**Usage Funnels with SQL**

**What is a funnel?**

In marketing analysis, a **funnel** refers to the customer journey from initial awareness to final conversion (e.g., making a purchase, signing up, or another desired action). It visualizes how potential customers move through different stages, helping businesses identify drop-off points and optimize strategies for better conversion rates.

Diagrama

El contenido generado por IA puede ser incorrecto.

**Why is Funnel Analysis Important?**

* Identify where users drop off in the journey.
* Help optimize marketing strategies for better engagement and conversions.
* Inform decision-making on ad spending, content creation, and customer experience improvements.

**Funnel Analysis**

Throughout this project, we will be working with data from a fictional company called Cushion Co. Using SQL, you can dive into complex funnels and event flow analysis to gain insights into their users’ behavior.

**Build a Funnel from a single table**

Cushion Co. users were asked to answer a five-question survey:

1. How likely are you to recommend Cushion Co. to a friend?
2. Which Cushion Co. location do you shop at?
3. How old are you?
4. What is your gender?
5. What is your annual household income?

We will be using a table called **survey\_responses** with the following columns:

* **question\_text** - the survey question
* **user\_id** - the user identifier
* **response** - the user answer

Let’s begin by exploring the **survey\_responses** table to understand its structure and contents.

**SQL Query**

**SELECT** \*

**FROM** survey\_responses

**LIMIT** 10;



Now, let’s build our first basic funnel. ***What is the number of responses for each question?***

**SQL Query**

**SELECT** question\_text, **COUNT**(**DISTINCT** user\_id) **AS** Users

**FROM** survey\_responses

**GROUP** BY 1;

|  |  |
| --- | --- |
| **question\_text** | **Users** |
| 1. How likely are you to recommend Cushion Co. to a friend? | 500 |
| 2. Which Mattresses and More location do you shop at? | 475 |
| 3. How old are you? | 380 |
| 4. What is your gender? | 361 |
| 5. What is your annual household income? | 270 |

**Survey Result.**

If we calculate the ratio of people completing each step to those who completed the previous step:

|  |  |  |
| --- | --- | --- |
| **Question** | **Users** | **Percent Completed this Question** |
| 1 | 500 | 100% |
| 2 | 475 | 95% |
| 3 | 380 | 80% |
| 4 | 361 | 95% |
| 5 | 270 | 75% |

Questions 2 and 4 show high completion rates, while Questions 3 and 5 have lower rates. This indicates that age and household income may be more sensitive topics, leading to reluctance in responding.

**Compare Funnels For A/B Tests**

Cushion Co. has an onboarding workflow for new website users, utilizing modal pop-ups to greet them and highlight key site features such as:

1. Welcome to Cushion Co.
2. Browse our bedding selection
3. Select items to add to your cart
4. View your cart by clicking on the icon
5. Press ‘Buy Now!’ when you’re ready to checkout

The Product team at Cushion Co. has created a new design for the pop-ups that they believe will lead more users to complete the workflow.

They’ve set up an **A/B test** where:

* 50% of users view the original **control** version of the pop-ups
* 50% of users view the new **variant** version of the pop-ups

Eventually, we want to answer the question:

***How is the funnel different between the two groups?***

We will be using a table called **onboarding\_modals** with the following columns:

* **user\_id** - the user identifier
* **modal\_text** - the modal step
* **user\_action** - the user response (Close Modal or Continue)
* **ab\_group** - the version (control or variant)

Let’s start by exploring the **onboarding\_modals** table.

**SQL Query**

**SELECT** \*

**FROM** onboarding\_modals

**LIMIT** 10;



Now, using GROUP BY we can count the number of distinct **user\_id**‘s for each value of **modal\_text**. This will tell us the number of users completing each step of the funnel.

**SQL Query**

**SELECT** modal\_text, **COUNT**(**DISTINCT** user\_id) **AS** Users

**FROM** onboarding\_modals

**GROUP** BY **1**;

|  |  |
| --- | --- |
| **modal\_text** | **Users** |
| 1 - Welcome to Cushion Co. | 1000 |
| 2 - Browse our bedding selection | 695 |
| 3 - Select items to add to your cart | 575 |
| 4 - View your cart by clicking on the icon | 447 |
| 5 - Press 'Buy Now!' when you're ready to checkout | 379 |

The previous query combined both the control and variant groups. We can use a **CASE** statement within our **COUNT()** aggregate function to count the  **user\_ids** whose **ab\_group** is equal to ‘**control**’ and ‘**variant**’.

**SQL Query**

**SELECT** modal\_text,

**COUNT**(**DISTINCT CASE**

**WHEN** ab\_group = 'control' **THEN** user\_id

**END**) **AS** 'control\_clicks',

**COUNT**(**DISTINCT CASE**

**WHEN** ab\_group = 'variant' **THEN** user\_id

**END**) **AS** 'variant\_clicks'

**FROM** onboarding\_modals

**GROUP BY** **1**

**ORDER BY** **1**;

|  |  |  |
| --- | --- | --- |
| **modal\_text** | **control\_clicks** | **variant\_clicks** |
| 1 - Welcome to Cushion Co. | 500 | 500 |
| 2 - Browse our bedding selection | 301 | 394 |
| 3 - Select items to add to your cart | 239 | 336 |
| 4 - View your cart by clicking on the icon | 183 | 264 |
| 5 - Press 'Buy Now!' when you're ready to checkout | 152 | 227 |

**A/B Tests Results**

After some quick math:

|  |  |  |
| --- | --- | --- |
| **Modal** | **Control Percent** | **Variant Percent** |
| 1 | 100% | 100% |
| 2 | 60% | 79% |
| 3 | 80% | 85% |
| 4 | 80% | 80% |
| 5 | 85% | 85% |

* In Modal 2, the variant achieves a 79% completion rate, surpassing the control's 60%.
* In Modal 3, the variant reaches 85% completion, compared to the control's 80%.
* All other steps maintain the same completion levels.

These results indicate that the variant leads to higher completion rates.

**Build a Funnel from Multiple Tables.**

Cushion Co. sell essential bedding from their e-commerce store. Their purchase funnel is:

1. The user browses products and adds them to their cart
2. The user proceeds to the checkout page
3. The user enters credit card information and makes a purchase

As a sales analyst, you want to analyze shopping trends in the days leading up to Christmas. You suspect that as the holiday nears, customers are more likely to complete their purchases, transitioning from browsing to buying gifts.

The data for Cushion Co. is spread across several tables:

* **browse (b)** - each row in this table represents an item that a user has added to his shopping cart.
* **checkout (c)** - each row in this table represents an item in a cart that has been checked out.
* **purchase (p)** - each row in this table represents an item that has been purchased.

Let’s examine each table.

**SQL**

**SELECT** \*

**FROM** browse

**LIMIT** **5**;

|  |  |  |
| --- | --- | --- |
| **user\_id** | **browse\_date** | **item\_id** |
| 336f9fdc-aaeb-48a1-a773-e3a935442d45 | 12/20/2017 | 3 |
| 336f9fdc-aaeb-48a1-a773-e3a935442d45 | 12/20/2017 | 22 |
| 336f9fdc-aaeb-48a1-a773-e3a935442d45 | 12/20/2017 | 25 |
| 336f9fdc-aaeb-48a1-a773-e3a935442d45 | 12/20/2017 | 24 |
| 4596bb1a-7aa9-4ac9-9896-022d871cdcde | 12/20/2017 | 0 |

**SELECT** \*

**FROM** checkout

**LIMIT** **5**;

|  |  |  |
| --- | --- | --- |
| **user\_id** | **checkout\_date** | **item\_id** |
| 2fdb3958-ffc9-4b84-a49d-5f9f40e9469e | 12/20/2017 | 26 |
| 2fdb3958-ffc9-4b84-a49d-5f9f40e9469e | 12/20/2017 | 24 |
| 3a3e5fe6-39a7-4068-8009-3b9f649cb1aa | 12/20/2017 | 7 |
| 3a3e5fe6-39a7-4068-8009-3b9f649cb1aa | 12/20/2017 | 6 |
| 3a3e5fe6-39a7-4068-8009-3b9f649cb1aa | 12/20/2017 | 12 |

**SELECT** \*

**FROM** purchase

**LIMIT** **5**;

|  |  |  |
| --- | --- | --- |
| **user\_id** | **purchase\_date** | **item\_id** |
| 2fdb3958-ffc9-4b84-a49d-5f9f40e9469e | 12/20/2017 | 26 |
| 2fdb3958-ffc9-4b84-a49d-5f9f40e9469e | 12/20/2017 | 24 |
| 3a3e5fe6-39a7-4068-8009-3b9f649cb1aa | 12/20/2017 | 7 |
| 3a3e5fe6-39a7-4068-8009-3b9f649cb1aa | 12/20/2017 | 6 |
| 3a3e5fe6-39a7-4068-8009-3b9f649cb1aa | 12/20/2017 | 12 |

Now we need to combine the information from the three tables into one table with the following schema:

|  |  |  |  |
| --- | --- | --- | --- |
| **browser\_date** | **user\_id** | **is\_checkout** | **is\_purchase** |
| 12/20/2017 | 6a7617321513 | True | False |
| 12/20/2017 | 022d871cdcde | False | False |

Each row will represent a single user:

* If the user has any entries in checkout, then **is\_checkout** will be True.
* If the user has any entries in purchase, then **is\_purchase** will be True.

If we use an **INNER JOIN** to create this table, we’ll lose information from any customer who does not have a row on the checkout or purchase table. Therefore, we’ll need to use a series of **LEFT JOIN** commands.

Let’s proceed with the LEFT JOIN for three tables, including the following columns to obtain the desired schema.

* DISTINCT **b.browse\_date**
* **b.user\_id**
* c.user\_id IS NOT NULL AS '**is\_checkout**'
* p.user\_id IS NOT NULL AS '**is\_purchase**'

**\*HINT**

For review, IS NOT NULL will return:

* 1 (True) if a non-empty value is found
* 0 (False) if a NULL value is found

If a **user\_id** is not in the **checkout** table (aliased as **c**), then **b.user\_id** will be filled in, but **c.user\_id** will be NULL because of our LEFT JOIN.

**SQL**

**SELECT** **DISTINCT** b.browse\_date,

b.user\_id,

c.user\_id **IS** **NOT** **NULL** **AS** 'is\_checkout',

p.user\_id **IS** **NOT** **NULL** **AS** 'is\_purchase'  
**FROM** browse **AS** 'b'  
**LEFT** **JOIN** checkout 'c'  
  **ON** c.user\_id = b.user\_id  
**LEFT** **JOIN** purchase 'p'  
  **ON** p.user\_id = c.user\_id  
**LIMIT 10**;

|  |  |  |  |
| --- | --- | --- | --- |
| **browse\_date** | **user\_id** | **is\_checkout** | **is\_purchase** |
| 12/20/2017 | 336f9fdc-aaeb-48a1-a773-e3a935442d45 | 0 | 0 |
| 12/20/2017 | 4596bb1a-7aa9-4ac9-9896-022d871cdcde | 0 | 0 |
| 12/20/2017 | 2fdb3958-ffc9-4b84-a49d-5f9f40e9469e | 1 | 1 |
| 12/20/2017 | fc394c75-36f1-4df1-8665-23c32a43591b | 0 | 0 |
| 12/20/2017 | 263e59f2-479b-4736-872c-302ad082b20f | 0 | 0 |
| 12/20/2017 | 58ff3291-84bf-4fc7-96cc-0bc1477adea9 | 0 | 0 |
| 12/20/2017 | d582b899-cace-43dc-84f3-a1df0c30e90c | 0 | 0 |
| 12/20/2017 | 3215212f-7a6f-4d95-937a-ee0ce911db04 | 0 | 0 |
| 12/20/2017 | d0768167-da9c-4209-b3e6-5c6fc446bece | 0 | 0 |
| 12/20/2017 | 182fcdb3-babd-4ade-ae6d-c3d6f30ffcde | 0 | 0 |

Let’s put the whole thing in a **WITH** statement so that we can continue building our query. We will give the temporary table the name **funnels**:

**SQL**

**WITH** funnels **AS** **(**

**SELECT** **DISTINCT** b.browse\_date,

b.user\_id,

c.user\_id **IS** **NOT** **NULL** **AS** **'is\_checkout'**,

p.user\_id **IS** **NOT** **NULL** **AS** **'is\_purchase'**

**FROM** browse **AS** 'b'

**LEFT** **JOIN** checkout **AS** 'c'

**ON** c.user\_id = b.user\_id

**LEFT** **JOIN** purchase **AS** 'p'

**ON** p.user\_id = c.user\_id**)**

Now, let’s add two columns that sum **is\_checkout** and **is\_purchase** in **funnels**. Alias these columns as ‘**num\_checkout**’ and ‘**num\_purchase**’. This will be the number of users in the “**checkout**” and “**purchase**” steps of the funnel.

**SQL**

**WITH** funnels **AS** **(**

**SELECT** **DISTINCT** b.browse\_date,

b.user\_id,

c.user\_id **IS** **NOT** **NULL** **AS** **'is\_checkout'**,

p.user\_id **IS** **NOT** **NULL** **AS** **'is\_purchase'**

**FROM** browse **AS** 'b'

**LEFT** **JOIN** checkout **AS** 'c'

**ON** c.user\_id = b.user\_id

**LEFT** **JOIN** purchase **AS** 'p'

**ON** p.user\_id = c.user\_id**)**

**SELECT** **COUNT(**\***)** **AS** **'num\_browse'**,

**SUM(**is\_checkout**)** **AS** **'num\_checkout'**,

**SUM(**is\_purchase**)** **AS** **'num\_purchase'**

**FROM** funnels;

|  |  |  |
| --- | --- | --- |
| **num\_browse** | **num\_checkout** | **num\_purchase** |
| 775 | 183 | 163 |

Finally, let’s do add some more calculations to make the results more in depth.

Let’s add these two columns:

* Percentage of users from browse to checkout.
* Percentage of users from checkout to purchase.

**WITH** funnels **AS** **(**

**SELECT** **DISTINCT** b.browse\_date,

b.user\_id,

c.user\_id **IS** **NOT** **NULL** **AS** **'is\_checkout'**,

p.user\_id **IS** **NOT** **NULL** **AS** **'is\_purchase'**

**FROM** browse **AS** 'b'

**LEFT** **JOIN** checkout **AS** 'c'

**ON** c.user\_id = b.user\_id

**LEFT** **JOIN** purchase **AS** 'p'

**ON** p.user\_id = c.user\_id**)**

**SELECT** **COUNT(**\***)** **AS** **'num\_browse'**,

**SUM(**is\_checkout**)** **AS** **'num\_checkout'**,

**SUM(**is\_purchase**)** **AS** **'num\_purchase',**

**1.0** \* **SUM**(is\_checkout) / **COUNT**(user\_id) **AS** **'% browse to checkout'**,

**1.0** \* **SUM**(is\_purchase) / **SUM**(is\_checkout) **AS** **'% checkout to purchase'**

**FROM** funnels;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **num\_browse** | **num\_checkout** | **num\_purchase** | **% browse to checkout** | **% checkout to purchase** |
| 775 | 183 | 163 | 0.236129032 | 0.890710383 |

The Marketing team and Service Analyst at Cushion Co. should focus their strategies on increasing the number of users who move from browsing items to completing a checkout. Since the conversion rate from checkout to purchase is already high, improving the browse-to-checkout transition can significantly boost overall sales.

**\*HINT**

By default, mathematical operations between integers will round down to the nearest whole number. To prevent this rounding, it's common practice to multiply the result by **1.0**. This approach ensures the outcome is a decimal value instead of an integer.

We’ve created a funnel for Cushion Co.’s purchase process. However, the management team suspects that conversion from checkout to purchase changes as the **browse\_date** gets closer to Christmas Day.

We can make a few edits to this code to calculate the funnel for each **browse\_date** using **GROUP BY**.

**SQL**

**WITH** funnels **AS** **(**

**SELECT** **DISTINCT** b.browse\_date,

b.user\_id,

c.user\_id **IS** **NOT** **NULL** **AS** **'is\_checkout'**,

p.user\_id **IS** **NOT** **NULL** **AS** **'is\_purchase'**

**FROM** browse **AS** 'b'

**LEFT** **JOIN** checkout **AS** 'c'

**ON** c.user\_id = b.user\_id

**LEFT** **JOIN** purchase **AS** 'p'

**ON** p.user\_id = c.user\_id**)**

**SELECT** browse\_date,

**COUNT(**\***)** **AS** **'num\_browse'**,

**SUM(**is\_checkout**)** **AS** **'num\_checkout'**,

**SUM(**is\_purchase**)** **AS** **'num\_purchase',**

**1.0** \* **SUM**(is\_checkout) / **COUNT**(user\_id) **AS** **'% browse to checkout'**,

**1.0** \* **SUM**(is\_purchase) / **SUM**(is\_checkout) **AS** **'% checkout to purchase'**

**FROM** funnels

**GROUP** **BY** browse\_date

**ORDER** **BY** browse\_date;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **browse\_date** | **num\_browse** | **num\_checkout** | **num\_purchase** | **browse\_to\_checkout** | **checkout\_to\_purchase** |
| 12/20/2017 | 100 | 20 | 16 | 0.2 | 0.8 |
| 12/21/2017 | 150 | 33 | 28 | 0.22 | 0.848484848 |
| 12/22/2017 | 250 | 62 | 55 | 0.248 | 0.887096774 |
| 12/23/2017 | 275 | 68 | 64 | 0.247272727 | 0.941176471 |

As Christmas approaches, the conversion rates from browsing to checkout and from checkout to purchase increase significantly. Cushion Co. can leverage this trend by implementing targeted promotions and customer loyalty strategies to attract more users during the holiday season—not just for Christmas, but for other key shopping periods as well.