Для выполнения следующих **SQL** запросов [**SELECT**](https://sql-language.ru/select.html) нам необходимо прежде всего изучить структуру таблиц.

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя таблицы** | **Имя поля** | **Тип поля** | **Примечание** |
| **FAKULTET** | KOD\_F | Integer | [PRIMARY KEY](https://sql-language.ru/primary-key.html) |
| NAZV\_F | Char, 30 |  |
| **SPEC** | KOD\_S | Integer | PRIMARY KEY |
| KOD\_F | Integer |  |
| NAZV\_S | Char, 50 |  |
| **STUDENT** | KOD\_STUD | Integer | PRIMARY KEY |
| KOD\_S | Integer |  |
| FAM | Char, 30 |  |
| IM | Char, 15 |  |
| OT | Char, 15 |  |
| STIP | Decimal, 3 |  |
| BALL | Decimal, 3 |  |

**Примеры запросов** SELECT на языке SQL:

**SQL Пример SELECT №1**  
*Выбрать студентов, получающих стипендию, равную 150.*

**SELECT** FAM FROM STUDENT WHERE STIP=150;

С помощью данного **SQL** запроса **SELECT** выбираются все значения из таблицы STUDENT, поле STIP которых строго равно 150.

**SQL Пример SELECT №2**  
*Выбрать студентов, имеющих балл от 82 до 90. Студенты должны быть отсортированы в порядке убывания балла.*

**SELECT** FAM *FROM* STUDENT *WHERE* BALL *BETWEEN* 81 AND 91 *ORDER BY* BALL *DESC*;

Как видно из **SQL** примера, чтобы выбрать студентов, которые имеют балл от 82 до 90, мы используем условие *BETWEEN*. Чтобы отсортировать в убывающем порядке *DESC*.

**SQL Пример SELECT №3**  
*Выбрать студентов, фамилии которых начинаются с буквы «А».*

**SELECT** FAM *FROM* STUDENT *WHERE* FAM *LIKE* ‘А%’;

Для того, чтобы выбрать фамилии, начинающиеся с буквы «А», мы используем оператор **SQL** *LIKE* для поиска значений по образцу.

**SQL SELECT Пример №4**  
*Подсчитать средний балл на каждом факультете.*

**SELECT** NAZV\_F *As* Название, *ROUND*(*AVG*(BALL), 2) As СредБалл *FROM* FAKULTET, SPEC, STUDENT *WHERE* STUDENT.KOD\_S=SPEC.KOD\_S *AND* SPEC.KOD\_F=FAKULTET.KOD\_F *GROUP BY* NAZV\_F;

Пример запроса **SQL SELECT** показывает нам использование функции **SQL** *AVG* для вычисления среднего значения, *ROUND* для округления значения, раздела *GROUP BY* для группировки столбцов.

**SQL SELECT**. Пример №5.  
*Подсчитать количество студентов, обучающихся на каждом факультете. Вывести в запросе название факультета, комментарий – «обучается», количество студентов, комментарий «человек».*

**SELECT** NAZV\_F||’ обучается ‘||*COUNT*(KOD\_STUD) *As*CountStudOnFakultet||’ человек’|| *FROM* FAKULTET, SPEC, STUDENT *WHERE*STUDENT.KOD\_S=SPEC.KOD\_S *AND* SPEC.KOD\_F=FAKULTET.KOD\_F *GROUP BY*NAZV\_F;

**SQL SELECT**. Пример №6.  
*Упорядочить студентов по факультетам, специальностям, фамилиям.*

**SELECT** NAZV\_F, NAZV\_S, FAM FROM FAKULTET, SPEC, STUDENT WHERE STUDENT.KOD\_S=SPEC.KOD\_S AND SPEC.KOD\_F=FAKULTET.KOD\_F *ORDER BY*NAZV\_F, NAZV\_S, FAM;

**SQL SELECT**. Пример №7.  
*Определить, кто учится на специальности, к которой относится студент «Петров».*

**SELECT** FAM FROM STUDENT WHERE STUDENT.KOD\_S=(**SELECT** KOD\_S FROM STUDENT WHERE FAM=’Петров’);

В данном **SQL** примере мы используем подзапрос **SQL SELECT**, который возвращает код специальности, на которой учится студент по фамилии Петров.

**SQL SELECT**. Пример №8.  
*Показать, какие специальности встречаются в таблице STUDENT. Дубликаты исключить. Вывести в запросе названия специальностей.*

**SELECT**DISTINCT NAZV\_S FROM SPEC, STUDENT WHERE STUDENT.KOD\_S=SPEC.KOD\_S;

В БД «b\_library**»** создадим таблицу авторов «**tAuthors»** с такими столбцами: AuthorId, AuthorFirstName, AuthorLastName, AuthorAge:

CREATE TABLE tAuthors (  
AuthorId             INT              IDENTITY (1, 1) NOT NULL,  
AuthorFirstName      NVARCHAR (20)    NOT NULL,  
AuthorLastName       NVARCHAR (20)    NOT NULL,  
AuthorAge            INT               NOT NULL   
);

Заполним нашу таблицу таким авторами: Александр Пушкин, Сергей Есенин, Джек Лондон, Шота Руставели и Рабиндранат Тагор. Для этого используем такой SQL запрос:

INSERT tAuthors VALUES  
('Александр', 'Пушкин', '37'),  
('Сергей', 'Есенин', '30'),  
('Джек', 'Лондон', '40'),  
('Шота', 'Руставели', '44'),  
('Рабиндранат', 'Тагор', '80');

**UPDATE**

**UPDATE** – SQL запрос, позволяющий внести изменения или дописывать новую информацию в те записи, которые уже существуют.

Внесем корректировки в шестую запись (**AuthorId = 6**). Значения изменим для полей имени, фамилии и возраста автора.

UPDATE tAuthors SET AuthorFirstName = 'Лев', AuthorLastName='Толстой', AuthorAge = '82' WHERE AuthorId = '6';

Затем, обратимся к БД, чтобы вывести все имеющиеся записи:

SELECT \* FROM tAuthors;

Мы видим изменения информации в записи автора под номером 6.

**DELETE**

**DELETE** – SQL запрос, выполняя который в СУБД производится операция удаления определенной строки из таблицы в БД.

Обратимся к «**tAuthors**» с командой на удаление строки, где **AuthorId** = 5:

DELETE FROM tAuthors WHERE AuthorId = '5';

Чтобы увидеть изменения, снова обратимся к базе для вывода всех записей:

SELECT \* FROM tAuthors;

Мы видим, что запись автора под номером 5 теперь отсутствует в «tAuthors» и, соответственно, не выводится с другими записями.

Воспользуемся нашей предыдущей таблицей «**tAuthors»**и создадим дополнительно еще одну таблицу с книгами этих авторов – «**tBooks»**. В качестве идентификатора авторов книг используем значение AuthorId из «**tAuthors»**, а название книги - BookTitle**.**

CREATE TABLE tBooks (  
BookId               INT              IDENTITY (1, 1) NOT NULL,  
BookTitle            NVARCHAR (20)    NOT NULL,  
Author               INT              NOT NULL  
);

Заполним «**tBooks»**такими книгами:

INSERT tBooks VALUES  
('Руслан и Людмила', '1'),  
('Кавказский пленник', '1'),  
('Евгений Онегин ', '1'),  
('Радуница', '2'),  
('Преображение', '2'),  
('Мартин Иден', '3'),  
('Морской волк', '3'),  
('Белый Клык', '3');

1) Сделаем выборку из БД всех книг, у которых имя автора - «Александр»:

SELECT BookId, BookTitle  
FROM tBooks  
WHERE AuthorId = (SELECT AuthorId FROM tAuthors WHERE AuthorFirstName = 'Александр');

2) Сделаем выборку данных из «**tBooks»**всех книг, авторами которых являются люди, с именами «Александр» или «Сергей»:

SELECT BookTitle  
FROM tBooks  
WHERE AuthorId = SOME (SELECT AuthorId FROM tAuthors   
WHERE AuthorFirstName IN ('Александр', 'Сергей'));

Результатом подзапроса является один столбец величин. Если хотя бы для одного значения V, получаемого из подзапроса, результат операции "<значение выражения> <[оператор сравнения](http://www.sql-tutorial.ru/ru/book_comparison_predicates.html)> V" равняется **TRUE**, то предикат **SOME** также равняется **TRUE**.

 3) Сделаем выборку по книгам из таблицы «**tBooks»**, у которых именами авторов являются НЕ «Сергей» и НЕ «Александр»:

SELECT \*  
FROM tBooks  
WHERE NOT AuthorId= ALL (SELECT AuthorId FROM tAuthors WHERE AuthorFirstName IN ('Александр', 'Сергей'));

Значение предиката ALL будет истинным, если для всех значений V, получаемых из подзапроса, предикат "<значение выражения> <оператор сравнения> V" дает TRUE.

4) Возьмем таблицу «**tBooks»**и сделаем из нее выборку всех книг с указанием как имен, так и фамилий авторов этих книг из «**tAuthors»:**

SELECT tBooks.BookId, tBooks.BookTitle, tAuthors.AuthorFirstName,  
tAuthors.AuthorLastName  
FROM tBooks, tAuthors  
JOIN tAuthors ON tAuthors.AuthorId = tBooks.AuthorId;

Или

SELECT BookId, BookTitle, AuthorFirstName,AuthorLastName  
FROM tBooks, tAuthors  
WHERE tAuthors.AuthorId = tBooks.AuthorId;

Задачи из экзаменационных билетов

1. Составить запрос на языке SQL: Найти код, ФИО и размер зарплаты сотрудников, получающих зарплату в размере от 21000 до 35000 руб. Указать необходимые отношения

SELECT Sotrud\_Id, Fam, Im, Otch, Zarplata

FROM SOTRUDNIK

WHERE Zarplata BETWEEN 20999 AND 35001;

1. Составить запрос на языке SQL: Найти код, ФИО и размер зарплаты сотрудников, получающих зарплату, размер которой не меньше средней зарплаты сотрудников отдела 123. Указать необходимые отношения

SELECT Sotrud\_Id, Fam, Im, Otch, Zarplata

FROM SOTRUDNIK

WHERE Zarplata > (SELECT AVG (Zaplata) As SredZarplata FROM SOTRUDNIK WHERE Otdel=’123’);