторых стран есть дистрикты с одинаковым названием
- Поэтому если сделать такой запрос, то результат будет неверным
- ```
SELECT district, SUM(Population)  
FROM city  
GROUP BY district
```
- Здесь в одну группу попадут города, относящиеся к разным странам

# GROUP BY по нескольким столбцам

- Чтобы все было правильно, надо еще сгруппировать и по коду страны
- `SELECT` countryCode, district, `SUM`(Population)  
`FROM` city  
`GROUP BY` countryCode, district
- Теперь в одну группу попадут только города, у которых одна страна, и при этом один дистрикт

# HAVING

- Есть возможность фильтрации групп после группировки при помощи **HAVING**
- Заметим, что **WHERE** имеет другую роль – он отсеивает строки до группировки
- Например, хотим в предыдущем запросе вывести информацию только для тех клиентов, у кого заказов не меньше 2
- **SELECT** clientName, COUNT(\*) **AS** ordersCount  
**FROM** orders  
**GROUP BY** clientName  
**HAVING** COUNT(\*) >= 2  
**ORDER BY** ordersCount **DESC**;

# Задача «Group by»

- По таблице городов
  - Вывести все страны вместе с количеством городов
  - Вывести все страны вместе с количеством городов, оставить только страны, в которых не менее 2 городов
  - Вывести все страны вместе с количеством городов, в которых не менее 1 млн человек, оставить только страны, в которых не менее 2 таких городов
  - Найти среднюю численность населения городов по каждой стране, вывести в порядке убывания

# Стандартные функции

# Стандартные функции

- В любой СУБД есть различные стандартные функции – для работы с числами, строками, датой и временем и др.
- Их можно использовать в запросах
- В каждой СУБД они могут быть разные, но мы рассмотрим самые основные в MySQL
- Ссылка на документацию:
- <https://dev.mysql.com/doc/refman/8.0/en/functions.html>

# Математические функции

- **ABS(x)** – модуль числа
- **MOD(x, y)** – остаток от деления x на y
- **Тригонометрия:** SIN(x), COS(x), TAN(x), COT(x). Требуют угол в радианах
- **Преобразование углов между градусами и радианами:** RADIANS(x), DEGREES(x)
- **PI()** – выдает константу Пи
- Также есть возведение в степень, округления, логарифмы, квадратный корень, обрезка числа до нужного числа знаков и др.
- <https://dev.mysql.com/doc/refman/8.0/en/mathematical-functions.html>



# Строковые функции

- **CONCAT(str1, str2, ...)** – конкатенация строк
- **LOWER(str), UPPER(str)** – перевод в нижний и верхний регистр
- **CHAR\_LENGTH(str)** – длина строки (количество символов)
- **SUBSTRING(str, pos, len)** – получение подстроки из строки
- **REPLACE(str, from, to)** – замена подстроки
- **LOCATE(substr, str)** – выдает индекс подстроки в строке (отсчитывается от 1). Если не нашлось, выдает 0
- И др.
- <https://dev.mysql.com/doc/refman/8.0/en/string-functions.html>

# Функции для даты и времени

- **NOW()** – получить текущую дату и время
- **DAY(date), MONTH(date), YEAR(date)** – получить день, месяц, год от даты соответственно
- **WEEKDAY(date)** – получить день недели в виде числа от 0 (понедельник) до 6 (воскресенье)
- **DATE\_ADD(date, expr), DATE\_SUB(date, expr)** – прибавить/вычесть указанное время
- **DATEDIFF(expr1, expr2)** – разница между днями в датах, время игнорируется
- **LAST\_DAY(date)** – выдает дату-последний день месяца из этой даты
- <https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html>

# Задача «Стандартные функции»

- Всю эту задачу нужно обязательно присылать в одной цепочке писем
- По таблице стран:
  - Выведите страны в порядке убывания длины названия. И пусть название страны будет в верхнем регистре, а название континента — в нижнем
  - Для каждой страны найдите радиус окружности, у которой площадь круга такая же, как площадь этой страны. Отсортируйте результат по убыванию радиуса
- По задаче про магазин:
  - Сделайте, чтобы в таблице товаров у вас было какое-либо поле с типом дата
  - Выведите список товаров, чтобы день, месяц и год выводились отдельными полями
  - Выведите сколько товаров было в каждый год, который присутствует в этой таблице

# Хороший стиль форматирования SELECT

- Если не `SELECT *`, то нужно переносить `FROM` на следующую строку
- Каждая новая часть `SELECT`'а должна начинаться с новой строки: `WHERE`, `GROUP BY`, `HAVING`, `ORDER BY` и др.