**Как ограничить выборку**

Оператор **LIMIT** выгружает первые записи таблицы. А если нужно вывести записи не с начала, а с середины таблицы?

Тогда к оператору **LIMIT** добавляют оператор **OFFSET**. Он позволяет пропустить определённое число записей от начала таблицы.

SELECT поле\_1,

поле\_2,

поле\_3

...

FROM таблица

**LIMIT** 10 **OFFSET** 5;

Такой запрос выведет записи с шестой по пятнадцатую, если начинать нумерацию с единицы.

Запрос ниже выведет записи с первой по десятую:

SELECT поле\_1,

поле\_2,

поле\_3

...

FROM таблица

**LIMIT** 10 **OFFSET** 0;

OFFSET — оператор самостоятельный. Если нужно отобразить записи всей таблицы, начиная с определённой, то нужно указать только оператор OFFSET без оператора LIMIT. Например, если нужно отобразить всю таблицу, начиная с шестой записи, то код будет выглядеть так:

SELECT поле\_1,

поле\_2,

поле\_3

...

FROM таблица

**OFFSET** 5;

**Урок: Как ограничить выборку.**

1. **Выгрузи из таблицы client поля email, first\_name, last\_name.**

SELECT

email,

first\_name,

last\_name -- перечисли поля тут

FROM client; -- укажи таблицу;

1. **Выгрузи первые четыре поля из таблицы invoice. Ограничь выгрузку первыми пятью записями.**

SELECT

invoice\_id,

customer\_id,

invoice\_date,

billing\_address

FROM invoice

**LIMIT** 5;

1. **Выведи первые 20 записей из таблицы track. Оставь в итоговой таблице только два поля: name и unit\_price.**

SELECT

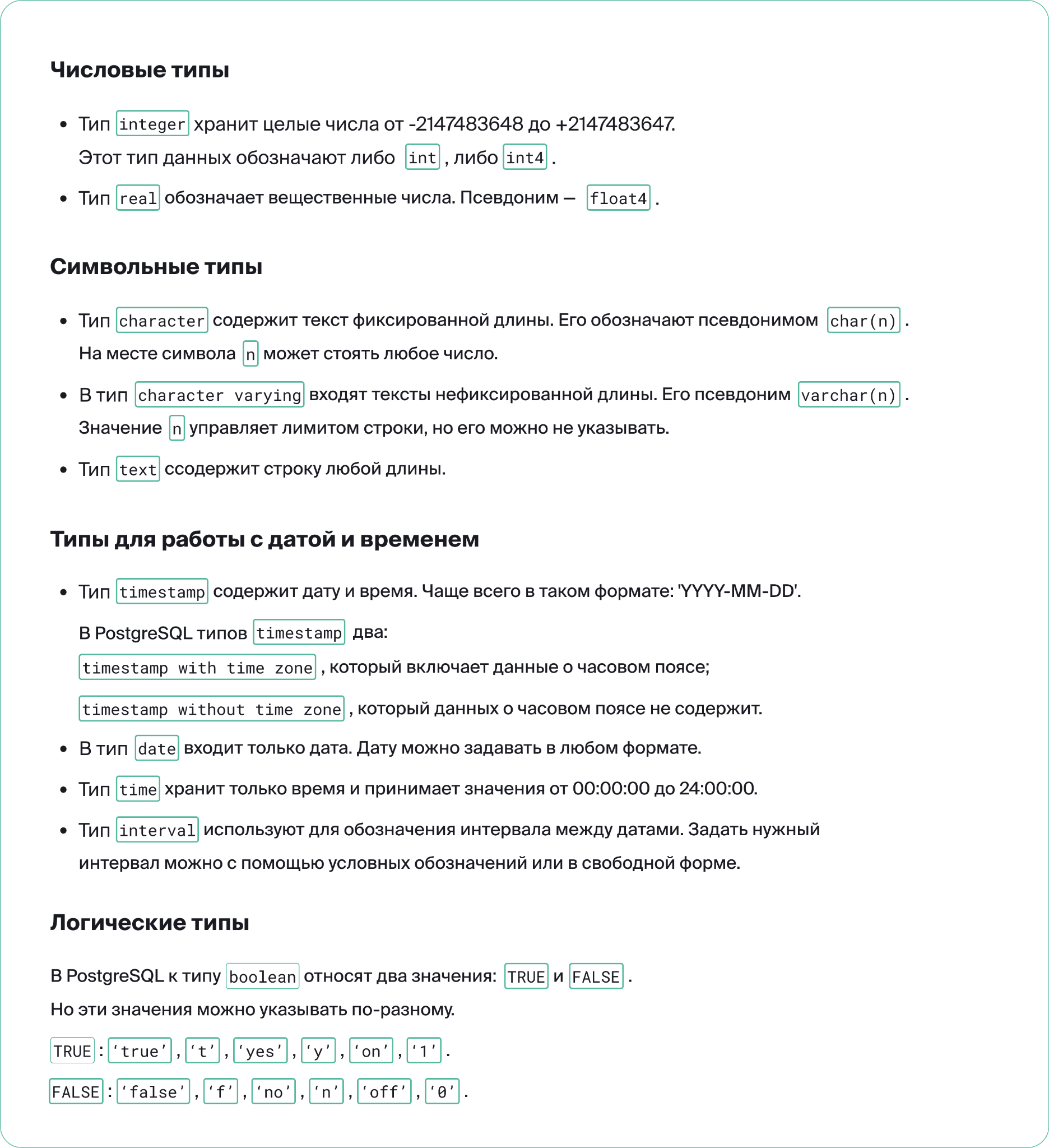
name,

unit\_price

FROM track

**LIMIT** 20;

**Типы данных в PostgreSQL**



**Изменение типов данных**

В PostgreSQL выделяют несколько типов данных. У некоторых есть специальные обозначения или псевдонимы.

Чтобы поменять тип данных при выгрузке, используют оператор CAST. В выгруженной таблице тип данных поля track\_id изменится на varchar. При этом тип данных поля в таблице track останется прежним.

SELECT **CAST**(track\_id AS varchar)

FROM track;

|  | **Тип данных** | **Псевдоним** | **Что входит** |
| --- | --- | --- | --- |
| **Числовые типы** | integer | int,int4 | Целые числа |
|  | real | float4 | Вещественные числа |
| **Символьные типы** | character | char(n) | Строка фиксированной длины |
|  | character varying | varchar(n) | Строка нефиксированной длины |
|  | text |  | Строка любой длины |
| **Типы для работы с датой и временем** | timestamp without timezone | timestamp | Дата и время без данных о часовом поясе |
|  | timestamp with timezone | timestamptz | Дата и время с данными о часовом поясе |
|  | date |  | Дата в разных форматах |
|  | time |  | Время 00:00:00 до 24:00:00 |
|  | interval |  | Интервал между датами |
| **Логические типы** | boolean | bool | TRUE и аналоги: 'true', 't', 'yes', 'y', 'on', 1. FALSE и аналоги: 'false', 'f', 'no', 'n', 'off', 0. |

**Урок: Изменение типов данных.**

1. Выгрузи из таблицы track поля milliseconds и bytes. Оба поля должны быть строками. Используй тип данных для строк нефиксированной длины.

SELECT -- выгрузи поля, изменив тип данных

**CAST**(milliseconds AS varchar),

**CAST**(bytes AS varchar)

FROM track;

1. Выгрузи из таблицы invoice поле total. Оставь только целую часть числа.

SELECT **CAST**(total AS integer)

FROM invoice;

1. Выгрузи из таблицы staff дни рождения сотрудников. Необходимые значения в виде строк хранит поле birth\_date. Сформируй запрос так, чтобы на экране отобразилась только дата.

SELECT **CAST**(birth\_date AS date)

FROM staff;

**Срез данных с помощью оператора WHERE**

В SQL используют похожие операторы сравнения, что и в Python: >, <, >=, <=. Равенство обозначают знаком =, а неравенство — знаками <> или !=.

Сравнение с числом прописывают просто: WHERE [поле] < 5. Но если сравнивают с символьным типом, набор символов берут в одинарные кавычки: WHERE [поле] = 'Иванов'. Это правило касается даты и времени: WHERE [поле] = '2013-07-01'.

SELECT track\_id, -- здесь указаны нужные поля

album\_id,

...

FROM track -- таблица, из которой выгружают данные

**WHERE** track\_id = 5; -- условие для среза данных

**Урок: Срез данных с помощью оператора WHERE**

1. Изучи таблицу invoice\_line, которая хранит информацию о купленных треках. Выгрузи все значения из таблицы, отфильтровав треки дороже 0.99. Нужные значения хранит столбец unit\_price.

SELECT \*

FROM invoice\_line

**WHERE** unit\_price > 0.99;--- сюда добавь условие;

1. Из таблицы client выгрузи несколько полей: first\_name с именем покупателя, last\_name с фамилией и city, в котором указан город проживания. Оставь записи только о тех покупателях, которые живут в Бразилии (англ. Brazil). Страна проживания хранится в поле country.

SELECT first\_name, -- здесь указаны нужные поля

last\_name,

city

FROM client -- таблица, из которой выгружают данные

**WHERE** country = 'Brazil'; -- условие для среза данных

1. Нужно проверить, где и когда совершали самые крупные покупки. Из таблицы invoice выгрузи поле billing\_address, оно хранит адрес выставления счёта, и поле invoice\_date с датой покупки. Дата указана в формате 'YYYY-MM-DD HH:MM:SS'. Нужно оставить только дату 'YYYY-MM-DD'. Отфильтруй записи, в которых значение поля total больше или равно 8.

SELECT billing\_address, -- здесь указаны нужные поля

CAST (invoice\_date AS date)

FROM invoice -- таблица, из которой выгружают данные

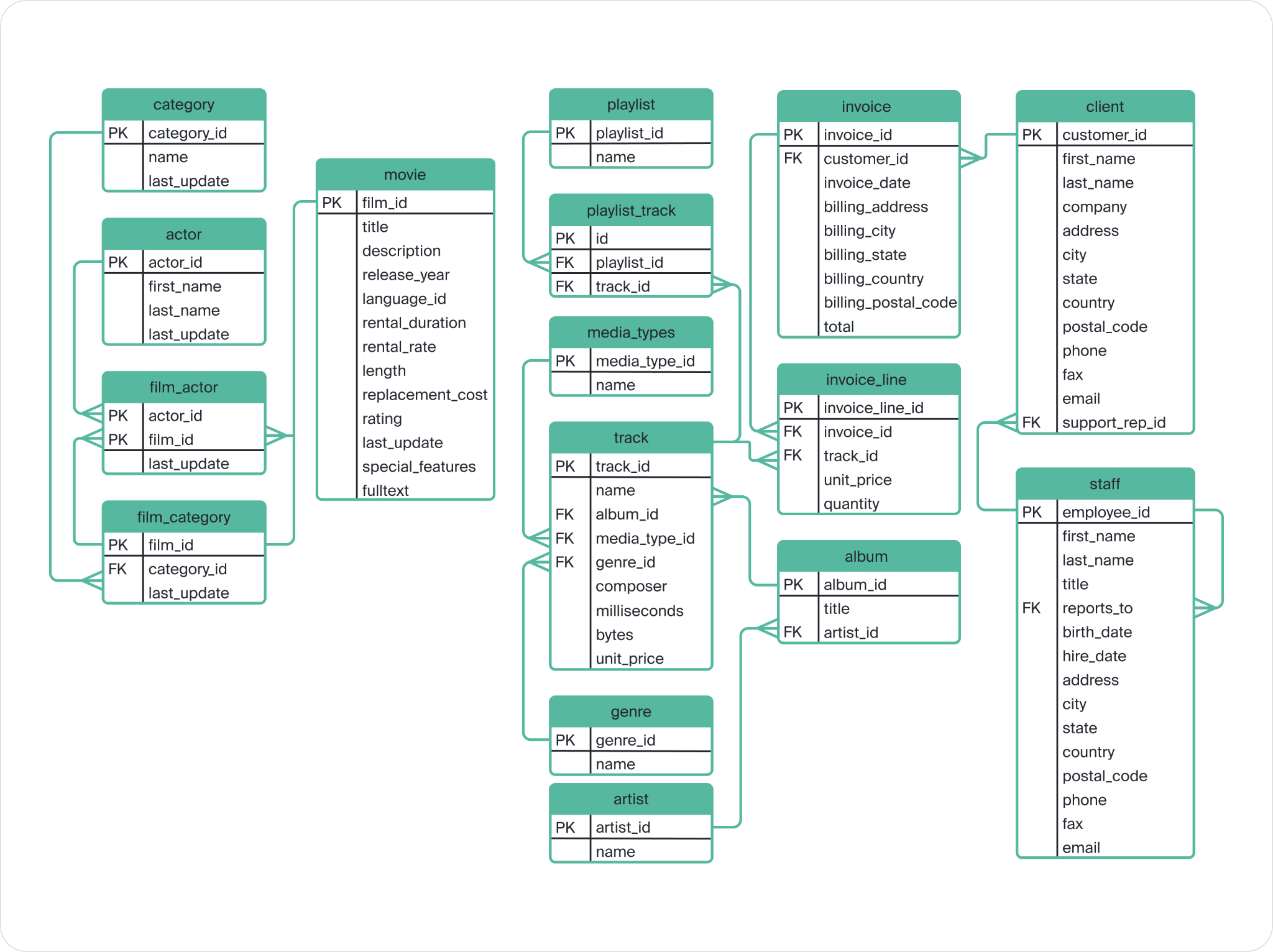
**WHERE** total >= 8; -- условие для среза данных

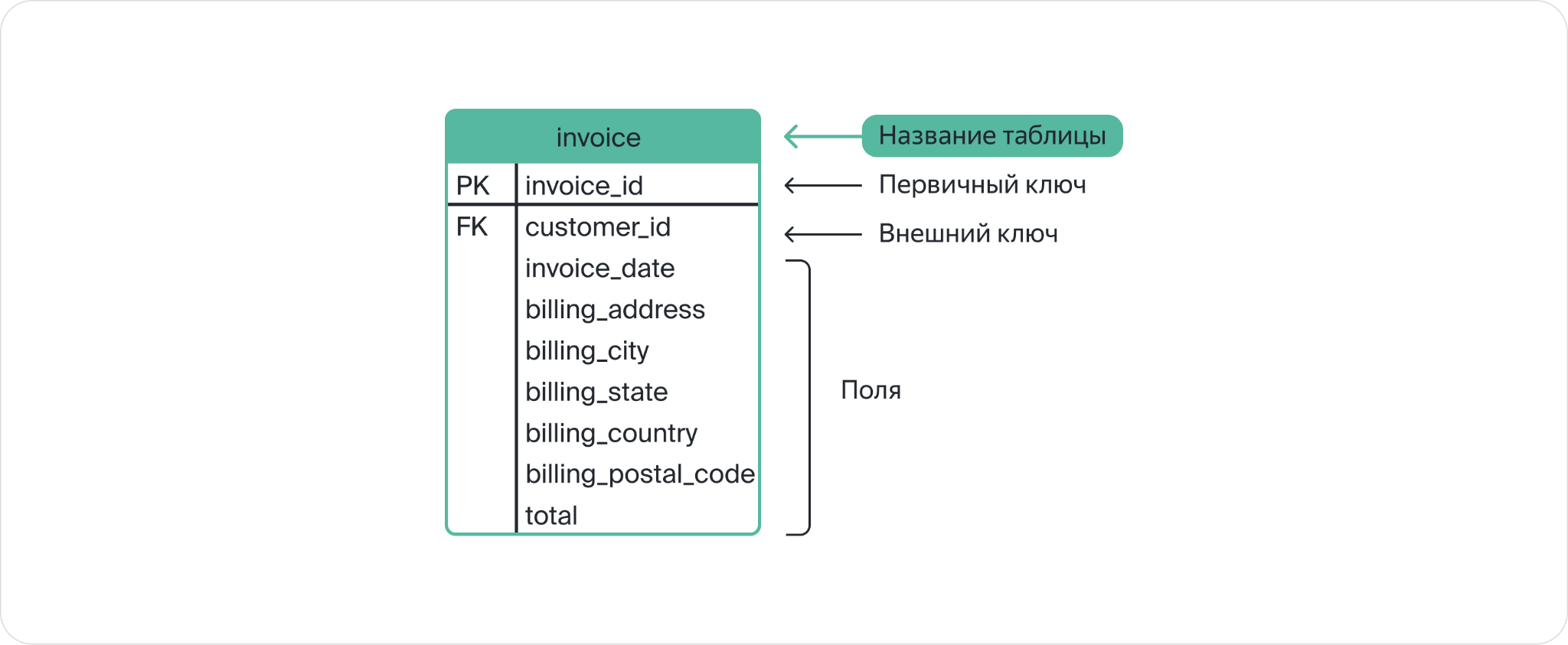
**Как работать с диаграммой и описанием базы данных**

Чтобы разобраться в структуре базы данных, тестировщик изучает требования. Туда аналитики или разработчики прикладывают:

1️⃣ Описание структуры БД.

2️⃣ Диаграмму базы, или ER-диаграмму (от англ. entity-relationship diagram, диаграмма «сущность — связь»). ER-диаграмма графически отображает то, как связаны таблицы в базе данных. Не забудь, что в базе все данные связаны — этим она отличается от обычного хранилища.

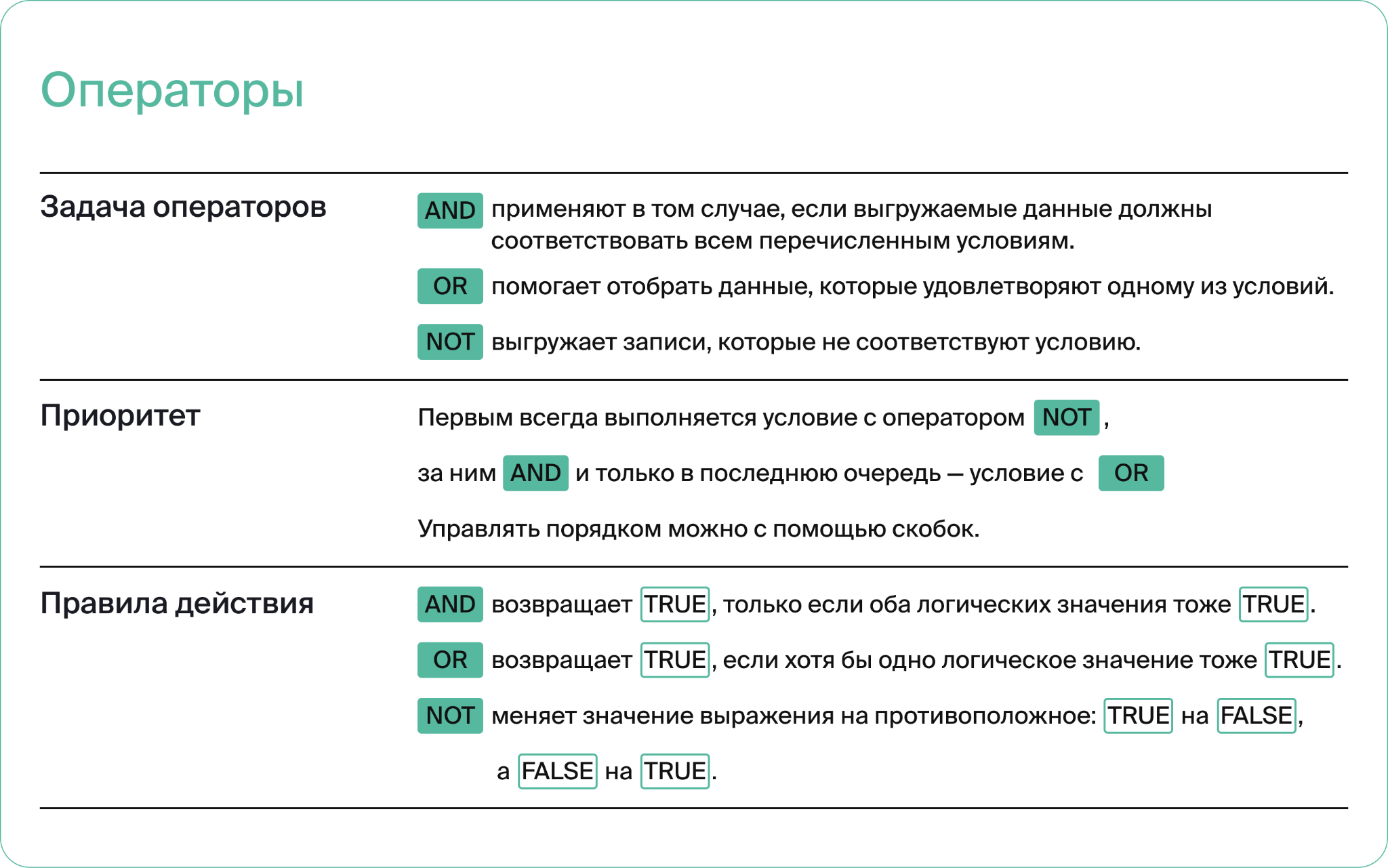




**Логические операторы**

Вот правила, по которым действуют логические операторы в SQL:

* AND возвращает TRUE, только если оба логических значения тоже TRUE.
* OR возвращает TRUE, если хотя бы одно логическое значение тоже TRUE.
* NOT меняет значение выражения на противоположное: TRUE на FALSE, а FALSE на TRUE.



**Урок: Срез данных с помощью логических операторов**

1. Теперь попробуй отфильтровать данные о счетах — их хранит таблица invoice. Выгрузи из таблицы поле с суммой заказа total и поле с идентификатором покупателя customer\_id. Счета должны быть оформлены в трёх городах: Дублине (англ. Dublin), Лондоне (англ. London) или Париже (англ. Paris). Информацию о городе хранит поле billing\_city.

SELECT total,

customer\_id

FROM invoice

WHERE billing\_city = 'Dublin'

**OR** billing\_city = 'London'

**OR** billing\_city = 'Paris';

1. Из таблицы invoice выгрузи поля total и customer\_id. Значение total должно быть больше или равно 5, а значение customer\_id должно равняться 40 или 46.

SELECT total,

customer\_id

FROM invoice

WHERE total >= 5

**AND** (customer\_id = 40

**OR** customer\_id = 46);

1. Теперь объедини условия. Из таблицы invoice выгрузи поля total и customer\_id. Счёт должен быть оформлен в Дублине (англ. Dublin), Лондоне (англ. London) или Париже (англ. Paris). Значение total должно быть больше или равно 5, а значение customer\_id должно равняться 40 или 46.

SELECT total,

customer\_id

FROM invoice

WHERE (billing\_city = 'Dublin'

**OR** billing\_city = 'London'

**OR** billing\_city = 'Paris')

**AND** total >= 5

**AND** (customer\_id = 40

**OR** customer\_id = 46);

1. В следующих заданиях не будет указаний на конкретные поля. Попробуй найти поля самостоятельно, пользуясь схемой базы данных и её описанием.

Выгрузи фамилии и номера телефонов пользователей из таблицы client. В итоговую таблицу должны войти данные о пользователях, которые живут в США (англ. USA) или Франции (англ. France). Убедись, что пользователь обращался в поддержку к сотруднику с идентификатором 3.

SELECT last\_name,

phone

FROM client

WHERE (country = 'USA'

**OR** country = 'France')

**AND** support\_rep\_id = 3;

1. Из таблицы movie выгрузи названия фильмов, стоимость аренды которых не превышает двух долларов, а срок аренды составляет больше шести дней. Выгруженные фильмы не должны относиться к рейтингам PG и PG-13.

SELECT title

FROM movie

WHERE rental\_rate < 2

AND rental\_duration > 6

AND (rating != 'PG'

AND rating != 'PG-13');

1. Потренируйся работать с диаграммой и описанием базы. Выгрузи адреса и города оформления заказов, сделанных в сентябре 2009 года. Отбери данные о заказах, оформленных во всех странах, кроме США и Бразилии. Стоимость заказа должна быть больше двух долларов.

SELECT billing\_address,

billing\_city

FROM invoice

WHERE CAST(invoice\_date AS date) > '2009-09-01'

**AND** CAST(invoice\_date AS date) < '2009-10-01'

**AND**  billing\_country != 'Brazil'

**AND**  billing\_country != 'USA'

**AND** total > 2;

**Операторы IN, LIKE, BETWEEN**

Оператор **IN**

Добавить в условие несколько значений можно с помощью оператора **IN**. Значения, по которым нужно фильтровать данные, перечисляют в скобках:

SELECT total,

customer\_id

FROM invoice

WHERE billing\_city **IN** ('Dublin',

'London',

'Paris',

'Boston',

'Berlin',

'Stuttgart');

Оператор **LIKE**

Чтобы отфильтровать данные по шаблону, используют оператор **LIKE**. С помощью знака % показывают, какую позицию в строке занимает шаблон.

Основные варианты шаблонов можно собрать в таблицу:

| Выражение | Что попадёт | Пример кода | Результат |
| --- | --- | --- | --- |
| 'text%' | Значения, которые начинаются с text | WHERE quotes LIKE 'кот%' | 'кот просил еды' 'котелок не варит' |
| '%text' | Значения, которые заканчиваются на text | WHERE quotes LIKE '%кот' | 'куда пропал кот' 'двойной апперкот' |
| '%text%' | Значения, в которых text занимает любую позицию | WHERE quotes LIKE '%кот%' | 'кота зовут василий' 'они танцевали котильон' |
| 'te%xt' | Значения, которые начинаются на te и заканчиваются на xt | WHERE quotes LIKE 'ко%т' | 'кот играет' 'консультант' |

Оператор **BETWEEN**

Выбрать значения из конкретного диапазона можно с помощью оператора **BETWEEN**. Границы диапазона указывают так: **WHERE** [ключевое слово] **BETWEEN** [начало диапазона] **AND** [конец диапазона]. Начало и конец диапазона включены в условие.

Указать промежуток с помощью **BETWEEN** можно для числовых типов данных:

SELECT \*

FROM invoice

WHERE customer\_id **BETWEEN** 23 AND 48;

Так выглядит запрос с оператором BETWEEN для промежутка времени:

SELECT last\_name

FROM staff

WHERE birth\_date **BETWEEN** '1970-01-01' AND '1970-12-31';

**Урок: Операторы IN, LIKE, BETWEEN**

1. Выгрузи из таблицы playlist все названия плейлистов из поля name, в которых есть подстрока 'Classic'.

SELECT name

FROM playlist

WHERE name **LIKE** '%Classic%';

1. Выгрузи из таблицы invoice адреса из поля billing\_address и страну из поля billing\_country. Отфильтруй данные по стране: оставьте только США (англ. USA), Индию (англ. India), Канаду (англ. Canada), Аргентину (англ. Argentina) и Францию (англ. France).

SELECT billing\_address,

billing\_country

FROM invoice

WHERE billing\_country **IN** ('USA',

'India',

'Canada',

'Argentina',

'France');

1. Дополни предыдущий запрос. Исключи из городов выставления счёта billing\_city Редмонд (англ. Redmond), Лион (англ. Lyon) и Дели (англ. Delhi). Само поле billing\_city выводить не нужно.

SELECT billing\_address,

billing\_country

FROM invoice

WHERE billing\_country **IN** ('USA',

'India',

'Canada',

'Argentina',

'France')

AND billing\_city **NOT IN** ('Redmond',

'Lyon',

'Delhi') -- сюда добавь условие;

1. Выгрузи все поля из таблицы invoice. Информацию о дате покупки хранит поле invoice\_date: выбери только те заказы, которые были оформлены в период с '2009-03-04' по '2012-02-09' включительно. Обрати внимание, что у поля invoice\_date тип данных varchar.

Сумма покупки total должна быть меньше 5. Кроме того, исключите из запроса Канаду (англ. Canada), Бразилию (англ. Brazil) и Финляндию (англ. Finland). Название страны содержит поле billing\_country.

SELECT \*

FROM invoice

WHERE CAST (invoice\_date AS date) **BETWEEN** '2009-03-04' AND '2012-02-09'

AND total < 5

AND billing\_country **NOT IN** ('Canada',

'Brazil',

'Finland');

1. Выгрузи из таблицы movie название фильмов, описание которых заканчивается подстрокой Mexico. Аренда фильмов из итоговой таблицы должна составлять меньше двух долларов либо их рейтинг не должен быть PG-13.

SELECT title

FROM movie

WHERE description **LIKE** '%Mexico'

AND (rental\_rate < 2

OR rating != 'PG-13');

1. Из таблицы с треками отбери названия композиций, которые соответствуют условиям. В итоговую таблицу должны войти:

* треки длиннее 300000 миллисекунд, в создании которых принимал участие Bono, с идентификаторами жанра 7, 8, 9, 10;
* либо треки, размер которых превышает 1000000000 байт.

SELECT name

FROM track

WHERE milliseconds > 300000

AND composer **LIKE** '%Bono%'

AND genre\_id **IN** (7,

8,

9,

10)

OR bytes > 1000000000;

**Условная конструкция с оператором CASE**

Условные конструкции в SQL состоят из проверяемого условия и результата, который возвращается в зависимости от условия. Так выглядит синтаксис условных конструкций в SQL:

* начало конструкции обозначают оператором CASE,
* после оператора WHEN пишут условие,
* после оператора THEN — возвращаемый результат;
* а в конце указывают оператор END.

SELECT total,

**CASE**

WHEN total < 5 THEN 'маленький'

WHEN total >= 5 AND total < 10 THEN 'средний'

WHEN total >= 10 THEN 'крупный'

**END**

FROM invoice

LIMIT 10;

Все условия необязательно прописывать. Условную конструкцию можно дополнить выражением с оператором ELSE: после оператора указывают результат на случай, если ни одно условие не выполнено.

SELECT total,

CASE

WHEN total >= 5 AND total < 10 THEN 'средний'

WHEN total >= 10 THEN 'крупный'

**ELSE** 'маленький'

END

FROM invoice

LIMIT 10;

Важный момент — условия в запросе проверяются по порядку. Если условие выполнено, сразу выводится соответствующий результат, а остальные условия не проверяются.

SELECT total,

**CASE**

WHEN total >= 5 THEN 'средний'

WHEN total >= 10 THEN 'крупный'

ELSE 'маленький'

**END**

FROM invoice

LIMIT 10;

Заказу под номером 5 присвоена категория средний, а не крупный. Дело в том, что проверяемое значение в поле total удовлетворяет двум условиям сразу. Первое условие было выполнено, поэтому остальные условия не проверялись. Такой ситуации можно избежать, если правильно указать границы диапазона

**Урок: Условная конструкция с оператором CASE**

1. Выдели категории в таблице staff, которая хранит информацию о сотрудниках. Выведи на экран поля last\_name, first\_name и title. Категории нужно выделить на основе значений в поле title — оно содержит информацию о должности сотрудника:

* Если в title встречается слово 'IT', в новом поле будет отображена категория 'разработка'.
* Если в title встречается слово 'Manager' и нет слова 'IT', в новом поле отобразится категория 'отдел продаж'.
* Если в title встречается слово 'Support', в новом поле появится категория 'поддержка'.

SELECT last\_name,

first\_name,

title, -- здесь укажи нужные поля

**CASE**

WHEN title LIKE '%IT%' THEN 'разработка'

WHEN (title LIKE '%Manager%' AND title NOT LIKE '%IT%') THEN 'отдел продаж'

WHEN title LIKE '%Support%' THEN 'поддержка'

**END** -- сюда запиши условия

FROM staff;

1. Теперь попробуй разделить на категории фильмы из таблицы movie. Выведи на экран поля title и rental\_rate. Выделить категории нужно по цене аренды фильма:

* Если значение rental\_rate меньше 1, в новом поле появится категория 'категория 1'.
* Если значение rental\_rate больше либо равно 1, но меньше 3, в новом поле отобразится категория 'категория 2'.
* Если значение rental\_rate больше либо равно 3, в новом поле появится категория 'категория 3'.

SELECT title,

rental\_rate,-- сюда добавь поля

**CASE**

WHEN rental\_rate < 1 THEN 'категория 1'

WHEN rental\_rate >= 1 AND rental\_rate < 3 THEN 'категория 2'

WHEN rental\_rate >= 3 THEN 'категория 3'

**END**-- здесь запиши условия

FROM movie;

**Работа с пропусками**

Специальное значение **NULL** в SQL обозначает пропуск. Проверить данные на пропуски можно только специальными операторами, а сделать это с помощью = и <> нельзя.

Для работы с NULL используют операторы **IS NULL** и **IS NOT NULL**. С помощью этих операторов можно искать пропуски или получать срезы.

Такое условие охватит все записи с пропусками в поле fax:

WHERE fax **IS NULL**;

Чтобы выбрать записи без пропусков, пишут такое условие:

WHERE fax **IS NOT NULL**;

**Урок: Работа с пропусками.**

1. Из таблицы invoice выгрузи поле billing\_city с городами оформления счёта. Выгрузи только те записи, в которых на месте индекса стоит пропуск. Данные с индексами хранит поле billing\_postal\_code.

SELECT billing\_city

FROM invoice

WHERE billing\_postal\_code **IS NULL**; -- сюда запиши условие;

1. Дополни предыдущий запрос. Исключи из выдачи записи с пропусками в поле billing\_state. Выбери записи, в которых сумма заказа в поле total не ниже 15 долларов.

SELECT billing\_city

FROM invoice

WHERE billing\_postal\_code **IS NULL**

AND billing\_state **IS NOT NULL**

AND total >= 15; -- дополни условие;

1. Найди треки длиннее 250000 миллисекунд, в названии которых есть слово Moon, но автор трека не указан.

Выгрузи номера альбомов, в которых находятся выбранные треки.

SELECT album\_id

FROM track

WHERE milliseconds > 250000

AND name LIKE '%Moon%'

AND composer **IS NULL**;

1. Отобрази имена, фамилии и страну проживания для покупателей, которые не указали информацию о месте работы, регионе проживания, а также телефон и факс.

SELECT first\_name,

last\_name,

country

FROM client

WHERE company **IS NULL**

AND state **IS NULL**

AND phone **IS NULL**

AND fax **IS NULL**;

**Агрегирующие функции**

Основные агрегирующие функции в SQL:

* **SUM**(поле) возвращает сумму значений в поле;
* **AVG**(поле) находит среднее арифметическое значений в поле;
* **MIN**(поле) возвращает минимальное значение в поле;
* **MAX**(поле) возвращает максимальное значение в поле;
* **COUNT**(поле) выводит количество записей в поле.

Агрегирующие функции применяют после оператора **SELECT**.

SELECT **COUNT**(billing\_postal\_code)

FROM invoice;

Отобрать уникальные значения поможет ключевое слово **DISTINCT**. Иногда его сочетают с функцией COUNT.

SELECT COUNT(**DISTINCT** billing\_country)

FROM invoice;

**Урок: Применение агрегирующих функций.**

1. Перед тем как применить агрегирующие функции, нужно получить срез данных. Выгрузи все поля из таблицы invoice, выбери записи о заказах, оформленных в сентябре 2009 года. Информацию о дате заказа содержит поле invoice\_date. Тип данных поля — varchar.

SELECT \*

FROM invoice

WHERE CAST(invoice\_date AS date) > '2009-09-01'

AND CAST(invoice\_date AS date) < '2009-10-01';

1. Дополни запрос. Оставь в таблице поля с идентификатором покупателя (поле customer\_id), датой заказа без времени (поле invoice\_date) и суммой заказа (поле total). Выбери покупателей с идентификаторами 11, 13, 44, 36, 48, 52, 54, 56.

SELECT customer\_id,

**CAST**(invoice\_date AS date), -- не забудь преобразовать дату в нужный формат

total -- сюда запиши нужные поля

FROM invoice

WHERE CAST(invoice\_date AS date) BETWEEN '2009-09-01' AND '2009-09-30'

AND customer\_id IN (11, 13, 44, 36, 48, 52, 54, 56); -- дополни условие

1. Дополни запрос. Найди минимальное и максимальное значения поля total. Условия для среза остаются прежними.

SELECT

MIN(total),

MAX(total)-- добавь расчёты с помощью агрегирующих функций

FROM invoice

WHERE CAST(invoice\_date AS date) BETWEEN '2009-09-01' AND '2009-09-30' AND customer\_id IN (11, 13, 44, 36, 48, 52, 54, 56);

1. Добавь к минимуму и максимуму несколько новых полей:

* среднее значение поля total;
* количество уникальных идентификаторов покупателей из поля customer\_id;
* суммарная выручка.

Не меняй условия для среза данных.

SELECT MIN(total),

MAX(total),

AVG(total),

COUNT(DISTINCT customer\_id),

SUM(total) -- добавь нужные поля

FROM invoice

WHERE CAST(invoice\_date AS date) BETWEEN '2009-09-01' AND '2009-09-30' AND customer\_id in (11, 13, 44, 36, 48, 52, 54, 56);

1. Посчитай, сколько пропусков содержит поле fax из таблицы client.

SELECT COUNT(\*)

FROM client

WHERE fax **IS NULL**;

**Применение оператора GROUP BY**

Группировка помогает анализировать данные. Например, с помощью группировки можно сравнить выручку в разных городах. В SQL данные группируют оператором GROUP BY — его указывают после условного оператора WHERE. Если WHERE в запросе нет, оператор GROUP BY пишут после FROM.

-- нужно указать поля и необходимые вычисления

SELECT billing\_city,

AVG(total)

-- не забыть таблицу

FROM invoice

-- здесь указывают поле, в котором группируют данные

**GROUP BY** billing\_city

LIMIT 5;

Не забудь вывести поле, по которому группируешь данные. Иначе таблица получится неинформативной.

SELECT AVG(total)

FROM invoice

**GROUP BY** billing\_city

LIMIT 5;

**Урок: Применение оператора GROUP BY**

1. Группировка помогает сравнивать данные. Но сначала нужно получить срез. Напиши запрос, который выгрузит общую выручку (поле total) в США (англ. USA). Информацию о стране хранит поле billing\_country.

-- добавь агрегирующую функцию

SELECT SUM(total)

FROM invoice

-- сюда напиши условие;

WHERE billing\_country = 'USA';

1. Теперь можно проверить, как отличаются данные по городам. Посчитай общую выручку, количество заказов, среднюю выручку для каждого города США. Нужное поле — billing\_city.

-- добавь агрегирующие функции

SELECT billing\_city,

SUM(total),

COUNT(customer\_id),

AVG(total)

FROM invoice

WHERE billing\_country = 'USA'

-- впиши категорию для группировки

**GROUP BY** billing\_city

1. Посчитай, сколько пользователей с почтовыми доменами yahoo и gmail обслужил каждый сотрудник. В итоговой таблице должны быть два поля:

* идентификатор сотрудника,
* количество пользователей.

SELECT support\_rep\_id,

COUNT(email)

FROM client

WHERE email LIKE '%yahoo%' OR email LIKE '%gmail%'

**GROUP BY** support\_rep\_id;

**Сортировка данных**

Для сортировки данных в SQL используют оператор **ORDER BY**. Его пишут в самом конце запроса, после него можно указать только оператор LIMIT.

По умолчанию оператор ORDER BY сортирует данные от меньшего к большему. Чтобы изменить порядок сортировки вручную, после названия поля указывают ключевое слово DESC: тогда данные будут отсортированы по убыванию. Если указать ASC, данные будут отсортированы по возрастанию — так же, как и по умолчанию.

**Урок: Сортировка данных**

1. Отбери пять самых **крупных** заказов из таблицы invoice.

SELECT \*

FROM invoice

-- впиши условие для сортировки

**ORDER BY** total **DESC**

-- добавь ограничение;

LIMIT 5;

1. Отбери пятерых самых активных клиентов в США с 25 мая 2011 по 25 сентября 2011. Дату хранит поле invoice\_date, тип данных поля — varchar.

SELECT customer\_id,

COUNT(customer\_id)

FROM invoice

-- напиши условие

WHERE CAST (invoice\_date AS date) BETWEEN '2011-05-25' AND '2011-09-25'

AND billing\_country = 'USA'

GROUP BY customer\_id

-- добавь данные для сортировки

**ORDER BY** COUNT(invoice\_id) DESC

LIMIT 5;

**Оператор HAVING**

**HAVING** и **WHERE**

Если нужно получить срез данных после группировки, используют оператор HAVING. HAVING похож на оператор WHERE, но с отличием: HAVING всегда идёт после GROUP BY.

| Name | Оператор WHERE | Оператор HAVING |
| --- | --- | --- |
| Когда используют | Используют, чтобы получить срез данных перед группировкой или отфильтровать записи для агрегирующих функций | Используют, чтобы получить срез данных после группировки |
| Можно применить без оператора GROUP BY | Да | Нет |
| Как сочетается с оператором GROUP BY | Используют перед оператором GROUP BY | Используют после оператора GROUP BY |
| Можно сочетать с агрегирующими функциями | Нет | Да |

Оператор HAVING используют для того, чтобы получить срез данных после группировки. У этого оператора есть несколько особенностей:

* HAVING нельзя применять без оператора GROUP BY;
* использовать HAVING без агрегирующих функций нет смысла: в таких случаях больше подойдёт WHERE.

Оператор HAVING всегда указывают после GROUP BY.

SELECT customer\_id,

SUM(total)

FROM invoice

GROUP BY customer\_id

**HAVING SUM**(total) > 41;

**Урок: Применение оператора HAVING**

1. Сравни фильмы разных возрастных рейтингов. Найди среднее значение цены аренды фильма в поле rental\_rate для каждого рейтинга (поле rating). Оставь в таблице только те записи, в которых среднее значение rental\_rate больше 3.

-- укажи нужные поля

SELECT AVG(rental\_rate),

rating

FROM movie

-- сгруппируй данные

GROUP BY rating

-- пропиши условие;

**HAVING** AVG(rental\_rate) > 3;

1. Изучи заказы, которые оформили в сентябре 2011 года. Сравни общую сумму выручки (поле total) за каждый день этого месяца: выведи день в формате '2011-09-01' и сумму. Информацию о дате заказа хранит поле invoice\_date. Не забудь изменить тип данных в этом поле, чтобы использовать операторы для работы с датой.

Оставь в таблице только те значения суммы, которые больше 1 и меньше 10.

-- укажи нужные поля

SELECT CAST(invoice\_date AS date),

SUM(total)

FROM invoice

-- пропиши условие

WHERE CAST(invoice\_date AS date) >= '2011-09-01'

AND CAST(invoice\_date AS date) < '2011-10-01'

-- сгруппируй данные

GROUP BY invoice\_date

-- добавь условие

**HAVING** SUM(total) BETWEEN 1 AND 10;

1. Посчитай пропуски в поле с почтовым индексом billing\_postal\_code для каждой страны (поле billing\_country). Получи срез: в таблицу должны войти только те записи, в которых поле billing\_address содержит слова Street, Way, Road или Drive.

Отобрази в таблице страну и число пропусков, если их больше 6.

-- укажи нужные поля

SELECT billing\_country,

COUNT(\*)

FROM invoice

-- пропиши условие

WHERE billing\_postal\_code IS NULL

AND (billing\_address LIKE '%Street%'

OR billing\_address LIKE '%Way%'

OR billing\_address LIKE '%Road%'

OR billing\_address LIKE '%Drive%')

-- сгруппируй данные

GROUP BY billing\_country

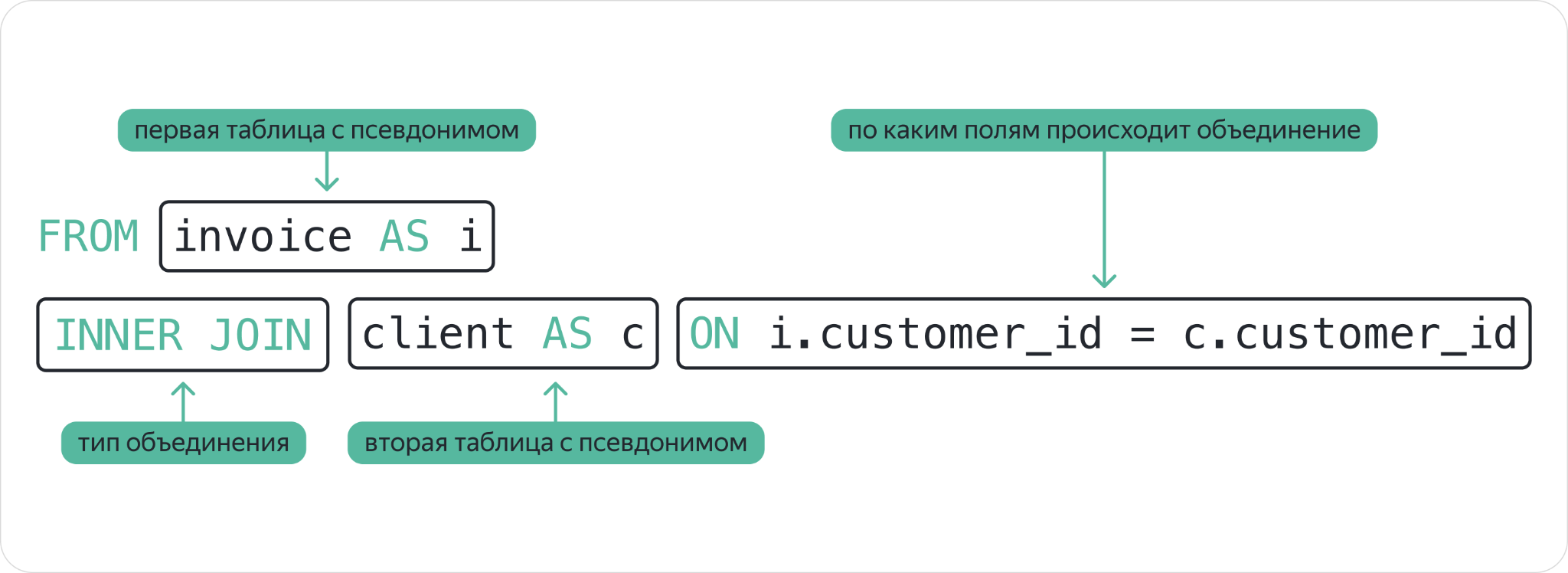
-- пропиши условие

HAVING COUNT(\*) > 6;

**Оператор INNER JOIN**

Вспомни материал предыдущих уроков:

* Таблицы в базе данных связаны, и все взаимосвязи между ними отражает ER-диаграмма.
* Связь между таблицами обеспечивается внешними ключами, и с их помощью таблицы можно объединить.
* Псевдонимы позволяют привязывать поле к конкретной таблице — это пригодится при объединении.
* Соединить таблицы можно по-разному, и от выбранного способа зависит результат слияния.
* Для объединения таблиц используют операторы INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN.



**Урок: Оператор INNER JOIN.**

1. Нужно объединить данные двух таблиц: track и invoice\_line. Таблица track хранит информацию о музыкальных треках в магазине, названия треков указаны в поле name. Таблица invoice\_line содержит данные о купленных треках, их стоимость указана в поле unit\_price. В обеих таблицах есть поле track\_id — в нём содержатся идентификаторы музыкальных треков.

Выгрузи таблицу, в которой названию трека будет соответствовать его стоимость. Отбери все уникальные записи. Если какой-либо из треков не покупали или у купленного трека нет названия — такие записи не должны войти в таблицу. Оставь в итоговой таблице первые 20 записей.

SELECT t.name,

i.unit\_price

FROM track AS t

**INNER JOIN** invoice\_line AS i ON t.track\_id=i.track\_id

WHERE t.name IS NOT NULL AND i.unit\_price IS NOT NULL

GROUP BY t.name, i.unit\_price

LIMIT 20;

1. Нужно дополнить запрос: добавь поле с идентификатором плейлиста playlist\_id. Такое поле можно получить из таблицы playlist\_track. В этой таблице собраны идентификаторы плейлистов и треков (поле track\_id). Условие остаётся прежним: если идентификаторы треков не совпадают во всех трёх таблицах, такие треки не должны войти в итоговую таблицу. Выведи первые 20 записей.

SELECT t.name,

i.unit\_price,

p.playlist\_id -- укажи необходимое поле

FROM track AS t

INNER JOIN invoice\_line AS i ON t.track\_id=i.track\_id

INNER JOIN playlist\_track AS p ON t.track\_id=p.track\_id -- добавь ещё одно присоединение;

LIMIT 20;

1. Идентификатор плейлиста теперь указан в итоговой таблице. Но что это за плейлисты — непонятно. Эту информацию можно взять в четвёртой таблице — playlist. Таблица содержит поле playlist\_id с идентификатором плейлиста и поле name — с его названием. Добавь в итоговую таблицу поле name. Условия те же: данные без совпадения не должны попасть в таблицу. Ограничь вывод первыми 20 записями.

SELECT t.name,

i.unit\_price,

pt.playlist\_id,

pl.name -- укажи необходимое поле

FROM track AS t

INNER JOIN invoice\_line AS i ON t.track\_id=i.track\_id

INNER JOIN playlist\_track AS pt ON t.track\_id=pt.track\_id

INNER JOIN playlist AS pl ON pt.playlist\_id=pl.playlist\_id -- добавь ещё одно присоединение

LIMIT 20;

**Операторы LEFT OUTER JOIN и RIGHT OUTER JOIN**

Операторы LEFT OUTER JOIN и RIGHT OUTER JOIN тебе уже знакомы.

* Оператор LEFT OUTER JOIN предполагает, что в результат слияния обязательно войдут все записи из левой таблицы. Записи из правой таблицы сохранятся только в том случае, если значения в поле, по которому происходит объединение, совпадают со значениями в левой таблице.
* Оператор RIGHT OUTER JOIN устроен аналогичным образом, но большим приоритетом обладает правая таблица.

**Урок: Операторы LEFT OUTER JOIN и RIGHT OUTER JOIN**

1. Выведи названия всех треков, добавив информацию о датах, в которые эти треки покупали. Ни один трек не должен потеряться, даже если его не покупали вообще. Чтобы получить нужный результат, надо соединить три таблицы сразу, ведь таблица invoice, которая хранит данные о дате заказа, не содержит информации о купленных треках.

Сначала соедини таблицы track и invoice\_line по ключу track\_id, а затем присоедини таблицу invoice по ключу invoice\_id. В итоговую таблицу помести два поля: name из таблицы track и invoice\_date из таблицы invoice. Приведи дату в нужный формат.

SELECT name,

CAST(invoice.invoice\_date AS date) -- помести поле с датой

FROM track -- определи левую таблицу

LEFT JOIN invoice\_line ON track.track\_id = invoice\_line.track\_id -- определи вариант присоединения первой таблицы

LEFT JOIN invoice ON invoice\_line.invoice\_id = invoice.invoice\_id;-- определи вариант присоединения второй таблицы;

1. Выгрузи таблицу из двух полей: первое поле с фамилией сотрудника, второе — с количеством пользователей, чьи запросы этот сотрудник обработал. Назови поля employee\_last\_name и all\_customers соответственно. Сгруппируй записи по идентификатору сотрудника. Отсортируй количество пользователей по убыванию.

SELECT s.last\_name AS employee\_last\_name,

COUNT(c.customer\_id) AS all\_customers

FROM staff as s

LEFT OUTER JOIN client AS c ON s.employee\_id=c.support\_rep\_id

GROUP BY s.employee\_id, s.last\_name

ORDER BY all\_customers DESC;

1. Отобрази названия фильмов, в которых снимались актёры и актрисы, не указанные в базе.

SELECT title

FROM movie AS m

LEFT JOIN film\_actor AS fa ON fa.film\_id = m.film\_id

LEFT JOIN actor AS a ON a.actor\_id = fa.actor\_id

WHERE a.actor\_id IS NULL;

**Оператор INSERT**

**INSERT** помогает добавить новую строку в таблицу (англ. «вводить»). Синтаксис запроса:

INSERT INTO

название\_таблицы

(название\_столбца\_1,название\_столбца\_2, название\_столбца\_3 ... )

-- блок с названиями столбцов необязательный

VALUES

(значение\_1,значение\_2,значение\_3...);

В запросе два ключевых оператора: **INSERT INTO** и **VALUES**.

INSERT INTO указывает, в какую таблицу добавить данные. VALUES перечисляет данные, которые нужно добавить.

**Оператор UPDATE**

Оператор UPDATE (англ. «обновить») меняет текущую информацию в таблице. Синтаксис запроса:

UPDATE

название\_таблицы

SET

имя\_столбца = значение\_поля

WHERE

условие; -- определяем условие, по которому будем отбирать строки

В запросе три ключевых оператора: **UPDATE**, **SET** и **WHERE**.

UPDATE указывает, в какой таблице изменить данные. SET указывает столбец, в котором необходимо поменять данные. WHERE выбирает строки, опционально.

**Оператор DELETE**

Оператор DELETE удаляет записи из таблицы (англ. «удалять»). Синтаксис запроса:

DELETE FROM

название\_таблицы

WHERE

условие; -- определяем условие, по которому будем отбирать строки

В запросе два ключевых оператора: **DELETE FROM** и **WHERE**.

DELETE FROM указывает, из какой таблицы удалить записи. WHERE выбирает строки, в которых нужно удалить данные. Это опционально.

**Проект**

**Задание 1**

Посчитай, сколько компаний закрылось.

SQL-Запрос:

SELECT COUNT(status)

FROM company

WHERE status LIKE '%closed%'

**Задание 2**

Отобрази количество привлечённых средств для новостных компаний США. Используй данные из таблицы company. Отсортируй таблицу по убыванию значений в поле funding\_total.

SQL-Запрос:

SELECT funding\_total

FROM company

WHERE country\_code = 'USA'

AND category\_code = 'news'

ORDER BY funding\_total DESC

**Задание 3**

Отобрази имя, фамилию и названия аккаунтов людей в поле network\_username, которые начинаются на 'Silver'.

SQL-Запрос:

SELECT first\_name, last\_name, network\_username

FROM people

WHERE network\_username LIKE 'Silver%'

**Задание 4**

Выведи на экран всю информацию о людях, у которых названия аккаунтов в поле network\_username содержат подстроку 'money', а фамилия начинается на 'K'.

SQL-Запрос:

SELECT \*

FROM people

WHERE network\_username LIKE '%money%'

AND last\_name LIKE 'K%'

**Задание 5**

Для каждой страны отобрази общую сумму привлечённых инвестиций, которые получили компании, зарегистрированные в этой стране. Страну, в которой зарегистрирована компания, можно определить по коду страны. Отсортируй данные по убыванию суммы.

SQL-Запрос:

SELECT SUM(funding\_total),

country\_code

FROM company

GROUP BY country\_code

ORDER BY SUM(funding\_total) DESC

**Задание 6**

Отобрази имя и фамилию всех сотрудников стартапов. Добавь поле с названием учебного заведения, которое окончил сотрудник, если эта информация известна.

SQL-Запрос:

SELECT e.instituition, p.first\_name, p.last\_name

FROM people AS p

LEFT OUTER JOIN education AS e ON p.id = e.person\_id

**Задание 7**

Найди общую сумму сделок по покупке одних компаний другими в долларах. Отбери сделки, которые осуществлялись только за наличные с 2011 по 2013 год включительно.

SQL-Запрос:

SELECT SUM(price\_amount)

FROM acquisition

WHERE term\_code = 'cash'

AND CAST(acquired\_at AS DATE) BETWEEN '2011-01-01' AND '2013-12-31'

**Задание 8**

Выясни, в каких странах находятся фонды, которые чаще всего инвестируют в стартапы. Для каждой страны посчитай минимальное, максимальное и среднее число компаний, в которые инвестировали фонды этой страны, основанные с 2010 по 2012 год включительно. Исключи страны с фондами, у которых минимальное число компаний, получивших инвестиции, равно нулю. Выгрузи десять самых активных стран-инвесторов: отсортируй таблицу по среднему количеству компаний от большего к меньшему. Затем добавь сортировку по коду страны в лексикографическом порядке.

SQL-Запрос:

SELECT country\_code,

MIN(invested\_companies),

MAX(invested\_companies),

AVG(invested\_companies)

FROM fund

WHERE CAST(founded\_at AS DATE) BETWEEN '2010-01-01' AND '2012-12-31'

GROUP BY country\_code

HAVING MIN(invested\_companies) > 0

ORDER BY AVG(invested\_companies) DESC,

country\_code

LIMIT 10;