Website Performance Analysis

In this section we'll be diving deeper into the fuzzy factory website to understand where customers are landing on the website and how they make their way through the conversion funnel on the path to placing an order.

Analyzing Top website content

Website content analysis is all about understanding which pages are seen the most so that we can identify where we should focus on improving the business.

Common Use Cases:

- Finding the most viewed pages.
- Identifying the most common entry pages for your website.
- Once we identify the most viewed pages and the most viewed entry pages we'll look at how well they're performing so that we can see if they're meeting our business objectives or if we should do some work on them so that they can improve for this section.

Top Website Pages | SQL

```sql
-- Top Website Pages

SELECT
    pageview_url,
    COUNT(DISTINCT website_pageview_id) AS views
FROM website_pageviews
WHERE created_at < '2012-06-09'
GROUP BY 1
ORDER BY 2 DESC;
```

100% 17:13

Result Grid | Filter Rows: Search | Export:

pageview_url	views
/home	10403
/products	4239
/the-original-mr-fuzzy	3037
/cart	1306
/shipping	869
/billing	716
/thank-you-for-your-order	306

pageview_url: url of the website page
website_pageviews: Pageviews table. A tuple is generated whenever a customer visits a page
website_pageview_id: Primary key in website pageview ids

Findings:

- What we see the home page gets the vast majority of the page used during this time period followed by the products and the original Mr. fuzzy page.

Top Entry Pages | SQL

```sql
/* Step 1 Find the website_session_id and
respective minimum page view id [Landing Page] */

CREATE TEMPORARY TABLE landing_pages
SELECT
    website_session_id,
    MIN(website_pageview_id) AS first_pageview_id
FROM website_pageviews
WHERE created_at < '2012-06-12'
GROUP BY 1;

SELECT * FROM landing_pages;
```

100% 1:22

Result Grid | Filter Rows: Search | Export: | Fetch

website_session...	first_pageview...
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	12

```sql
/* Step 2 Identify the pageview_url for the
first_pageview_id */
CREATE TEMPORARY TABLE landing_urls
SELECT
    L.website_session_id,
    W.pageview_url AS landing_page
FROM landing_pages AS L
    LEFT JOIN
        website_pageviews AS W
            ON L.first_pageview_id = W.website_pageview_id;
    -- The above join gets the pageview_url of landing sessions
    -- only

SELECT * FROM landing_urls;
```

100% 28:46

Result Grid | Filter Rows: Search | Export: | Fetch rows:

website_session...	landing_page
1	/home
2	/home
3	/home
4	/home
5	/home
6	/home
7	/home
8	/home

```sql
SELECT
    landing_page,
    COUNT(website_session_id) AS sessions
FROM landing_urls
GROUP BY 1;
```

100% 12:54

Result Grid | Filter Rows: Search | Export:

landing_page	sessions
/home	10714

.Findings:

- The landing page URL of the home page is getting all of the sessions at this point in the life of the business.

So that's a pretty obvious place again to focus where every customer is seeing for the first time you probably want to spend a good amount of time making sure that page is as great as it could be.

Landing Page Performance and Testing

Landing page performance and optimization testing landing page analysis and testing is all about understanding the performance of your key landing pages and then running experiments or tests to move the needle on your key business objectives.

In a real business we might say could we run another landing page that would perform better the way that we figure this out is we take our customers and we randomize some of them going to landing page A and some of them going to another experience in this case landing page B
By introducing landing page B and by running this experiment we can figured out how to improve our business.
Common Use Cases:

- Landing page bounce rate analysis to identify your top opportunities for optimization
- Can set up AB experiments of your live traffic, you and analyze test results to see if that will improve bounce rates and session to order conversion rates.

Bounce Rate Analysis | SQL

```sql
-- Bounce Rate Analysis

/* Step 1 Need to identify the landing sessions */

CREATE TEMPORARY TABLE landing_sessions
SELECT
    website_session_id,
    MIN(website_pageview_id) AS first_pageview_id
    /*MIN(website_pageview_id)
     Identifies the first page view id of each website session
     AKA landing page view */
FROM website_pageviews
WHERE created_at < '2012-06-14'
GROUP BY 1;

SELECT * FROM landing_sessions;
```

100% 30:75

Result Grid Filter Rows: Search Export: Fetch rows:

website_session...	first_pageview...
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	12

```sql
/* Step 2
Left Join to the website_pageviews table
Get the landing page url of each website session */

CREATE TABLE landing_pages
SELECT
    L.website_session_id AS website_session_id,
    W.pageview_url AS pageview_url
FROM landing_sessions AS L
    LEFT JOIN website_pageviews AS W
    ON L.first_pageview_id = W.website_pageview_id;

SELECT * FROM landing_pages;
```

100% 24:89

Result Grid Filter Rows: Search Export: Fetch rows

website_session...	pageview_url
1	/home
2	/home
3	/home
4	/home
5	/home
6	/home
7	/home

```sql
/* Step 3
Identifying bounce sessions, by left joining to website_pageviews */
CREATE TABLE bounced_sessions
SELECT
    L.website_session_id AS website_session_id,
    L.pageview_url AS pageview_url,
    COUNT(W.website_pageview_id) AS pageviews
    -- the above column will give us no pageviews for each session
FROM landing_pages AS L
     LEFT JOIN website_pageviews AS W
     ON L.website_session_id = W.website_session_id
GROUP BY 1,2
HAVING COUNT(W.website_pageview_id) = 1;
-- We are limiting to only one sessions
-- bounce means they visited the landing_page but didn't go anyfuther in funnel.
SELECT * FROM bounced_sessions;
```

100% 28:106

Result Grid Filter Rows: Search Export: Fetch rows:

website_session...	pageview_url	pageviews
1	/home	1
2	/home	1
3	/home	1
4	/home	1
5	/home	1
7	/home	1

```sql
/* Step 4
Left joining the landing_pages and bounced_sessions
And using case and pivot to find bounce rate analysis  */

SELECT
    L.pageview_url AS URL,
    COUNT(L.website_session_id) AS sessions,
    COUNT(B.website_session_id) AS bounced_sessions,
    COUNT(B.website_session_id)/COUNT(L.website_session_id) AS bounce_rate
FROM landing_pages AS L
     LEFT JOIN bounced_sessions AS B
     ON L.website_session_id = B.website_session_id
GROUP BY 1;
```

0% 12:120

esult Grid Filter Rows: Search Export:

URL	sessions	bounced_sessio...	bounce_rate
/home	11048	6538	0.5918

Findings:

- we see that the home page has a 59.18% bounce rate .

A/B Test Analysis | SQL

```sql
-- A new custom lander ['/lander-1'] was developed
-- A/B test was run against home page for gsearch non brand traffic
-- Now lets look at the bounce_rates of both landing_pages

-- Step 1 Find when the lander_1 was live
SELECT
    MIN(website_session_id)
FROM website_pageviews
WHERE pageview_url = '/lander-1';
-- 11683 was the first session when '/lander-1' got live
```

Result Grid | Filter Rows: Search | Export:

MIN(website_session_...
11683

```sql
/* Step 2 Need to identify the landing sessions */
CREATE TEMPORARY TABLE landing_ids
SELECT
    P.website_session_id,
    MIN(P.website_pageview_id) AS website_pageview_id
    -- The First pageview_id for each session
FROM website_pageviews AS P
    LEFT JOIN website_sessions AS S
    ON P.website_session_id = S.website_session_id
WHERE P.website_session_id >= '11683'
AND P.created_at < '2012-07-28'
AND S.utm_source = 'gsearch'
AND S.utm_campaign = 'nonbrand'
GROUP BY 1;
SELECT * FROM landing_ids;
```

Result Grid | Filter Rows: Search | Export: | Fetch rows

website_session_id	website_pageview...
11683	23504
11684	23505
11685	23506
11686	23507
11687	23509

```sql
/* Step 3
Left Join to the website_pageviews table
Get the landing page url of each website session */
CREATE TEMPORARY TABLE landing_pages
SELECT
    L.website_session_id AS website_session_id,
    W.pageview_url AS pageview_url
FROM landing_ids AS L
    LEFT JOIN website_pageviews AS W
    ON L.website_pageview_id = W.website_pageview_id;
SELECT * FROM landing_pages;
```

Result Grid | Filter Rows: Search | Export: | Fetch rows

website_session...	pageview_url
11683	/lander-1
11684	/home
11685	/lander-1
11686	/lander-1
11687	/home
11688	/home

```sql
/* Step 4
Identifying bounce sessions, by left joining to website_pageviews */
CREATE TEMPORARY TABLE bounced_sessions
SELECT
    L.website_session_id AS website_session_id,
    L.pageview_url AS pageview_url,
    COUNT(W.website_pageview_id) AS page_views
FROM landing_pages AS L
    LEFT JOIN website_pageviews AS W
    ON L.website_session_id = W.website_session_id
GROUP BY 1,2
HAVING page_views = 1;
SELECT * FROM bounced_sessions;
```

Result Grid | Filter Rows: Search | Export: | Fetch rows:

website_session...	pageview_url	page_views
11683	/lander-1	1
11684	/home	1
11685	/lander-1	1
11687	/home	1
11688	/home	1
11690	/home	1
11692	/lander-1	1

```sql
/* Step 5
Left joining the landing_pages and bounced_sessions
And using case and pivot to find bounce rate analysis*/

SELECT
    L.pageview_url AS landing_pages,
    COUNT(L.website_session_id) AS sessions,
    COUNT(B.website_session_id) AS bounced_sessions,
    COUNT(B.website_session_id)/COUNT(L.website_session_id) AS bounce_rate
FROM landing_pages AS L
    LEFT JOIN bounced_sessions AS B
    ON L.website_session_id = B.website_session_id
GROUP BY 1;
```

Result Grid | Filter Rows: Search | Export:

landing_pag...	sessions	bounced_sessio...	bounce_rate
/lander-1	2316	1233	0.5324
/home	2261	1319	0.5834

Findings:

- The new lander has a bounce rate of 53% versus the home page for the same traffic was that 58% so it does look like this was an improvement in terms of performance.

Landing Page Bounce Rate Trend Analysis | SQL

```sql
-- Step 1
-- Identify the landing pageview_ids
CREATE TEMPORARY TABLE landing_ids
SELECT
    S.website_session_id,
    MIN(P.website_pageview_id) AS landing_id,
    COUNT(P.website_pageview_id) AS page_views
FROM website_sessions AS S
    LEFT JOIN website_pageviews AS P
    ON P.website_session_id = S.website_session_id
WHERE S.created_at> '2012-06-01'
AND S.created_at < '2012-08-31'
AND S.utm_source = 'gsearch'
AND S.utm_campaign = 'nonbrand'
GROUP BY 1;
SELECT * FROM landing_ids;
```

27:209

Result Grid | Filter Rows: Search | Export: | Fetch ro

website_session_id	landing_id	page_views
9350	18598	3
9351	18600	3
9352	18601	4
9354	18611	1
9356	18616	6
9357	18622	1
9358	18623	3
9359	18626	1

```sql
-- Step2
-- Left join to website_pageviews
-- and identify the landing page urls for each session
CREATE TEMPORARY TABLE landing_urls
SELECT
    L.website_session_id,
    L.landing_id,
    L.page_views,
    W.pageview_url AS landing_url,
    W.created_at
FROM landing_ids AS L
    LEFT JOIN website_pageviews AS W
    ON L.landing_id = W.website_pageview_id;
SELECT * FROM landing_urls;
```

100% 28:226

Result Grid | Filter Rows: Search | Export: | Fetch rows:

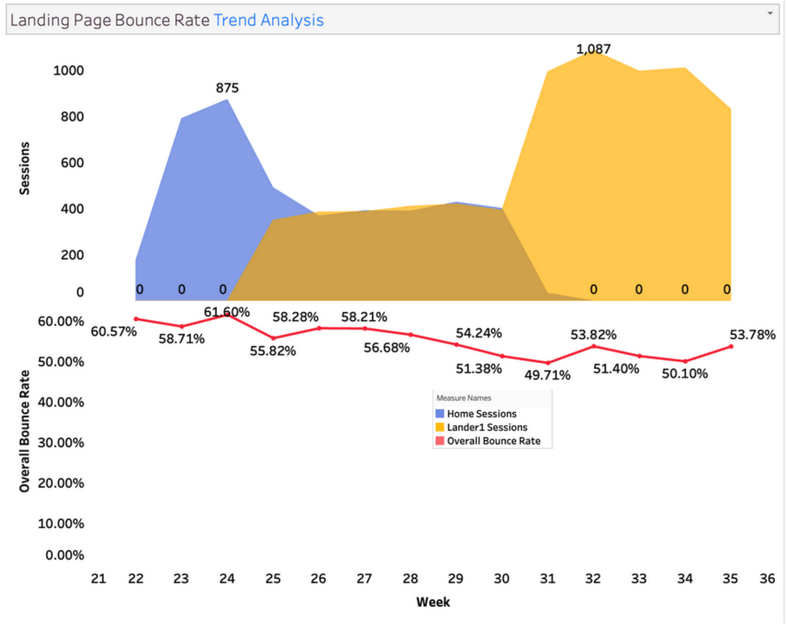
website_session...	landing_id	page_views	landing_url	created_at
9350	18598	3	/home	2012-06-01 00:05:11
9351	18600	3	/home	2012-06-01 00:06:39
9352	18601	4	/home	2012-06-01 00:08:27
9354	18611	1	/home	2012-06-01 01:08:43
9356	18616	6	/home	2012-06-01 01:37:31
9357	18622	1	/home	2012-06-01 02:29:33
9358	18623	3	/home	2012-06-01 02:39:16

```sql
-- Step 3 Perform Trend Analysis
SELECT
    YEAR(created_at) AS year,
    WEEK(created_at) AS week,
    MIN(DATE(created_at)) AS week_start_date,
    COUNT(DISTINCT website_session_id) AS website_sessions,
    COUNT(CASE WHEN page_views = 1 THEN website_session_id ELSE NULL END) AS bounced_sessions,
    COUNT(CASE WHEN page_views = 1 THEN website_session_id ELSE NULL END)/COUNT(DISTINCT website_session_id) AS overall_bounce_
    COUNT(DISTINCT CASE WHEN landing_url = '/home' THEN website_session_id ELSE NULL END) AS home_sessions,
    COUNT(DISTINCT CASE WHEN landing_url = '/lander-1' THEN website_session_id ELSE NULL END) AS lander1_sessions
FROM landing_urls
GROUP BY 1,2;
```

1:231

ult Grid | Filter Rows: Search | Export:

year	week	week_start_date	website_sessions	bounced_sessio...	overall_bounce_r...	home_sessions	lander1_sessions
2012	22	2012-06-01	175	106	0.6057	175	0
2012	23	2012-06-03	792	465	0.5871	792	0
2012	24	2012-06-10	875	539	0.6160	875	0
2012	25	2012-06-17	842	470	0.5582	492	350
2012	26	2012-06-24	755	440	0.5828	369	386
2012	27	2012-07-01	780	454	0.5821	392	388
2012	28	2012-07-08	801	454	0.5668	390	411
2012	29	2012-07-15	850	461	0.5424	429	421
2012	30	2012-07-22	796	409	0.5138	402	394
2012	31	2012-07-29	1028	511	0.4971	33	995
2012	32	2012-08-05	1087	585	0.5382	0	1087
2012	33	2012-08-12	998	513	0.5140	0	998
2012	34	2012-08-19	1012	507	0.5010	0	1012
2012	35	2012-08-26	833	448	0.5378	0	833



Findings:

- We can see the bounce rate starting in 61% and then overtime as traffic switched over to the Lander we're seeing bounce rate closer to the 50% range.

Analyzing Conversion Funnels

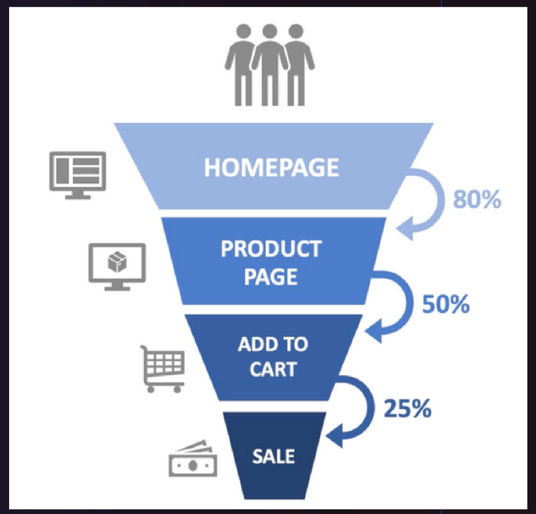
Conversion Funnel Analysis is all about understanding and optimizing each step of your users experience on their journey towards purchasing your products.

When we build our conversion funnels we'll be looking at our users the pages that they land on and then we'll see what percentage of those customers move on to the next step in our flow.

Some common use cases:

- Identifying the most common paths customers may take.
- Identifying how many of your users will continue to each next step in your conversion flow and where your largest abandonment points.

Optimizing critical pain points where your users are abandoning so that business can convert more users and sell more products.



Conversion Funnel Analysis | SQL

```sql
-- Conversion Funnel Analysis

/* Funnel: lander-1, products, fuzzy, cart, shipping, billing, thankyou */

-- Step 1
-- Flagging all the website_sessions with each page viewed
SELECT
    S.website_session_id,
    P.pageview_url,
    P.created_at,
    CASE WHEN P.pageview_url = '/products' THEN 1 ELSE 0 END AS products_page,
    CASE WHEN P.pageview_url = '/the-original-mr-fuzzy' THEN 1 ELSE 0 END AS fuzzy_page,
    CASE WHEN P.pageview_url = '/cart' THEN 1 ELSE 0 END AS cart_page,
    CASE WHEN P.pageview_url = '/shipping' THEN 1 ELSE 0 END AS shipping_page,
    CASE WHEN P.pageview_url = '/billing' THEN 1 ELSE 0 END AS billing_page,
    CASE WHEN P.pageview_url = '/thank-you-for-your-order' THEN 1 ELSE 0 END AS thankyou_page
FROM website_sessions AS S
    LEFT JOIN website_pageviews P
    ON S.website_session_id = P.website_session_id
WHERE S.utm_source = 'gsearch'
AND S.utm_campaign = 'nonbrand'
AND P.created_at > '2012-08-05'
AND P.created_at < '2012-09-05'
ORDER BY 1,3;
```

75% 7:249

Result Grid | Filter Rows: Search | Export: | Fetch rows:

website_session_id	pageview_url	created_at	products_pa...	fuzzy_page	cart_page	shipping_pa...	billing_pa...	thankyou_page
18243	/lander-1	2012-08-05 00:11:26	0	0	0	0	0	0
18244	/lander-1	2012-08-05 00:44:06	0	0	0	0	0	0
18244	/products	2012-08-05 00:46:01	1	0	0	0	0	0
18244	/the-original-mr-fuzzy	2012-08-05 00:46:32	0	1	0	0	0	0
18244	/cart	2012-08-05 00:47:26	0	0	1	0	0	0
18244	/shipping	2012-08-05 00:51:00	0	0	0	1	0	0
18244	/billing	2012-08-05 00:53:17	0	0	0	0	1	0

```sql
-- Step 2
-- Using the step 1 as subquery, identifying the customer journey
-- Now Grouping by session_id to understand each customers journey

CREATE TEMPORARY TABLE session_level_funnel
SELECT
    website_session_id,
    MAX(products_page) AS products_page,
    MAX(fuzzy_page) AS fuzzy_page,
    MAX(cart_page) AS cart_page,
    MAX(shipping_page) AS shipping_page,
    MAX(billing_page) AS billing_page ,
    MAX(thankyou_page) AS thankyou_page
FROM (    -- Sub Query here from step 1
GROUP BY 1;

SELECT * FROM session_level_funnel;
```

6:303

Result Grid | Filter Rows: Search | Export: | Fetch rows:

website_session...	products_pa...	fuzzy_page	cart_page	shipping_pa...	billing_pa...	thankyou_page
18243	0	0	0	0	0	0
18244	1	1	1	1	1	0
18245	0	0	0	0	0	0
18246	1	0	0	0	0	0
18247	1	1	0	0	0	0
18249	0	0	0	0	0	0
18250	0	0	0	0	0	0

```sql
-- Step3 funnel flow
SELECT
    COUNT(website_session_id) AS sessions,
    SUM(products_page ) AS to_products,
    SUM(fuzzy_page) AS to_fuzzy,
    SUM(cart_page) AS to_cart,
    SUM(shipping_page) AS to_shipping,
    SUM(billing_page) AS to_billing,
    SUM(thankyou_page) AS to_thankyou
FROM session_level_funnel;
```

27:314

Result Grid | Filter Rows: Search | Export:

sessions	to_produc...	to_fuzzy	to_cart	to_shipping	to_billing	to_thankyou
4493	2115	1567	683	455	361	158

```sql
-- Step 4 Click rate from one page to another
SELECT
    SUM(products_page)/COUNT(website_session_id) AS lander_rate,
    SUM(fuzzy_page)/SUM(products_page) AS products_rate,
    SUM(cart_page)/SUM(fuzzy_page) AS fuzzy_rate,
    SUM(shipping_page)/SUM(cart_page) AS cart_rate,
    SUM(billing_page)/SUM(shipping_page) AS shipping_rate,
    SUM(thankyou_page)/SUM(billing_page) AS thankyou_rate
FROM session_level_funnel;
```

27:324

Result Grid | Filter Rows: Search | Export:

lander_rate	products_ra...	fuzzy_rate	cart_rate	shipping_ra...	thankyou_ra...
0.4707	0.7409	0.4359	0.6662	0.7934	0.4377

Findings:

- We see that the lander clicking-through rate is 47%, the fuzzy clicking-through rate is 43%, and the final thank you click rate is 43%, which is concerning.

Billing A/B Analysis | SQL

```sql
-- /billing-2 Was introduced, did final billing % increase for the new funnel ?
-- A/B test was run against billing and billing-2 for all the traffic
-- Now lets look at the funnels of both the billing sessions

-- First setp, when was the first live session of the /billing-2
SELECT
    MIN(created_at),
    MIN(website_session_id),
    MIN(website_pageview_id)
FROM website_pageviews
WHERE pageview_url = '/billing-2';

-- website_session_id : '25325',
-- website_pageview_id : '53550' for the first /billing-2 session
```

59:322

Result Grid | Filter Rows: Search | Export:

MIN(created_at)	MIN(website_session_...	MIN(website_pageview_...
2012-09-10 00:13:05	25325	53550

```sql
-- Step 2
-- Now identfying the sessions which went till billing and billing 2

CREATE TEMPORARY TABLE billing
SELECT
    website_session_id,
    pageview_url
FROM website_pageviews
WHERE website_pageview_id >= 53550
AND created_at < '2012-11-10'
AND pageview_url in ('/billing', '/billing-2')
ORDER BY 1;

SELECT * FROM billing;
```

23:355

Result Grid | Filter Rows: Search | Export: | Fetch rows:

website_session...	pageview_url
25325	/billing-2
25343	/billing
25353	/billing
25358	/billing-2
25368	/billing
25393	/billing

```sql
-- Left Joing to oders and calculating the conversion rate
SELECT
    B.pageview_url,
    COUNT(DISTINCT B.website_session_id) AS billing_sessions,
    COUNT(DISTINCT O.order_id) AS orders,
    COUNT(DISTINCT O.order_id)/COUNT(DISTINCT B.website_session_id) AS conversion_rate
FROM billing AS B
    LEFT JOIN ORDERS AS O ON B.website_session_id = O.website_session_id
GROUP BY 1;
```

1:366

Result Grid | Filter Rows: Search | Export:

pageview_url	billing_sessio...	orders	conversion_rate
/billing	657	300	0.4566
/billing-2	654	410	0.6269

Findings:

- It looks like billing two has a lot more orders and the conversion rate to order is substantially higher.