Урок 2.1. Условные конструкции в SQL: операторы CASE и IF

Оператор CASE

Важно!

На уроке будет использоваться база данных с доступом на чтение:
hostname: ich-db.edu.itcareerhub.de
username: ich1
password: password

Оператор CASE — это инструмент для выполнения условной логики в SQL.

Он работает аналогично оператору IF-THEN-ELSE в языках программирования, позволяя возвращать разные значения на основе заданных условий.

```
None
CASE
    WHEN условие1 THEN результат1
    WHEN условие2 THEN результат2
    ...
    ELSE результатN
END
```

- **WHEN** — задает условие, которое проверяется. Условий может быть несколько.
- **THEN** — определяет результат, который возвращается, если условие выполнено.
- **ELSE** — (необязательно) задает значение, которое будет возвращено, если ни одно из условий не выполнено.
- **END** — завершает блок CASE.

Оператор CASE не обязательно может содержать ELSE, в таком случае все строки не попадающие под условие прописанное в WHEN ... THEN будут иметь пропуска NULL в столбце.

Этот оператор создает новый столбец в выборке, но никаким образом не влияет на исходные данные в таблице.

IT CAREER HUB

Алиасы

При создании новых столбцов им автоматически присваиваются названия, включающие текст формулы, по которой столбцы были вычислены.
Важно присваивать столбцам понятные имена с помощью алиасов.

Алиасы (псевдонимы) — это функции, которые позволяют сделать результат запроса более понятным и удобным для использования.

- Алиасы задаются с помощью ключевого слова AS, но его можно и опустить.
- Алиасы особенно полезны, когда вы работаете с вычисляемыми значениями или используете функции, и хотите дать результату понятное имя.

```
None
SELECT
    выражение1 AS имя1,
    выражение2 AS имя2,
    ...
FROM
    таблица;
```

Выражение — это имя столбца, арифметическое выражение, функция или подзапрос.

Имя — это алиас (псевдоним), который вы присваиваете результату выражения.

IT CAREER HUB

Задание для закрепления

1. Классифицировать товары таблицы products по их стоимости standard_cost, присваивая каждому из них категорию, например, "Дорогой« от 50, "Средний« от 20 до 50 включая 50 или "Дешевый« до 20 включительно.

```
None
SELECT product_name,standard_cost,
CASE WHEN standard_cost> 50 THEN 'Дорогой'
WHEN standard_cost> 20 AND  standard_cost <= 50 THEN 'Средний'
ELSE 'Дешевый' END AS price_category FROM products;
```

2. Подумайте как можно оптимизировать запрос Всегда ли стоит указывать все интервалы. Помните, что SQL как и Питон читает код сверху вниз.

```
None
SELECT product_name, standard_cost,
CASE WHEN standard_cost > 50 THEN 'Дорогой'
WHEN standard_cost > 20 THEN 'Средний'
ELSE 'Дешевый' END AS price_category FROM products;
```

3. Предположим, что вы хотите предоставить разные скидки discount клиентам таблица customers в зависимости от их региона.

- 'WA', 'CA' - 5% скидка
- 'ID', 'OR' - 7% скидка
- 'UT', 'NV' - 13% скидка
- Остальные - без скидки

```
None
SELECT *,
CASE WHEN state_province IN ('WA', 'CA') THEN 5
WHEN state_province IN ('ID', 'OR') THEN 7
WHEN state_province IN ('UT', 'NV') THEN 13
ELSE 0 END as discount
FROM customers
```

4. Вы хотите установить статус для заказов таблица orders в зависимости от даты отправки (shipped_date) и даты заказа (order_date).

   Если нет данных в shipped_dateто статус 'Ожидание отправки', если shipped_date = order_date, то 'Отправлено в день заказа'. В остальных случаях -'Отправлено'.

```
None
SELECT
    id,
    order_date,
    shipped_date,
    CASE
        WHEN shipped_date IS NULL THEN 'Ожидание отправки'
        WHEN shipped_date = order_date THEN 'Отправлено в день заказа'
        ELSE 'Отправлено'
    END AS order_status
FROM
    orders;
```

IT CAREER HUB

Оператор IF

Оператор IF — это функция, которая является простым способом выполнения условной логики, однако он поддерживается только My SQL.

```
None
SELECT IF(condition, true_result, false_result) AS new_column_name
FROM table_name;
```

- **condition:** Условие, которое проверяется.
- **true_result:** Возвращаемое значение, если условие истинно.
- **false_result:** Возвращаемое значение, если условие ложно.

В SQL оператор CASE используется вместо IF по нескольким причинам:

- Стандарт SQL

CASE является стандартным оператором в SQL и поддерживается во всех основных СУБД (например, MySQL, SQL Server, PostgreSQL, Oracle). В то время как IF часто ограничен определенными СУБД (например, MySQL поддерживает IF, но не все СУБД).

- Удобство работы с множественными условиями

Когда нужно проверить несколько условий, CASE обеспечивает более читабельный и логичный подход по сравнению с вложенными IF.

Использование нескольких вложенных IF может привести к сложным, трудно читаемым и поддерживаемым запросам, тогда как CASE предлагает более понятную структуру.

IT CAREER HUB

Задание для закрепления

1. Вывести имя продукта из таблицы products а также его категорию.
   Если standard_cost > 20 то 'Expensive'.
   В обратном случае 'Affordable' с применением оператора IF.

```
None

SELECT

  product_name,

  IF(standard_cost > 20, 'Expensive', 'Affordable') AS PriceCategory

FROM products;
```

Другие способы для создания столбцов на основе имеющихся

С помощью CASE и IF мы создаем новые столбцы, значения в которых зависят от значений в столбцах исходной таблице. Это не единственный способ создавать нужные вам столбцы.

Математические операции

SQL поддерживает основные математические операции, такие как сложение, вычитание, умножение и деление. Эти операции могут быть использованы для создания новых столбцов. При этом важно применять все математические операции только к числовым столбцам.

Предположим, мы хотим вычислить общую стоимость каждого товара в заказе (цена умноженная на количество) таблица order_details:

```
None

SELECT

  product_id,

  unit_price,

  quantity,

  (unit_price * quantity) AS total_price

FROM order_details;
```

📎**Строковые функции — это функции, которые позволяют манипулировать строками данных, например, объединять строки, извлекать подстроки, преобразовывать регистр символов и т.д.**

Функции **LEFT/RIGHT** и **SUBSTRING** используются для извлечения части строки.

- **Функция LEFT**: Извлекает определенное количество символов с начала строки.
- **Функция RIGHT**: Извлекает определенное количество символов с конца строки.

Выбрать название компании, а также укороченное до 2 букв с конца название компании из таблицы customers:

```
None

SELECT

  company,

  RIGHT (company, 2) AS ShortName

FROM customers;
```

Функция SUBSTRING — это функция, которая извлекает подстроку, начиная с определенной позиции и на определенную длину.

Выбрать Три цифры рабочего телефона клиента после кода Таблица customers. Код - первые три цифры в business_phone, написанные в скобках:

```
None
SELECT

* , SUBSTRING(business_phone, 6, 3)

FROM customers;
```

IT CAREER HUB

Конкатенация строк

Оператор CONCAT — это оператор, который может объединить несколько строковых столбцов в один.

1. Выбрать полное имя состоящее из имени и фамилии из таблицы employees

```
None

SELECT

  CONCAT(first_name, ' ', last_name) AS full_name

FROM employees;
```

Использование функции COALESCE

Функция COALESCE — это функция, которая возвращает первое ненулевое (NOT NULL) значение из списка аргументов.

Это полезно для замены NULL значений.

В таблице employees замените все пропуски в столбце notes на значение 'Not filled' Выведите полученный столбец и столбец notes.

```
None
SELECT notes, COALESCE(notes, 'Not filled')
FROM employees;
```

Существуют другие специфические функции работы со столбцами, которые можно посмотреть в документации.

Здесь дополнительные примеры:

Функция	**Описание**	**Пример использования**	**Результат**
CONCAT	Объединяет строки вместе.	`SELECT CONCAT(first_name, ' ', last_name) AS full_name FROM employees;`	Полное имя, объединенное из имени и фамилии.
SUBSTRING / SUBSTR	Извлекает подстроку из строки, начиная с определенной позиции.	`SELECT SUBSTRING(product_name, 1, 3) AS short_name FROM products;`	Первые три символа из названия продукта.
LEFT	Возвращает указанное количество символов с начала строки.	`SELECT LEFT(product_name, 5) AS short_name FROM products;`	Первые пять символов из названия продукта.
RIGHT	Возвращает указанное количество символов с конца строки.	`SELECT RIGHT(product_name, 4) AS end_chars FROM products;`	Последние четыре символа из названия продукта.
LENGTH	Возвращает длину строки в байтах.	`SELECT LENGTH(product_name) AS name_length FROM products;`	Длина названия продукта.
TRIM	Удаляет пробелы с начала и конца строки.	`SELECT TRIM(' Sample Text ') AS trimmed_text;`	"Sample Text" без пробелов в начале и конце.
LTRIM	Удаляет пробелы с начала строки.	`SELECT LTRIM(' Sample Text ') AS trimmed_text;`	"Sample Text " без пробелов в начале.
RTRIM	Удаляет пробелы с конца строки.	`SELECT RTRIM(' Sample Text ') AS trimmed_text;`	"Sample Text" без пробелов в конце.
REPLACE	Заменяет все вхождения подстроки на другую подстроку.	`SELECT REPLACE(product_name, 'Pen', 'Marker') AS new_name FROM products;`	Замена "Pen" на "Marker" в названии продукта.

UPPER	Преобразует все символы строкового выражения в верхний регистр (заглавные буквы).	SELECT UPPER('sql функции');	“SQL ФУНКЦИИ”
LOWER	Преобразует все символы строкового выражения в нижний регистр (строчные буквы).	SELECT LOWER('SQL ФУНКЦИИ');	“sql функции”