б.кузнецов; место жительства-брянск; родился: 8-дек-1981.

4. Составьте запрос для таблицы STUDENT таким образом, чтобы выходная таблица содержала всего один столбец в следующем виде:
   Борис Кузнецов родился в 1981 году.
5. Вывести фамилии, имена студентов и величину получаемых ими стипендий, при этом значения стипендий должны быть увеличены в 100 раз.
6. То же, что и в задаче 4, но только для студентов 1, 2 и 4-го курсов и таким образом, чтобы фамилии и имена были выведены прописными буквами.
7. Составьте запрос для таблицы UNIVERSITY таким образом, чтобы выходная таблица содержала всего один столбец в следующем виде:
   Код-10; ВГУ-г.ВОРОНЕЖ; Рейтинг=296.
8. Тоже, что и в задаче 7, но значения рейтинга требуется округлить до первого знака (например, значение 382 округляется до 400).

2.4. 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** вычисляет среднее значение для всех выбранных значений данного поля;
- **MAX** вычисляет наибольшее из всех выбранных значений 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 всем 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 EXAM_MARKS;
```

Аргументы **DISTINCT** и **ALL** позволяют, соответственно, исключать и включать дубликаты обрабатываемых функцией **COUNT** значений, при этом необходимо учитывать, что при использовании опции **ALL** значения **NULL** все равно не войдут в число подсчитываемых значений.

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

Предложение **GROUP BYGROUP 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 группирование, но и указывает порядок разбиения столбцов на группы.

Следует иметь в виду, что в предложении **GROUP BY** должны быть указаны все выбираемые столбцы, приведенные после ключевого слова **SELECT**, кроме столбцов, указанных в качестве аргумента в агрегирующей функции.

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**, указывают только поля или выражения, которые на выходе имеют единственное значение для каждой выводимой группы.

УПРАЖНЕНИЯ

9. Напишите запрос для подсчета количества студентов, сдававших экзамен по предмету обучения с идентификатором, равным 20.
10. Напишите запрос, который позволяет подсчитать в таблице EXAM_MARKS количество различных предметов обучения.
11. Напишите запрос, который выполняет выборку для каждого студента

значения его идентификатора и минимальной из полученных им оценок.

12. Напишите запрос, который выполняет выборку для каждого студента значения его идентификатора и максимальной из полученных им оценок.
13. Напишите запрос, выполняющий вывод фамилии первого в алфавитном порядке (по фамилии) студента, фамилия которого начинается на букву “И”.
14. Напишите запрос, который выполняет вывод для каждого предмета обучения наименование предмета и максимальное значение номера семестра, в котором этот предмет преподается.
15. Напишите запрос, который выполняет вывод данных для каждого конкретного дня сдачи экзамена о количестве студентов, сдававших экзамен в этот день.
16. Напишите запрос для получения среднего балла для каждого курса по каждому предмету.
17. Напишите запрос для получения среднего балла для каждого студента.
18. Напишите запрос для получения среднего балла для каждого экзамена.
19. Напишите запрос для определения количества студентов, сдававших каждый экзамен.
20. Напишите запрос для определения количества изучаемых предметов на каждом курсе.

2.5. Пустые значения (NULL) в агрегирующих функциях

Наличие пустых (**NULL**) значений в полях таблицы накладывает особенности на выполнение агрегирующих операций над данными, которые следует учитывать при их использовании в SQL-запросах.

2.5.1. Влияние NULL–значений в функции COUNT

Если аргумент функции **COUNT** является константой или столбцом без пустых значений, то функция возвращает количество строк, к которым применимо определенное условие или группирование.