**Осваиваем SQL на примере данных интернет-магазина Google**

АВТОР: [РОМАН РОМАНЧУК](http://thisisdata.ru/blog/author/roman-romanchuk/)

На заре своей веб-аналитической молодости, работая над составлением отчетов, я использовал только Excel.

То есть, например, бизнес говорит:

«Хочу увидеть отчет с воронкой продаж, начиная от посещения сайта и заканчивая получением товара в офисе».

Это сейчас я знаю Data Studio, практикую Power BI, SQL и еще много чего. А раньше, что я делал в таком случае? Открывал Google Analytics, создавал кастомный отчет с необходимыми параметрами и показателями и выгружал его в Excel. Далее шел к аналитику (не веб, а обычному, в банках такие есть) и просил выгрузить из базы все в тот же Excel, данные по клиентам посетившим офис, а после сводил две таблицы в единый отчет.

Так вот, этот путь тупиковый и если для малого бизнеса еще может подойти, то для больших данных не прокатит.

А как надо, спросите вы? «Изучайте SQL» — отвечу я!

Google BigQuery

Осваивать SQL мы будем на примере реальных данных электронной торговли магазина Google Merchandise Store, который продает товары под торговой маркой Google. Публичный датасет совсем недавно был выложен в BigQuery — облачную базу данных, которая позволяет обрабатывать терабайты данных за считанные секунды.

Чтобы получить доступ к набору данных:

1. Перейдите на страницу [http://bigquery.cloud.google.com](http://bigquery.cloud.google.com/).
2. Если вы новичок в BigQuery или у вас еще нет проекта, вам нужно будет [создать проект](https://console.cloud.google.com/projectcreate).
3. После того как проект создан, можно переходить непосредственно к [датасету Google Merchandise Store](https://bigquery.cloud.google.com/table/bigquery-public-data:google_analytics_sample.ga_sessions_20170801)

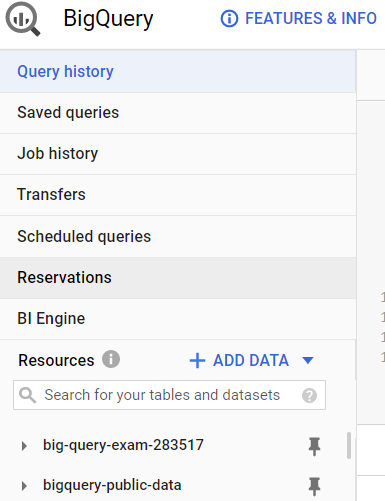
<https://cloud.google.com/bigquery/public-data#public-ui>



## Доступ к общедоступным наборам данных в Cloud Console: чтобы открыть bigquery-public-data проект с помощью Cloud Console, введите в браузере следующий URL-адрес:

## <https://console.cloud.google.com/bigquery?project=bigquery-public-data&page=project>

В результате в Вашем распоряжении будут данные из bigquery-public-data



Анализируемый набор данных содержит информацию о трафике, взаимодействии с контентом и транзакциях за период с 1 августа 2016 года по 1 августа 2017 года. Для каждого дня в наборе данных создается по одной таблице с названием в формате «ga\_sessions\_ГГГГММДД», а каждая строка таблицы содержит данные об одном сеансе ([схема данных](https://support.google.com/analytics/answer/3437719?hl=ru) в помощь).

Пишем SQL-запросы

SQL или structured query language — это язык структурированных запросов применяемый для создания, модификации и управления данными в реляционной базе данных.

В данной статье создавать и удалять мы ничего не будем, так что основным нашим оператором будет SELECT, оператор позволяющий выбирать данные, удовлетворяющие заданным условиям.

**Оператор SELECT состоит из нескольких предложений:**

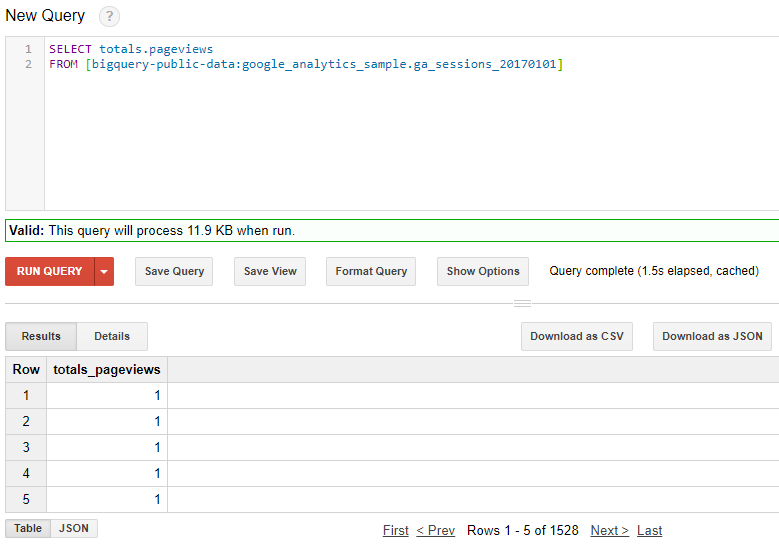
1. SELECT — определяет список возвращаемых столбцов (как существующих, так и вычисляемых).
2. FROM — указывает откуда (из какой таблицы и какого датасета) брать данные.

Все операторы и их параметры лучше писать заглавными буквами, для удобства восприятия кода. Но если напишите строчными, то код все равно будет работать.

Давайте потренируемся и попробуем получить общее количество просмотров страниц за 2017-01-01. Смотрим в схему данных, находим поле totals.pageviews и пишем запрос:

1. SELECT totals.pageviews
2. FROM [bigquery-public-data:google\_analytics\_sample.ga\_sessions\_20170101]

И получаем вот такой результат:

[](http://thisisdata.ru/wp-content/uploads/2018/04/img_5adb207e7726a.png)

Но что это? Вы ведь ожидали увидеть одну цифру с общим количеством просмотров страниц, а не 1528 строк. А все дело в том, что любая БД работает как КЭП (капитан очевидность) — что попросили, то и получили 🙂

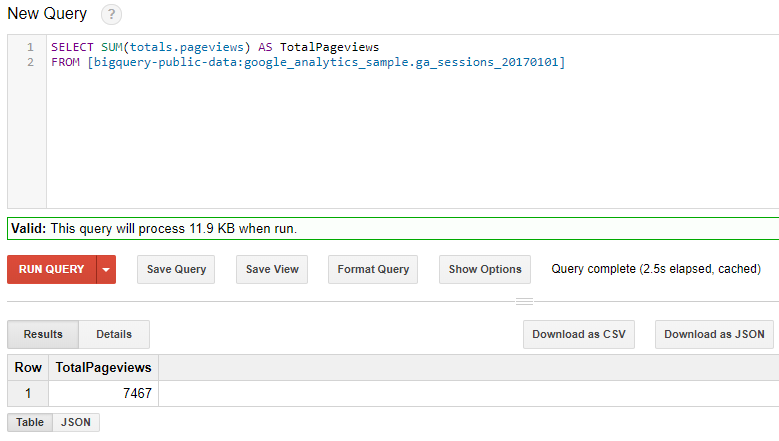
В  запросе вы сказали: «Выведи мне поле всего просмотров страниц из таблицы за 2017-01-01». На что в ответ и получили все строки таблицы, содержащие данные о количестве просмотров страниц.

А нужно было сформулировать запрос так: «Выведи мне СУММУ поля всего просмотров страниц из таблицы за 2017-01-01».

Пробуем:

1. SELECT SUM(totals.pageviews) AS TotalPageviews
2. FROM [bigquery-public-data:google\_analytics\_sample.ga\_sessions\_20170101]

То что нужно:

[](http://thisisdata.ru/wp-content/uploads/2018/04/img_5adb251584beb.png)

Также в запросе, помимо агрегируещей функции SUM, вы могли заметить параметр AS, который отвечает за пользовательское название столбца (если его не указать, то у столбца не будет имени, точнее будет примерно такое «f0\_»).

А что делать, если мы хотим вывести сумму просмотров страниц не за один день, а допустим за три дня? На помощь нам приходят операторы фильтрации, группировки и сортировки.

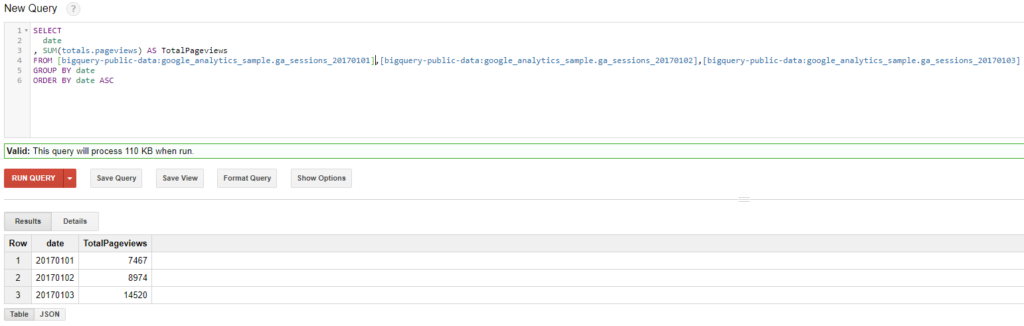
**Параметры оператора SELECT**

1. WHERE — фильтрует данные по заданным вами условиям.
2. GROUP BY — группирует строки по результатам агрегатных функций (MAX, SUM, AVG, …).
3. ORDER BY — сортирует значения по одному или более столбцам. Сортировка может производиться как по возрастанию, так и по убыванию значений. Параметр ASC (по умолчанию) устанавливает порядок сортировки по возрастанию, DESC по убыванию.

Дополняем наш запрос:

1. SELECT
2. date
3. , SUM(totals.pageviews) AS TotalPageviews
4. FROM [bigquery-public-data:google\_analytics\_sample.ga\_sessions\_20170101],[bigquery-public-data:google\_analytics\_sample.ga\_sessions\_20170102],[bigquery-public-data:google\_analytics\_sample.ga\_sessions\_20170103]
5. GROUP BY date
6. ORDER BY date ASC

И получаем просмотры страниц по дням, отсортированные по возрастанию:

[](http://thisisdata.ru/wp-content/uploads/2018/04/img_5adc78db06b4b.png)

**Standard SQL**

Давайте включим стандартный диалект. Для этого нужно нажать на кнопку «Show Options» и убрать галочку «Use Legacy SQL».

И попробуем выбрать тот же диапазон дат.

SELECT

date

, SUM(totals.pageviews) AS TotalPageviews

FROM `bigquery-public-data.google\_analytics\_sample.ga\_sessions\_\*`

WHERE \_TABLE\_SUFFIX BETWEEN '20170101' AND '20170107'

GROUP BY date

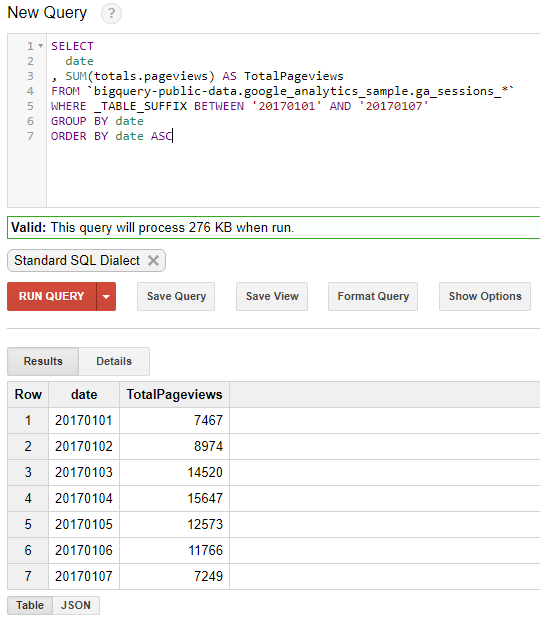
ORDER BY date ASC

В данном случае, вместо функции TABLE\_DATE\_RANGE в FROM используется шаблон названия таблицы, а в WHERE фильтр \_TABLE\_SUFFIX, который выбирает данные находящиеся в заданном диапазоне дат (BETWEEN).

Плюс обратите внимание на немного изменившуюся пунктуацию в предложении FROM.

Вместо квадратных скобок [ ], используйте ` `. А для того, чтобы отделить название проекта от названия таблицы, используйте точку ., вместо двоеточия :.

Результат тот же:

[](http://thisisdata.ru/wp-content/uploads/2018/04/img_5adca4c602ef1.png)

Подробнее о различиях между диалектами [читайте в справке Google](https://cloud.google.com/bigquery/docs/reference/standard-sql/migrating-from-legacy-sql) (на вражеском языке).

Теперь давайте применим все полученные навыки и сконструируем более сложный отчет.

Собираем отчет

Давайте посчитаем количество просмотров страниц, транзакций, конверсию из просмотра в транзакцию, средний чек и доход за неделю:

SELECT

date

, SUM(totals.pageviews) AS Pageviews

, ROUND((SUM(totals.transactions)/SUM(totals.pageviews))\*100,2) AS CR

, SUM(totals.transactions) AS Transactions

, ROUND(AVG(totals.totalTransactionRevenue)/1000000,2) AS AverageCheck

, ROUND(SUM(totals.totalTransactionRevenue)/1000000,2) AS Revenue

FROM `bigquery-public-data.google\_analytics\_sample.ga\_sessions\_\*`

WHERE \_TABLE\_SUFFIX BETWEEN '20170101' AND '20170107'

GROUP BY date

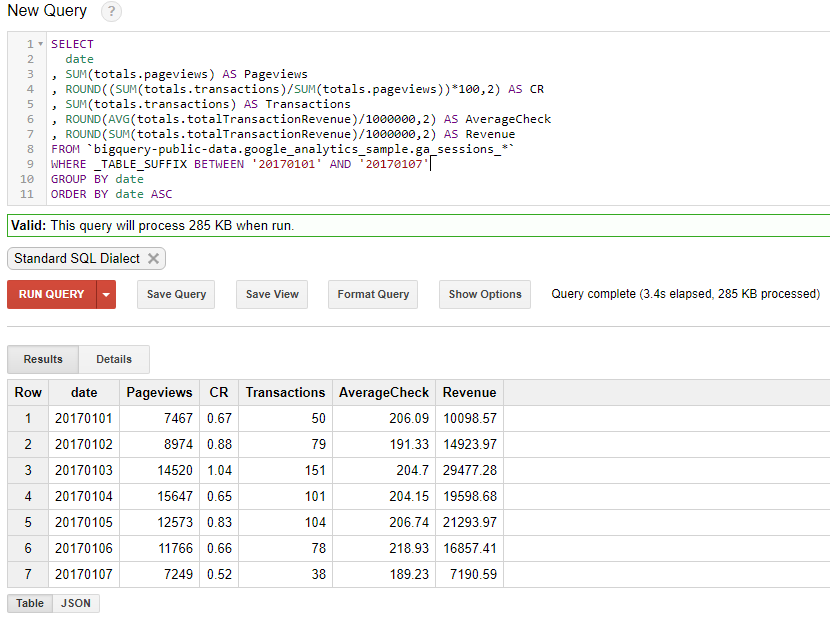
ORDER BY date ASC

Из нового:

ROUND — округляет число до стольких знаков после запятой, сколько указано во втором аргументе функции (в нашем случае до 2).

AVG — выводит среднее значение диапазона чисел.

Уже похоже на отчет:

[](http://thisisdata.ru/wp-content/uploads/2018/04/img_5adca609931a7.png)

Что дальше с этим делать? Пробуйте построить более сложные отчеты, скачивайте в CSV или экспортируйте в Google Sheets и Data Studio для анализа.

А я пока буду готовить [следующую статью](http://thisisdata.ru/blog/osvaivayem-sql-na-primere-dannykh-internet-magazina-google-2/), в которой расскажу о более продвинутых возможностях SQL.

**Полезные ссылки:**

* [Осваиваем SQL на примере данных интернет-магазина Google. Ч.2](http://thisisdata.ru/blog/osvaivayem-sql-na-primere-dannykh-internet-magazina-google-2/)
* [Справочник по BigQuery](https://support.google.com/analytics/answer/4419694?hl=ru)
* [Стандартные функции и операторы SQL](https://cloud.google.com/bigquery/docs/reference/standard-sql/functions-and-operators)
* [0x0b приемов работы с BigQuery на Standard SQL](https://burgerdata.com/blog/priemy-raboty-v-bigquery/)

**Осваиваем SQL на примере данных интернет-магазина Google. Ч.2**

АВТОР: [РОМАН РОМАНЧУК](http://thisisdata.ru/blog/author/roman-romanchuk/)

В [первой части](http://thisisdata.ru/blog/osvaivayem-sql-na-primere-dannykh-internet-magazina-google/) данной статьи мы учились подключаться к Google BigQuery, рассмотрели оператор SELECT и его параметры, попробовали написать простые запросы. Сегодня я хотел бы рассказать о более продвинутых возможностях SQL.

Давайте вспомним, что хотел от нас бизнес в первой части, а хотел он следующее:

«Хочу увидеть отчет с воронкой продаж, начиная от посещения сайта и заканчивая получением товара в офисе».

Чаще всего под воронкой продаж подразумевается путь пользователя по сайту перед оплатой заказа, сама оплата, а также статусы по оплаченному заказу (например из CRM).

Строим воронку продаж

Чтобы построить воронку продаж, нам нужно посчитать сколько пользователей побывали на каждом из ее этапов, а именно:

* посмотрели каталог товаров,
* перешли в корзину,
* далее на страницу оплаты,
* и как итог страницу «Спасибо за покупку».

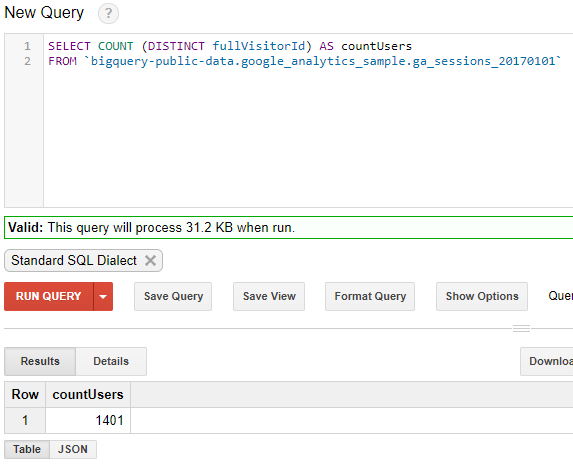
Я взял только эти страницы, как наиболее очевидные этапы воронки. На самом деле этапов может быть гораздо больше (регистрация, авторизация, сервисные страницы и прочее).

Для подсчета общего количества пользователей (поле fullVisitorId) достаточно воспользоваться функцией COUNT, которая возвращает количество значений. Если добавить к функции аргумент DISTINCT, то будет посчитано количество **уникальных** значений.

SELECT COUNT (DISTINCT fullVisitorId) AS countUsers

FROM `bigquery-public-data.google\_analytics\_sample.ga\_sessions\_20170101`

Посчитали:

[](http://thisisdata.ru/wp-content/uploads/2018/05/img_5aef1774aff00.png)

Отлично! Теперь нам нужно посчитать сколько пользователей побывало на каждом этапе воронки. А для этого нам потребуется комбинация из типов страниц (параметров) и количества пользователей (показателя). Поможет нам в этом функция CASE, которая в зависимости от указанных условий возвращает одно из множества возможных значений.

**Синтаксис CASE:**

CASE WHEN условия поиска THEN результат ELSE результат, если условия не найдены END

Чтобы посчитать страницы, нужно запросить поля hits.page.pagePath, однако вместо подсчета страниц вы можете использовать события расширенной электронной торговли hits.eCommerceAction.action\_type (более подробно [в схеме данных](https://support.google.com/analytics/answer/3437719?hl=ru)).

SELECT

CASE

WHEN h.page.pagePath LIKE '/google+redesign%' THEN 'Catalog'

WHEN h.page.pagePath LIKE '/basket%' THEN 'Basket'

WHEN h.page.pagePath LIKE '/payment%' THEN 'Payment'

WHEN h.page.pagePath LIKE '/ordercompleted%' THEN 'OrderCompleted'

ELSE 'Other'

END AS funnelStep

, COUNT (DISTINCT fullVisitorId) AS countUsers

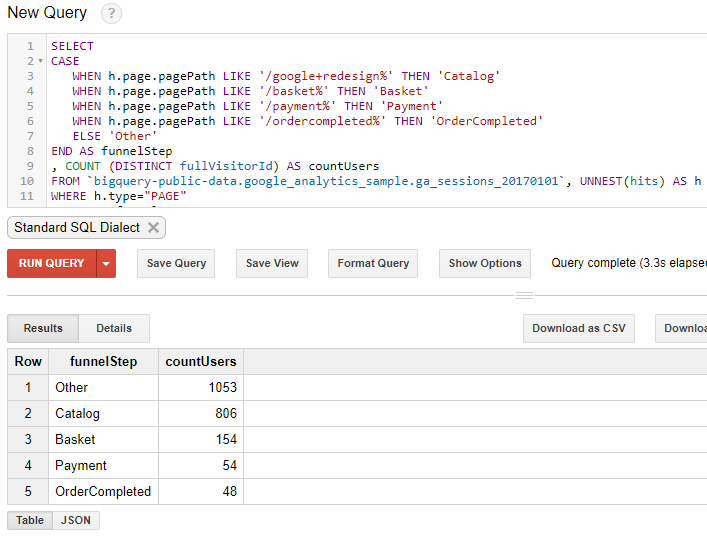
FROM `bigquery-public-data.google\_analytics\_sample.ga\_sessions\_20170101`, UNNEST(hits) AS h

WHERE h.type="PAGE"

GROUP BY funnelStep

ORDER BY countUsers DESC

Уже похоже на воронку:

[](http://thisisdata.ru/wp-content/uploads/2018/05/img_5af849a72007f.png)

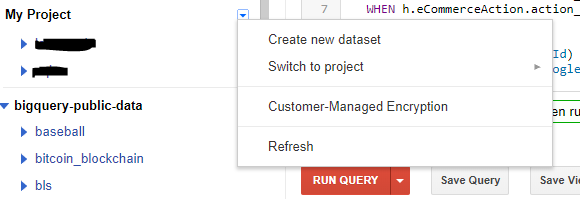
К шагу воронки «Other» относятся все остальные страницы, не попавшие в заданные условия (главная, сервисные, авторизация, регистрация и прочее), избавимся от него позже.

Теперь мы подошли к самому интересному, в воронку нужно добавить оффлайн статусы по оплаченным заказам (доставляются ли они или уже получены клиентом). Так как в датасете от Google такая возможность вообще не предусмотрена, предлагаю загрузить данные в собственный набор данных и потом использовать их при построении воронки.

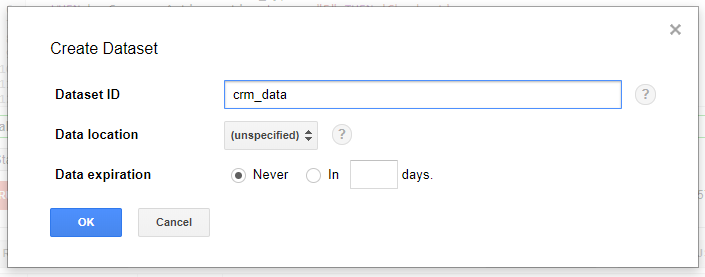
Загружаем данные в Google BigQuery

Представим, что у нас есть CRM, данные из которой мы хотим соединить с данными в BigQuery. Чтобы их загрузить, будем использовать Google Sheets.

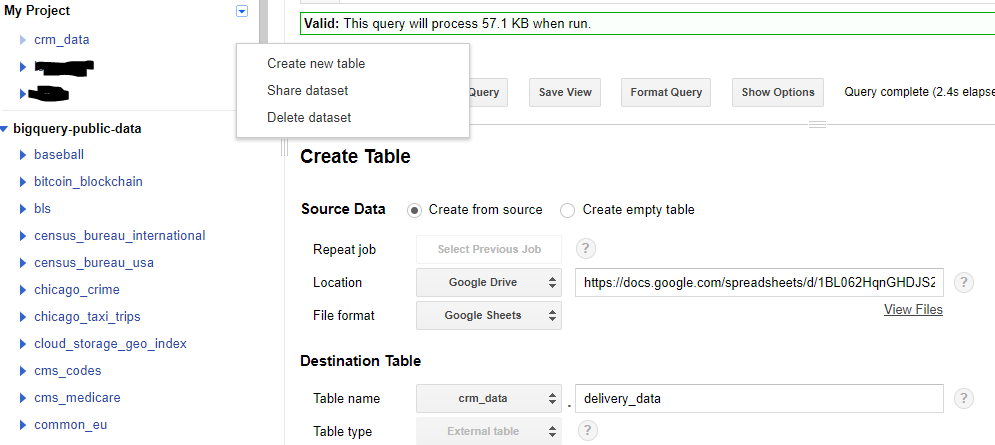
1. Заходим в [BigQuery](http://bigquery.cloud.google.com/).
2. Создаем новый датасет.

[](http://thisisdata.ru/wp-content/uploads/2018/05/img_5aef6088c3397.png)

1. Придумываем название и нажимаем «Ок».

[](http://thisisdata.ru/wp-content/uploads/2018/05/img_5aef61041cca7.png)

1. Теперь добавим в датасет таблицу с данными из CRM (ссылка на [файл с тестовыми данными](https://docs.google.com/spreadsheets/d/1BL062HqnGHDJS2s-gEQkV1MZbp06zGhHa3PjK-c68cw/edit?usp=sharing)).

[](http://thisisdata.ru/wp-content/uploads/2018/05/img_5aef6607b36ba.png)

1. Теперь нужно описать структуру таблицы, то есть ввести название и типы данных для каждого поля.

[](http://thisisdata.ru/wp-content/uploads/2018/05/img_5aef6481a8140.png)

1. Готово! Теперь данные из нашей «CRM» можно использовать для JOIN.

Соединяем данные нескольких таблиц

Чтобы достроить нашу воронку до конца, необходимо соединить данные из таблиц ga\_sessions и delivery\_data. А для этого нужно использовать оператор JOIN, который соединяет данные из разных таблиц по какому-либо единому для этих таблиц идентификатору.

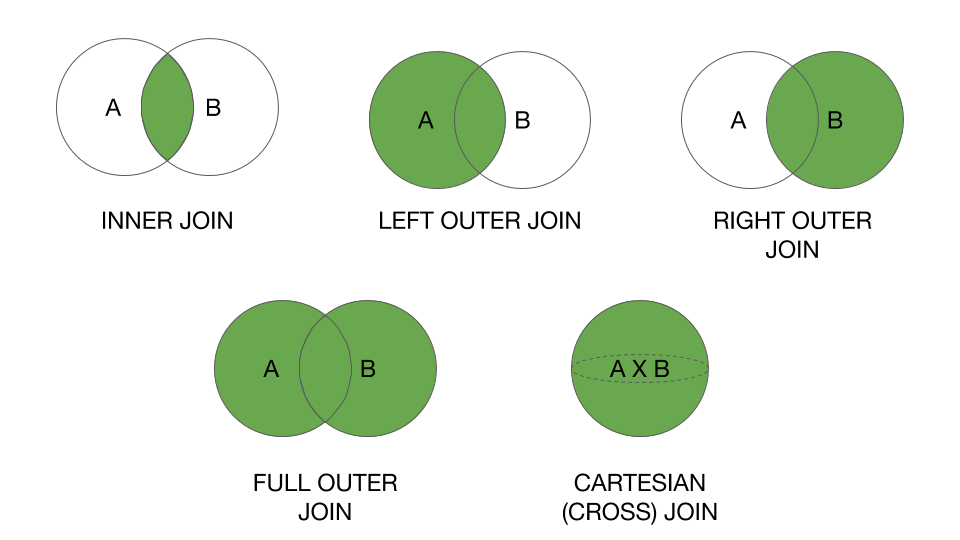
Чтобы связать две таблицы в предложении FROM нужно написать:

table\_1 LEFT JOIN table\_2 ON table\_1.id = table\_2.id

**Виды оператора JOIN:**

1. INNER JOIN — при помощи этого соединения, вы получите записи присутствующие в обеих таблицах.
2. OUTER JOIN — соединение,  в результат которого входят все записи либо одной, либо обеих таблиц:
   * LEFT OUTER JOIN — возвращает все записи из таблицы слева и соединяет их со связанными записями из правой таблицы (именно такое соединение мы используем).
   * RIGHT OUTER JOIN — тоже самое, что и предыдущее соединение, только все записи возвращаются из присоединяемой таблицы.
   * FULL OUTER JOIN — возвращает все записи обеих таблиц, там где нет пересечений возвращается NULL.
3. CROSS JOIN — перекрестное соединение при котором каждая запись одной таблицы соединяется с каждой записью второй таблицы, давая тем самым в результате все возможные сочетания строк двух таблиц.

Визуально это выглядит так:

[](http://thisisdata.ru/wp-content/uploads/2018/05/join-types.png)

Подробнее читайте в [документации](https://cloud.google.com/bigquery/docs/reference/standard-sql/query-syntax#join-types).

Теперь можем смело добавить в нашу воронку статусы заказов.

SELECT

CASE

WHEN h.page.pagePath LIKE '/google+redesign%' THEN 'Catalog'

WHEN h.page.pagePath LIKE '/basket%' THEN 'Basket'

WHEN h.page.pagePath LIKE '/payment%' THEN 'Payment'

WHEN h.page.pagePath LIKE '/ordercompleted%' THEN 'OrderCompleted'

WHEN deliveryStatus = 'inTransit' THEN 'InTransit'

WHEN deliveryStatus = 'delivered' THEN 'Delivered'

ELSE 'Other'

END AS funnelStep

, COUNT (DISTINCT `bigquery-public-data.google\_analytics\_sample.ga\_sessions\_20170101`.fullVisitorId) AS countUsers

FROM `bigquery-public-data.google\_analytics\_sample.ga\_sessions\_20170101`, UNNEST(hits) AS h

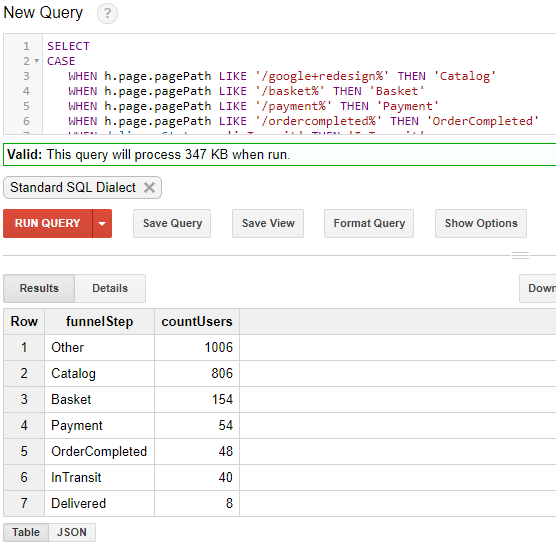
LEFT JOIN `alert-snowfall-167320.crm\_data.delivery\_data` ON `bigquery-public-data.google\_analytics\_sample.ga\_sessions\_20170101`.fullVisitorId = `alert-snowfall-167320.crm\_data.delivery\_data`.fullVisitorId

WHERE h.type="PAGE"

GROUP BY funnelStep

ORDER BY countUsers DESC

Получилось:

[](http://thisisdata.ru/wp-content/uploads/2018/05/img_5af86375df9bd.png)

Осталось только немного оптимизировать запрос, так как в текущем виде он работает достаточно медленно. Для этого разобьем наш сложный запрос на более простые подзапросы, а также воспользуемся оператором WITH, который позволяет именовать подзапросы (подробнее [у Севы](https://burgerdata.com/blog/priemy-raboty-v-bigquery/#%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0-%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D0%B0) и [в доке](https://cloud.google.com/bigquery/docs/reference/standard-sql/query-syntax#with-clause)).

WITH table\_1 AS (

SELECT

h.page.pagePath AS pageType

, fullVisitorId AS id

FROM `bigquery-public-data.google\_analytics\_sample.ga\_sessions\_20170101`, UNNEST(hits) AS h

WHERE h.type="PAGE"

),

table\_2 AS (

SELECT

deliveryStatus

, fullVisitorId **as** id

FROM `alert-snowfall-167320.crm\_data.delivery\_data`

),

table\_join AS (

SELECT

CASE

WHEN pageType LIKE '/google+redesign%' THEN 'Catalog'

WHEN pageType LIKE '/basket%' THEN 'Basket'

WHEN pageType LIKE '/payment%' THEN 'Payment'

WHEN pageType LIKE '/ordercompleted%' THEN 'OrderCompleted'

WHEN deliveryStatus = 'inTransit' THEN 'InTransit'

WHEN deliveryStatus = 'delivered' THEN 'Delivered'

ELSE 'Other'

END AS funnelStep

, COUNT (DISTINCT table\_1.id) AS countUsers

FROM table\_1

LEFT JOIN table\_2 ON table\_1.id = table\_2.id

GROUP BY funnelStep

ORDER BY countUsers DESC

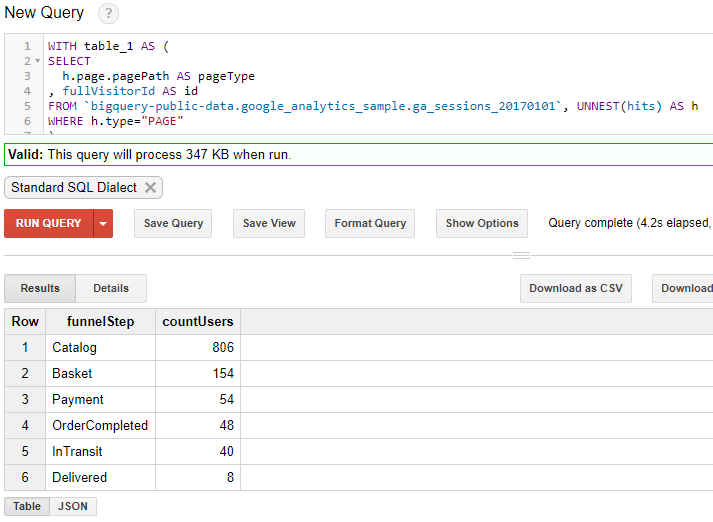
)

SELECT \*

FROM table\_join

WHERE funnelStep <> 'Other'

Окончательная воронка:

[](http://thisisdata.ru/wp-content/uploads/2018/05/img_5af86d0e3189f.png)

Отлично, а что дальше?

Моей задачей было познакомить вас с основными операторами языка SQL и показать как работает BigQuery (надеюсь справился).

Дальше вы можете попробовать строить свои воронки и отчеты, детализировать пути пользователей по сайту, делать аналитику по конкретному пользователю или визуализировать все это в Data Studio. А самое главное попытаться применить эти знания на практике. Успехов!

**Полезные ссылки:**

* [Осваиваем SQL на примере данных интернет-магазина Google. Ч.1](http://thisisdata.ru/blog/osvaivayem-sql-na-primere-dannykh-internet-magazina-google/)
* [Как визуализировать воронку продаж в Google Data Studio?](http://thisisdata.ru/blog/kak-vizualizirovat-voronku-prodazh-v-google-data-studio/)
* [0x0b приемов работы с BigQuery на Standard SQL](https://burgerdata.com/blog/priemy-raboty-v-bigquery/)
* [Синтаксис запросов для Standard SQL](https://cloud.google.com/bigquery/docs/reference/standard-sql/query-syntax)
* [Интерактивный учебник по SQL](http://www.sql-tutorial.ru/ru/content.html)
* [Понимание SQL](http://www.sql.ru/docs/sql/u_sql/)
* [Разговорный SQL. Практический курс](https://minyurov.com/2015/03/15/talking-sql/)