**Cодержание урока**

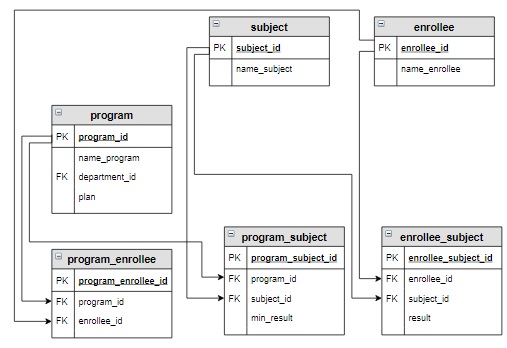
В данном уроке с помощью запросов корректировки данных для базы данных «Абитуриент» формируется список абитуриентов, рекомендованных к зачислению в университет:

* [создается таблица с суммой баллов абитуриентов по предметам ЕГЭ в соответствии с поданными заявлениями](https://stepik.org/lesson/310420/step/2?unit=292726);
* [из таблицы удаляются абитуриенты, если они не набрали минимального балла по предмету, необходимому для поступления на образовательную программу](https://stepik.org/lesson/310420/step/3?unit=292726);
* [абитуриентам, у которых есть медаль или значок ГТО, добавляются дополнительные баллы](https://stepik.org/lesson/310420/step/4?unit=292726);
* [абитуриенты сортируются в соответствии с набранными баллами по каждой образовательной программе](https://stepik.org/lesson/310420/step/5?unit=292726);
* формируется список абитуриентов, рекомендованных к зачислению ( [вставляется столбец для нумерации](https://stepik.org/lesson/310420/step/6?unit=292726), [осуществляется нумерация студентов по образовательной программе](https://stepik.org/lesson/310420/step/7?unit=292726), [выбираются абитуриенты с наибольшими баллами в соответствии с планом набора](https://stepik.org/lesson/310420/step/8?unit=292726)).

## Задание

Создать вспомогательную таблицу **applicant**,  куда включить **id** образовательной программы, **id** абитуриента, сумму баллов абитуриентов (столбец **itog**) в отсортированном сначала по **id** образовательной программы, а потом по убыванию суммы баллов виде (использовать [запрос](https://stepik.org/lesson/310418/step/10?unit=292724) из предыдущего урока).

**Фрагмент логической схемы базы данных:**



**Пояснение**

**Связанные шаги**

**Текст задания**(чтобы не прокручивать страницу) :

*Создать вспомогательную таблицу****applicant****,  куда включить****id****образовательной программы,****id****абитуриента, сумму баллов абитуриентов (столбец****itog****) в отсортированном сначала по****id****образовательной программы, а потом по убыванию суммы баллов виде (использовать*[*запрос*](https://stepik.org/lesson/310418/step/10?unit=292724)*из предыдущего урока).*

**Результат**

1

CREATE TABLE applicant AS

2

SELECT program\_enrollee.program\_id,

3

      program\_enrollee.enrollee\_id,

4

        SUM(result) AS 'itog'

5

FROM

6

  /\* program

7

    JOIN program\_enrollee

8

    ON program.program\_id = program\_enrollee.program\_id

9

      JOIN enrollee

10

      ON enrollee.enrollee\_id = program\_enrollee.enrollee\_id

11

        JOIN enrollee\_subject

12

        ON enrollee.enrollee\_id = enrollee\_subject.enrollee\_id

13

          JOIN subject

14

          ON subject.subject\_id = enrollee\_subject.subject\_id

15

            JOIN program\_subject

16

            ON subject.subject\_id = program\_subject.subject\_id

17

              AND program.program\_id = program\_subject.program\_id

18

              Option 2\*/

19

              program\_enrollee

20

               JOIN enrollee\_subject

21

                ON program\_enrollee.enrollee\_id = enrollee\_subject.enrollee\_id

22

                 JOIN program\_subject

23

                  ON program\_subject.program\_id = program\_enrollee.program\_id

24

                   AND program\_subject.subject\_id = enrollee\_subject.subject\_id

25

GROUP BY 1, 2

26

ORDER BY 1, 3 DESC;

27

​

28

SELECT \* FROM applicant;

## Задание

Из таблицы **applicant**, созданной на предыдущем шаге, удалить записи, если абитуриент на выбранную образовательную программу не набрал минимального балла хотя бы по одному предмету (использовать [запрос](https://stepik.org/lesson/310418/step/11?unit=292724) из предыдущего урока).

**Фрагмент логической схемы базы данных:**

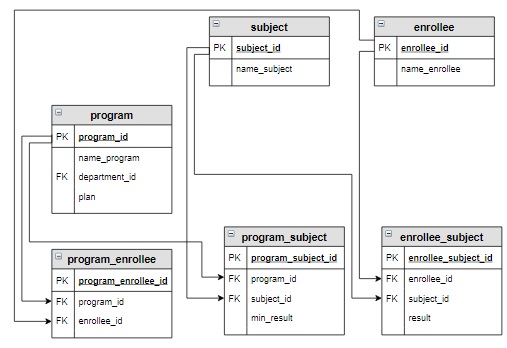
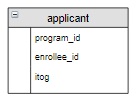


Таблица **applicant**:



**Пояснение**

**Связанные шаги**

**Текст задания**(чтобы не прокручивать страницу) :

*Из таблицы****applicant****,  созданной на предыдущем шаге, удалить записи, если абитуриент на выбранную образовательную программу не набрал минимального балла хотя бы по одному предмету (использовать*[*запрос*](https://stepik.org/lesson/310418/step/11?unit=292724)*из предыдущего урока).*

1

DELETE FROM applicant

2

USING applicant JOIN program\_subject USING(program\_id)

3

               JOIN enrollee\_subject USING(enrollee\_id, subject\_id)

4

WHERE result < min\_result;

5

​

6

DELETE FROM applicant

7

WHERE (program\_id, enrollee\_id) IN

8

(

9

   SELECT program.program\_id, enrollee.enrollee\_id

10

   FROM program

11

    JOIN program\_enrollee

12

     ON program.program\_id = program\_enrollee.program\_id

13

      JOIN enrollee

14

       ON enrollee.enrollee\_id = program\_enrollee.enrollee\_id

15

        JOIN enrollee\_subject

16

         ON enrollee.enrollee\_id = enrollee\_subject.enrollee\_id

17

          JOIN subject

18

           ON subject.subject\_id = enrollee\_subject.subject\_id

19

            JOIN program\_subject

20

             ON subject.subject\_id = program\_subject.subject\_id

21

              AND program.program\_id = program\_subject.program\_id

22

   WHERE result < min\_result

23

);

24

​

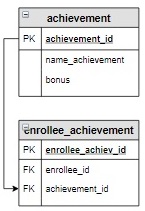
25

SELECT \* FROM applicant;

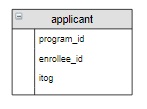
## Задание

Повысить итоговые баллы абитуриентов в таблице **applicant** на значения дополнительных баллов (использовать [запрос](https://stepik.org/lesson/310418/step/7?unit=292724) из предыдущего урока).

**Фрагмент логической схемы базы данных:**



**Структура корректируемой таблицы:**



**Пояснение**

**Связанные шаги**

**Текст задания**(чтобы не прокручивать страницу) :

*Повысить итоговые баллы абитуриентов в таблице****applicant****на значения дополнительных баллов (использовать*[*запрос*](https://stepik.org/lesson/310418/step/7?unit=292724)*из предыдущего урока).*

**Результат**

1

UPDATE

2

  applicant, (SELECT enrollee.enrollee\_id, SUM(bonus) AS 'Бонус'

3

   FROM

4

        enrollee

5

         JOIN enrollee\_achievement

6

          ON enrollee.enrollee\_id = enrollee\_achievement.enrollee\_id

7

           JOIN achievement

8

            ON achievement.achievement\_id = enrollee\_achievement.achievement\_id

9

             GROUP BY enrollee.enrollee\_id) AS wert

10

SET applicant.itog = applicant.itog + wert.Бонус

11

WHERE wert.enrollee\_id=applicant.enrollee\_id;

12

​

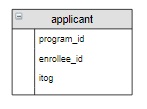
13

SELECT \* FROM applicant;

## Задание

Поскольку при добавлении дополнительных баллов, абитуриенты по каждой образовательной программе могут следовать не в порядке убывания суммарных баллов, необходимо создать новую таблицу **applicant\_order** на основе таблицы **applicant**.При создании таблицы данные нужно отсортировать сначала по **id** образовательной программы, потом по убыванию итогового балла. А таблицу **applicant**, которая была создана как вспомогательная, необходимо удалить.

**Структура корректируемой таблицы:**



**Пояснение**

**Связанные шаги**

**Текст задания**(чтобы не прокручивать страницу) :

*Поскольку при добавлении дополнительных баллов, абитуриенты по каждой образовательной программе могут следовать не в порядке убывания суммарных баллов, необходимо создать новую таблицу****applicant\_order****на основе таблицы****applicant****.**При создании таблицы данные нужно отсортировать сначала по****id****образовательной программы, потом по убыванию итогового балла. А таблицу****applicant****, которая была создана как вспомогательная, необходимо удалить.*

**Результат**

1

CREATE TABLE applicant\_order AS

2

SELECT program\_id, enrollee\_id, itog FROM applicant

3

ORDER BY 1, 3 DESC;

4

​

5

SELECT \* FROM applicant\_order;

6

​

7

DROP TABLE applicant;

## Зачисление студентов

Рассмотрим, как происходит формирование списка абитуриентов, проходящих по конкурсу на образовательные программы.

В таблицу **applicant\_order** для пояснения включены столбцы «План набора» и «Результат». Каждая программа имеет свой план набора, например, план набора образовательной программы с**id** 1 – 2 человека.  Проходящими по конкурсу считаются первые два человека в списке  отсортированных по итоговому баллу абитуриентов, подавших заявление на  образовательную программу. Это абитуриенты с **id** 3 и 2. Аналогично отбираются абитуриенты на остальные образовательные программы. В таблице все проходящие по конкурсу выделены зеленым цветом.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **program\_id** | **план набора** | **enrollee\_id** | **itog** | **Результат** |
| 1 | 2 | 3 | 235 | рекомендован к зачислению |
| 1 | 2 | 2 | 226 | рекомендован к зачислению |
| 1 | 2 | 1 | 219 |  |
| 2 | 1 | 6 | 276 | рекомендован к зачислению |
| 2 | 1 | 3 | 235 |  |
| 2 | 1 | 2 | 226 |  |
| 3 | 2 | 6 | 270 | рекомендован к зачислению |
| 3 | 2 | 4 | 239 | рекомендован к зачислению |
| 3 | 2 | 5 | 200 |  |
| 4 | 3 | 6 | 270 | рекомендован к зачислению |
| 4 | 3 | 3 | 247 | рекомендован к зачислению |
| 4 | 3 | 5 | 200 | рекомендован к зачислению |

Для отбора рекомендованных к зачислению студентов можно разработать различные алгоритмы. Мы реализуем несколько SQL запросов. В первом запросе вставим в таблицу **applicant\_order** новый столбец для последовательной нумерации строк.

Для изменения структуры таблицы используется оператор ALTER TABLE. С его помощью можно вставить новый столбец, удалить существующий, переименовать столбец и пр.

Для вставки нового столбца используется SQL запросы:

ALTER TABLE таблица ADD имя\_столбца тип; - вставляет столбец после последнего

ALTER TABLE таблица ADD имя\_столбца тип FIRST; - вставляет столбец перед первым

ALTER TABLE таблица ADD имя\_столбца тип AFTER имя\_столбца\_1; - вставляет столбец после укзанного столбца

Для удаления столбца используется SQL запросы:

ALTER TABLE таблица DROP COLUMN имя\_столбца; - удаляет столбец с заданным именем

ALTER TABLE таблица DROP имя\_столбца; - ключевое слово COLUMN не обязательно указывать

ALTER TABLE таблица DROP имя\_столбца,

DROP имя\_столбца\_1; - удаляет два столбца

Для переименования столбца используется  запрос (тип данных указывать обязательно):

ALTER TABLE таблица CHANGE имя\_столбца новое\_имя\_столбца ТИП ДАННЫХ;

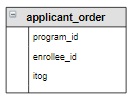
Для изменения типа  столбца используется запрос (два раза указывать имя столбца обязательно):

ALTER TABLE таблица CHANGE имя\_столбца имя\_столбца НОВЫЙ\_ТИП\_ДАННЫХ;

## Задание

Включить в таблицу **applicant\_order** новый столбец **str\_id** целого типа , расположить его перед первым.

**Структура корректируемой таблицы:**



**Результат**

1

ALTER TABLE applicant\_order ADD str\_id int FIRST; /\*вставляет столбец перед первым\*/

2

SELECT \* FROM applicant\_order;

3

ALTER TABLE applicant\_order ADD str1 varchar(50); /\*вставляет столбец после последнего\*/

4

SELECT \* FROM applicant\_order;

5

ALTER TABLE applicant\_order ADD str2 int AFTER str1; /\*вставляет столбец после укзанного столбца\*/

6

SELECT \* FROM applicant\_order;

7

ALTER TABLE applicant\_order DROP COLUMN str2; /\*удаляет столбец с заданным именем\*/

8

SELECT \* FROM applicant\_order;

9

ALTER TABLE applicant\_order DROP str1,

10

                   DROP str\_id; /\* удаляет два столбца\*/

11

SELECT \* FROM applicant\_order;

12

ALTER TABLE applicant\_order CHANGE itog itog2 int;/\*Для переименования столбца используется                                                                           запрос (тип данных указывать обязательно)\*/

13

SELECT \* FROM applicant\_order;

14

ALTER TABLE applicant\_order CHANGE itog2 itog2 varchar(50);/\*Для изменения типа столбца используется                                                                     запрос (два раза указывать имя столбца обязательно)\*/

15

SELECT \* FROM applicant\_order;

оператора IF выглядит следующим образом:

IF(*condition***,** *set\_this\_value\_if\_TRUE***,** *set\_this\_value\_if\_FALSE*)

Где находится AND конечно же является важным.

IF(*condition***,** *set\_this\_value\_if\_TRUE AND do\_something\_if\_TRUE***,** *set\_this\_value\_if\_FALSE*) - случай 1

IF(*condition***,** *set\_this\_value\_if\_TRUE***,** *set\_this\_value\_if\_FALSE AND do\_something\_if\_FALSE*) - случай 2

В первом случае, выражение после AND (как и выражение перед AND) сработает, если условие выполнится.

Во втором случае, выражение после AND (как и выражение перед AND) сработает, если условие НЕ выполнится.

**Нумерация строк**

Номер строки в таблице или запросе в некоторых версиях SQL можно получить с помощью оконной функции  **row\_number()**.  Когда создавался этот шаг версии SQL на платформе Stepik эта функция не поддерживалась. Сейчас версию изменили, оконные функции рассматриваются в следующем уроке.

На этом шаге нумерацию реализуем с помощью переменных.  Переменные задаются с помощью ключевого слова **SET**,  перед именем указывается символ @. Например, создадим переменную **@row\_num** и присвоим ей значение 1:

SET @row\_num := 1;

Теперь эту переменную можно использовать в запросах,  кроме того в запросах можно изменить ее значение.

**Пример**

Пронумеруем записи в таблице **applicant\_order**.

*Запрос:*

SET @row\_num := 0;

SELECT \*, (@row\_num := @row\_num + 1) AS str\_num

FROM applicant\_order;

*Результат:*

+------------+-------------+------+---------+

| program\_id | enrollee\_id | itog | str\_num |

+------------+-------------+------+---------+

| 1 | 3 | 235 | 1 |

| 1 | 2 | 226 | 2 |

| 1 | 1 | 219 | 3 |

| 2 | 6 | 276 | 4 |

| 2 | 3 | 235 | 5 |

| 2 | 2 | 226 | 6 |

| 3 | 6 | 270 | 7 |

| 3 | 4 | 239 | 8 |

| 3 | 5 | 200 | 9 |

| 4 | 6 | 270 | 10 |

| 4 | 3 | 247 | 11 |

| 4 | 5 | 200 | 12 |

+------------+-------------+------+---------+

**Пояснение**

**Пример**

Создадим нумерацию, которая начинается заново для каждой образовательной программы. Для этого можно использовать алгоритм, в котором в переменную **@row\_num** заносится 1, если**id** программы в предыдущей записи не равен **id** программы в текущей:

* объявить переменную **@num\_pr**, задать ей начальное значение;
* запомнить **id** образовательной программы для текущей записи в переменной **@num\_pr**;
* для следующей записи сравнить значение переменной **@num\_pr** с **id** образовательной программы;
* если они равны, то продолжить нумерацию **@row\_num := @row\_num + 1**;
* в противном случае начать нумерацию снова, для этого установить **@row\_num := 1**.

*Запрос:*

SET @num\_pr := 0;

SET @row\_num := 1;

SELECT \*,

if(program\_id = @num\_pr, @row\_num := @row\_num + 1, @row\_num := 1) AS str\_num,

@num\_pr := program\_id AS add\_var

from applicant\_order;

*Результат:*

+------------+-------------+------+---------+---------+

| program\_id | enrollee\_id | itog | str\_num | add\_var |

+------------+-------------+------+---------+---------+

| 1 | 3 | 235 | 1 | 1 |

| 1 | 2 | 226 | 2 | 1 |

| 1 | 1 | 219 | 3 | 1 |

| 2 | 6 | 276 | 1 | 2 |

| 2 | 3 | 235 | 2 | 2 |

| 2 | 2 | 226 | 3 | 2 |

| 3 | 6 | 270 | 1 | 3 |

| 3 | 4 | 239 | 2 | 3 |

| 3 | 5 | 200 | 3 | 3 |

| 4 | 6 | 270 | 1 | 4 |

| 4 | 3 | 247 | 2 | 4 |

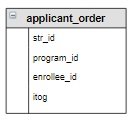
| 4 | 5 | 200 | 3 | 4 |

+------------+-------------+------+---------+---------+

**Задание**

Занести в столбец**str\_id**таблицы **applicant\_order** нумерацию абитуриентов, которая начинается с 1 для каждой образовательной программы.

**Структура корректируемой таблицы:**



**Пояснение**

**Пояснение от Илья Бодня**

**Связанные шаги**

1

SET @num\_pr := 1; /\*создадим переменную @num\_pr и присвоим ей значение 1\*/

2

SET @row\_num := 0; /\*создадим переменную @row\_num и присвоим ей значение 0\*/

3

UPDATE applicant\_order

4

SET str\_id = if(program\_id = @num\_pr, @row\_num := @row\_num + 1, @row\_num := 1

5

               AND @num\_pr := program\_id);

6

SELECT \* FROM applicant\_order;

7

​

8

SET @nr := 1; /\* Переменная для номера строки \*/

9

SET @np := 0; /\* Переменная для номера program\_id \*/

10

​

11

UPDATE applicant\_order

12

  SET str\_id =

13

      IF(program\_id = @np, /\* Если номер program\_id равен значению переменной (для первой строки таблицы 1 = 0, для второй 1 = 1, смотри AND), то: \*/

14

      @nr := @nr + 1, /\* установи str\_id = 1 + 1, а если нет, то: \*/

15

      @nr := 1 /\* установи str\_id = 1 \*/

16

      AND @np := program\_id); /\* а для следующей строки измени значение переменной на номер program\_id \*/

17

​

18

SELECT \*

19

 FROM applicant\_order;

20

​

21

/\*SET @num\_pr := 0;

22

SET @row\_num := 1;

23

UPDATE applicant\_order

24

SET str\_id =

25

  CASE

26

      WHEN program\_id = @num\_pr THEN @row\_num := @row\_num + 1

27

      ELSE @row\_num := 1 AND @num\_pr := program\_id

28

  END;

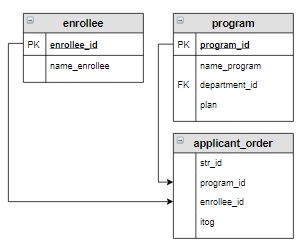
29

SELECT \* FROM applicant\_order;\*/

**Задание**

Создать таблицу **student**,  в которую включить абитуриентов, которые могут быть рекомендованы к зачислению  в соответствии с планом набора. Информацию отсортировать сначала в алфавитном порядке по названию программ, а потом по убыванию итогового балла.

**Фрагмент логической схемы базы данных:**



**Текст задания**(чтобы не прокручивать страницу):

*Создать таблицу****student****,  в которую включить абитуриентов, которые могут быть рекомендованы к зачислению  в соответствии с планом набора. Информацию отсортировать сначала в алфавитном порядке по названию программ, а потом по убыванию итогового балла.*

**Результат**

1

CREATE TABLE student AS

2

SELECT name\_program, name\_enrollee, itog

3

FROM

4

  program

5

    JOIN applicant\_order

6

     ON program.program\_id = applicant\_order.program\_id

7

       JOIN enrollee

8

        ON enrollee.enrollee\_id = applicant\_order.enrollee\_id

9

WHERE str\_id <= plan

10

ORDER BY 1, 3 DESC;

11

​

12

SELECT \* FROM student;