Лабораторная работа №4

Вложенные подзапросы

SQL позволяет использовать одни запросы внутри других запросов, т.е. 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');
```

Следует обратить внимание, что этот корректен только в том случае, если в результате выполнения указанного в скобках *подзапроса* возвращается *единственное значение.* Если в результате выполнения подзапроса будет возвращено несколько значений, то при выполнении запроса будет зафиксирована ошибка. В данном примере это произойдет, если в таблице STUDENT будет несколько записей со значениями поля SURNAME = 'Петров'.

В некоторых случаях для гарантии получения единственного значения в результате выполнения подзапроса используется DISTINCT. Одним из видов функций, которые автоматически *всегда* выдают в результате единственное значение для любого количества строк, являются агрегирующие функции.

Оператор IN также широко применяется в подзапросах. Он задаёт список значений, с которыми сравниваются другие значения для определения истинности задаваемого этим оператором предиката.

Данные обо всех оценках (таблица EXAM_MARKS) студентов из Воронежа можно выбрать с помощью следующего запроса:

```
SELECT *
FROM EXAM_MARKS
WHERE STUDENT_ID IN
(SELECT STUDENT_ID
```

```
FROM STUDENT
WHERE CITY=’Воронеж’);
```

Подзапросы можно применять внутри предложения HAVING. Пусть требуется определить количество предметов обучения с оценкой, превышающей среднее значение оценки студента с идентификатором 301:

```
SELECT COUNT(DISTINCT SUBJ_ID), MARK
FROM EXAM_MARKS
GROUP BY MARK
HAVING MARK >
(SELECT AVG(MARK)
 FROM EXAM_MARKS
WHERE STUDENT_ID =301);
```

УПРАЖНЕНИЕ 1

1. Напишите запрос, выводящий список студентов, получающих максимальную стипендию, отсортировав его в алфавитном порядке по фамилиям.

2. Напишите запрос, выводящий список студентов, получающих стипендию, превышающую среднее значение стипендии.

3. Напишите запрос, выводящий список, студентов, обучающихся в Воронеже, с последующей сортировкой по идентификаторам университетов и курсам.

4. Напишите запрос, выводящий список предметов, на изучение которых отведено максимальное количество часов.

5. Напишите запрос, выполняющий вывод имен и фамилий студентов, место проживания которых не совпадает с городом, в котором находится их университет.

6. Напишите запрос, выводящий список университетов, расположенных в Москве и имеющих рейтинг меньший, чем у ВГУ.

Формирование связанных подзапросов

При использовании подзапросов во внутреннем запросе можно ссылаться на таблицу, имя которой указано в предложении FROM внешнего запроса. Такие подзапросы называются связанными.

Связанный подзапрос выполняется по одному разу для каждой строки таблицы основного запроса, а именно:

• 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.

Пример.

Выбрать сведения обо всех предметах обучения, по которым проводится экзамен 20 января 2012 г.

```
SELECT *
  FROM SUBJECT SU
  WHERE '20/01/2012' IN
        (SELECT EXAM_DATE
          FROM EXAM_MARKS EX
          WHERE SU.SUBJ_ID = EX.SUBJ_ID);
```

В некоторых СУБД для выполнения этого запроса, возможно, потребуется преобразование значения даты в символьный тип. В приведенном запросе SU и EX являются псевдонимами (алиасами), т.е. специально вводимыми именами, которые могут быть использованы в данном запросе вместо настоящих имен. В приведенном примере они используются вместо имен таблиц SUBJECT и EXAM_MARKS.

Эту же задачу можно решить с помощью операции соединения таблиц:

```
SELECT DISTINCT FIRST.SUBJ_ID, SUBJ_NAME,
                                HOUR SEMESTER
FROM SUBJECT FIRST EXAM_MARKS SECOND
WHERE FIRST.SUBJ_ID = SECOND.SUBJ_ID
AND SECOND.EXAM_DATE = '20/01/2012';
```

В этом выражении алиасами таблиц являются имена FIRST и SECOND;

Можно использовать подзапросы, связывающие таблицу со своей собственной копией. Например, надо найти идентификаторы, фамилии и стипендии студентов, получакиих стипендию выше средней на курсе, на котором они учатся:

```
SELECT DISTINCT STUDENT_ID, SURNAME, STIPEND
FROM STUDENT E1
WHERE STIPEND >
(SELECT AVG (STIPEND)
FROM STUDENT E2
WHERE EI.KURS =E2.KURS);
```

Тот же результат можно получить с помощью следующего запроса:

```
SELECT DISTINCT STUDENT_ID, SURNAME, STIPEND
FROM STUDENT E1,
(SELECT KURS, AVG(STIPEND) AS AVG_STIPEND
  FROM STUDENT E2
GROUP BY E2.KURS) E3
WHERE E1.STIPEND > AVG_STIPEND AND
                                        E1.KURS=E3.KURS;
```

Обратите внимание — второй запрос должен выполниться гораздо быстрее. Дело в том, что в первом варианте запроса агрегирующая функция AVG выполняется над таблицей, указанной в подзапросе, для каждой строки внешнего запроса. В другом варианте вторая таблица (алиас E2) обрабатывается агрегирующей функцией один раз, в результате чего формируется

вспомогательная таблица (в запросе она имеет алиас E3), со строками которой затем соединяются строки первой таблицы (алиас E1). Следует иметь в виду, что на самом деле реальное время выполнения запроса в большой степени зависит от оптимизатора запросов конкретной СУБД, и, вполне возможно, что такое преобразование запроса будет выполнено оптимизатором.

УПРАЖНЕНИЕ 2

1. Напишите запрос для получения списка студентов, которые учатся в своем городе.

2. Напишите запрос для получения списка иногородних студентов (обучающихся не в своем городе), с последующей сортировкой по идентификаторам университетов и курсам.

3. Напишите запрос для получения списка преподавателей, работающих не в своем городе, с последующей сортировкой по идентификаторам университетов и городам проживания преподавателей.

4. Напишите запрос для получения списка предметов, на изучение которых отведено максимальное количество часов среди всех предметов, изучаемых в том же семестре. Список упорядочить по семестрам.

5. Напишите запрос для получения списка студентов, получающих стипендию, превосходящую среднее значение стипендии на их курсе.

6. Напишите запрос для получения списка студентов, получающих минимальную стипендию в своем университете, с последующей сортировкой по значениям идентификатора университета и стипендии.

7. Напишите запрос для получения списка университетов, в которых учится более 50 студентов, с последующей сортировкой по рейтингам.

8. Напишите запрос для. получения списка университетов, в которых работает более 5 преподавателей, с последующей сортировкой по рейтингам университетов.

Связанные подзапросы в HAVING

Ранее указывалось, что предложение GROUP BY позволяет группировать выводимые SELECT-запросом записи по значению некоторого поля. Использование предложения HAVING позволяет при выводе осуществлять фильтрацию таких групп. Предикат предложения HAVING оценивается не для каждой строки результата, а для каждой группы выходных записей, сформированной предложением GROUP BY внешнего запроса.

Пусть, например, необходимо по данным из таблицы EXAM MARKS определить сумму полученных студентами оценок (значений поля MARK), сгруппировав значения оценок по датам экзаменов и исключив те дни, когда число студентов, сдававших в течение дня экзамены, было меньше 10:

```
SELECT EXAM_DATE, SUM(MARK)
FROM EXAM MARKS A
GROUP BY EXAM_DATE
HAVING 10 <
(SELECT COUNT(MARK)
FROM EXAM_MARKS B
WHERE A.EXAM_DATE B EXAM_DATE);
```

Подзапрос вычисляет количество строк, у которых значения поля EXAM_DATE (дата экзамена) совпадает с датой, для которой сформирована очередная группа основного запроса.

УПРАЖНЕНИЕ 3

1. Напишите запрос с подзапросом для получения всех оценок студента с фамилией Иванов. Предположим, что его персональный номер неизвестен. Всегда ли такой запрос будет корректным?

2. Напишите запрос, выбирающий имена всех студентов, имеющих по предмету с идентификатором 101 балл выше общего среднего балла.

3. Напишите запрос, который выполняет выборку имен всех студентов, имеющих по предмету с идентификатором 102 балл ниже общего среднего балла.

4. Напишите запрос, выполняющий вывод количества предметов, по которым экзаменовался каждый студент, сдававший более 20 предметов.

5. Напишите команду SELECT, использующую связанные подзапросы и выполняющую вывод имен и идентификаторов студентов, у которых стипендия совпадает с максимальным значением стипендии для города, в котором живет студент.

6.-Напишите запрос, который позволяет вывести имена и идентификаторы всех студентов, о которых точно известно, что они проживают в городе, где нет ни одного университета.

7. Напишите два запроса, которые позволяют .вывести имена и идентификаторы всех студентов, о которых точно известно, что они проживают не в том городе, где расположен их университет: один запрос с использованием связанного подзапроса, а другой - с использованием соединения.

Соединение таблиц. Оператор JOIN

Если в операторе SELECT после ключевого слова FROM указывается не одна, а две таблицы, то в результате выполнения запроса, в котором отсутствует предложение WHERE, каждая строка одной таблицы будет соединена с каждой строкой второй таблицы. Такая опёрация называется декартовым произведением, или полным соединением таблиц базы данных. Сама по себе эта операция не имеет практического значения, более того, при ошибочном использовании. она может привести к неожиданным нештатным ситуациям, так как в этом случае в ответе на запрос количество записей будет равно произведению числа записей в соединяемых таблицах, т. е. может оказаться чрезвычайно большим. Соединение таблиц имеет смысл тогда, когда соединяются не все строки исходных таблиц, а только те, которые интересуют пользователя. Такое ограничение может быть осуществлено с помощью использования в запросе соответствующего условия в предложении WHERE. Таким образом, SQL позволяет выводить информацию из нескольких таблиц, связывая их по значениям определенных полей.

Например, если необходимо получить фамилии студентов (таблица STUDENT) и для каждого студента - названия университетов (таблица UNIVERSITY), расположенных в городе, где живет студент, то необходимо получить все комбинации записей о студентах и университетах в обеих таблицах, в которых значение поля CITY совпадает. Это можно сделать с помощью следующего запроса:

```
SELECT STUDENT. SURNAME, UNIVERSITY.UNIV_NAME,
STUDENT.CITY
FROM STUDENT, UNIVERSITY
WHERE STUDENT.CITY = UNIVERSITY.CУTY;
```

Соединение, использующее предикаты, основанные на равенствах, называется эквисоединением. Рассмотренный пример соединения таблиц относятся к виду так называемого внутреннего (INNER) соединения. При таком типе соединения соединяются только те строки таблиц, для которых является истинным предикат, задаваемый в предложении ON выполняемого запроса.

Приведенный выше запрос может бить записаниначе, с использованием ключевого--слова JOIN:

```
SELECT STUDENT.SURNAME, UNIVERSITYUNIV_NAME,
STUDENT.CITY
FROM STUDENT INNER JOIN UNIVERSITY
ON STUDENT.CITY = UNIVERSITY.CITY;
```

Ключевое слово INNER в запросе может быть опущено, так как эта опция в операторе JOIN действует по умолчанию.

-

УПРАЖНЕНИЕ 4

1. Напишите зарос для получения списка предметов -вместе-с фамилиями студентов, изучающих их на соответствующем курсе.

2. Напишите запрос, выполняющий вывод имел и фамилий студентов, имеющих весь набор положительных (тройки, четверки и пятерки) оценок.

Операции -соединения таблиц посредством ссылочной целостности.

Информация в таблицах STUDENT и EXAM_MARKS уже связана посредствам поля STUDENT_ID. В таблице STUDENT поле STUDENT_ID является первичным ключом, а в таблице EXAM_MARKS — ссылающимся на него внешним ключом. Состояние связанных таким образом таблиц называется состоянием ссылочной целостности. В данном случае ссылочная .целостность этих таблиц подразумевает, что каждому значению поля STUDENT_ID в таблице EXAM_MARKS обязательно соответствует какое же значение поля STUDENT_ID в таблице STUDENT. Другими словами, в таблице EXAM_MARKS не может быть записей, имеющих идентификаторы студентов, которых нет в таблице STUDENT. Стандартное применение операции соединения состоит в извлечении данных в терминах этой связи.

Чтобы получить список фамилий студентов с полученными ими оценками и идентификаторами предметов можно. использовать следующий запрос:

```
SELECT SURNAME, MARK, SUBJ_ID
FROM STUDENT, EXAM_MARKS
WHERE STUDENT.STUDENT_ID =
EXAM_MARKS.STUDENT_ID;
```

Тот же самый результат может быть получен при использовании в запросе для задания операции соединения таблиц ключевого слова JOIN. Запрос с оператором JOIN выглядит следующим образом:

```
SELECT SURNAME, MARK
FROM STUDENT JOIN EXAM_MARKS
ON STUDENT.STUDENT_ID EXAM_MARKS.STUDENT_ID;
```

Для такого рода запросов, когда соединение таблиц осуществляется по одноименным столбцам, можно использовать так называемое естественное соединение, .задаваемое ключевым словом NATURAL. В этом случае в запросе не указывается предложение ON условия отбора записей. Приведенный выше запрос будет выглядеть: следующим образов

```
SELECT SURNAME, MARK
```

```
FROM STUDENT NATURAL JOIN EXAM_MARKS;
```

Хотя выше речь шла о соединении двух таблиц, можно сформировать запросы путем соединения более чем двух таблиц.

Пусть требуется найти фамилии всех студентов, получивших неудовлетворительную оценку, вместе с названиями предметов обучения, по которым получена эта оценка.

```
SELECT SUBJ_NAME, SURNAME, MARK
FROM STUDENT, SUBJECT, EXAM_MARKS
WHERE STUDENT.STUDENT_ID = EXAM_MARKS.STUDENT_ID
AND SUBJECT.SUBJ_ID = EXAM_MARKS.SUBJ_ID
AND EXAM_MARKS.MARK = 2;
```

То же самое с использованием оператора JOIN:

```
SELECT SUBJ_NAME, SURNAME, MARK
FROM STUDENT JOIN EXAM_MARKS
ON STUDENT.STUDENT_ID = EXAM_MARKS.STUDENT_ID
JOIN SUBJECT
ON SUBJECT.SUBJ_ID =EXAM_MARKS.SUBJ_ID
WHERE MARK = 2;
```

УПРАЖНЕНИЕ 5

1:. Напишите запрос для получения списка университетов с указанием количества студентов, обучающихся на каждом курсе.

2. Напишите запрос для получения списка преподавателей с указанием их учебных предметов.

3. Напишите запрос для получения списка преподавателей с.указанием нагрузки (суммарного количества часов) в каждом семестре.

4. Напишите запрос для получения списка университетов вместе с названиями преподаваемых в них предметов.

5. Напишите запрос для получения списка университетов с указанием суммарного количества аудиторных часов в каждом семестре.

6. Напишите запрос для получения списка университетов с указанием суммарного количества часов, отводимых на изучение каждого предмета.

7. Напишите запрос для получения списка преподавателей с указанием суммарного количества часов, отведенных для обучения каждому из предметов.

8. Напишите запрос для сортировки списка университетов по значениям максимальной стипендии, выплачиваемой студентам.

9. Напишите запрос для получения списка университетов вместе с фамилиями самых молодых студентов, обучаемых в них.

10. Напишите запрос для получения списка университетов вместе с фамилиями студентов, получающих максимальную для каждого университета стипендию.

Ответы:упражнение 1 (2.8)

1. SELECT *
   FROM STUDENT
   WHERE STIPEND =
   (SELECT MAX(STIPEND) FROM STUDENT)
   ORDER BY SURNAME;
2. SELECT *
   FROM STUDENT
   WHERE STIPEND>
   (SELECT AVG(STIPEND FROM STUDENT);
3. SELECT *
   FROM STUDENT
   WHERE UNIV_ID IN
   (SELECT UNIV_ID FROM UNIVERSITY
   WHERE CITY= 'Воронеж')
   ORDER BY UNIV_ID, KURS;
4. SELECT *
   FROM SUBJECT
   WHERE HOUR=(SELECT MAX(HOUR) FROM SUBJECT);
5. SELECT SURNAME, NAME
   FROM STUDENT S
   WHERE CITY <> (SELECT CITY FROM UNIVERSITY
   WHERE UNIV_ID =S.UNIV_ID);

6. SELECT * FROM UNIVERSITY
WHERE CITY=’Москва’
AND RATING < (SELECT RATING FROM UNIVERSITY
WHERE UNIV_NAME= 'ВГУ');

Упражнение 2

```
1.  SELECT * FROM STUDENT S
     WHERE CITY=(SELECT CITY FROM UNIVERSITY
                    WHERE UNIV_ID=S.UNIV_ID);
2.  SELECT * FROM STUDENT S
     WHERE CITY<>(SELECT CITY FROM UNIVERSITY
                    WHERE UNIV_ID=S.UNIV_ID)
     ORDER BY UNIV_ID, KURS;
3. SELECT * FROM LECTURER L
     WHERE CITY<>(SELECT CITY FROM UNIVERSITY
                    WHERE UNIV_ID=L.UNIV_ID)
     ORDER BY UNIV_ID, CITY;
4. SELECT * FROM SUBJECT S
     WHERE HOUR = (SELECT MAX(HOUR) FROM SUBJECT
                    WHERE SEMESTER=S.SEMESTER)
     ORDER BY SEMESTER;
5. SELECT * FROM STUDENT S
     WHERE STIPEND >
        (SELECT AVG(STIPEND) FROM STUDENT
         WHERE KURS=S.KURS);
6. SELECT * FROM STUDENT S
     WHERE STIPEND =
       (SELECT MIN(STIPEND) FROM STUDENT
        WHERE UNIV_ID=S.UNIV_ID)
     ORDER BY UNIV_ID, STIPEND;

7. SELECT * FROM UNIVERSITY U
     WHERE 50<(SELECT COUNT(*) FROM STUDENT
                    WHERE UNIV_ID=U.UNIV_ID)
     ORDER BY RATING;
8. SELECT * FROM UNIVERSITY U
     WHERE 5<(SELECT COUNT(*) FROM LECTURER
                    WHERE UNIV_ID=U.UNIV_ID)
     ORDER BY RATING;
```

Упражнение 3

1.
```
SELECT *
   FROM EXAM_MARKS
   WHERE STUDENT_ID =
      (SELECT STUDENT_ID FROM STUDENT
       WHERE SURNAME = 'Иванов');
```

2.
```
SELECT DISTINCT NAME
   FROM STUDENT, EXAM_MARKS
   WHERE MARK > (SELECT AVG(MARK) FROM EXAM_MARKS)
   AND STUDENT.STUDENT_ID = EXAM_MARKS.STUDENT_ID
   AND SUBJ_ID = 101;
```

3.
```
SELECT DISTINCT NAME
   FROM STUDENT, EXAM_MARKS
   WHERE MARK < (SELECT AVG(MARK) FROM EXAM_MARKS)
   AND STUDENT.STUDENT_ID = EXAM_MARKS.STUDENT_ID
   AND SUBJ_ID = 102;
```

4.
```
SELECT COUNT(SUBJ_ID)
   FROM EXAM_MARKS
   GROUP BY STUDENT_ID
   HAVING COUNT(SUBJ_ID) > 20;
```

5.
```
SELECT NAME, STUDENT_ID
   FROM STUDENT A
   WHERE STIPEND = (SELECT MAX(STIPEND)
                    FROM STUDENT B
                    WHERE A.CITY = B.CITY);
```

6.
```
SELECT NAME, STUDENT_ID
   FROM STUDENT A
   WHERE A.CITY IS NOT NULL
   AND A.CITY NOT IN
      (SELECT B.CITY FROM UNIVERSITY B);
```

7.
```
SELECT NAME, STUDENT_ID
   FROM STUDENT A
   WHERE A.CITY IS NOT NULL
     AND A.UNIV_ID IS NOT NULL
     AND A.CITY NOT IN
        (SELECT B.CITY
         FROM UNIVERSITY B
         WHERE A.UNIV_ID = B.UNIV_ID);
```

```
SELECT DISTINCT NAME, A.UNIV_ID
   FROM STUDENT A, UNIVERSITY B
   WHERE A.CITY <> B.CITY
     AND A.CITY IS NOT NULL
     AND A.UNIV_ID = B.UNIV_ID;
```

Упражнение 4

1.
```
SELECT SUBJ_NAME, SURNAME, NAME, KURS, SEMESTER
   FROM SUBJECT, STUDENT
   WHERE SEMESTER=KURS*2-1 OR SEMESTER=KURS*2;
```
2.
```
SELECT DISTINCT SURNAME, NAME
   FROM STUDENT S, EXAM_MARKS E
   WHERE S.STUDENT_ID=E.STUDENT_ID AND MARK>2;
```

То же с использованием несвязанного вложенного подзапроса с **IN**:

```
SELECT SURNAME, NAME FROM STUDENT
   WHERE STUDENT_ID IN (SELECT STUDENT_ID
                        FROM EXAM_MARKS
                        WHERE MARK>2);
```

Упражнение 5

```
1. SELECT UNIV_NAME, KURS, COUNT (STUDENT_ID)
     FROM STUDENT S, UNIVERSITY U
     WHERE S.UNIV_ID=U.UNIV_ID
     GROUP BY UNIV_NAME, KURS;
2. SELECT SUBJ_NAME, SURNAME, NAME
     FROM LECTURER L, SUBJ_LECT SL, SUBJECT SB
     WHERE L.LECTURER_ID=SL.LECTURER_ID
       AND SL.SUBJ_ID=SB.SUBJ_ID;
3. SELECT SURNAME, NAME, SEMESTER, SUM(HOUR)
     FROM LECTURER L, SUBJ_LECT SL, SUBJECT SB
     WHERE L.LECTURER_ID=SL.LECTURER_ID
       AND SL.SUBJ_ID=SB.SUBJ_ID
     GROUP BY SURNAME, NAME, SEMESTER;
4. SELECT UNIV_NAME, SUBJ_NAME
     FROM UNIVERSITY U, LECTURER L, SUBJ_LECT SL,
          SUBJECT SB
     WHERE U.UNIV_ID=L.UNIV_ID
       AND L.LECTURER_ID=SL.LECTURER_ID
       AND SL.SUBJ_ID=SB.SUBJ_ID;
5. SELECT UNIV_NAME, SEMESTER, SUM(HOUR)
     FROM UNIVERSITY U, LECTURER L, SUBJ_LECT SL,
          SUBJECT SB
     WHERE U.UNIV_ID=L.UNIV_ID
       AND L.LECTURER_ID=SL.LECTURER_ID
       AND SL.SUBJ_ID=SB.SUBJ_ID
     GROUP BY UNIV_NAME, SEMESTER;
6. SELECT UNIV_NAME, SUBJ_NAME, SUM(HOUR)
     FROM UNIVERSITY U, LECTURER L, SUBJ_LECT SL,
          SUBJECT SB
     WHERE U.UNIV_ID=L.UNIV_ID
       AND L.LECTURER_ID=SL.LECTURER_ID
       AND SL.SUBJ_ID=SB.SUBJ_ID
     GROUP BY UNIV_NAME, SUBJ_NAME;
7. SELECT SURNAME, NAME, SUBJ_NAME, SUM(HOUR)
     FROM LECTURER L, SUBJ_LECT SL, SUBJECT SB
     WHERE L.LECTURER_ID=SL.LECTURER_ID
       AND SL.SUBJ_ID=SB.SUBJ_ID
     GROUP BY SURNAME, NAME, SUBJ_NAME;
8. SELECT UNIV_NAME, MAX(STIPEND)
     FROM STUDENT S, UNIVERSITY U
     WHERE S.UNIV_ID=U.UNIV_ID
     GROUP BY UNIV_NAME
     ORDER BY 2;
```

```
9. SELECT UNIV_NAME, SURNAME, NAME, BIRTHDAY
     FROM STUDENT S, UNIVERSITY U
     WHERE S.UNIV_ID=U.UNIV_ID
       AND BIRTHDAY=(SELECT MAX(BIRTHDAY)
                     FROM STUDENT
                     WHERE UNIV_ID=U.UNIV_ID);
10. SELECT UNIV_NAME, SURNAME, NAME, STIPEND
     FROM STUDENT S, UNIVERSITY U
     WHERE S.UNIV_ID=U.UNIV_ID
      AND STIPEND=(SELECT MAX(STIPEND) FROM STUDENT
                   WHERE UNIV_ID=U.UNIV_ID);
11. SELECT SURNAME, NAME, SUBJ_NAME, MARK
     FROM STUDENT ST, EXAM_MARKS E, SUBJECT SB
     WHERE ST.STUDENT_ID=E.STUDENT_ID
       AND E.SUBJ_ID=SB.SUBJ_ID;
12. SELECT SUBJ_NAME, SURNAME, NAME, MARK
     FROM STUDENT ST, EXAM_MARKS E, SUBJECT SB
     WHERE ST.STUDENT_ID=E.STUDENT_ID
       AND E.SUBJ_ID=SB.SUBJ_ID
       AND MARK=(SELECT MAX(MARK) FROM EXAM_MARKS
                 WHERE SUBJ_ID=SB.SUBJ_ID);
13. SELECT SUBJ_NAME, SURNAME, NAME, EXAM_DATE
     FROM STUDENT ST, EXAM_MARKS E, SUBJECT SB
     WHERE ST.STUDENT_ID=E.STUDENT_ID
       AND E.SUBJ_ID=SB.SUBJ_ID
       AND EXAM_DATE=(SELECT MAX(EXAM_DATE)
                      FROM EXAM_MARKS
                      WHERE SUBJ_ID=SB.SUBJ_ID);
14. SELECT SUBJ_NAME, SURNAME, NAME, EXAM_DATE, MARK
     FROM STUDENT ST, EXAM_MARKS E, SUBJECT SB
     WHERE ST.STUDENT_ID=E.STUDENT_ID
       AND E.SUBJ_ID=SB.SUBJ_ID
       AND MARK>2
       AND EXAM_DATE=(SELECT MIN(EXAM_DATE)
                      FROM EXAM_MARKS
                      WHERE SUBJ_ID=SB.SUBJ_ID);
15. SELECT * FROM LECTURER L, SUBJ_LECT S
     WHERE L.LECTURER_ID=S.LECTURER_ID
       AND EXISTS (SELECT * FROM SUBJ_LECT
                   WHERE LECTURER_ID=S.LECTURER_ID
                     AND SUBJ_ID<>S.SUBJ_ID);
16. SELECT * FROM LECTURER L, SUBJ_LECT S
     WHERE L.LECTURER_ID=S.LECTURER_ID
       AND NOT EXISTS (SELECT * FROM SUBJ_LECT
                   WHERE LECTURER_ID=S.LECTURER_ID
                     AND SUBJ_ID<>S.SUBJ_ID);
```