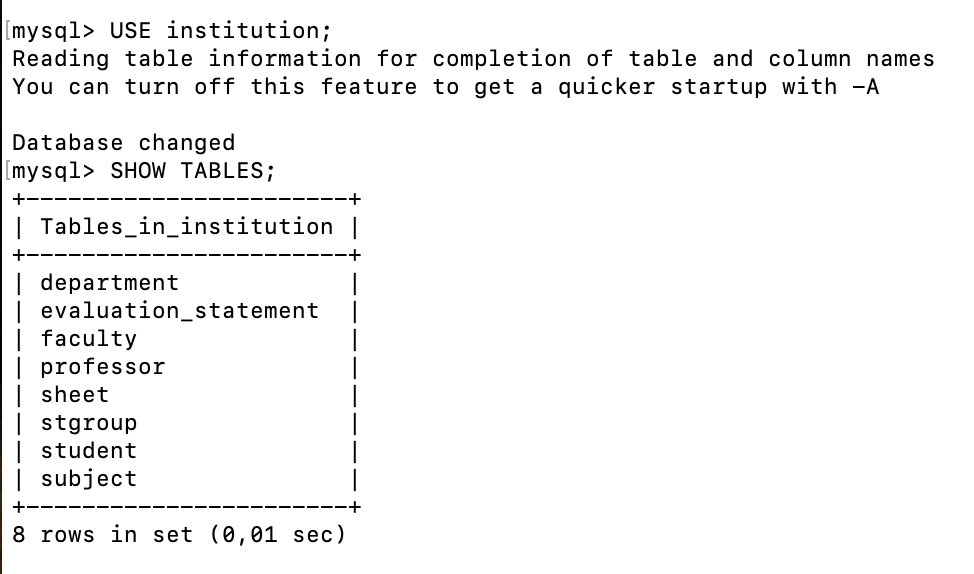
|  |  |  |
| --- | --- | --- |
| **Список сущностей (таблицы)** | | |
| № | Название | Назначение |
| 1 | Факультет | Описание факультета и его деканата |
| 2 | Кафедра | Описание кафедры |
| 3 | Преподаватели | Описание состава сотрудников кафедр |
| 4 | Группы | Перечень групп каждой кафедры |
| 5 | Студенты | Перечень студентов каждой группы |
| 6 | Предметы | Перечень предметов каждой кафедры |
| 7 | Ведомости | Экзаменационно-зачетные ведомости с перечнем студентов и их оценками |
| 8 | Оценочная ведомость | Таблица внутри таблицы ведомости. Отражает связь один-ко-многим. Каждая ведомость дается каждой группе с множеством студентов |

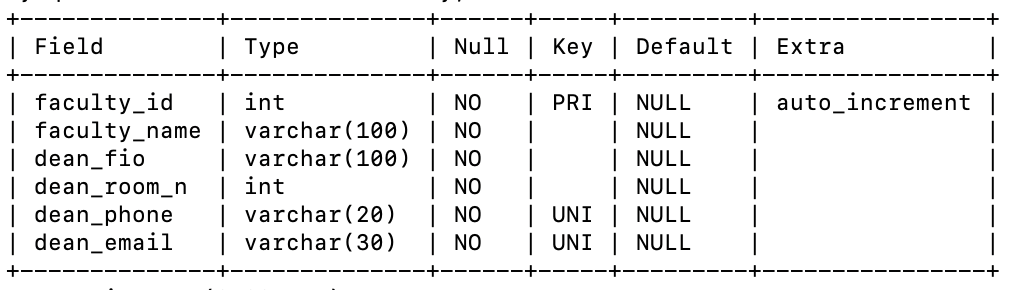
**Отчет Фокиной Д.И.(МЛТ221) по итоговому заданию по дисциплине «База данных»**

**База данных «Учебное заведение» (Institution)**

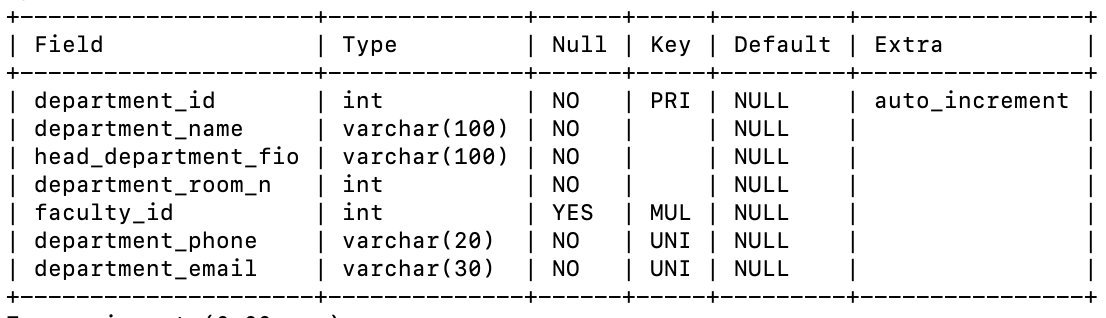
****

**Описание сущностей (таблиц). Описание атрибутов таблиц и их идентификаторов**

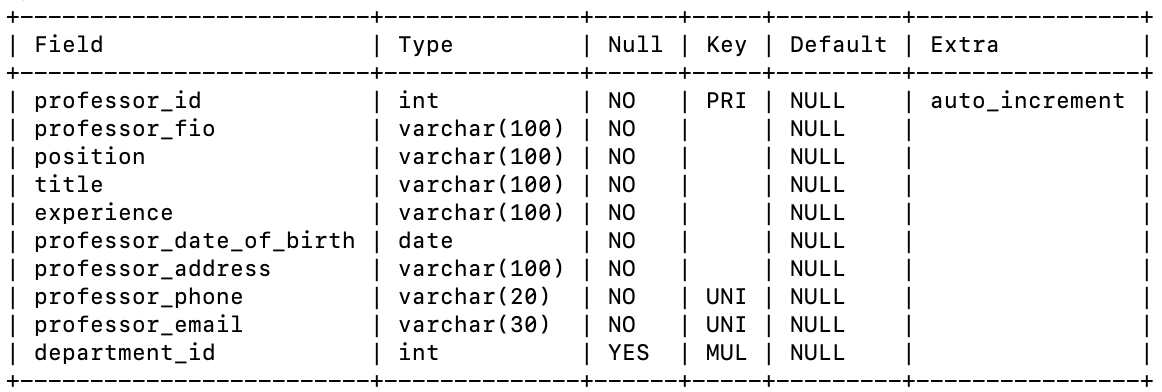
|  |  |  |  |
| --- | --- | --- | --- |
| **Список атрибутов таблицы «Факультеты» (faculty)** | | | |
| Ключевое поле | Название | Идентификатор | Назначение |
| Первичный ключ | Код факультета | faculty\_id | Первичный ключ – уникальное значение, соответствующее каждому факультету. |
|  | Название факультета | faculty\_name |  |
|  | ФИО декана | dean\_fio |  |
|  | Номер кабинета деканата | dean\_room\_n |  |
|  | Телефон деканата | dean\_phone |  |
|  | E-mail деканата | dean\_email |  |



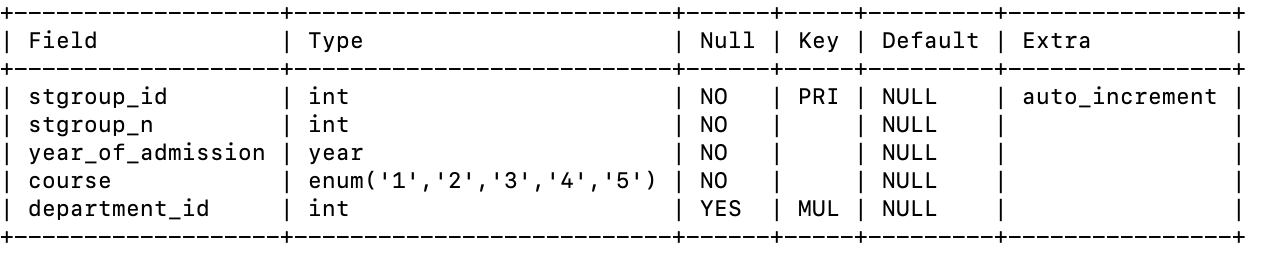
|  |  |  |  |
| --- | --- | --- | --- |
| **Список атрибутов таблицы «Кафедра»** **(department)** | | | |
| Ключевое поле | Название | Идентификатор | Назначение |
| Первичный ключ | Код кафедры | department\_id | Первичный ключ – уникальное значение, соответствующее каждой кафедре. |
| Внешний ключ | Код факультета | faculty\_id | С помощью внешнего ключа определяется какая кафедра к какому факультету принадлежит. |
|  | Название кафедры | department\_name |  |
|  | ФИО завкафедры | head\_department\_fio |  |
|  | Номер кабинета кафедры | department\_room\_n |  |
|  | Телефон кафедры | department\_phone |  |
|  | E-mail кафедры | department\_email |  |



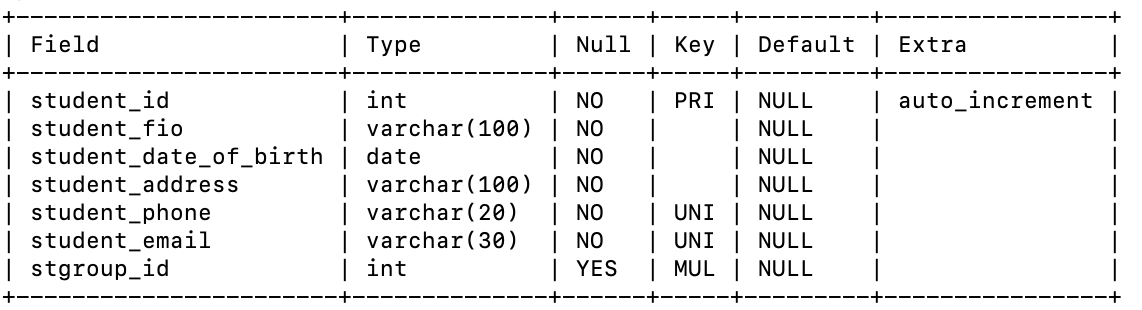
|  |  |  |  |
| --- | --- | --- | --- |
| **Список атрибутов таблицы «Преподаватели» (professor)** | | | |
| Ключевое поле | Название | Идентификатор | Назначение |
| Первичный ключ | Код преподавателя | professor\_id | Первичный ключ – уникальное значение, соответствующее каждому преподавателю. |
| Внешний ключ | Код кафедры | department\_id | С помощью внешнего ключа определяется какой преподаватель к какой кафедре принадлежит. |
|  | ФИО | professor\_FIO |  |
|  | Должность | position |  |
|  | Научное звание | title |  |
|  | Стаж работы | experience |  |
|  | Дата рождения | professor\_date\_of\_birth |  |
|  | Адрес проживания | professor\_address |  |
|  | Телефон преподавателя | professor\_phone |  |
|  | E-mail преподавателя | professor\_email |  |



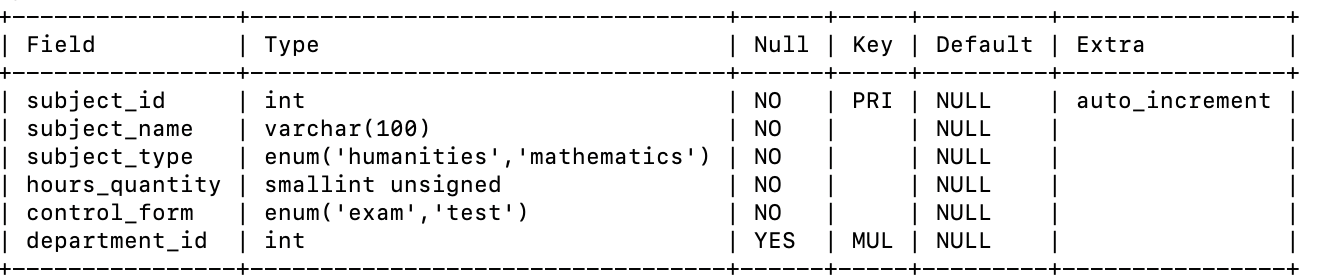
|  |  |  |  |
| --- | --- | --- | --- |
| **Список атрибутов таблицы «Группы»** **(stgroup)** | | | |
| Ключевое поле | Название | Идентификатор | Назначение |
| Первичный ключ | Код группы | stgroup\_id | Первичный ключ – уникальное значение, соответствующее каждой группе. |
| Внешний ключ | Код кафедры | department\_id | С помощью внешнего ключа определяется какая группа к какой кафедре принадлежит. |
|  | Номер группы | stgroup\_n |  |
|  | Год поступления | year\_of\_admission |  |
|  | Курс обучения | course |  |



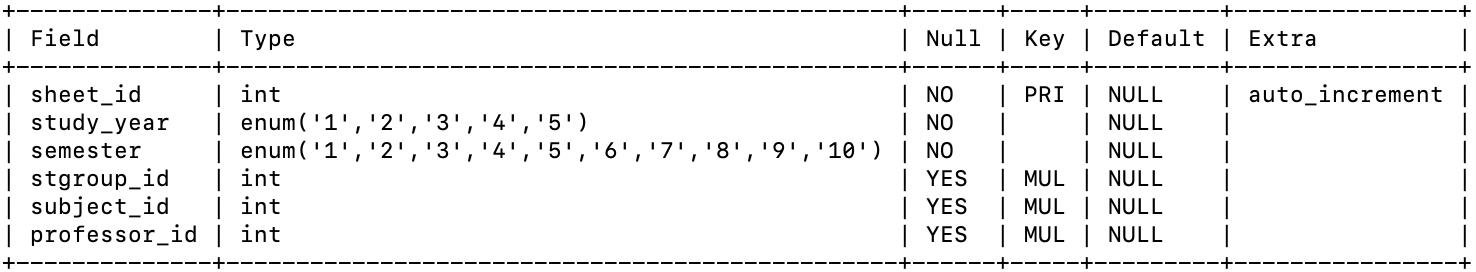
|  |  |  |  |
| --- | --- | --- | --- |
| **Список атрибутов таблицы «Студенты»** **(student)** | | | |
| Ключевое поле | Название | Идентификатор | Назначение |
| Первичный ключ | Код студента | student\_id | Первичный ключ – уникальное значение, соответствующее каждому студенту. |
| Внешний ключ | Код группы | stgroup\_id | С помощью внешнего ключа определяется какой студент к какой группе принадлежит. |
|  | ФИО | student\_fio |  |
|  | Дата рождения | student\_date\_of\_birth |  |
|  | Адрес проживания | student\_address |  |
|  | Телефон студента | student\_phone |  |
|  | E-mail студента | student\_email |  |



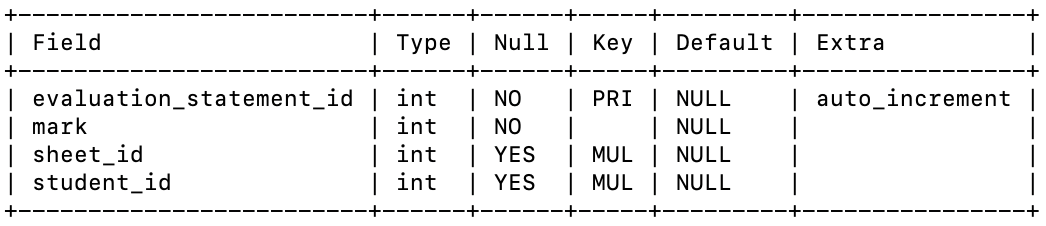
|  |  |  |  |
| --- | --- | --- | --- |
| **Список атрибутов таблицы «Предметы» (subject)** | | | |
| Ключевое поле | Название | Идентификатор | Назначение |
| Первичный ключ | Код предмета | subject\_id | Первичный ключ – уникальное значение, соответствующее каждому предмету. |
| Внешний ключ | Код кафедры | department\_id | С помощью внешнего ключа определяется какой предмет к какой кафедре принадлежит. |
|  | Название предмета | subject\_name |  |
|  | Тип предмета | subject\_type |  |
|  | Количество часов | hours\_quantity |  |
|  | Форма контроля | control\_form |  |



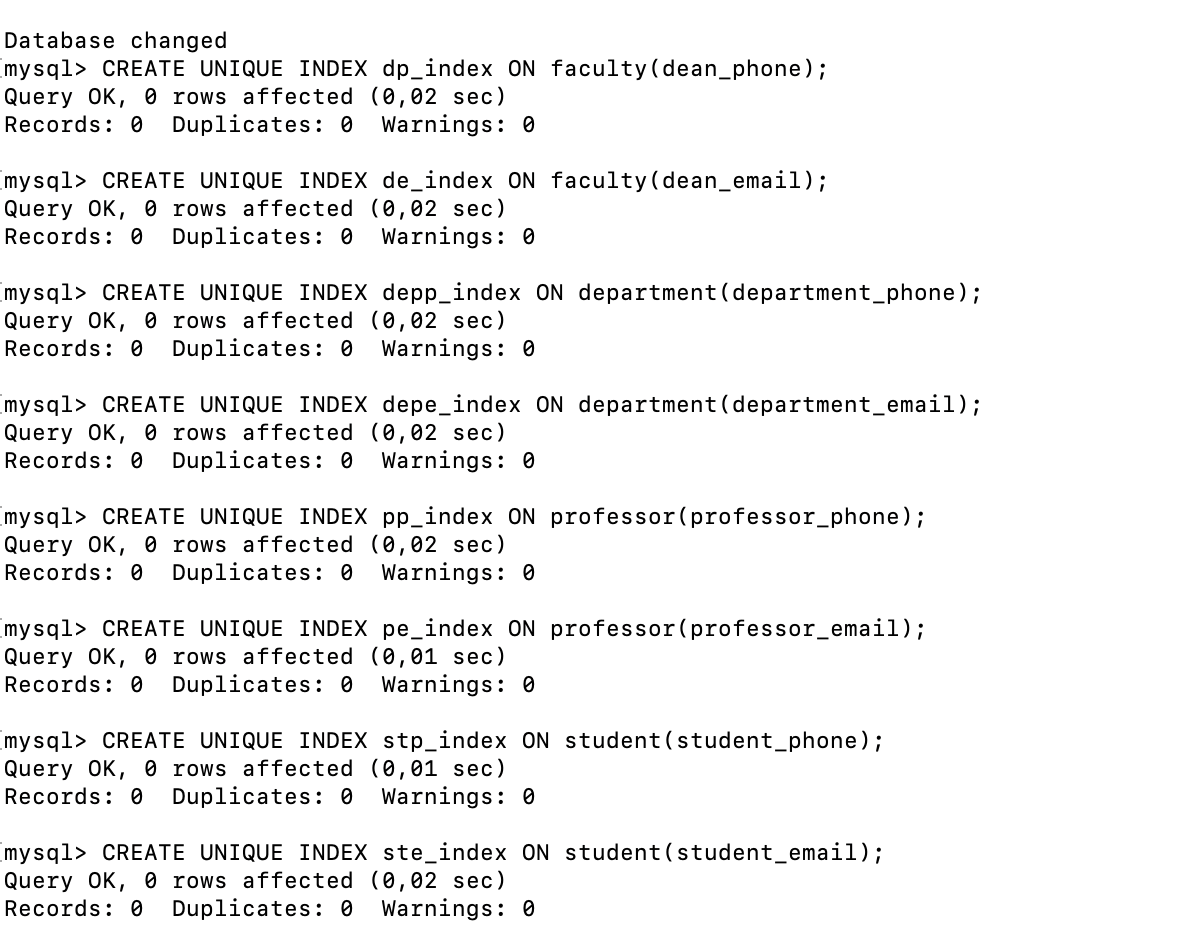
|  |  |  |  |
| --- | --- | --- | --- |
| **Список атрибутов таблицы «Ведомости» (sheet)** | | | |
| Ключевое поле | Название | Идентификатор | Назначение |
| Первичный ключ | Код ведомости | sheet\_id | Первичный ключ – уникальное значение, соответствующее каждой учебной ведомости. |
| Внешний ключ | Код группы | stgroup\_id | С помощью внешнего ключа определяется какая ведомость какой группе выдана. |
| Внешний ключ | Код предмета | subject\_id | С помощью внешнего ключа определяется какая ведомость для какого предмета выдана. |
| Внешний ключ | Код преподавателя | professor\_id | С помощью внешнего ключа определяется какая ведомость какому преподавателю выдана. |
|  | Учебный год | study\_year |  |
|  | Семестр | semester |  |



|  |  |  |  |
| --- | --- | --- | --- |
| **Список атрибутов таблицы «Оценочная ведомость» (evaluation\_statement)** | | | |
| Ключевое поле | Название | Идентификатор | Назначение |
| Первичный ключ | Код оценочной ведомости | evaluation\_statement\_id | Первичный ключ – уникальное значение, соответствующее каждой оценочной ведомости. |
| Внешний ключ | Код ведомости | sheet\_id | С помощью внешнего ключа эта таблица связывается с таблицей «Ведомости». |
| Внешний ключ | Код студента | student\_id | С помощью внешнего ключа определяется студент. |
|  | Оценка | mark |  |



**Для ограничения допустимых значений используется ENUM (например, для типа предмета, года обучения, семестра, курса, формы контроля), для ограничения переменной длины строк – VARCHAR.**

**Для предотвращения дублирования записей установлены уникальные индексы для телефонов и адресов электронной почты деканов, кафедр, преподавателей и студентов. **

**Описание связей**

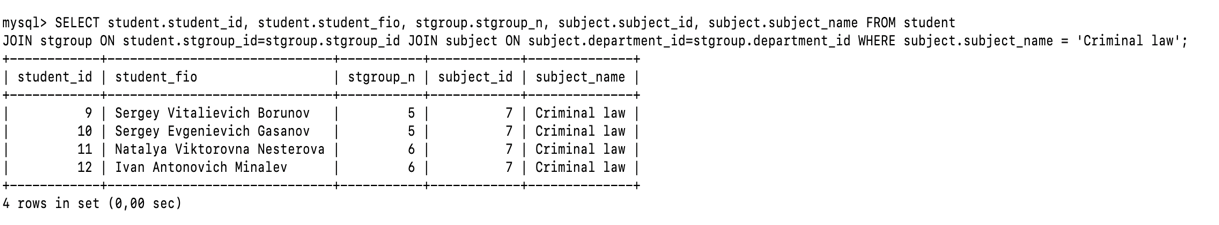
|  |  |
| --- | --- |
| **Связанные таблицы по типу один-ко-многим** | **Описание** |
| Факультет - Кафедра | Одному факультету могут принадлежать несколько кафедр |
| Кафедра - Группа | Одной кафедре может принадлежать несколько групп |
| Кафедра - Предметы | На одной кафедре может преподаваться несколько предметов |
| Кафедра – Преподаватели | На одной кафедре могут работать несколько преподавателей |
| Группа- Студенты | В каждой группе учится несколько студентов |
| Группа - Ведомости | Каждой группе выдают несколько ведомостей |
| Предметы – Ведомости | Из множества предметов составляется одна ведомость |
| Преподаватели - Ведомости | Ведомость выдается конкретному преподавателю |
| Ведомости – Оценочная ведомость | Оценочная ведомость принадлежит одной определенной ведомости |
| Студенты – Оценочная ведомость | В оценочной ведомости перечислены все студенты группы |

**Возможности:**

* **выводить список студентов по определённому предмету:**

**Если запросом:**

SELECT student.student\_id, student.student\_fio, stgroup.stgroup\_n, subject.subject\_id, subject.subject\_name FROM student JOIN stgroup ON student.stgroup\_id=stgroup.stgroup\_id JOIN subject ON subject.department\_id=stgroup.department\_id WHERE subject.subject\_name='Criminal law';



**Если хранимой процедурой:**

DELIMITER //

CREATE PROCEDURE GetStudentsBySubject(IN subject\_name VARCHAR(100))

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

RESIGNAL;

END;

START TRANSACTION;

SELECT student.student\_id, student.student\_fio,

stgroup.stgroup\_n, subject.subject\_id, subject.subject\_name

FROM student

JOIN stgroup ON student.stgroup\_id=stgroup.stgroup\_id

JOIN subject ON subject.department\_id=stgroup.department\_id

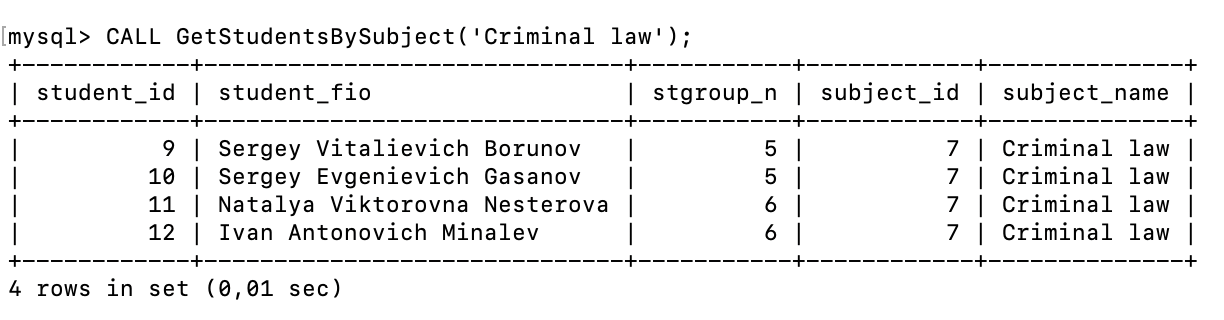
WHERE subject.subject\_name = subject\_name;

COMMIT;

END //

DELIMITER ;

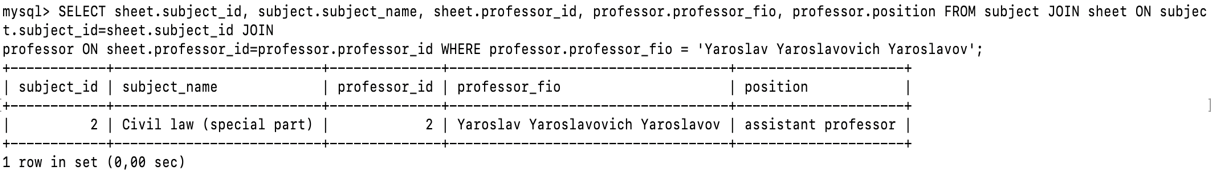
CALL GetStudentsBySubject('Criminal law');



* **выводить список предметов, которые преподаёт конкретный преподаватель:**

**Если запросом:**

SELECT sheet.subject\_id, subject.subject\_name, sheet.professor\_id, professor.professor\_fio, professor.position FROM subject JOIN sheet ON subject.subject\_id=sheet.subject\_id JOIN professor ON sheet.professor\_id=professor.professor\_id WHERE professor.professor\_fio = 'Yaroslav Yaroslavovich Yaroslavov';



**Если хранимой процедурой:**

DELIMITER //

CREATE PROCEDURE GetSubjectByProfessorFio(IN professor\_fio VARCHAR(100))

BEGIN

SELECT sheet.subject\_id, subject.subject\_name,

sheet.professor\_id, professor.professor\_fio, professor.position

FROM subject

JOIN sheet ON subject.subject\_id = sheet.subject\_id

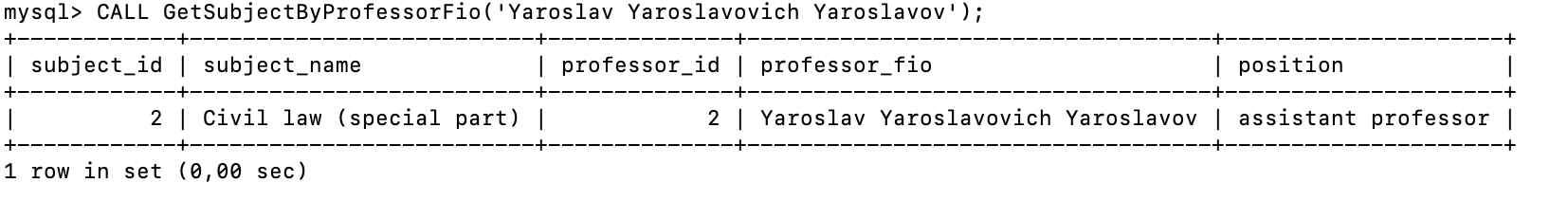
JOIN professor ON sheet.professor\_id = professor.professor\_id

WHERE professor.professor\_fio = professor\_fio;

END //

DELIMITER ;

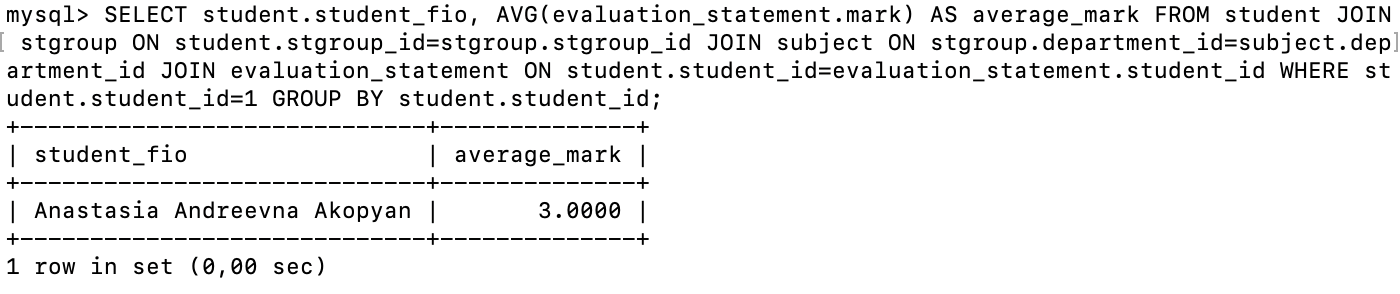
CALL GetSubjectByProfessorFio('Yaroslav Yaroslavovich Yaroslavov');



**- выводить средний балл студента по всем предметам:**

**Если запросом:**

SELECT student.student\_fio, AVG(evaluation\_statement.mark) AS average\_mark FROM student JOIN stgroup ON student.stgroup\_id=stgroup.stgroup\_id JOIN subject ON stgroup.department\_id=subject.department\_id JOIN evaluation\_statement ON student.student\_id=evaluation\_statement.student\_id WHERE student.student\_id=1 GROUP BY student.student\_id;



**Если хранимой процедурой:**

DELIMITER //

CREATE PROCEDURE GetAverageMarkByStudentId(IN student\_id INT)

BEGIN

SELECT student.student\_fio, AVG(evaluation\_statement.mark) AS average\_mark

FROM student

JOIN stgroup ON student.stgroup\_id = stgroup.stgroup\_id

JOIN subject ON stgroup.department\_id = subject.department\_id

JOIN evaluation\_statement ON student.student\_id = evaluation\_statement.student\_id

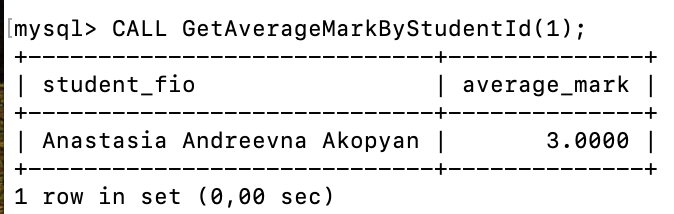
WHERE student.student\_id = student\_id

GROUP BY student.student\_id;

END //

DELIMITER ;

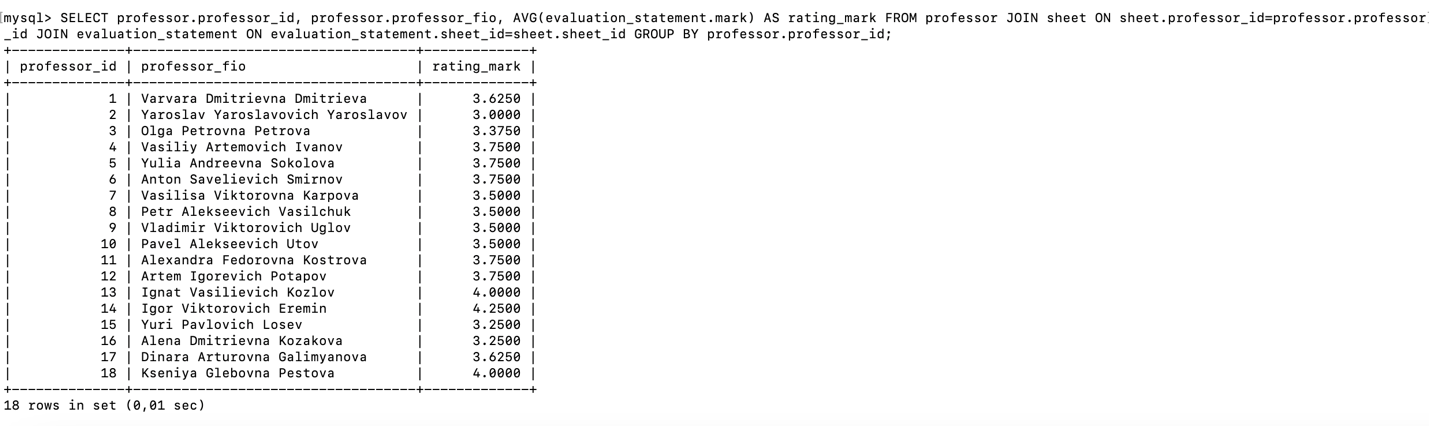
CALL GetAverageMarkByStudentId(1);



- **выводить рейтинг преподавателей по средней оценке студентов:**

**Если запросом:**

SELECT professor.professor\_id, professor.professor\_fio, AVG(evaluation\_statement.mark) AS rating\_mark FROM professor JOIN sheet ON sheet.professor\_id=professor.professor\_id JOIN evaluation\_statement ON evaluation\_statement.sheet\_id=sheet.sheet\_id GROUP BY professor.professor\_id;



**Если хранимой процедурой:**

DELIMITER //

CREATE PROCEDURE GetAverageRatingByProfessor()

BEGIN

SELECT professor.professor\_id, professor.professor\_fio, AVG(evaluation\_statement.mark) AS rating\_mark

FROM professor

JOIN sheet ON sheet.professor\_id = professor.professor\_id

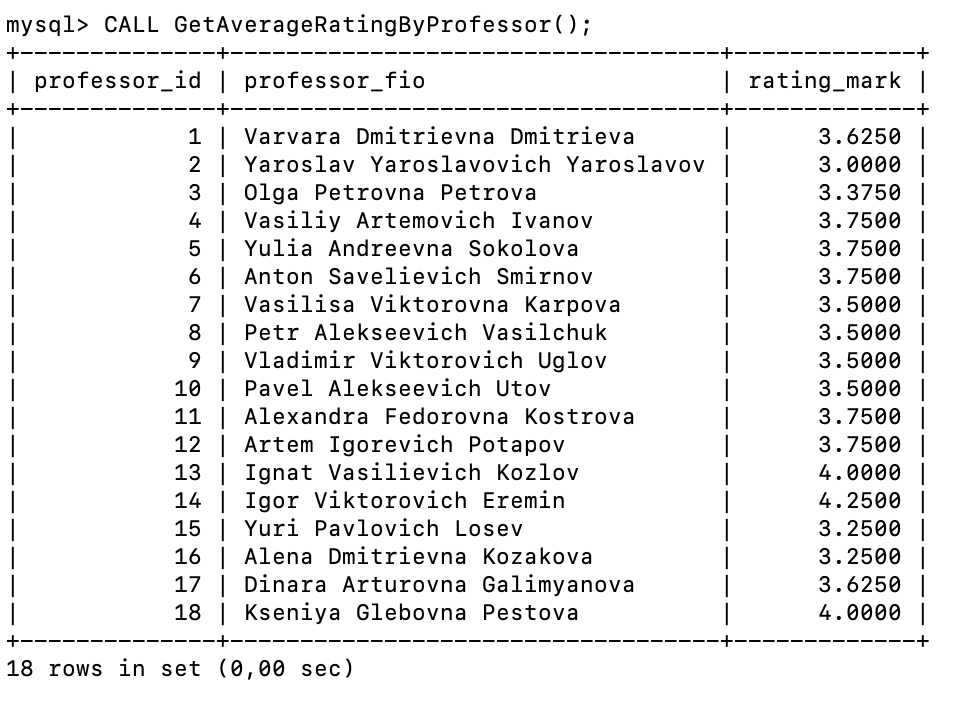
JOIN evaluation\_statement ON evaluation\_statement.sheet\_id = sheet.sheet\_id

GROUP BY professor.professor\_id;

END //

mysql> DELIMITER ;

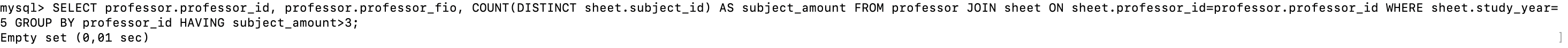
CALL GetAverageRatingByProfessor();

****

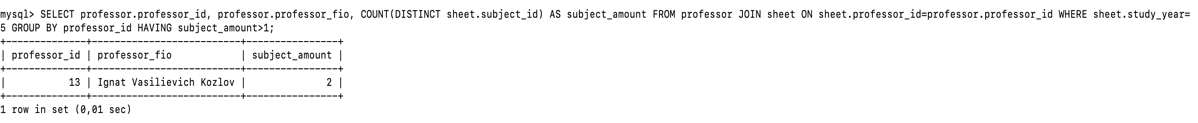
- **выводить список преподавателей, которые преподавали более 3 предметов за последний год:**

**Если запросом:**

SELECT professor.professor\_id, professor.professor\_fio, COUNT(DISTINCT sheet.subject\_id) AS subject\_amount FROM professor JOIN sheet ON sheet.professor\_id=professor.professor\_id WHERE sheet.study\_year=5 GROUP BY professor\_id HAVING subject\_amount>3;



*В базе данных есть преподаватели, ведущие два предмета, более 3-х предметов в текущей версии – нет.*



**Если хранимой процедурой:**

DELIMITER //

CREATE PROCEDURE GetProfessorsWithSubjectAmount()

BEGIN

SELECT professor.professor\_id, professor.professor\_fio, COUNT(DISTINCT sheet.subject\_id) AS subject\_amount

FROM professor

JOIN sheet ON sheet.professor\_id = professor.professor\_id

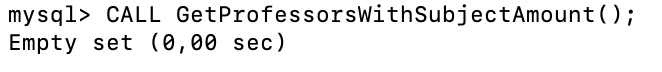
WHERE sheet.study\_year = 5

GROUP BY professor.professor\_id

HAVING subject\_amount > 3;

END //

DELIMITER ;

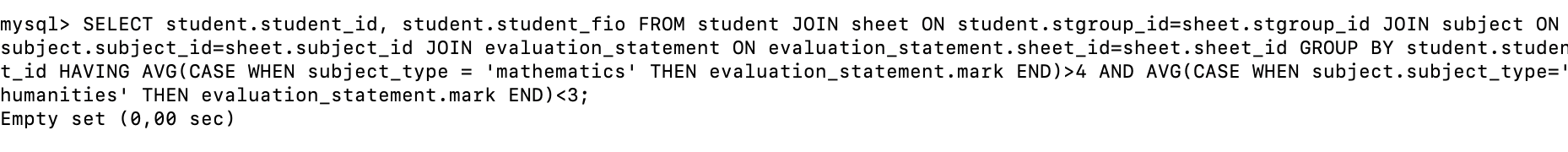
CALL GetProfessorsWithSubjectAmount();

- **выводить список студентов, которые имеют средний балл выше 4 по математическим предметам, но ниже 3 по гуманитарным:**

**Если запросом:**

SELECT student.student\_id, student.student\_fio FROM student JOIN sheet ON student.stgroup\_id=sheet.stgroup\_id JOIN subject ON subject.subject\_id=sheet.subject\_id JOIN evaluation\_statement ON evaluation\_statement.sheet\_id=sheet.sheet\_id GROUP BY student.student\_id HAVING AVG(CASE WHEN subject\_type = 'mathematics' THEN evaluation\_statement.mark END)>4 AND AVG(CASE WHEN subject.subject\_type='humanities' THEN evaluation\_statement.mark END)<3;

*В текущей версии нет студентов, которые имеют средний балл выше 4 по математическим предметам, но ниже 3 по гуманитарным.*



**Если хранимой процедурой:**

DELIMITER //

CREATE PROCEDURE filter\_students()

BEGIN

SELECT student.student\_id, student.student\_fio

FROM student

JOIN sheet

ON student.stgroup\_id = sheet.stgroup\_id

JOIN subject

ON subject.subject\_id = sheet.subject\_id

JOIN evaluation\_statement

ON evaluation\_statement.sheet\_id = sheet.sheet\_id

GROUP BY student.student\_id

HAVING AVG(CASE WHEN subject\_type = 'mathematics' THEN evaluation\_statement.mark END) > 4

AND AVG(CASE WHEN subject.subject\_type = 'humanities' THEN evaluation\_statement.mark END) < 3;

END //

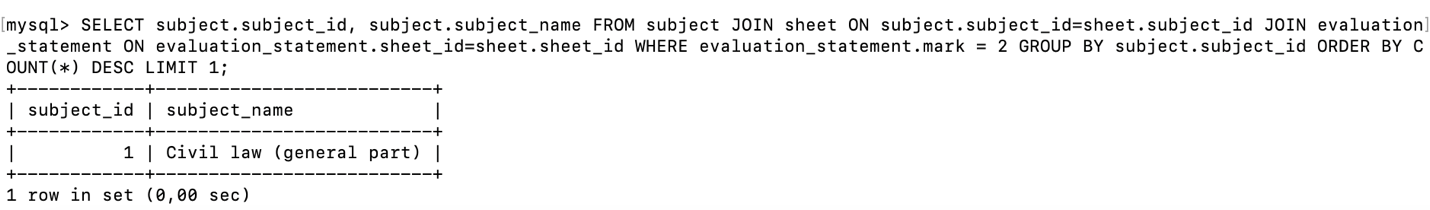
DELIMITER ;

CALL filter\_students();

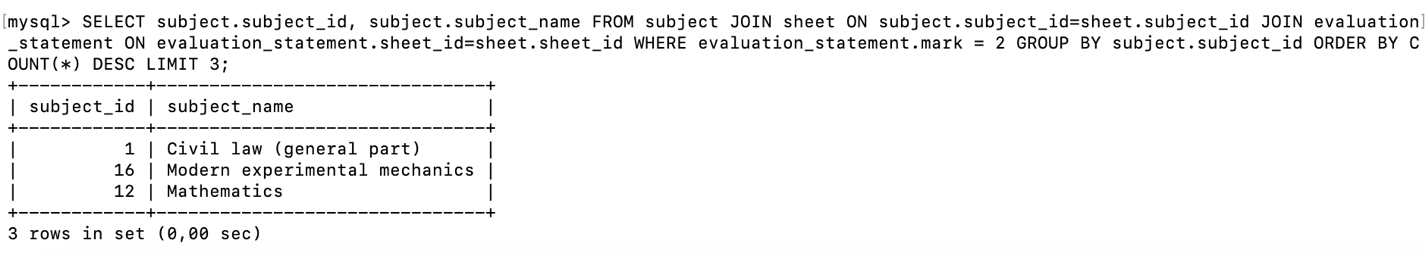


**- определить предметы, по которым больше всего двоек в текущем семестре:**

**Если запросом:**

SELECT subject.subject\_id, subject.subject\_name FROM subject JOIN sheet ON subject.subject\_id=sheet.subject\_id JOIN evaluation\_statement ON evaluation\_statement.sheet\_id=sheet.sheet\_id WHERE evaluation\_statement.mark = 2 GROUP BY subject.subject\_id ORDER BY COUNT(\*) DESC LIMIT 1; 

SELECT subject.subject\_id, subject.subject\_name FROM subject JOIN sheet ON subject.subject\_id=sheet.subject\_id JOIN evaluation\_statement ON evaluation\_statement.sheet\_id=sheet.sheet\_id WHERE evaluation\_statement.mark = 2 GROUP BY subject.subject\_id ORDER BY COUNT(\*) DESC LIMIT 3;



**Если хранимой процедурой:**

DELIMITER //

CREATE PROCEDURE GetSubjectWithMostMarks()

BEGIN

SELECT subject.subject\_id, subject.subject\_name

FROM subject

JOIN sheet ON subject.subject\_id = sheet.subject\_id

JOIN evaluation\_statement ON evaluation\_statement.sheet\_id = sheet.sheet\_id

WHERE evaluation\_statement.mark = 2

GROUP BY subject.subject\_id

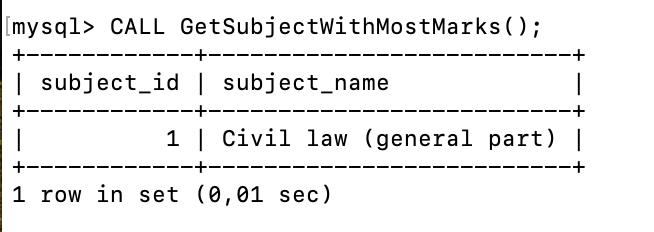
ORDER BY COUNT(\*) DESC

LIMIT 1;

END //

DELIMITER ;

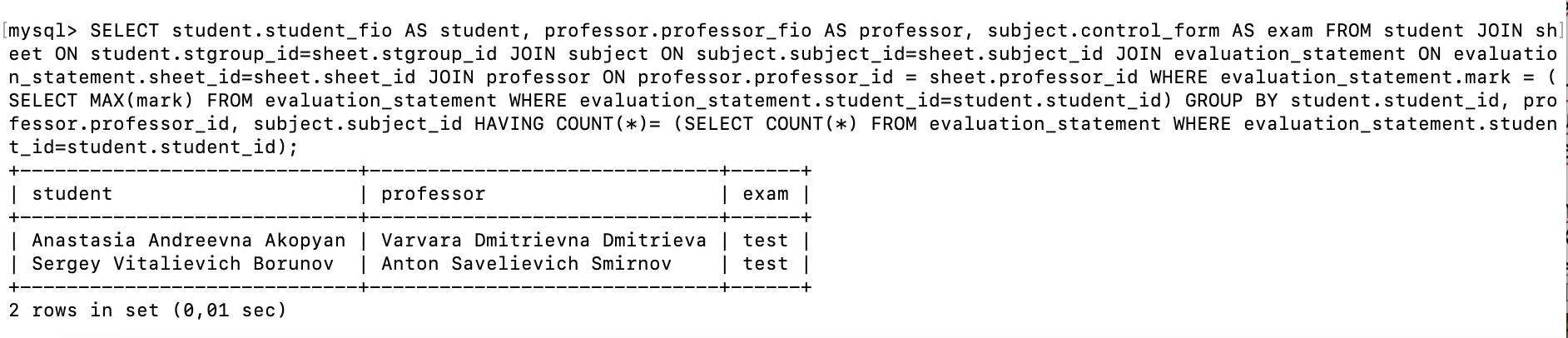
CALL GetSubjectWithMostMarks();



**- выводить студентов, которые получили высший балл по всем своим экзаменам, и преподавателей, которые вели эти предметы:**

**Если запросом:**

SELECT student.student\_fio AS student, professor.professor\_fio AS professor, subject.control\_form AS exam FROM student JOIN sheet ON student.stgroup\_id=sheet.stgroup\_id JOIN subject ON subject.subject\_id=sheet.subject\_id JOIN evaluation\_statement ON evaluation\_statement.sheet\_id=sheet.sheet\_id JOIN professor ON professor.professor\_id = sheet.professor\_id WHERE evaluation\_statement.mark = (SELECT MAX(mark) FROM evaluation\_statement WHERE evaluation\_statement.student\_id=student.student\_id) GROUP BY student.student\_id, professor.professor\_id, subject.subject\_id HAVING COUNT(\*)= (SELECT COUNT(\*) FROM evaluation\_statement WHERE evaluation\_statement.student\_id=student.student\_id);



**Если хранимой процедурой:**

DELIMITER //

CREATE PROCEDURE GetExamResults()

BEGIN

SELECT student.student\_fio AS student, professor.professor\_fio AS professor, subject.control\_form AS exam

FROM student

JOIN sheet ON student.stgroup\_id = sheet.stgroup\_id

JOIN subject ON subject.subject\_id = sheet.subject\_id

JOIN evaluation\_statement ON evaluation\_statement.sheet\_id = sheet.sheet\_id

JOIN professor ON professor.professor\_id = sheet.professor\_id

WHERE evaluation\_statement.mark = (SELECT MAX(mark) FROM evaluation\_statement WHERE evaluation\_statement.student\_id = student.student\_id)

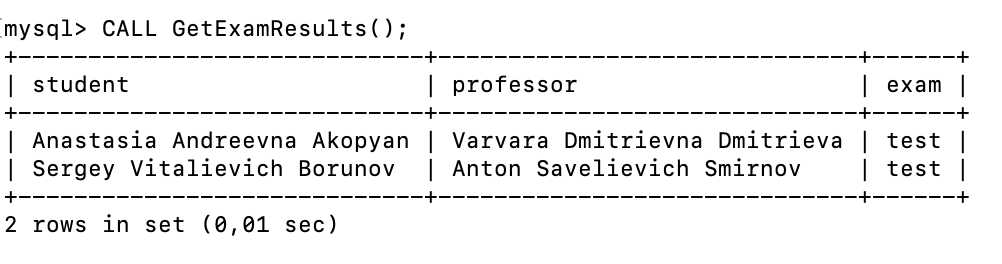
GROUP BY student.student\_id, professor.professor\_id, subject.subject\_id

HAVING COUNT(\*) = (SELECT COUNT(\*) FROM evaluation\_statement WHERE evaluation\_statement.student\_id = student.student\_id);

END //

DELIMITER ;

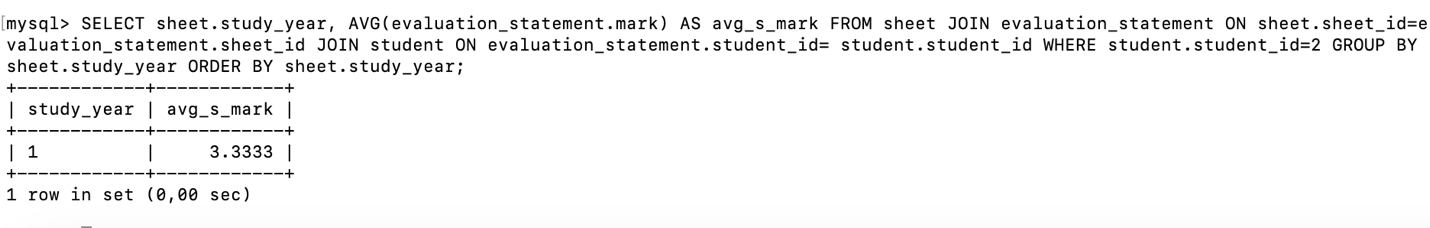
CALL GetExamResults();



**- просматривать изменение среднего балла студента по годам обучения:**

**Если запросом:**

SELECT sheet.study\_year, AVG(evaluation\_statement.mark) AS avg\_s\_mark FROM sheet JOIN evaluation\_statement ON sheet.sheet\_id=evaluation\_statement.sheet\_id JOIN student ON evaluation\_statement.student\_id= student.student\_id WHERE student.student\_id=2 GROUP BY sheet.study\_year ORDER BY sheet.study\_year;



**Если хранимой процедурой:**

DELIMITER //

CREATE PROCEDURE calculateAvgMarkByStudyYear(IN studentId INT)

BEGIN

SELECT sheet.study\_year, AVG(evaluation\_statement.mark) AS avg\_s\_mark

FROM sheet

JOIN evaluation\_statement ON sheet.sheet\_id = evaluation\_statement.sheet\_id

JOIN student ON evaluation\_statement.student\_id = student.student\_id

WHERE student.student\_id = studentId

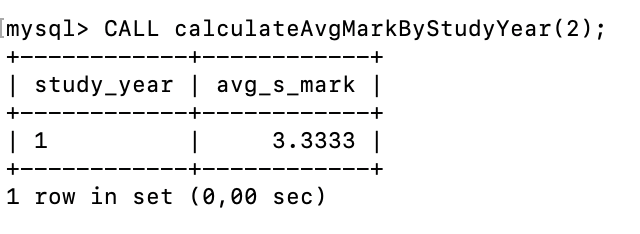
GROUP BY sheet.study\_year

ORDER BY sheet.study\_year;

END//

DELIMITER ;

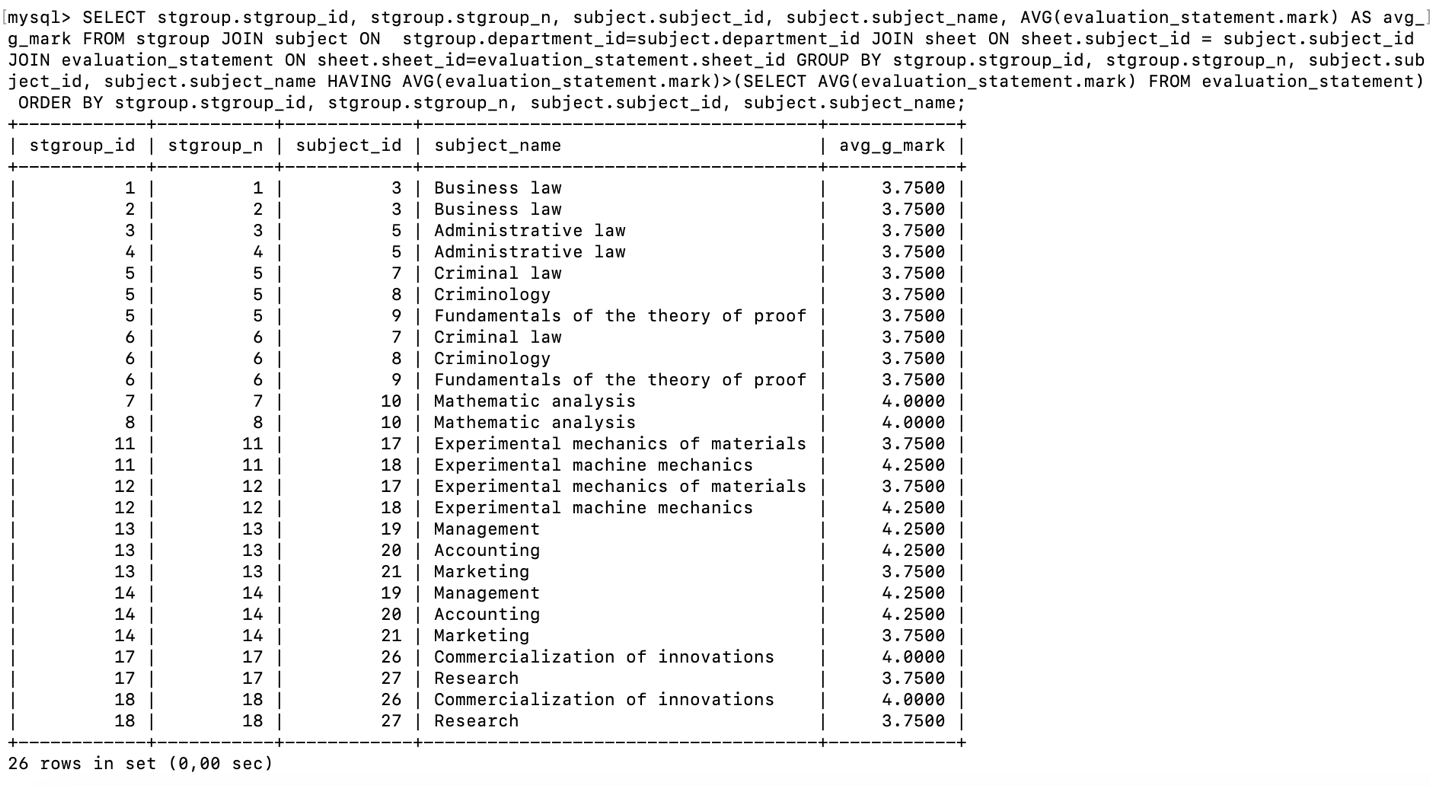
CALL calculateAvgMarkByStudyYear(2);

****

**- определить группы, в которых средний балл выше, чем в других, по аналогичным предметам, чтобы выявить лучшие методики преподавания или особенности состава группы:**

**Если запросом:**

SELECT stgroup.stgroup\_id, stgroup.stgroup\_n, subject.subject\_id, subject.subject\_name, AVG(evaluation\_statement.mark) AS avg\_g\_mark FROM stgroup JOIN subject ON stgroup.department\_id=subject.department\_id JOIN sheet ON sheet.subject\_id = subject.subject\_id JOIN evaluation\_statement ON sheet.sheet\_id=evaluation\_statement.sheet\_id GROUP BY stgroup.stgroup\_id, stgroup.stgroup\_n, subject.subject\_id, subject.subject\_name HAVING AVG(evaluation\_statement.mark)>(SELECT AVG(evaluation\_statement.mark) FROM evaluation\_statement) ORDER BY stgroup.stgroup\_id, stgroup.stgroup\_n, subject.subject\_id, subject.subject\_name;



**Если хранимой процедурой:**

DELIMITER //

CREATE PROCEDURE calculateAvgMarkByGroup()

BEGIN

SELECT stgroup.stgroup\_id, stgroup.stgroup\_n, subject.subject\_id, subject.subject\_name,

AVG(evaluation\_statement.mark) AS avg\_g\_mark

FROM stgroup

JOIN subject ON stgroup.department\_id = subject.department\_id

JOIN sheet ON sheet.subject\_id = subject.subject\_id

JOIN evaluation\_statement ON sheet.sheet\_id = evaluation\_statement.sheet\_id

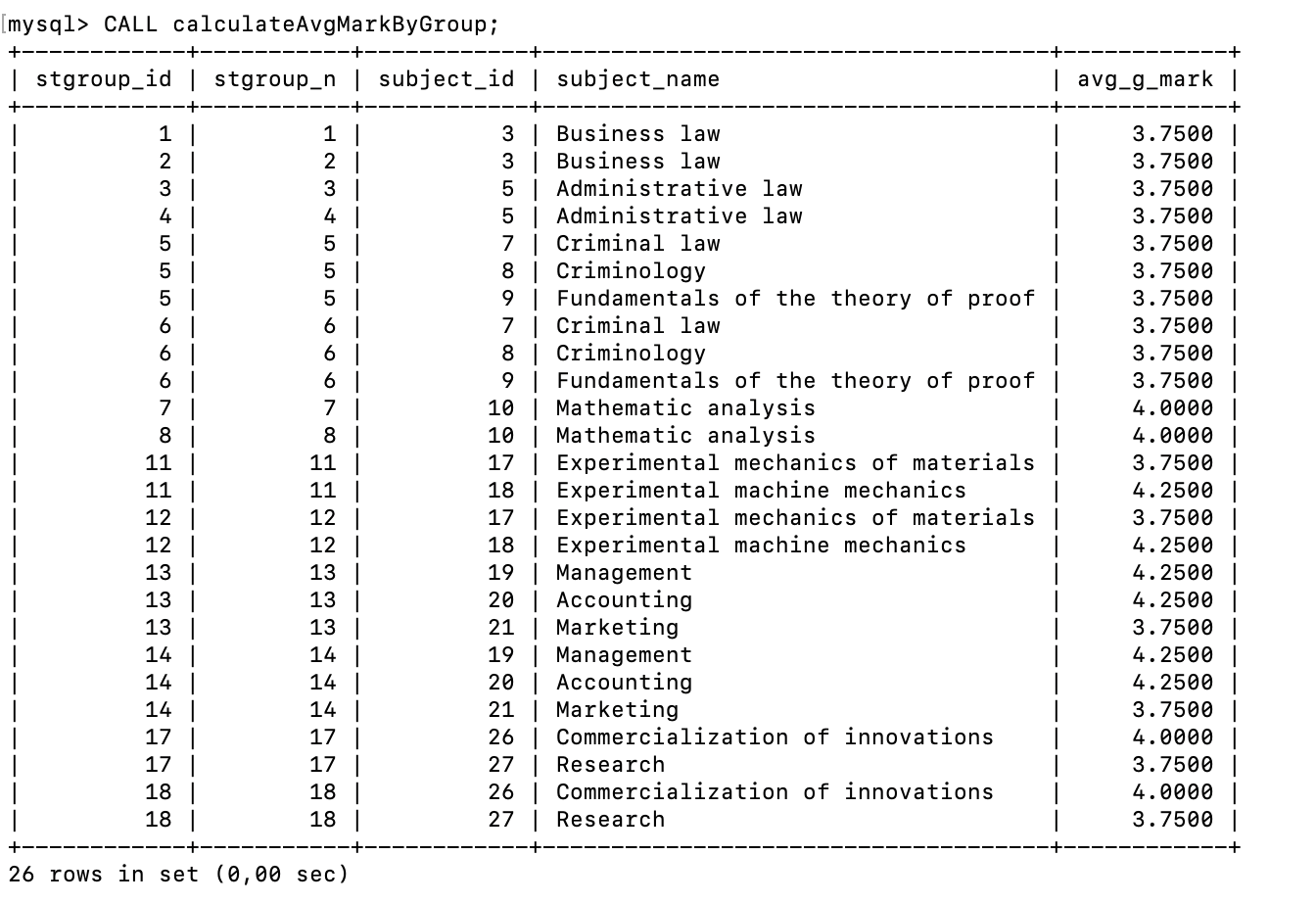
GROUP BY stgroup.stgroup\_id, stgroup.stgroup\_n, subject.subject\_id, subject.subject\_name

HAVING AVG(evaluation\_statement.mark) > (SELECT AVG(evaluation\_statement.mark) FROM evaluation\_statement)

ORDER BY stgroup.stgroup\_id, stgroup.stgroup\_n, subject.subject\_id, subject.subject\_name;

END//

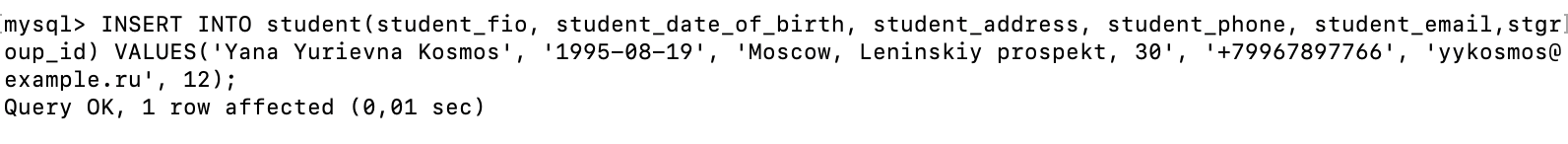
DELIMITER ;

CALL calculateAvgMarkByGroup;

**- вставьте записи о новом студенте с его личной информацией: ФИО, дата рождения, контактные данные и др.**

**Если запросом:**

INSERT INTO student(student\_fio, student\_date\_of\_birth, student\_address, student\_phone, student\_email,stgroup\_id) VALUES('Yana Yurievna Kosmos', '1995-08-19', 'Moscow, Leninskiy prospekt, 30', '+79967897766', 'yykosmos@example.ru', 12);



**Если хранимой процедурой:**

DELIMITER //

CREATE PROCEDURE insert\_student (

IN student\_fio VARCHAR(100),

IN student\_date\_of\_birth DATE,

IN student\_address VARCHAR(100),

IN student\_phone VARCHAR(20),

student\_email VARCHAR(30),

IN stgroup\_id INT

)

BEGIN

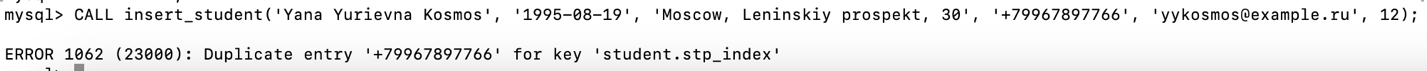
INSERT INTO student (student\_fio, student\_date\_of\_birth, student\_address, student\_phone, student\_email, stgroup\_id)

VALUES (student\_fio, student\_date\_of\_birth, student\_address, student\_phone, student\_email, stgroup\_id);

END//

DELIMITER ;

CALL insert\_student('Yana Yurievna Kosmos', '1995-08-19', 'Moscow, Leninskiy prospekt, 30', '+79967897766', 'yykosmos@example.ru', 12);

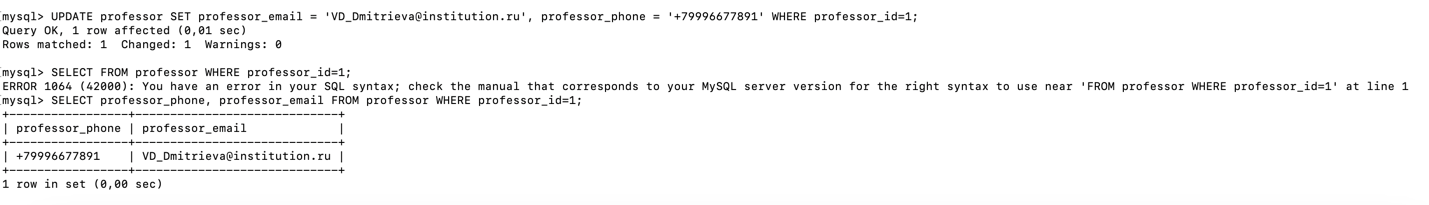


*Те же данные не внести из-за механизма защиты дублирования записей (stp\_index).*

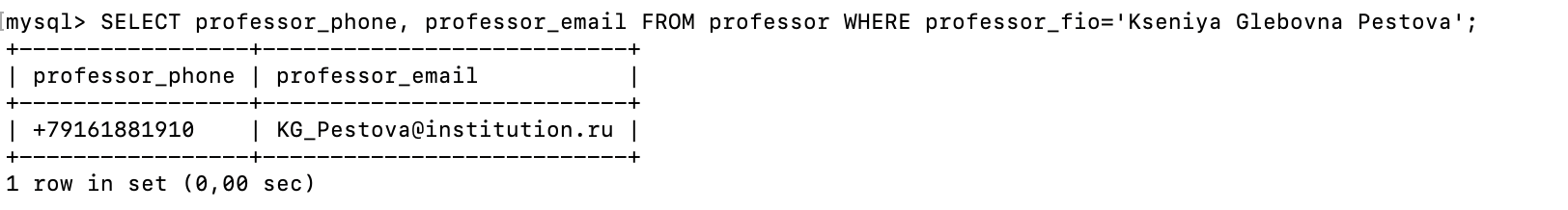
- **возможность обновления контактной информации преподавателя, например, электронной почты или номера телефона, на основе его идентификационного номера или ФИО:**

**Если запросом:**

UPDATE professor SET professor\_email = 'VD\_Dmitrieva@institution.ru', professor\_phone = '+79996677891' WHERE professor\_id=1;



UPDATE professor SET professor\_email = 'KG\_Pestova@institution.ru', professor\_phone = '+79161881910' WHERE professor\_fio='Kseniya Glebovna Pestova';



**Если хранимой процедурой:**

DELIMITER //

CREATE PROCEDURE UpdateProfessor(

IN professor\_id INT,

IN professor\_email VARCHAR(30),

IN professor\_phone VARCHAR(20))

BEGIN

UPDATE professor

SET professor\_email = professor\_email,

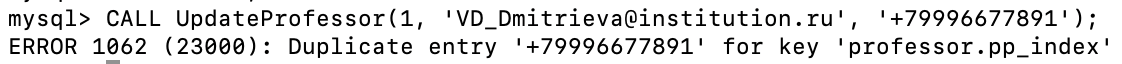
professor\_phone = professor\_phone

WHERE professor\_id = professor\_id;

END //

DELIMITER ;

CALL UpdateProfessor(1, 'VD\_Dmitrieva@institution.ru', '+79996677891');



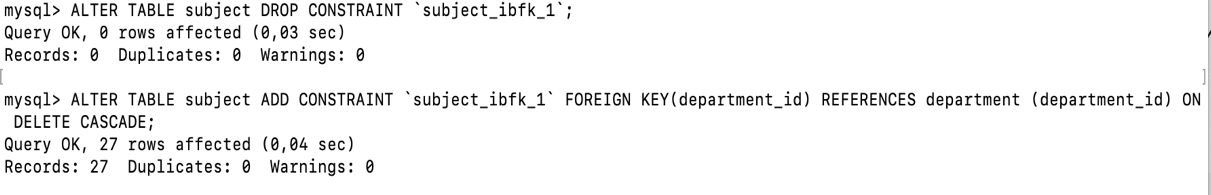
*Те же данные не внести из-за механизма защиты от дублирования записей (pp\_index).*

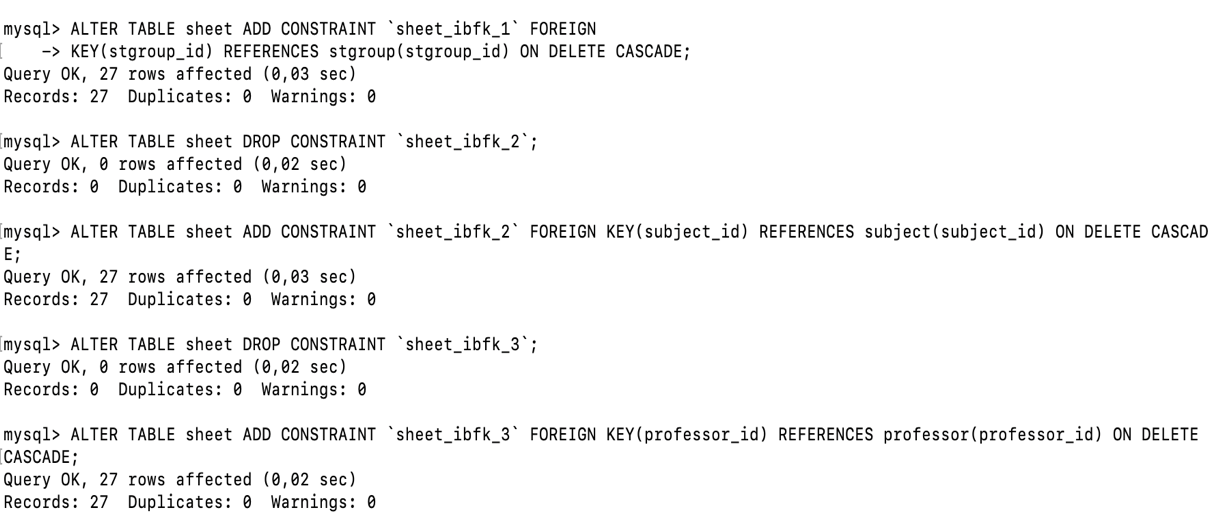
**-возможность удаления записи о предмете, который больше не преподают в учебном заведении. Учтите возможные зависимости, такие как оценки студентов по этому предмету;**

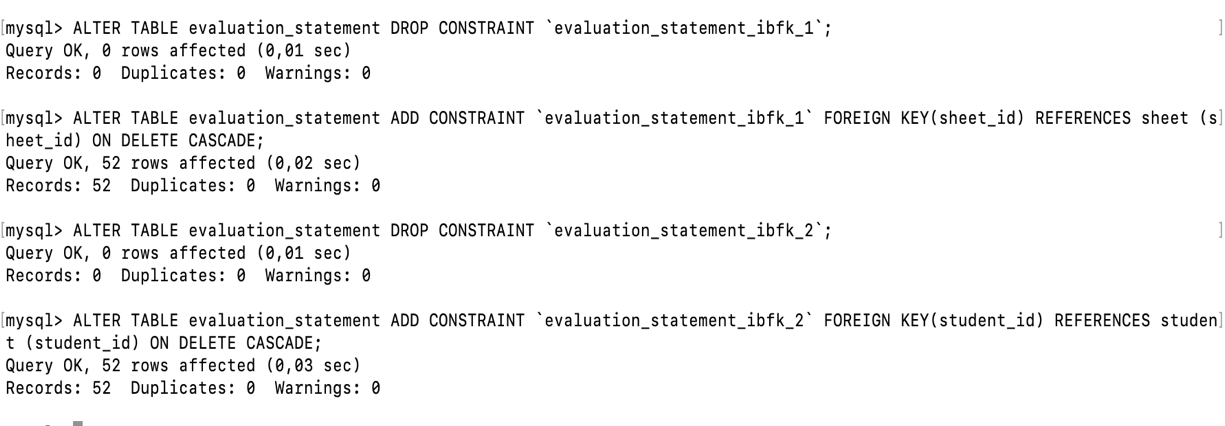
**Если запросом:**

Добавлено ON DELETE CASCADE к внешним ключам.

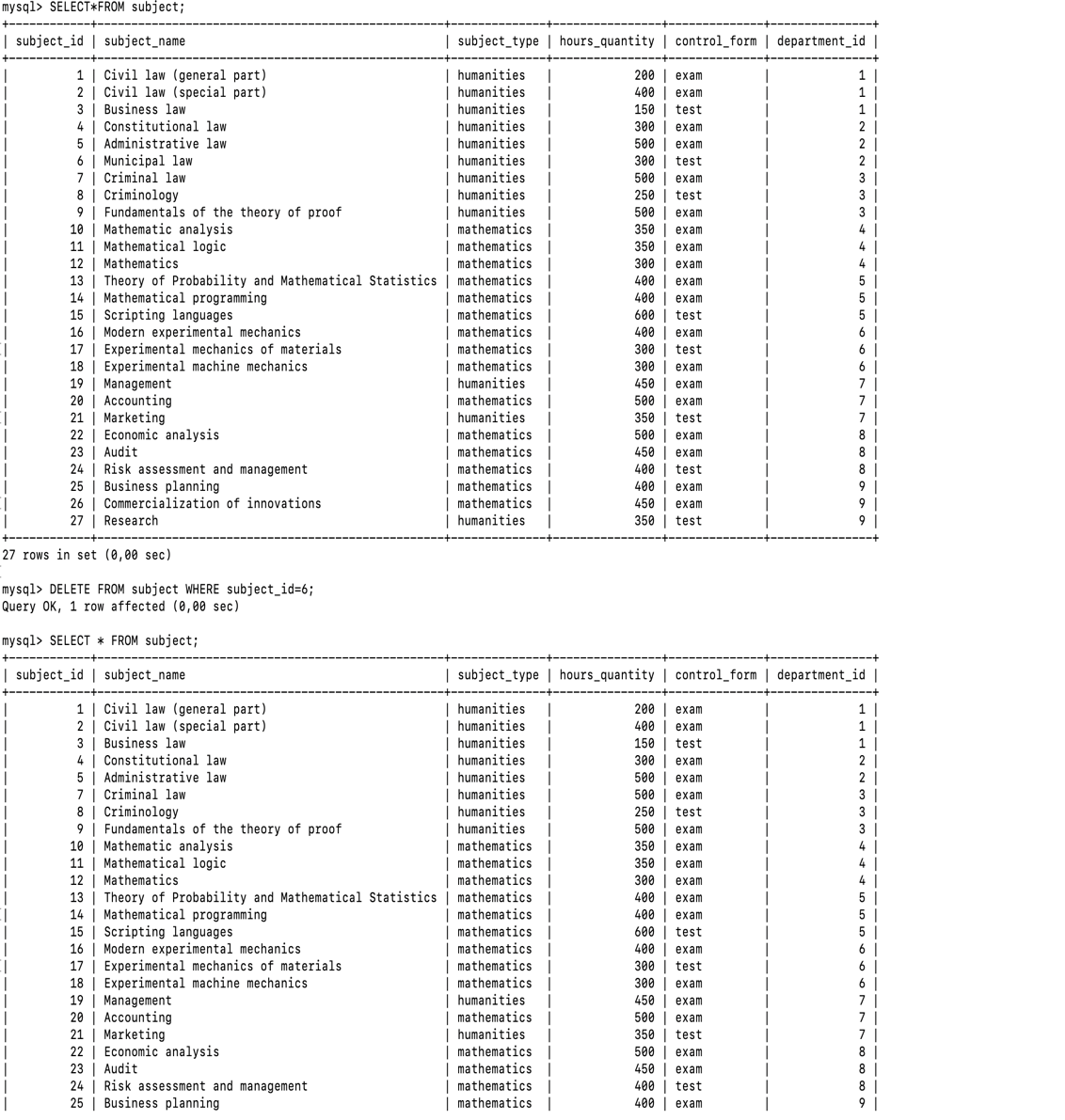
Например, ALTER TABLE sheet DROP CONSTRAINT `sheet\_ibfk\_1`;

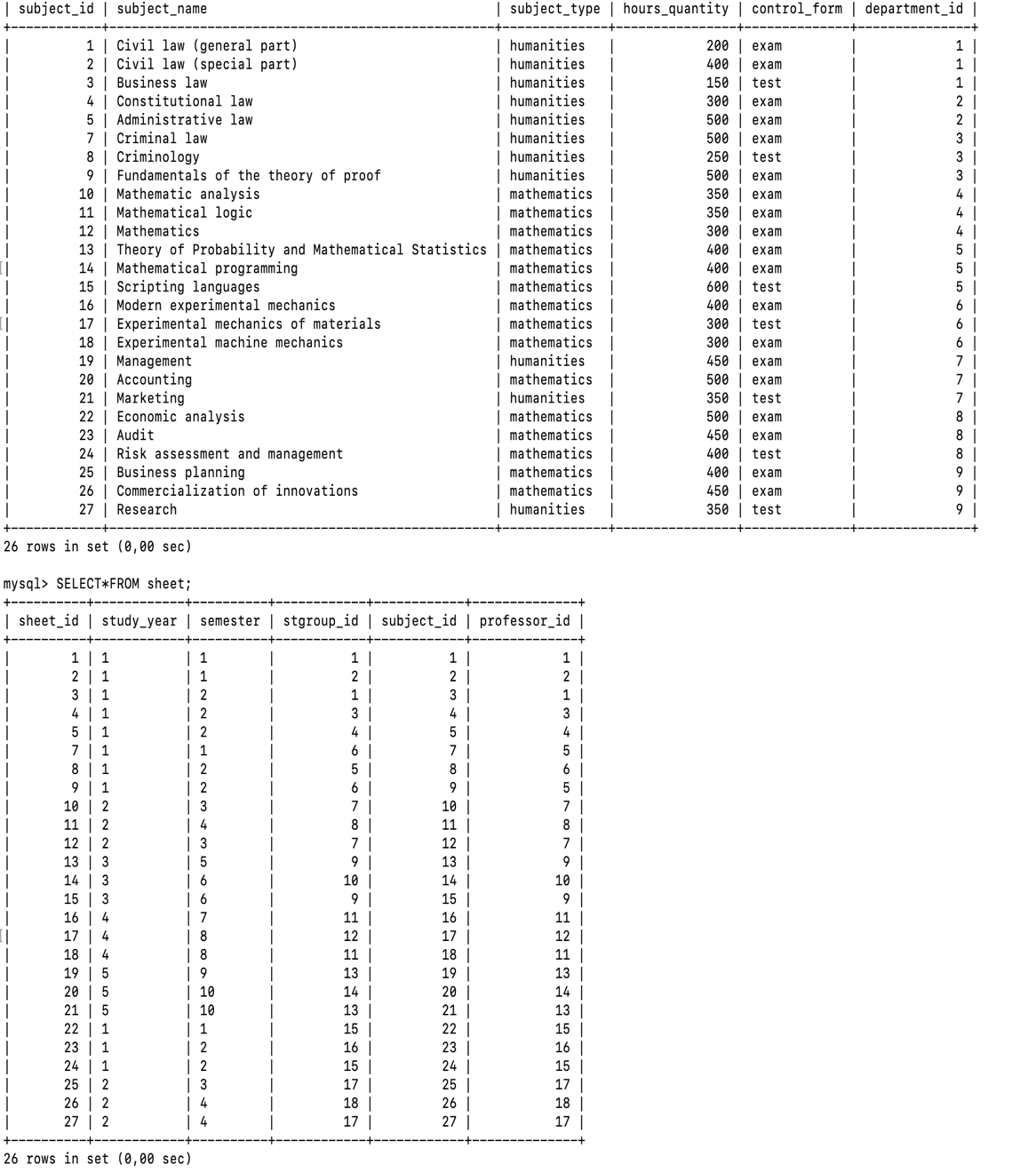
ALTER TABLE sheet ADD CONSTRAINT `sheet\_ibfk\_1` FOREIGN KEY(stgroup\_id) REFERENCES stgroup(stgroup\_id) ON DELETE CASCADE;

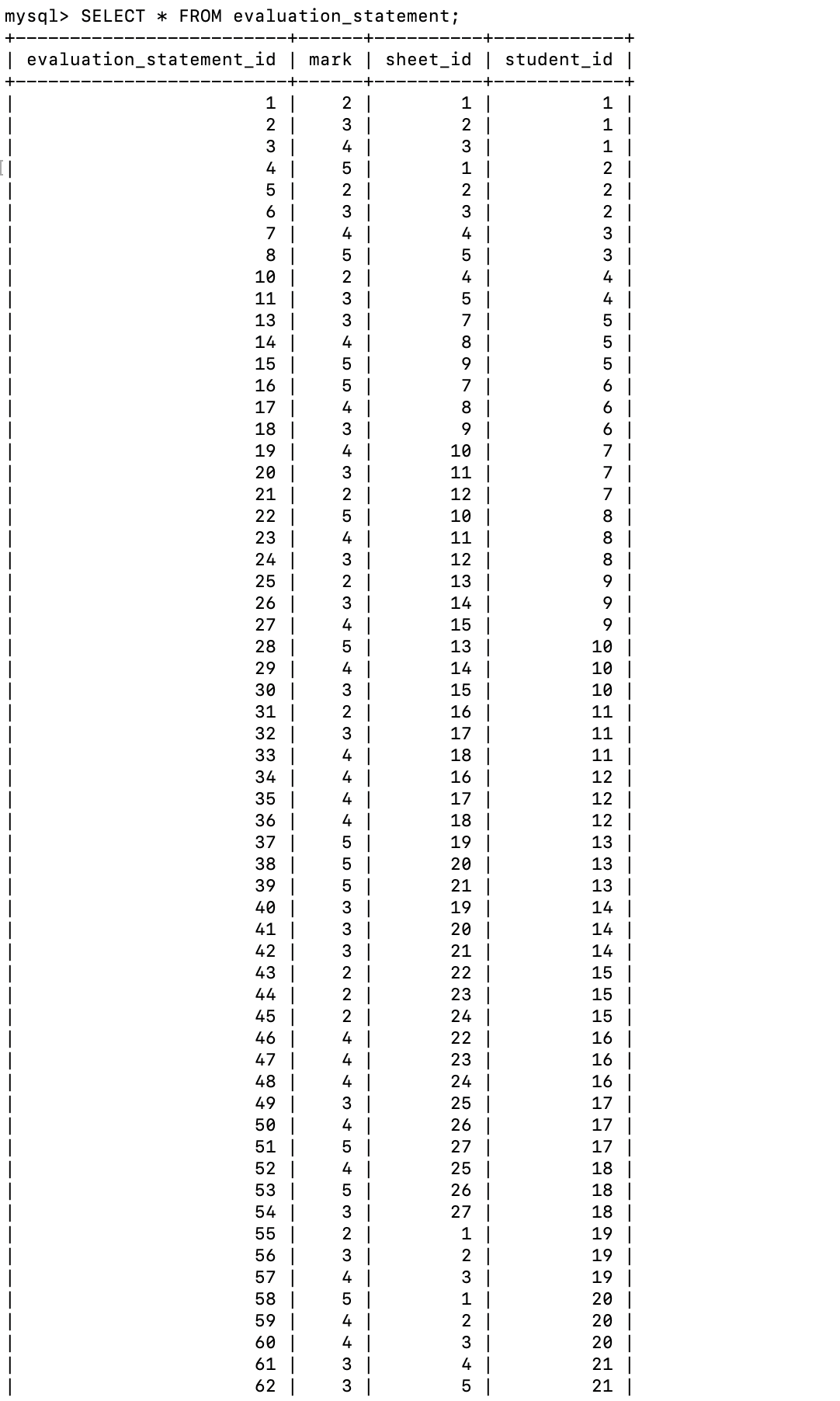


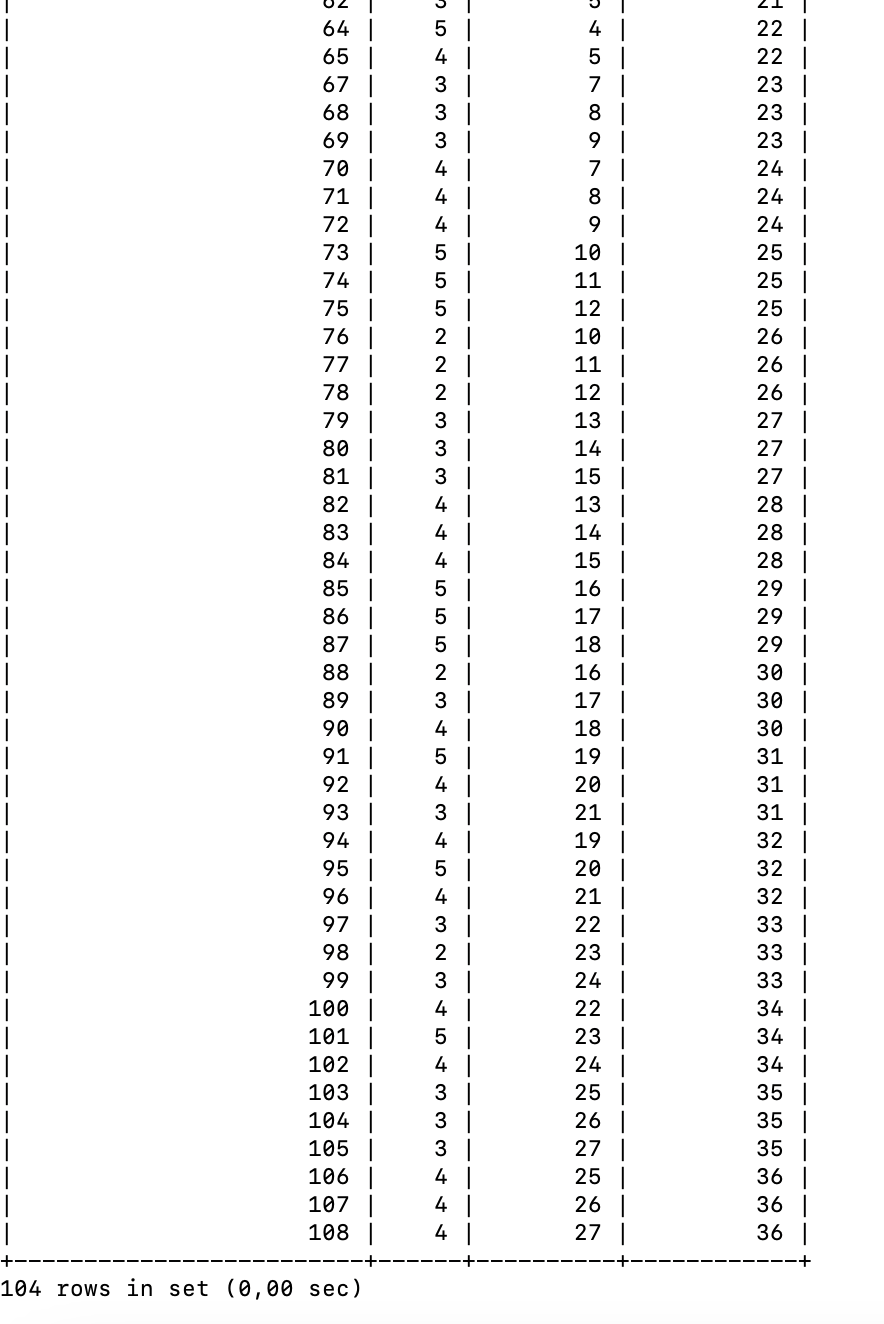


**Удаление:** DELETE FROM subject WHERE subject\_id=6;







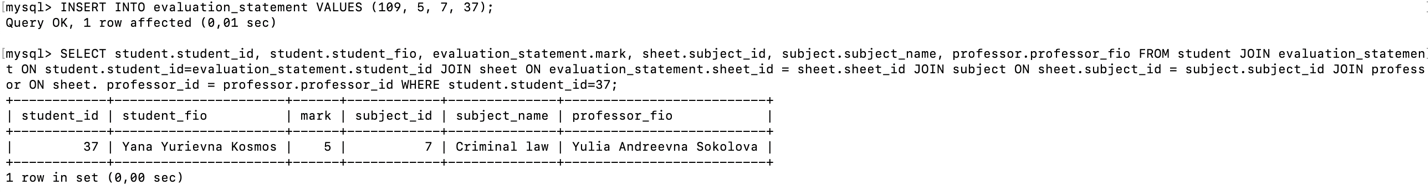
****

**-возможность вставки новой записи об оценке, выставленной студенту по определённому предмету, с указанием даты, преподавателя и полученной оценки.**

**Если запросом:**

INSERT INTO evaluation\_statement VALUES (109, 5, 7, 37);

SELECT student.student\_id, student.student\_fio, evaluation\_statement.mark, sheet.subject\_id, subject.subject\_name, professor.professor\_fio FROM student JOIN evaluation\_statement ON student.student\_id=evaluation\_statement.student\_id JOIN sheet ON evaluation\_statement.sheet\_id = sheet.sheet\_id JOIN subject ON sheet.subject\_id = subject.subject\_id JOIN professor ON sheet. professor\_id = professor.professor\_id WHERE student.student\_id=37;

****

**Если хранимой процедурой:**

*Сначала вставка оценки для студента, затем вывод данных о предмете, преподавателе, студенте и оценке. В текущей версии базы данных нет дат оценок.*

DELIMITER //

CREATE PROCEDURE AddMark(IN mark INT, IN sheet\_id INT, IN student\_id INT)

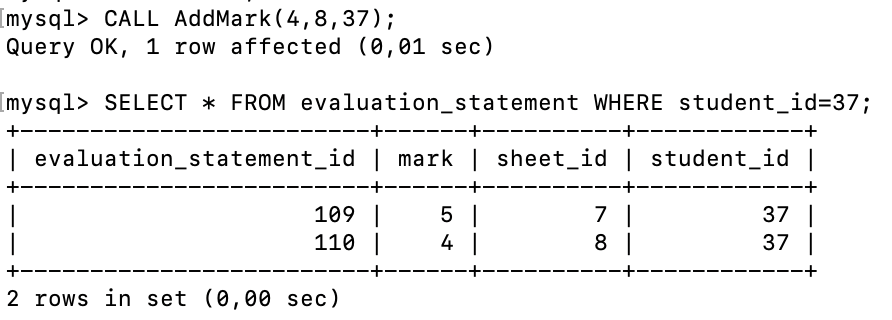
BEGIN

INSERT INTO evaluation\_statement (mark, sheet\_id, student\_id) VALUES (mark, sheet\_id, student\_id);

END //

DELIMITER ;

CALL AddMark(4,8,37);



DELIMITER //

CREATE PROCEDURE NewMarkData(IN student\_id INT)

BEGIN

SELECT

student.student\_id,

student.student\_fio,

evaluation\_statement.mark,

sheet.subject\_id,

subject.subject\_name,

professor.professor\_fio

FROM

student

JOIN evaluation\_statement ON student.student\_id = evaluation\_statement.student\_id

JOIN sheet ON evaluation\_statement.sheet\_id = sheet.sheet\_id

JOIN subject ON sheet.subject\_id = subject.subject\_id

JOIN professor ON sheet.professor\_id = professor.professor\_id

WHERE

student.student\_id = student\_id;

END //

DELIMITER ;

CALL NewMarkData(37);

