Операторы IN, BETWEEN, LIKE, IS NULL

При задании логического условия в предложении **WHERE** могут быть использованы операторы **IN, BETWEEN, LIKE, IS NULL.**

Операторы **IN** (РАВЕН ЛЮБОМУ ИЗ СПИСКА) и **NOT IN** (НЕ РАВЕН НИ ОДНОМУ ИЗ СПИСКА) используются для сравнения проверяемого значения поля с заданным списком. Этот список значений указывается в скобках справа от оператора **IN.** Список значений не обязательно задается в явном виде, он может представлять собой результат подзапроса.

Построенный с использованием **IN** предикат (условие) считается истинным, если значение поля, имя которого указано слева от **IN,** в точности *совпадает с* одним из значений, перечисленных в списке, указанном в скобках справа от **IN.**

Предикат, построенный с использованием **NOT IN,** считается истинным, если значение поля, имя которого указано слева от **NOT** IN, *не совпадает* ни с одним из значений, принадлежащих списку, указанному в скобках справа от **NOT IN.**

Пример 1.

Получить из таблицы EXAM_MARKS сведения о студентах; *имеющих* экзаменационные оценки только 4 и 5.

```
SELECT *
FROM EXAM_MARKS
WHERE MARK IN (4,5);
```

Пример 2.

Получить сведения о студентах, *не имеющих* ни одной экзаменационной оценки, равной 4 или 5

```
SELECT *
FROM EXAM_MARKS
WHERE MARK NOT IN (4, 5);
```

Оператор **BETWEEN** используется для проверки условия вхождения значения поля в заданный интервал, т.е. вместо списка значений атрибута этот оператор задает границы его изменения.

Например, запрос, выполняющий вывод записей о предметах обучения, для которых количество отводимых часов лежит в пределах между 30 и 40, имеет вид:

```
SELECT *
```

FROM SUBJECT

WHERE HOUR **BETWEEN** 30 **AND** 40;

Граничные значения, в данном случае значения: 30 и 40, входят во множество значений, с которыми производится сравнение. Оператор **BETWEEN** может использоваться как для числовых, так и для символьных типов полей.

Оператор **LIKE** применим только к символьным полям типа **CHAR** или **VARCHAR**. Этот оператор осуществляет просмотр строковых значений полей с целью определения, входит ли заданная в операторе **LIKE** подстрока (образец поиска) в символьную строку, являющуюся значением проверяемого поля.

Для того чтобы осуществлять выборку строковых значений по заданному образцу подстроки, можно применять шаблон искомого образца строки, использующий следующие символы:

- символ подчеркивания "_", указанный в шаблоне образца, определяет возможность наличия в указанном месте одного любого символа,
- символ "%" допускает присутствие в указанном месте проверяемой строки последовательности любых символов произвольной длины.

Пример.

Написать запрос, выбирающий из таблицы STUDENT сведения о студентах, у которых фамилии начинаются на букву "P".

SELECT *

FROM STUDENT

WHERE SURNAME **LIKE** 'P%';

Так как символы "_" и "%" выполняют в языке **SQL** указанные выше специальные функции, возникает проблема, когда необходимо их указывать в текстовом образце в качестве обычных, а не служебных символов. В этих случаях применяют специальный механизм, позволяющий при интерпретации системой строки-образца отключить управляющие функции этих символов. Отключить служебные функции сиволов "_" и "%" можно путем вставки непосредственно перед ними так называемого escape-символа (escape character). Этот символ, который можно еще назвать знаком перехода или знаком отключения, является служебным знаком, используемым для указания того, что должен быть изменен характер интерпретации следующего непосредственно за ним символа. В нашем случае если такой символ предшествует знаку "_" или "%", то этот знак будет интерпретироваться уже буквально, как любой другой символ, а не как служебный символ. В

SQL в качестве такого переключающего: (escape) символа может быть назначен любой символ. для этих целей. Предназначено специальное ключевое слово **ESCAPE**.

Например можно задать образец поиска с. помощью следующего выражения:

LIKE '__P' **ESCAPE** '\'

В этом выражении символ "\" с помощью ключевого слова **ESCAPE** объявляется **escape**-символом. Первый символ "_" в заданном шаблоне поиска "__ P" будет соответствовать, как и ранее, любому набору символов в проверяемой строке. Однако второй символ "_", следующий после символа "\", объявленного escape-символом, уже будет интерпретироваться буквально как обычный символ, так же как и символ P в заданном шаблоне.

Обращаем ваше внимание на то, что в операторах сравнения =, <, >, <=, >=, <> и операторах **IN, BETWEEN** и **LIKE** при использовании **NULL** в качестве операнда будет возвращаться также **NULL**; В связи с этим, для проверки содержимого поля на наличие (отсутствие): в нем пустого значения **NULL** следует использовать специально предназначенные для этого операторы **IS NULL** (ЯВЛЯЕТСЯ ПУСТЫМ) и **IS NOT NULL** (НЕ ЯВЛЯЕТСЯ ПУСТЫМ), а не выражения **=NULL** или **<>NULL**

УПРАЖНЕНИЯ

1. Напишите запрос, выполняющий вывод находящихся в таблице EXAM_MARKS номеров предметов обучения, экзамены по которым сдавались между 10 и 20 января 2005 г.

2. Напишите запрос, выбирающий данные обо всех предметах обучения, экзамены по которым сданы студентами, имеющими идентификаторы 12 и 32.

3. Напишите запрос, который выполняет :вывод названий. предметов обучения, начинающихся на букву 'И'-;

4. Напишите запрос, выбирающий сведения о студентах, у которых имена начинаются на букву 'И' или 'С'.

5. Напишите запрос для выбора из таблицы EXAM_MARKS записей, для которых отсутствуют значения оценок (поле MARK).

6. Напишите запрос, выполняющий вывод из таблицы EXAM_MARKS записей, для которых в поле NARK проставлены значения оценок.

7. Напишите запрос для получения списка преподавателей, проживающих в городах, в названиях которых присутствует дефис.

8. Напишите запрос для получения списка учебных заведений, в названиях которых использованы кавычки.

9. Напишите запрос для получения списка предметов, названия которых оканчиваются на 'ия'.

10. Напишите запрос для получения списка учебных заведений, в названиях которых содержится слово 'университет'.

11. Напишите запрос для получения списка студентов, фамилии которых начинаются на 'Ков' или на 'Куз'.

12. Напишите запрос для получения списка предметов обучения, названия которых состоят из более одного слова.

13. Напишите запрос для получения списка учебных заведений, названия которых состоят как минимум из 7 слов.

14. Напишите запрос для получения списка студентов, фамилии которых состоят из трех букв.

Преобразование вывода и встроенные функции

В SQL реализованы операторы преобразования данных и встроенные функции, предназначенные для работы со значениями столбцов и/или константами в выражениях. Использование этих операторов допустимо в запросах везде, где можно использовать выражения.

Числовые, символьные и строковые константы.

Несмотря на то, что SQL работает с данными в понятиях строк и столбцов таблиц, имеется возможность применения значений выражений, построенных с использованием встроенных функций, констант, имен столбцов, которые определяются как своего рода виртуальные столбцы. Они помещаются в списке столбцов и могут сопровождаться псевдонимами,

Если в запросе вместо спецификации столбца SQL обнаруживает числа, то оно интерпретируется как числовая константа.

Символьные константы должны указываться в одинарных кавычках. Если одинарная кавычка должна выводиться как часть строковой константы, то ее нужно предварить другой одинарной кавычкой.

Например, результатом выполнения запроса

SELECT 'Фамилия', SURNAME, 'Имя', NAME, 100

FROM STUDENT;

является таблица следующего вида:

	SURNAME		NAME	
Фамилия	Иванов	ИМЯ	Иван	100
Фамилия	Петров	ИМЯ	Пётр	100
Фамилия	Сидоров	ИМЯ	Вадим	100
Фамилия	Кузнецов	ИМЯ	Артём	100

Арифметические операции для преобразования числовых данных.

• Унарный (одиночный) оператор - (знак минус) изменяет знак числового значения, перед которым он стоит, на противоположный.

• Бинарные операторы +, -, * и / предоставляют возможность выполнения арифметических операций сложения, вычитания, умножения и деления.

Например, результат запроса

SELECT SURNAME, NAME, STIPEND, - (STIPEND*KURS)/2
FROM STUDENT
WHERE KURS = 4 **AND** STIPEND >0;

Будет выглядеть следующим образом.

SURNAME	NAME	STIPEND	KURS	
Сидоров	Вадим	150	4	-300
Петров	Антон	200	4	-400

Символьная операция конкатенации строк.

Операция конкатенации, обозначаемая символом "||", позволяет соединять ("склеивать") значения двух или более столбцов символьного типа или символьных констант в одну строку.

Эта операция имеет синтаксис

<значимое символьное выражение> || <значимое символьное выражение>.

Например:

SELECT SURNAME || '_' || NAME, STIPEND
FROM STUDENT
WHERE KURS = 4 AND STIPEND > O;

Результат запроса будет выглядеть следующим образом:

	STIPEND
Сидоров_Вадим	150
Петров_Антон	200

Символьные функции преобразования букв различных слов в строке.

- LOWER — перевод в строчные символы (нижний регистр)
  LOWER (<строка>) •
- UPPER — перевод в прописные символы (верхний. Регистр)
  UPPER (<строка >)
- INITCAP — перевод первой буквы каждого слова строки в заглавную (прописную)

  INITCAP (<строка>

Например:

SELECT LOWER(SURNAME), **UPPER**(NAME)
FROM STUDENT

WHERE KURS = 4 **AND** STIPEND > 0;

Результат запроса будет выглядеть следующим образом

SURNAME	NAME
Сидоров Петров	Вадим Антон

Символьные. строковые функции.

- LPAD —. дополнение строки слева.

  LPAD (<строка>, <длина> [,<подстрока.>]

  <строка> дополняется слева указанной в <подстроке> последовательностью символов до указанной <длины>.(возможно, с повторением последовательности);

  если <подстрока> не указана, то по умолчанию <строка> дополняется пробелами;

  если <длина> меньше длины <строки>, то исходная <строка> усекается слева до заданной <длины>.

- RPAD — дополнение строки cnрава

  RPAD(<строка>,<длина>[, <подстрока>])

  <строка> дополняется справа указанной в <подстроке>

  последовательностью символов до указанной <длины> (возможно, с повторением последовательности);

  — если <подстрока> не указана, то по умолчанию <строка> дополняется пробелами;

  — если <длина> меньше длины <строки>, то исходная <строка> усекается справа до заданной <длины>.

- LTRIM — удаление левых граничных символов

  LTRIM (<строка> [, <подстрока>])

  — из <строки> удаляются слева символы, указанные в <подстроке>;

  — если <подстрока> не указана; то по умолчанию удаляются пробелы;

  — в <строку> справа добавляется столько пробелов, сколько символов слева было удалено, т.е. длина <строки> остается неизменной.

- RTRIM — удаление правых граничных символов

  RTRIN(<строка>[,<подстрока>])

  —из <строки> удаляются справа символы, указанные в <подстроке>;

  - если <подстрока> не указана, то по умолчанию удаляются пробелы;

  - в <строку> слева добавляется, столько пробелов, сколько символов справа было удалено, т.е. длина <строки> остается неизменной.

Функции LTRIM и RTRIM рекомендуется использовать при написании условных выражений, в которых сравниваются текстовые строки. Дело в том, что наличие начальных или конечных пробелов в сравниваемых операндах может исказить результат сравнения.

Например, константы ' AAA' и ' AAA ' не равны друг другу.

• SUBSTR — выделение подстроки

SUBSTR(<строка>, <начало>[, <количесиио>])

— из <строки> выбирается заданное <количество> символов,

начиная с указанной позиции в строке <начало>;

— если <количество> не задано, символы выбираются с <начала> и до конца <строки>;

— возвращается подстрока, содержащая число символов, заданное параметром <количество>, либо число символов от позиции, заданной параметром <начало>, до конца <строки>;

— если указанное <начало> превосходит длину <строки>, то возвращается строка, состоящая из пробелов. Длина этой строки будет равна заданному <количеству> или исходной длине <строки> (при не заданном <количестве>),

INSTR — поиск подстроки

INSTR(<строка>, <подстрока> [, <начало поиска>'[,<номер вхождения>]])

— <начало поиска> задает начальную позицию в строке для поиска <подстроки>; если не задано, то по умолчанию принимается значение 1;

— <номер вхождения> задает порядковый номер искомой подстроки; если не задан, то по умолчанию принимается значение 1;

—. значимые выражения в, <начале поиска> или в <номере вхождения> должны иметь беззнаковый целый тип или приводиться к этому типу.

- LENGTH — определение длины строки
  LENGTH(<cmpoxa >)

— длина <строки>, тип возвращаемого значения — INT;

— функция возвращает NULL, если <строка> имеет NULL-значение.

Пример 1

Результат запроса

SELECT LPAD(SURNAME, 10, '@'), **RPAD**(NAME,10, '$'.)
FROM STUDENT
WHERE KURS = 3 **AND** STIPEND > 0;

будет выглядеть следующим образом

@@@@Петров	Пётр$$$$$$
@@@@Павлов	Андрей$$$$$$
@@@@Лукин	Артём$$$$$$

Пример 2. Запрос

SELECT SUBSTR(NAME, 1,1) ||'.' || SURNAME, CITY,
LENGTH(CITY)
FROM STUDENT
WHERE KURS IN(2, 3, 4) AND STIPEND > 0;

выдаст результат

	CITY	
П.Петров	Курск	5
С.Сидоров	Москва	6
0.Зайцева	Липецк	6
А. Лукин	Воронеж	7
А. Петров	NULL	NULL

Функции работы с числами.

• ABS — абсолютное значение

ABS (<значимое числовое выражение>)

• FLOOR — наибольшее целое, не превосходящее заданное число с плавающей точкой

FLOOR(<значимое числовое выражение>)

• CEIL — наименьшее целое, которое равно или больше заданного числа

CEIL(<значимое числовое выражениё>)

— Функция округления — ROUND

ROUND (<значимое числовое выражение>, <точность>)

аргумент <точность> задает точность округления (см. пример ниже).
• Функция усечения — TRUNC
TRUNC (<значимое числовое выражение>, <точность>)
• Тригонометрические функции — COS, SIN, TAN
COS(<значимое числовое выражение>)
SIN(<значимое числовое выражение>)
TAN(значимое числовое выражение>)
• Гиперболические функции — COSH, SINH, TANH
COSH(<значимое числовое выражение>)
SINH(<значимое числовое выражение>)
TANH(<значимое числовое выражение>)
• Экспоненциальная функция — EXP
EXP*(<значимое числовое выражение>)*
• Логарифмические функции — ***LN****,* LOG
LN*(<значимое числовое выражение>)*
LOG*(<значимое числовое выражение>)*
• Функция возведения в степень — POWER
POWER *(<значимое числовое выражение>,<показатель степени>)*
• Определение знака числа — SIGN
SIGN*(<значимое числовое выражение>)*
• Вычисление квадратного корня — SQRT
SQRT *(<значимое числовое выражение>)*

Пример. Запрос
SELECT UNIV_NAME, RATING,
ROUND(RATING, -1), **TRUNC**(RATING, -1)
FROM UNIVERSITY;
вернет результат

UNIV_NAME	RATING		
МГУ	610	610	600
ВГУ	296	300	290
НГУ	345	350	340
РГУ	421	420	410

Функции преобразования значений.
• Преобразование в символьную строку — **TO_CHAR**
TO_CHAR *(<значимое выражение> [, <символьный формат>])*
— *<значимое выражение>* должно представлять числовое значение или значение типа дата-время;
— для числовых значений <символьный формат> должен иметь синтаксис [S]9[9…][.9[9…]], где S – представление знака числа (при отсутствии предполагается без отображения знака), 9 — представление цифр-знаков числового значения (для каждого знакоместа). Символьный формат определяет вид отображения чисел. По умолчанию для числовых значений используется формат '999999.99';
для значений типа ДАТА-ВРЕМЯ <символьный формат> имеет вид (т. е. вид отображения значений даты и времени)
— в части даты:
'DD-Mon-YY'
'DD-Mon-YYYY'

'MM/DD/YY'
'MM/DD/YYYY'
'MM.DD.YY'
'MM.DD.YYYY'
В части времени:
'HH24'
'HH24:Ml'
'HH24:MI.SS'
'HH24:Ml:SS.FF'
где: - HH24 — часы в диапазоне от 0 до 24;
Ml — минуты;
SS — секунды;
FF — тики (сотые доли секунды)

При выводе времени в качестве разделителя по умолчанию используется двоеточие (":"), но при желании можно использовать любой другой символ.

Возвращаемое значение — символьное представление <значимого выражения> в соответствии с заданным <символьным форматом> преобразования.

• Преобразование из символьного значения в числовое – TO_NUMBER

TO_NUMBER(<значимое символьное выражение>)

При этом <значимое символьное выражение> должно задавать символьное значение числового типа.

• Преобразование символьной строки в дату — TO_DATE

TO_DATE (<значимое символьное выражение> [,
<символьный формат>])

- <значимое символьное выражение> должно задавать символьное значение типа ДАТА-ВРЕМЯ;

— <символьный формат> должен описывать представление значения типа ДАТА-ВРЕМЯ в <значимом символьном выражении>. Допустимые форматы (в том числе и формат по умолчанию) приведены выше.

Возвращаемое значение — <значимое символьное выражение> во внутреннем представлении. Тип возвращаемого значения — DATE. Над значениями типа DATE разрешены следующие операции:

• к значению типа DATE можно прибавлять значения типа INTERVAL, в результате чего получается значение типа DATE;

• при вычитании двух значений типа DATE получается значение типа INTERVAL;

• при вычитании из значения типа DATE значения типа INTERVAL получается значение типа DATE.

В бинарных операциях один из операндов должен иметь значение отдельного элемента даты: только год, или только месяц, или только день.

Пример. Запрос

SELECT SURNAME, MAME, BIRTHDAY,
TO_CHAR(BIRTHDAY, 'DD-Mon-YYYY'),
TO_CHAR(BIRTHDAY, 'DD.MM.YY')
FROM STUDENT;

вернет результат

SURNAME	NAME	BIRTHDAY		
Иванов	Иван	3/12/1988	3-де к-1988	3.12.88
Петров	Петр	11/12/1986	11-дек-1986	11.12.86
Сидоров	Вадим	7/06/1985	7-июн-1985	7.06.85

Кузнецов	Борис	8/12/1987	8-дек-1987	8.12.87
Зайцева	Ольга	21/05/1987	21-май-1987	21.05.87
Павлов	Андрей	5/11/1985	5-ноя-1985	5.11.85
Котов	Павел	NULL	NULL	NULL
Лукин	Артем	11/12/1987	11-дек1987	11.12.87
Петров	Антон	5/08/1987	5-авг-1987	5.08.87
Белкин	Вадим	20/01/1986	20-янв-1986	20.01.86

Функция CAST является средством явного преобразования данных из одного типа в другой. Синтаксис этой команды имеет вид

CAST <значимое выражение> AS <тип данных>

• <значимое выражение> должно иметь числовой или символьный тип языка SQL (возможно, с указанием длины, точности и масштаба) или быть NULL-значением.

• Любое числовое выражение может быть явно преобразовано в любой другой числовой тип.

• Символьное выражение может быть преобразовано в любой числовой тип. При этом в результате такого преобразования отсекаются начальные и конечные пробелы, а остальные символы преобразуются в числовое значение по правилам языка SQL.

• Если заданная явным образом длина символьного типа недостаточна и преобразованное значение не размещается в нем, то результативное значение усекается справа.

• Возможно явное преобразование символьного типа в символьный с другой длиной. Если длина результата больше длины аргумента, то значение дополняется пробелами; если меньше, то усекается.

• NULL преобразуется в NULL.

• Числовое выражение может быть преобразовано в символьный тип.

Пример.

```
SELECT CAST STUDENT_ID AS CHAR(10)
FROM STUDENT;
```

УПРАЖНЕНИЯ

1. Составьте запрос для таблицы STUDENT таким образом, чтобы выходная таблица содержала один столбец, содержащий последовательность разделенных символом ";" (точка с запятой) значений всех столбцов этой таблицы; при этом текстовые значения должны отображаться прописными символами (верхний регистр), т. е. быть представленными в следующем виде:

10; КУЗНЕЦОВ; БОРИС; 0; БРЯНСК; 8.12.1987; 10.

2. Составьте запрос для таблицы STUDENT таким образом, чтобы выходная таблица содержала всего один столбец в следующем виде: Б. КУЗНЕЦОВ; место жительства — БРЯНСК; родился — 8.12.87.

3. Составьте запрос для таблицы STUDENT таким образом, чтобы выходная таблица содержала всего один столбец в следующем виде: б. кузнецов; место жительства — брянск; родился: 8-дек-1987.

4. Составьте запрос для таблицы STUDENT таким образом, чтобы выходная таблица содержала всего один столбец в следующем виде: Борис Кузнецов родился в 1987 году.

5. Составьте запрос, выводящий фамилии, имена студентов и величину получаемых ими стипендий, при этом значения стипендий должны быть увеличены в 100 раз.

6. То же, что и в упр. 4, но только для студентов 1, 2 и 4 курсов и таким образом, чтобы фамилии и имена были выведены прописными буквами.

7 Составьте запрос для таблицы UNIVERSITY таким образом, что - бы выходная таблица содержала всего один столбец в следующем виде: Код-10; ВГУ-г. ВОРОНЕЖ; Рейтинг=296.

8. То же; что и в упр. 7, но значения рейтинга требуется округлить до первого знака (например, значение -382 округляется-до 400).

Агрегирование и групповые функции

Агрегирующие функции позволяют получать из таблицы сводную (агрегированную) информацию, выполняя операции над группой строк таблицы. Для задания в SELECT-запросе агрегирующих операций используются следующие ключевые слова:

COUNT определяет количество строк или значений поля, выбранных посредством запроса и не являющихся NULL-значениями;

SUM вычисляет арифметическую сумму всех выбранных значений данного поля;

AVG вычисляете среднее значение :для всех выбранных: значений данного поля.

MAX-вычисляет-наибольшее из всех выбранных значении данного поля;

MIN вычисляет наименьшее из всех выбранных значений данного поля.

В SELECT-запросе агрегирующие функции используются аналогично именам полей, при этом последние (имена полей) используются в качестве аргументов этих функций.

Функция AVG предназначена для подсчета среднего значения поля на множестве записей таблицы.

Например, для определения среднего значения поля MARK (оценки) по веем записям таблицы EXAM_MARKS можно использовать запрос с функцией AVG следующего вида:

```
SELECT AVG(MARK)
FROM EXAM_MARKS;
```

Для подсчета общего количества строк в таблице следует использовать функцию COUNT со звездочкой:

```
SELECT COUNT (*)
FROM XAM_MARKS;
```

При подсчете значений конкретных атрибутов аргументы DISTINCT и ALL позволяют соответственно исключать и включать дубликаты обрабатываемых функцией COUNT значений. При этом необходимо учитывать, что при использовании опции ALL неопределенные значения атрибута (NULL) все равно нё войдут в число подсчитываемых значений.

```
SELECT COUNT(DISTINCT SUBJ_ID)
FROM SUBJECT;
```

Предложение GROUP BY (ГРУППИРОВАТЬ ПО) позволяет группировать записи в подмножества, определяемые значениями какого либо поля, и применять агрегирующие функции уже не ко всем записям таблицы, а раздельно к каждой сформированной группе.

Предположим, требуется найти максимальное значение оценки, полученной каждым студентом. Запрос будет выглядеть следующим образом:

```
SELECT STUDENT_ID, MAX(MARK)
FROM EXAM_MARKS
GROUP BY  STUDENT_ID;
```

Выбираемые из таблицы EXAM_MARKS записи группируются по значениям поля STUDENT_ID указанного в предложении GROUP BY, и для каждой группы находится максимальное значение поля MARK. Предложение GROUP BY позволяет применять агрегирующие функции к каждой группе, определяемой общим значением поля или полей, указанных в этом предложении. В приведенном запросе рассматриваются группы записей, сгруппированные по идентификаторам студентов.

В конструкции GROUP BY для группирования может быть использовано более одного столбца. Например:

```
SELECT STUDENT_ID, SUBJ_ID, MAX(MARK)
```

FROM EXAM_MARKS
GROUP BY STUDENT_ID, SUBJ_ID;

В этом случае строки вначале группируются по значениям первого столбца, а внутри этих групп — в подгруппы по значениям второго столбца. Таким образом, GROUP BY не только устанавливает столбцы по которым осуществляется группировка но и указывает. порядок разбиения столбцов на группы.

Следует иметь в виду, что после ключевого слова SELECT должны быть использованы. только те имена столбцов, которые указаны в предложении GROUP BY.

При необходимости часть сформированных с помощью GROUP BY групп может быть исключена с помощью предложения HAVING.

Предложение HAVING определяет критерий, по которому группы следует включать в выходные-данные (по аналогии с предложением WHERE, которое осуществляет это для отдельных строк).

```
SELECT SUBJ_NAME, MAX(HOUR)
FROM SUBJECT
GROUP BY SUBJ_NAME
HAVING MAX(HOUR) > 72;
```

В условии, задаваемом предложением HAVING, должны быть указаны только поля или выражения, которые на выходе имеют единственное значение для каждой выводимой группы.

УПРАЖНЕНИЯ

1. Напишите запрос для подсчета количества студентов, сдававших экзамен по предмету обучения с идентификатором 20.
2. Напишите запрос, который позволяет подсчитать в таблице EXAM_MARKS количество различных предметов обучения.
3. Напишите запрос, который для каждого студента выполняет выборку его идентификатора и минимальной из полученных им оценок.
4. Напишите запрос, который для каждого студента выполняёт выборку его идентификатора и максимальной из полученных им оценок.
5. Напишите запрос, выполняющий вывод первой по алфавиту фамилии студента, начинающейся на букву 'И'.
6. Напишите запрос, который для каждого предмета обучения выводит наименование предмета и максимальное значение номера семестра, в котором этот предмет преподается.
7. Напишите запрос, который для каждого конкретного дня сдачи экзамена выводит данные о количестве студентов, сдававших экзамен в этот день.
8. Напишите запрос, выдающий средний балл для Каждого студента.
9. Напишите запрос, выдающий средний балл для каждого экзамена.
10. Напишите запрос, определяющий количество сдававших студентов для каждого экзамена.
11. Напишите запрос для определения количества предметов, изучаемых на каждом курсе.
12. Для каждого университета напишите запрос выводящий суммарную стипендию обучающихся в нем студентов, с последующей сортировкой списка по этому значению.
13. Для каждого семестра напишите запрос, выводящий общее количество часов, отводимое на изучение соответствующих предметов.
14. Для каждого студента напишите запрос выводящий среднее значение оценок, полученных им на всех экзаменах.
15. Для каждого студента напишите запрос, выводящий среднее значение оценок полученных им по каждому предмету.

16. Напишите запрос, выводящий количество студентов, проживающих в каждом городе. Список отсортировать в порядке убывания количества студентов.

количества студентов

17. Для каждого университета напишите запрос, выводящий количество обучающихся в нем студентов, с последующей сортировкой списка по этому количеству.

18. Для каждого университета напишите запрос, выводящий количество работающих в нем преподавателей, с последующей сортировкой списка по этому количеству.

19. Для каждого университета напишите запрос, выводящий сумму стипендии, выплачиваемой студентам каждого курса.

20. Для каждого города напишите запрос, выводящий максимальный рейтинг университетов, в нем расположенных, с последующей сортировкой списка по значениям рейтингов.

21. Для каждого дня сдачи экзаменов напишите запрос, выводящий среднее значение всех экзаменационных оценок.

22. Для каждого дня сдачи экзаменов напишите запрос, выводящий максимальные оценки, полученные по каждому предмету.

23. Для каждого дня сдачи экзаменов напишите запрос, выводящий общее количество студентов, сдававших экзамены.

24. Для каждого дня сдачи экзаменов напишите запрос, выводящий общее количество экзаменов, сдававшихся каждым студентом.

25. Для каждого преподавателя напишите запрос, выводящий количество преподаваемых им предметов.

26. Для каждого предмета напишите запрос, выводящий количество преподавателей, ведущих по нему занятия

27. Напишите запрос, выполняющий вывод количества студентов, имеющих только отличные оценки.

28. Напишите запрос, выполняющий вывод количества экзаменов; сданных (с положительной оценкой) студентом с идентификатор ром 32.