# Добро пожаловать на курс по обучению PostgreSQL!

Этот курс направлен на изучение основ работы с языком SQL в системе управления базами данных PostgreSQL. На протяжении курса вы изучите основные команды и приемы работы с данными, такие как выборки, сортировки, объединения данных и использование агрегатных функций. Мы будем работать с учебной базой данных, чтобы изучить каждую тему на практике.

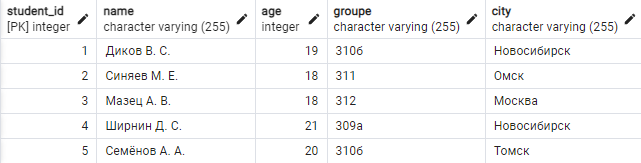
После прохождения курса вы сможете самостоятельно создавать запросы для работы с базами данных, что является ключевым навыком в области программирования и анализа данных.

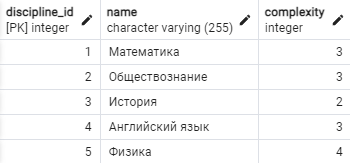
# Описание базы данных

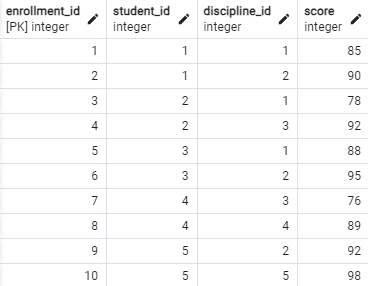
База данных состоит из трех таблиц: students, disciplines и enrollments. Они содержат следующую информацию:

1. Таблица students хранит информацию о студентах: их уникальный идентификатор, имя, возраст, группу и город.  
2. Таблица disciplines содержит информацию о дисциплинах: уникальный идентификатор, название и уровень сложности.  
3. Таблица enrollments связывает студентов с дисциплинами и содержит их оценки.

**Содержание таблиц**:

students:  


disciplines:  


enrollments:  


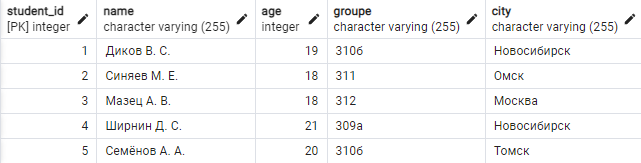
**Урок 1: Команда SELECT**

**Определение:**  
Команда **SELECT** используется для извлечения данных из одной или нескольких таблиц базы данных. Она предоставляет возможность гибкого формирования запросов для получения данных в необходимом формате. **SELECT** — это основная команда в SQL, позволяющая запрашивать данные из базы. Она поддерживает выбор конкретных столбцов, применение арифметических операций, использование агрегатных функций и объединение данных из нескольких таблиц. В сочетании с другими операторами, такими как **WHERE**, **ORDER BY** и **GROUP BY**, **SELECT** становится мощным инструментом для анализа данных.

**Пример:**

****

**Результат:**



*Попробуйте извлечь данные из другой таблицы, например enrollments.*

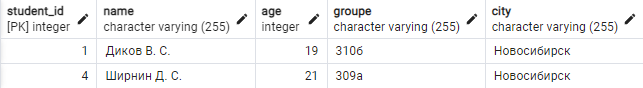
**Урок 2: Условия выборки (WHERE)**

**Определение:**  
Условие **WHERE** используется для фильтрации данных в запросах SQL. Оно ограничивает строки, возвращаемые запросом, в соответствии с заданным критерием. **WHERE** позволяет задавать сложные условия, такие как сравнение значений (**>**, **<**, **=**), текстовые фильтры (**LIKE**), проверки на принадлежность (**IN**) или диапазоны (**BETWEEN**). Эта команда незаменима при работе с большими объемами данных, так как позволяет извлекать только те записи, которые действительно нужны.

**Пример:**

****

**Результат:**

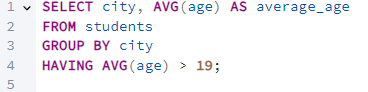
****

*Попробуйте извлечь данные с другими критериями, например выбрать всех, кому меньше 20.*

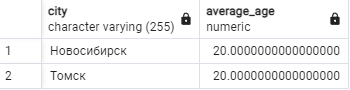
**Урок 3: Условия выборки (HAVING)**

**Определение:**  
**HAVING** используется для фильтрации данных после выполнения группировки в запросах SQL. Команда **HAVING** позволяет наложить условия на результат агрегатных функций (**SUM**, **COUNT**, **AVG**, и т.д.), которые не могут быть обработаны оператором **WHERE**. Используется вместе с **GROUP BY**.

**Пример:**

****

**Результат:**

****

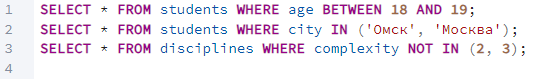
*Попробуйте поменять условия.*

**Урок 4: BETWEEN/IN/NOT IN**

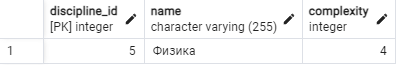
**Определение:**  
Операторы **BETWEEN**, **IN** и **NOT IN** используются для проверки диапазонов значений и принадлежности к списку.

* **BETWEEN** проверяет, попадает ли значение в заданный диапазон.
* **IN** позволяет указать список значений для проверки совпадений.
* **NOT IN** проверяет, не входит ли значение в указанный список.

**Пример:**

****

**Результат:**

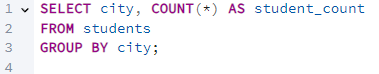
****

*Попробуйте поменять условия.*

**Урок 5: GROUP BY**

**Определение:**  
**GROUP BY** используется для группировки строк с одинаковыми значениями в указанных столбцах. Этот оператор часто используется с агрегатными функциями, такими как **SUM**, **AVG**, **COUNT**, чтобы получить итоговые значения для каждой группы.

**Пример:**

****

**Результат:**

****

*Попробуйте поменять условия или функции и посмотреть, что будет.*

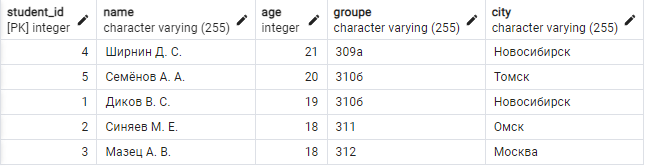
**Урок 6: ORDER BY**

**Определение:**  
**ORDER BY** используется для сортировки результата запроса по одному или нескольким столбцам.  
Можно указать порядок сортировки — по возрастанию (**ASC**, по умолчанию) или по убыванию (**DESC**).

**Пример:**

****

**Результат:**

****

*Попробуйте поменять порядок сортировки и посмотреть, что будет.*

**Урок 7: LIMIT/OFFSET**

**Определение:**  
Операторы **LIMIT** и **OFFSET** используются для ограничения количества возвращаемых строк и указания начальной позиции, с которой начинается выборка.

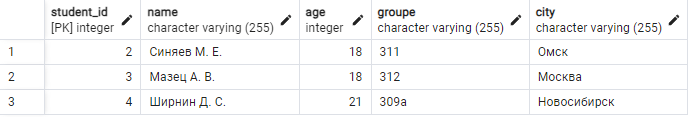
* **LIMIT** задает максимальное количество строк, которое будет возвращено.
* **OFFSET** указывает, сколько строк нужно пропустить перед началом выборки.

Эти операторы полезны для реализации постраничного вывода данных.

**Пример:**

****

**Результат:**

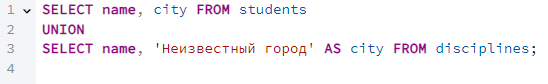
****

*Попробуйте поменять значения функций.*

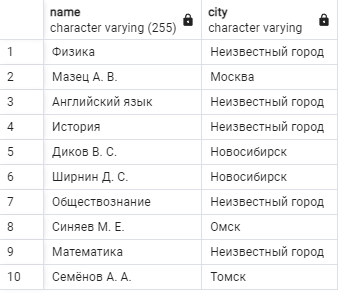
**Урок 8: UNION**

**Определение:**  
Оператор **UNION** используется для объединения результатов двух или более запросов в одну результирующую выборку. **UNION** удаляет дубликаты строк в объединенных результатах. Если требуется сохранить дубликаты, используется **UNION ALL**. Количество и порядок столбцов в объединяемых запросах должны совпадать.

**Пример:**

****

**Результат:**

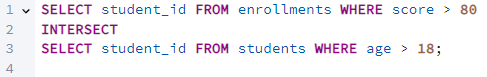
****

*Попробуйте поменять данные.*

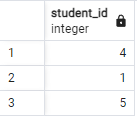
**Урок 9: INTERSECT**

**Определение:**  
Оператор **INTERSECT** возвращает только те строки, которые присутствуют в обоих запросах. **INTERSECT** выполняет пересечение результатов двух запросов, исключая дубли. Это полезно для нахождения общих данных между выборками.

**Пример:**

** **

**Результат:**

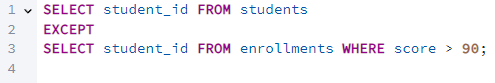
****

*Попробуйте извлечь данные из других таблиц с другими запросами и данными.*

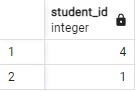
**Урок 10: EXCEPT**

**Определение:**  
Оператор **EXCEPT** возвращает строки, которые присутствуют в первом запросе, но отсутствуют во втором. **EXCEPT** используется для поиска уникальных строк в первой выборке. Дубли исключаются автоматически.

**Пример:**

****

**Результат:**

****

*Попробуйте поменять условия и данные.*

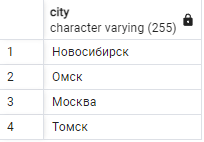
**Урок 11: DISTINCT**

**Определение:**  
**DISTINCT** используется для удаления дубликатов из результата запроса, возвращая только уникальные строки. Этот оператор часто применяется, если требуется найти уникальные значения в столбце или комбинации столбцов.

**Пример:**

****

**Результат:**

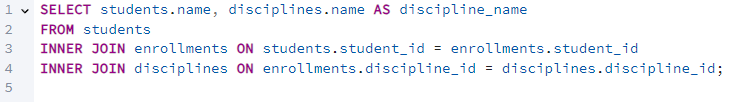
****

*Попробуйте поменять условия выборки.*

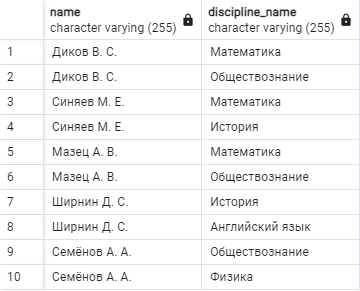
**Урок 12: INNER JOIN**

**Определение:**  
**INNER JOIN** используется для объединения строк из двух таблиц на основе условий, определенных в запросе. Этот тип соединения возвращает строки, которые соответствуют условию соединения, заданному через **ON**.

**Пример:**

****

**Результат:**

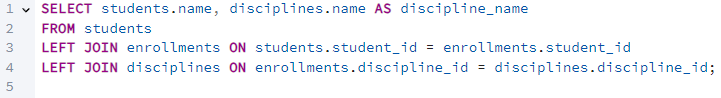
****

*Попробуйте поменять условия объединения и посмотреть, что получится.*

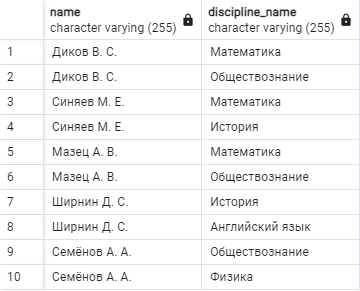
**Урок 13: LEFT JOIN**

**Определение:**  
**LEFT JOIN** возвращает все строки из левой таблицы и соответствующие строки из правой таблицы.  
Если совпадения в правой таблице нет, для таких строк в результирующей выборке устанавливаются значения **NULL**.

**Пример:**

****

**Результат:**

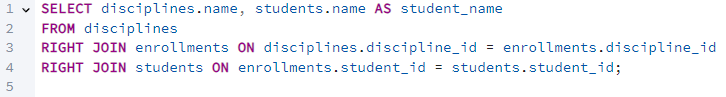
****

*Попробуйте поменять условия объединения и посмотреть, что получится.*

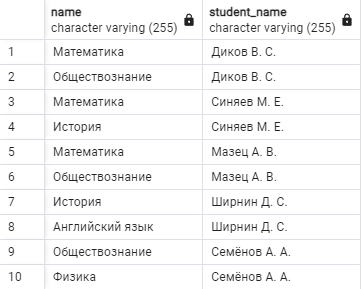
**Урок 14: RIGHT JOIN**

**Определение:**  
**RIGHT JOIN** возвращает все строки из правой таблицы и соответствующие строки из левой таблицы. Если совпадения в левой таблице нет, значения заменяются на **NULL**.

**Пример:**

****

**Результат:**

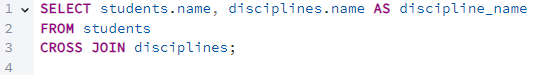


*Попробуйте поменять условия объединения и посмотреть, что получится.*

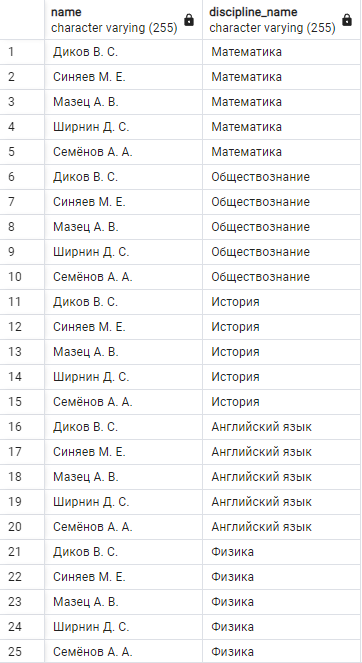
**Урок 15: CROSS JOIN**

**Определение:**  
**CROSS JOIN** создает декартово произведение двух таблиц, объединяя каждую строку из первой таблицы с каждой строкой второй. Этот тип соединения используется редко, так как результат может быть очень большим.

**Пример:**

****

**Результат:**

****

*Попробуйте поменять условия объединения и посмотреть, что получится.*

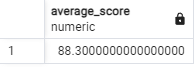
**Урок 16: Функция AVG**

**Определение:**  
**AVG** — это агрегатная функция, которая вычисляет среднее арифметическое значение для указанного столбца. Эта функция используется для анализа числовых данных, например, для определения среднего балла, возраста и т.д. **AVG** игнорирует значения **NULL**.

**Пример:**

****

**Результат:**

****

*Попробуйте посчитать среднее значение других данных, например вычислить средний возраст всех студентов.*

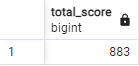
**Урок 17: Функция SUM**

**Определение:**  
**SUM** — это агрегатная функция, которая вычисляет сумму всех значений в указанном столбце.  
Функция применяется к числовым данным и может использоваться для вычисления общих значений, таких как общая сумма баллов, расходов и т.д. **SUM** игнорирует значения **NULL**.

**Пример:**

****

**Результат:**

****

*Попробуйте вычислить общее значение других данных.*

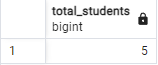
**Урок 18: Функция COUNT**

**Определение:**  
**COUNT** — это агрегатная функция, которая подсчитывает количество строк в результате запроса. Функция **COUNT** может подсчитывать как все строки, так и только строки, не содержащие **NULL** в указанном столбце.

**Пример:**

****

**Результат:**

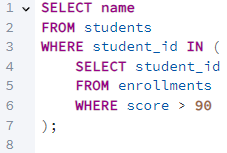
****

*Попробуйте посчитать общее количество других данных, например количество предметов.*

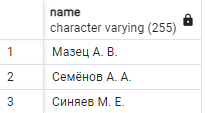
**Урок 19: Подзапросы**

**Определение:**  
**Подзапрос** — это запрос SQL, который встраивается внутрь другого запроса. **Подзапросы** используются для выполнения операций, результат которых будет применен в основном запросе. Они могут возвращать отдельные значения, строки или наборы данных.

**Пример:**

****

**Результат:**

****

*Попробуйте создать другие подзапросы или вывести другие значения.*

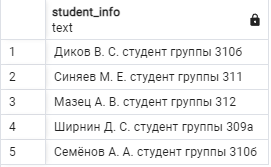
**Урок 20: Конкатенация столбцов**

**Определение:**  
**Конкатенация столбцов** — это операция объединения значений из нескольких столбцов в один результат. Для объединения строковых значений используется оператор **||** или функция **CONCAT**. Это полезно для создания комбинированных строк, например, полного имени или адреса.

**Пример:**

****

**Результат:**

****

*Попробуйте произвести конкатенацию с другими значениями и посмотреть, что получится.*