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.

Если аргументом функции является *столбец*, содержащий пустое значение, то **COUNT** вернет число строк, не содержащих пустые значения, и к которым применимо определенное условие или группирование.

Если бы механизм **NULL** не был доступен, то неприменимые и отсутствующие значения пришлось бы исключать с помощью конструкции **WHERE**.

Поведение функции **COUNT**(*) не зависит от пустых значений. Она возвратит общее количество строк в таблице.

2.5.2. Влияние NULL-значений в функции AVG

Среднее значение множества чисел равно сумме чисел, деленной на число элементов множества. Однако, если некоторые элементы пусты, то есть их значения неизвестны или не существуют, то деление на количество всех элементов множества приведет к неправильному результату.

Функция **AVG** вычисляет среднее значение всех *известных* значений множества элементов, то есть эта функция подсчитывает сумму *известных* значений и делит ее на количество *этих* значений, а не на общее количество значений, среди которых могут быть **NULL**-значения. Если столбец состоит только из пустых значений, то функция **AVG** также возвратит **NULL**.

2.6. Результат действия трехзначных условных операторов

Условные операторы при отсутствии пустых значений возвращают либо **TRUE** (истина), либо **FALSE** (ложь). Если же в столбце присутствуют пустые значения, то может быть возвращено и третье значение: **UNKNOWN** (неизвестно). В этой схеме, например, условие **WHERE** A=2, где A – имя столбца, значения которого могут быть неизвестны, при A=2 будет соответствовать **TRUE**, при A=4 в результате будет получено значение **FALSE**, а при отсутствующем значении A (**NULL**-значение) результат будет **UNKNOWN**. Пустые значения оказывают влияние на использование логических операторов **NOT**, **AND** и **OR**.

Оператор NOT

Обычный унарный оператор **NOT** обращает оценку **TRUE** в **FALSE** и наоборот. Однако **NOT NULL** по прежнему будет возвращать пустое значение **NULL**. При этом следует отличать случай **NOT NULL** от условия **IS NOT NULL**, которое является противоположностью **IS NULL**, отделяя известные значения от неизвестных.

Оператор AND

- Если результат двух условий, объединенных оператором **AND**, известен, то применяются правила булевой логики, то есть при обоих утверждениях **TRUE** составное утверждение также будет **TRUE**. Если же хотя бы одно из двух утверждений будет **FALSE**, то составное утверждение будет **FALSE**.
- Если результат одного из утверждений неизвестен, а другой оценивается как **TRUE**, то состояние неизвестного утверждения является определяющим, и, следовательно, итоговый результат также неизвестен.
- Если результат одного из утверждений неизвестен, а другой оценивается как **FALSE**, итоговый результат будет **FALSE**.
- Если результат обоих утверждений неизвестен, то результат также остается неизвестным.

Оператор OR

- Если результат двух условий, объединенных оператором **OR**, известен, то применяются правила булевой логики, а именно: если хотя бы одно из двух утверждений соответствует **TRUE**, то и составное утверждение будет **TRUE**, если оба утверждения оцениваются как **FALSE**, то составное утверждение будет **FALSE**.
- Если результат одного из утверждений неизвестен, а другой оценивается как **TRUE**, итоговый результат будет **TRUE**.
- Если результат одного из утверждений неизвестен, а другой оценивается как **FALSE**, то состояние неизвестного утверждения играет роль. Следовательно, итоговый результат также неизвестен.
- Если результат обоих утверждений неизвестен, то результат также остается неизвестным.

Примечание.

Отсутствующие (**NULL**) значения целесообразно использовать в столбцах, предназначенных для агрегирования, чтобы извлечь преимущества из способа обработки пустых значений в функциях **COUNT** и **AVG**. Практически во всех остальных случаях пустых значений следует избегать, так как при их наличии существенно усложняется корректное построение условий отбора, приводя иногда к непредсказуемым результатам выборки. Для индикации же отсутствующих, неприменимых или по какой-то причине неизвестных данных можно использовать значения по умолчанию, устанавливаемые заранее (например, с помощью команды **CREATE TABLE** (раздел 4.1).

2.7. Упорядочение выходных полей (ORDER BY)

Как уже отмечалось, записи в таблицах реляционной базы данных неупорядочены. Однако, данные, выводимые в результате выполнения запроса, могут быть упорядочены. Для этого используется оператор **ORDER BY**, который позволяет упорядочивать выводимые записи в соответствии со значениями одного или нескольких выбранных столбцов. При этом можно задать возрастающую (**ASC**) или убывающую (**DESC**) последовательность сортировки для каждого из столбцов. По умолчанию принята возрастающая последовательность сортировки.

Запрос, позволяющий выбрать все данные из таблицы предметов обучения SUBJECT, с упорядочиванием по наименованиям предметов, выглядит следующим образом:

```
SELECT *
    FROM SUBJECT
    ORDER BY SUBJ_NAME;
```

Тот же список, но упорядоченный в обратном порядке, можно получить запросом:

```
SELECT *
    FROM SUBJECT
    ORDER BY SUBJ_NAME DESC;
```

Можно упорядочить выводимый список предметов обучения по значениям семестров, а внутри семестров – по наименованиям предметов.

```
SELECT *
    FROM SUBJECT
    ORDER BY SEMESTR, SUBJ_NAME;
```

Предложение **ORDER BY** может использоваться с **GROUP BY** для упорядочивания групп записей. При этом оператор **ORDER BY** в запросе *всегда должен быть* ***последним***.

```
SELECT SUBJ_NAME, SEMESTR, MAX(HOUR)
    FROM SUBJECT
    GROUP BY SEMESTR, SUBJ_NAME
    ORDER BY SEMESTR;
```

При упорядочивании вместо наименований столбцов можно указывать их номера, имея, однако, в виду, что в данном случае это – номера столбцов, указанные при определении выходных данных в запросе, а не номера столбцов в таблице. Полем с номером 1 является первое поле, указанное в предложении **ORDER BY** – независимо от его расположения в таблице.

```
SELECT SUBJ_ID, SEMESTR
    FROM SUBJECT
    ORDER BY 2 DESC;
```

В этом запросе выводимые записи будут упорядочены по полю SEMESTR.

Если в поле, которое используется для упорядочивания, существуют **NULL-**значения, то все они размещаются в конце или предшествуют всем остальным значениям этого поля.

УПРАЖНЕНИЯ

21. Предположим, что стипендия всем студентам увеличена на 20%. Напишите запрос к таблице STUDENT, выполняющий вывод номера студента, фамилию студента и величину увеличенной стипендии. Выходные данные упорядочить: а) по значению последнего столбца (величине стипендии); б) в алфавитном порядке фамилий студентов.

22. Напишите запрос, который по таблице EXAM_MARKS позволяет найти а) максимальные и б) минимальные оценки каждого студента и выводит их вместе с идентификатором студента.

23. Напишите запрос, выполняющий вывод списка предметов обучения в

порядке а) убывания семестров и б) возрастания отводимых на предмет часов. Поле семестра в выходных данных должно быть первым, за ним должны следовать имя предмета обучения и идентификатор предмета.

24. Напишите запрос, который выполняет вывод суммы баллов всех студентов для каждой даты сдачи экзаменов и представляет результаты в порядке убывания этих сумм.

25. Напишите запрос, который выполняет вывод а) среднего, б) минимального, в) максимального баллов всех студентов для каждой даты сдачи экзаменов, и представляет результаты в порядке убывания этих значений.

2.8. Вложенные подзапросы

SQL позволяет использовать одни запросы внутри других запросов, то есть вкладывать запросы друг в друга. Предположим, известна фамилия студента (“Петров”), но неизвестно значение поля STUDENT_ID для него. Чтобы извлечь данные обо всех оценках этого студента, можно записать следующий запрос:

```
SELECT *
    FROM EXAM_MARKS
    WHERE STUDENT_ID =
        ( SELECT STUDENT_ID
          FROM STUDENT  SURNAME = ‘Петров’);
```

Как работает запрос SQL со связанным подзапросом?

- Выбирается строка из таблицы, имя которой указано во внешнем запросе.
- Выполняется подзапрос и полученное в результате его выполнения значение применяется для анализа этой строки в условии предложения **WHERE** внешнего запроса.
- По результату оценки этого условия принимается решение о включении или не включении строки в состав выходных данных.
- Процедура повторяется для следующей строки таблицы внешнего запроса.