Администрирование баз данных

Выборка данных из таблиц. Оператор *select*

Создать таблицу cars на основе sql dump

cars.sql

Файл cars.sql скопировать в папку

c:/mysql/dump/

Развернуть дамп таблицы в базу lesson2:

1 вариант

```
c:/mysql/bin>mysql -u root -p lesson2 < c:/mysql/dump/cars.sql
```

2 вариант

```
mysql>use lesson2;
```

```
mysql>source c:/mysql/dump/cars.sql;
```

select *select_expr* from *tbl_name*

“*” – символ * используется после оператора select если необходимо выбрать все столбцы из таблицы

`select_expr` – набор полей, разделенных запятой, которые будут возвращены в результате запроса

`tbl_name` – таблица, из которой извлекаются данные

Выбираем все колонки из таблицы

```
select * from cars;
```

Указываем имена колонок для выборки из таблицы

```
select make, model, petrol_type, year, price from cars;
```

Указываем имена колонок для выборки из таблицы.
Порядок выборки колонок может быть произвольным

```
select year, price, make, model from cars;
```

Условия WHERE

Для изменения количества выводимых строк используется ключевое слово *WHERE*, после которого следует логическое условие. Если запись удовлетворяет такому условию, то она попадает в выборку, в противном случае запись отбрасывается.

```
select * from cars where make='BMW'
```

Выбрать из таблицы все записи, где маркой является BMW.

```
select * from cars where make<>'BMW'
```

Выбрать из таблицы все записи, где маркой является все кроме BMW.

```
select * from cars where year>'2010'
```

Выбрать из таблицы все записи, где поле year больше 2010.

!!! Будет ли учитываться регистр в where части зависит от кодировки колонок таблицы

Суффиксы кодировок и зависимость от регистра

ci – case insensitive utf8_general_**ci**
cs – case sensitive cp1251_general_**cs**
bin – binary case sensitive utf8_**bin**

Просмотр кодировок для колонок таблицы:

```
show select full columns from tbl_name;
```

Составные логические условия

Условие может быть составным и объединяться при помощи нескольких логических операторов

Необходимо выбрать машины старше 2008 года и младше 2011 года включительно:

```
select * from cars where year>'2008' and year<='2011';
```

Аналогично выборку можно сделать при помощи конструкции

```
select * from cars where year between '2008' and '2011';
```

Конструкция, противоположенная конструкции `between` – `NOT BETWEEN`. Выбирает записи, не входящие в указанный диапазон:

```
select * from cars where year not between '2008' and '2011';
```

Извлечение записей, удовлетворяющих условию из списка (конструкция IN)

Необходимо выбрать машины марки BMW, AUDI, VOLKSWAGEN

```
select * from cars where make in ('BMW', 'AUDI', 'VOLKSWAGEN');
```

Обратная конструкция NOT
IN

Необходимо выбрать все машины кроме марки FORD,

```
select * from cars where make not in ('FORD', 'TOYOTA');
```

Сортировка записей (конструкция *ORDER BY*)

Изначально записи выбираются из таблицы в том порядке, в котором они хранятся в базе данных. Для сортировки данных необходимо после конструкции ORDER BY указать столбец, по которому происходит сортировка.

Необходимо отсортировать машины по

```
select * from cars order by price;
```

Сортировку можно проводить по нескольким столбцам.

Необходимо отсортировать машины по марки и

```
select * from cars order by make, price;
```

Необходимо отсортировать машины по марки и цене, выбрать толькj марки AUDI, BMW,

```
select * from cars where make in ('AUDI', 'BMW', 'TOYOTA')
order by make, price;
```

Сортировка записей (конструкция *ORDER BY*)

Обратная сортировка (по убыванию) выполняется с использованием ключевого слова *DESC*

Необходимо отсортировать машины по цене, начиная с самой большой

```
select * from cars order by price desc;
```

Прямая сортировка выполняется с помощью ключевого слова ***ASC***. Прямая сортировка используется по умолчанию, поэтому ASC можно не использовать

Ограничение выборки

Результатом выборки часто могут быть тысячи записей. Иногда бывает смыл предоставлять информацию порциями. Ограничивать выборку порциями можно с помощью ключевого слова ***LIMIT,*** за которым следует количество выбираемых записей.

Необходимо выбрать первые 20

```
select * from cars LIMIT 20;
```

Необходимо выбрать следующих 20 машин(выбираем 20 записей, начиная с

```
select * from cars LIMIT 20,20;
```

Необходимо выбрать 20 самых новых машин

```
select * from cars order by year desc LIMIT 20;
```

Необходимо выбрать 20 самых дорогих машин среди FORD и

```
select * from cars  where make in ('FORD', 'TOYOTA')
order by price desc
LIMIT 20;
```

Практические задания

1. Из таблицы cars выбрать все записи, где машины марки OPEL, цена которых не больше 10000 и пробег меньше 160000;

2. Выбрать всю информацию о 5 самых дешевых машинах марки AUDI;

3. Необходимая информация: год, марка, модель, цена.
   Условие: машины марки VOLKSWAGEN, BMW, MAZDA не старше 2010 года с пробегом меньше 150000, с дизельным двигателем. Отсортировать по цене, начиная с самой высокой;

Использование встроенных функций MySQL

При выполнении запросов часто требуется выполнить специфические задачи, для решения которых удобнее воспользоваться встроенными функциями MySQL

Функция имеет уникальное имя и может содержать несколько аргументов (или ни одного), которые перечисляются через запятую в круглых скобках. Если аргументы отсутствуют, круглые скобки все равно необходимо указывать. Между именем функции и круглыми скобками **НЕТ** пробела.

```
NOW()

version()
```

```
select now(), version() from dual;
```

```
select now(), 2+2 from dual;
```

Dual – псевдотаблица, на самом деле не существует

Функция count()

Функция в качестве аргументов принимает имена столбцов или символ *. Функция возвращает количество строк значения столбца в которых отличны от NULL

```
select count(make) from cars;
```

```
select count(*) from cars;
```

Функция min(), max()

Функция в качестве аргументов принимает имена столбцов. Функция возвращает минимальное или максимальное значение столбца, переданного в качестве аргумента

```
select min(price) from cars;
```

```
select max(price) from cars;
```

Функция count()

Функция в качестве аргументов принимает имена столбцов или символ *. Функция возвращает количество строк значения столбца в которых отличны от NULL

```
select count(make) from cars;
```

```
select count(*) from cars;
```

Функция min(), max()

Функция в качестве аргументов принимает имена столбцов. Функция возвращает минимальное или максимальное значение столбца, переданного в качестве аргумента

```
select min(price) from cars;
```

```
select max(price) from cars;
```

Использование оператора AS

Одной из особенностей использования функций является то, что в результате запроса название колонки совпадает с названием функции и ее параметрами. Это неудобно, особенно в прикладных программах, где после выполнения запроса обращение к результату происходит по имени столбца. В Select запросе столбцу можно присвоить имя, для этого предназначен оператор **AS**.

```
select count(make) as total from cars;
```

```
select min(price) as minimum from cars;
```

Группировка записей

Необходимо определить марки машин, находящиеся в нашей таблице

```
select make from cars order by make;
```

Результат запроса не удобен для восприятия. Нужно, чтоб отбирались только уникальные записи. Для этого перед именем столбца используем ключевое слово *DISTINCT*

```
select distinct(make) from cars order by make;
```

В результате запрос вернет только уникальные значения колонки *make*

Совместное использование distinct и

```
select count(distinct(make)) from cars order by make;
```

Группировка записей

Для группировки записей также используют конструкцию GROUP BY. GROUP BY в SELECT запросах располагается перед ORDER BY и LIMIT.

```
select make from cars
GROUP BY make
order by make;
```

В отличии от DISTINCT совместное использование GROUP BY совместно с COUNT() приводит не к подсчету всех строк в таблице, а к выводу количества записей, соответствующих каждому уникальному значению указанному в GROUP BY

```
select make, count(*) from cars
GROUP BY make
order by make;
```

Группировка записей

Иногда требуется ограничить выборку по результату функции, например, выбрать марки машины, которых на складе больше 5. Использование конструкции WHERE, в таком случае приведет, к **ошибке**

```
select make, count(*) as total from cars
where count(*)>5
GROUP BY make
order by make;
```

Для решения данной проблемы вместо ключевого слова *WHERE* используется ключевое слово *HAVING*, которое располагается **после** конструкции GROUP BY

```
select make, count(*) as total from cars
GROUP BY make
having count(*)>5
order by make;
```

Группировка записей

В условии HAVING можно использовать все столбцы результирующей таблицы, не только вычисляемые. Например, сгруппировать по годам количество машин, которые моложе 2010 года

```
select reg_year, count(*) as total from cars
GROUP BY reg_year
having reg_year>2010
order by reg_year
```

Для не вычисляемых столбцов для ограничения выборки можно использовать конструкцию WHERE вместо HAVING.

```
select reg_year, count(*) as total from cars
where reg_year>2010
GROUP BY reg_year
order by reg_year
```

Группировка записей

Группировку можно проводить по нескольким столбцам. В конструкцию GROUP BY необходимо указывать все столбцы участвующие в сортировке

Вывести количество машин соответствующих конкретной марке и году выпуска

```
select make, reg_year, count(*) as total from cars

where reg_year>2010

GROUP BY make,reg_year

order by make, reg_year
```

Задание

Создать отчет со следующими колонками

Make	Model	Quantity

Условия выборки:

1. Машины моложе 2010 года с пробегом меньше 160000
2. Марка: AUDI, BMW, TOYOTA, VOLKSWAGEN
3. В отчет попадают только марки и модели, количество которых минимум 2
4. Сгруппировать по марке, модели

Объединение результатов разных запросов в одну таблицу

Оператор SELECT возвращает результат в виде таблиц. Если формат результирующих таблиц (количество, порядок следования и тип столбцов) совпадают, то результат можно объединить в одну таблицу.

Запрос 1

```
select make, model, reg_year, price
from cars
where reg_year=2018
   and km<5000
order by make, model;
```

Запрос 2

```
select make, model, reg_year price
from cars
where price<40000
   and reg_year>2016
   and make in ('BMW', 'AUDI')
order by make, model;
```

Объединение нескольких запросов осуществляется с помощью оператора ***UNION***

```
select make, model, km, reg_year, price from cars
where reg_year=2018 and km<5000
UNION
select make, model, km, reg_year, price from cars
where price<40000
    and reg_year>=2017
    and make in ('BMW', 'AUDI')
order by make, model;
```

При использовании оператора ***UNION*** в результирующую таблицу выбираются только уникальные строки. В нашем случае оба изначальных запроса содержат одну строку с

```
select make, model, reg_year, km, price
from cars
where reg_year=2018
    and km<5000
order by make, model;
```

```
+-----------+-------+----------+------+----------+
| make      | model | reg_year | km   | price    |
+-----------+-------+----------+------+----------+
| AUDI      | A8    |     2018 | 3760 | 39000.00 |
| AUDI      | A8    |     2018 | 4000 | 47600.00 |
| BMW       | 325   |     2018 | 4300 | 38500.00 |
| TOYOTA    | AURIS |     2018 | 1500 | 18400.00 |
| TOYOTA    | CH-R  |     2018 | 3500 | 23800.00 |
| VOLKSWAGEN| PASSAT|     2018 | 4500 | 26200.00 |
| VOLVO     | XC90  |     2018 |  500 | 44700.00 |
+-----------+-------+----------+------+----------+
7 rows in set (0.00 sec)
```

```
select make, model, reg_year, km, price
from cars
where price<40000
    and reg_year>2016
    and make in ('BMW', 'AUDI')
order by make, model;
```

```
+------+-------+----------+-------+----------+
| make | model | reg_year | km    | price    |
+------+-------+----------+-------+----------+
| AUDI | A8    |     2018 |  3760 | 39000.00 |
| AUDI | A8    |     2018 | 12000 | 33200.00 |
| AUDI | Q7    |     2017 | 26000 | 36500.00 |
| AUDI | Q7    |     2017 | 26000 | 33500.00 |
| BMW  | 325   |     2018 |  4300 | 38500.00 |
+------+-------+----------+-------+----------+
5 rows in set (0.00 sec)
```

При использовании оператора ***UNION*** в результирующей таблице дублирующая строка

Чтобы отобразить в результирующей таблице все строки исходных запросов используется оператор ***UNION ALL*** (дублирующиеся строки не убираются)

```
select make, model, km, reg_year, price from cars
where reg_year=2018 and km<5000
UNION ALL
select make, model, km, reg_year, price from cars
where price<40000
    and reg_year>=2017
    and make in ('BMW', 'AUDI')
order by make, model;
```

Создание таблицы на основе *select* запроса

Создадим новые таблицы на основе запроса. Созданная таблица будет состоять из колонок и их типов таких же как в исходной таблице.

```
CREATE TABLE tbl_name
AS
SELECT COLUMN1, COLUMN2,…COLUMN_N FROM other_tbl_name
WHERE condition_1 and condition_2;
```

Создадим 2 таблицы (car_branch1 и cars_branch2) на основе данных из таблицы ***cars***

```
create table car_branch1
as
select * from cars
where make in ('AUDI', 'BMW', 'VOLKSWAGEN', 'VOLVO', 'MERCEDES');
```

```
create table car_branch2
as
select * from cars
where make not in ('AUDI', 'BMW', 'VOLKSWAGEN', 'VOLVO', 'MERCEDES');
```

Создание только структуры таблицы на основе другой таблицы

Создается только структура таблицы, без данных. Все колонки и их типы соответствуют исходной таблице

```
CREATE TABLE new_table LIKE old_table;
```

Объединение запросов из двух таблиц с помощью *union*

Объединим запросы и таблиц car_branch1 и car_branch2

```
 select make, model, reg_year, km, price
 from car_branch1
 where price<12000
union all
 select make, model, reg_year, km, price
 from car_branch2
 where price<15000 and reg_year>2012
```

Задание.

Построить отчет по филиалам car_branch1 и car_branch2. Необходима информация о наличии машин марки VOLVO и TOYOTA, с пробегом до 90000км, не старше 2012 года, цена до 20000.