Maven fuzzy factory is an online retailer which launched their first product in March 2012. This E-commerce Data Analysis project aims to analyze and gain insights from this e-commerce database(mavenfuzzy database) using MySQL. The project focuses on extracting meaningful information from the database to understand various aspects of the e-commerce business, including monthly & quarterly trends, campaign performance, impact of launching new products, adding new landing pages and product cross-selling.

This project has been divided into two parts. First part looks at business performance for the initial 8 months of business life. Second part of the project includes analysis related to the overall growth and relevant metrics during the first 3 years of business in order to secure the next round of funding.

Key Objectives:

1. Monthly Trends: Analyze and visualize the monthly trends in key metrics such as sessions, orders, revenue, and conversion rate. Identify patterns, seasonality, and any significant changes over time.
2. Campaign Performance: Evaluate the effectiveness of paid campaigns by analyzing metrics such as click-through rate, conversion rate, and return on ad spend (ROAS). Identify which campaigns are performing well and driving sales.
3. Cross-Selling Analysis: Identify products that are frequently purchased together to understand cross-selling opportunities. Analyze customer behavior and purchase patterns to suggest relevant product recommendations or improve product bundling strategies.
4. A/B tests : Analyzing the impact of adding new landing pages, new products in order to understand consumer behavior and inturn improve customer engagement and interaction with the platform

By leveraging MySQL and conducting thorough data analysis, this e-commerce project aims to provide valuable insights into the business's performance, customer behavior, and marketing strategies, ultimately helping to drive data-driven decision-making and optimize overall business operations.

PART -1 : This part contains 4 questions.

It's been 8 months since the Mavenfuzzy factory went live and the company's CEO Henry is due to present the performance metric to the board of the company next week in order to secure the member's trust in the company. For this, Henry needs to showcase the company's performance for the past 8 months and sell a good growth story. As an Analyst, I am tasked with preparing relevant metrics to show the company's promising growth. I received the email from the CEO on 27th Nov, 2012, therefore, I will be analyzing the consumer's data before this date.

Q.1 : Pull the monthly trends for all the channels to understand how each of our channels our performing.

Approach: Using website_sessions table as it contains utm_source attribute. After filtering the data before 27th Nov, Using CASE pivot method and group by to get the count of monthly sessions for each channel

SQL code for getting the result -

```
SELECT

Year,
Mon,
COUNT(DISTINCT(website_session_id)) AS total_sessions,
COUNT(DISTINCT(CASE WHEN utm_source = 'gsearch' THEN website_session_id ELSE  NULL
END)) AS  gsearch_sessions,
COUNT(DISTINCT(CASE WHEN utm_source = 'bsearch' THEN website_session_id ELSE NULL
END)) AS bsearch_sessions,
COUNT(DISTINCT(CASE WHEN utm_source IS NULL AND http_referer IS NOT NULL THEN
website_session_id ELSE NULL END)) AS organic_sessions,
COUNT(DISTINCT(CASE WHEN utm_source IS NULL AND http_referer IS NULL THEN
website_session_id ELSE NULL END)) AS direct_typein_sessions

FROM
(SELECT  YEAR(WS.created_at) Year,
            MONTH(WS.created_at) Mon,
            website_session_id,
            utm_source,
            utm_campaign,
```

```
        http_referer
FROM website_sessions WS
WHERE WS.created_at < '2012-11-27') AS session_with_source
GROUP BY 1,2;
```

Result : Please find below the result of the above query. It looks like most of our sessions are coming through the gsearch channel. Monthly session growth looks promising. Its also good to see the month-on-month increase in Organic & direct type in sessions as company doesn't have to pay for these sessions.

Year	Mon	total_sessions	gsearch_sessions	bsearch_sessions	organic_sessions	direct_typein_sessions
2012	3	1879	1860	2	8	9
2012	4	3734	3574	11	78	71
2012	5	3736	3410	25	150	151
2012	6	3963	3578	25	190	170
2012	7	4249	3811	44	207	187
2012	8	6097	4877	705	265	250
2012	9	6546	4491	1439	331	285
2012	10	8183	5534	1781	428	440
2012	11	12750	8889	2840	536	485

Q.2 : Gserach seems to be the biggest driver of the business. Pull monthly trends for gsearch sessions, orders and conversion rate.

Approach : Using common table expression(cte) to create the temporary result set. Using GROUP BY in subsequent SELECT statement referencing cte to get the desired result.

SQL code for getting the result -

```
WITH cte AS(
SELECT
    YEAR(WS.created_at) Year,
    MONTH(WS.created_at) Mon,
    WS.website_session_id,
    orders.order_id
FROM website_sessions WS
LEFT JOIN orders
ON WS.website_session_id = orders.website_session_id
WHERE WS.created_at < '2012-11-27'
```

```
AND utm_source = 'gsearch')

SELECT Year,
        Mon,
        COUNT(DISTINCT(website_session_id)) gsearch_session,
        COUNT(DISTINCT(order_id)) sessions_with_order,
        COUNT(DISTINCT(order_id))*100/COUNT(DISTINCT(website_session_id)) AS
session_to_order_rt
FROM cte
GROUP BY Year,Mon;
```

Result : Monthly growth for gsearch sessions looks promising. Session to order conversion rate has also jumped from 3.23% to 4.2%.

	Year	Mon	gsearch_session	sessions_with_order	session_to_order_rt
▶	2012	3	1860	60	3.2258
	2012	4	3574	92	2.5741
	2012	5	3410	97	2.8446
	2012	6	3578	121	3.3818
	2012	7	3811	145	3.8048
	2012	8	4877	184	3.7728
	2012	9	4491	188	4.1862
	2012	10	5534	234	4.2284
	2012	11	8889	373	4.1962

Q.3 : Pull monthly sessions & orders for gsearch split by device type.

Approach : Left Join orders table to website_sessions table to get all the sessions common to both tables and sessions unique to website_sessions table. Using CASE pivot method and GROUP BY to get sessions and orders for both mobile and desktop devices.

SQL code for getting the result -

```
SELECT
   Year(WS.created_at) Year,
   Month(WS.created_at) Mon,
   COUNT(DISTINCT(WS.website_session_id)) gsearch_sessions,
   COUNT(DISTINCT(CASE WHEN WS.device_type = 'mobile' THEN WS.website_session_id ELSE
NULL END)) mobile_sessions,
   COUNT(DISTINCT(CASE WHEN WS.device_type = 'desktop' THEN WS.website_session_id ELSE
NULL END)) desktop_sessions,
```

```
   COUNT(DISTINCT(CASE WHEN WS.device_type = 'mobile' THEN O.order_id ELSE NULL END))
mobile_orders,
   COUNT(DISTINCT(CASE WHEN WS.device_type = 'desktop' THEN O.order_id ELSE NULL
END)) desktop_orders,
   COUNT(DISTINCT(CASE WHEN WS.device_type = 'mobile' THEN O.order_id ELSE NULL
END))/
   COUNT(DISTINCT(CASE WHEN WS.device_type = 'mobile' THEN WS.website_session_id ELSE
NULL END)) mobile_order_cnv_rt,
   COUNT(DISTINCT(CASE WHEN WS.device_type = 'desktop' THEN O.order_id ELSE NULL
END))/
   COUNT(DISTINCT(CASE WHEN WS.device_type = 'desktop' THEN WS.website_session_id ELSE
NULL END)) desktop_order_cnv_rt

FROM website_sessions WS
LEFT JOIN orders O ON O.website_session_id = WS.website_session_id
WHERE WS.utm_source = 'gsearch'
AND WS.created_at < '2012-11-27'
GROUP BY 1,2;
```

<u>Result :</u> Looks like monthly session volume for desktop is roughly 3 times more than mobile. Sessions to order conversion rate for desktop also outperforms mobile.

Year	Mon	gsearch_sessions	mobile_sessions	desktop_sessions	mobile_orders	desktop_orders	mobile_order_cnv_rt	desktop_order_cnv_rt
2012	3	1860	727	1133	10	50	0.0138	0.0441
2012	4	3574	1393	2181	12	80	0.0086	0.0367
2012	5	3410	1062	2348	10	87	0.0094	0.0371
2012	6	3578	817	2761	8	113	0.0098	0.0409
2012	7	3811	955	2856	15	130	0.0157	0.0455
2012	8	4877	1238	3639	10	174	0.0081	0.0478
2012	9	4491	1166	3325	19	169	0.0163	0.0508
2012	10	5534	1386	4148	21	213	0.0152	0.0514
2012	11	8889	2211	6678	37	336	0.0167	0.0503

<u>Q.4 :</u> Analyse the impact of adding a new landing page '/lander-1'. Here utm_source = 'gsearch' and utm_campaign = 'nonbrand'

Approach - A new landing page '/lander-1' is added alongside '/home landing page. For a fair comparison, first getting the date when '/lander-1' was added and getting date for latest website_session that landed on '/home' page before being rerouted to '/lander-1' page and then pulling sessions, orders &

session to order conversion rate using website_pageviews, wesbite_sessions and orders tables for sessions landing on the home or lander-1 page between the date range.

Next, getting count of all sessions landing on '/lander-1' page for the remaining period to get the revenue gen since all our sessions are now landing on this page only.

```
SELECT MIN(created_at) FROM website_pageviews
WHERE pageview_url = '/lander-1';
```

We get created_at = '2012-06-19 00:35:54'. Date for First session landing on '/lander-1' page.

```
SELECT MAX(website_pageviews.created_at) FROM website_pageviews
INNER JOIN website_sessions ON website_sessions.website_session_id =
website_pageviews.website_session_id
WHERE pageview_url = '/home'
AND website_sessions.utm_source = 'gsearch' AND website_sessions.utm_campaign = 'nonbrand'
```

We get the date for last session which landed on '/home' page = '2012-07-29 23:48:16'

Following query gives a fair comparison of both the landing pages

```
SELECT
        WP.pageview_url,
        COUNT(DISTINCT(WS.website_session_id)) sessions,
        COUNT(DISTINCT(O.order_id)) orders,
        COUNT(DISTINCT(O.order_id))/COUNT(DISTINCT(WS.website_session_id)) AS
session_order_cnv_rate
FROM Website_sessions WS
INNER JOIN website_pageviews WP
ON WP.website_session_id = WS.website_session_id
LEFT JOIN orders O
ON O.website_session_id = WS.website_session_id
WHERE WS.utm_source = 'gsearch' AND WS.utm_campaign ='nonbrand'
AND WS.created_at BETWEEN '2012-06-19 00:35:54' AND '2012-07-29 23:48:16'
AND WP.pageview_url IN ('/home','/lander-1')
GROUP BY 1;
```

pageview_url	sessions	orders	session_order_cnv_r...
/home	2328	75	0.0322
/lander-1	2381	96	0.0403

Can clearly see ‘/lander-1’ having better session to order conversion rate.
‘/lander-1’ is getting 0.0088 extra orders per session as compared to ‘/home’ page. Therefore, lift generated by new landing page is 0.0088

Now, counting total sessions we have had since '2012-07-29 23:48:16' till 26th Nov,2012. All these sessions have been rerouted to ‘/lander-1’ page.

```
SELECT COUNT(DISTINCT(website_session_id)) total_sessions
FROM website_sessions
WHERE created_at> '2012-07-29 23:48:16'
AND created_at < '2012-11-27'
AND utm_source = 'gsearch'
AND utm_campaign = 'nonbrand';
```

total_sessions
22972

<u>Result :</u> We have had a total of 22972 sessions since then.
incremental rate was 0.0088 for lander-1.
(22972 session)*0.0088(order/session) = 202 extra/incremental orders since 29 July
We are getting roughly 50 extra orders every month.
Looks like rerouting all the sessions to ‘/lander-1’ was a great decision.

PART - 2 : This part contains 5 questions.

It's been 3 years since the Mavenfuzzy factory is in business. Company has seen enough growth to raise much higher VC funding and it is close to securing a large sum from one of the best firms. As an Analyst, I have been tasked to analyze the database to paint a higher growth picture and data driven performance optimization. This part will majorly focus on evaluating revenue generated, analyzing the impact of new product launches, looking for seasonality trends & cross selling of products.

Q.5 :Pull quarterly trends for session to order conversion rate, revenue per session and revenue per order.

Approach : Using website_sessions and left joining it with the orders table. Extracting Dayofyear to explain for the lesser sessions and revenue generated in the first and the last quarters of this period

```
SELECT
   Year(WS.created_at) yr,
    QUARTER(WS.created_at) quarter,
    COUNT(DISTINCT(Dayofyear(WS.created_at))) days_in_quarter,
    COUNT(DISTINCT(WS.website_session_id)) sessions,
    COUNT(DISTINCT(O.order_id)) orders,
    COUNT(DISTINCT(O.order_id))/  COUNT(DISTINCT(WS.website_session_id))
session_to_order_cnv_rt,
    SUM(O.price_usd)/COUNT(DISTINCT(O.order_id)) revenue_per_order,
    SUM(O.price_usd)/COUNT(DISTINCT(WS.website_session_id))  revenue_per_session
FROM website_sessions WS
LEFT JOIN orders O
ON O.website_session_id = WS.website_session_id
GROUP BY 1,2;
```

	yr	quarter	days_in_quarter	sessions	orders	session_to_order_cnv...	revenue_per_ord...	revenue_per_session
▶	2012	1	13	1879	60	0.0319	49.990000	1.596275
	2012	2	91	11433	347	0.0304	49.990000	1.517233
	2012	3	92	16892	684	0.0405	49.990000	2.024222
	2012	4	92	32266	1495	0.0463	49.990000	2.316217
	2013	1	90	19833	1273	0.0642	52.142396	3.346809
	2013	2	91	24745	1718	0.0694	51.538312	3.578211
	2013	3	92	27663	1840	0.0665	51.734533	3.441114
	2013	4	92	40540	2616	0.0645	54.715688	3.530741
	2014	1	90	46779	3069	0.0656	62.160684	4.078136
	2014	2	91	53129	3848	0.0724	64.374207	4.662462
	2014	3	92	57141	4035	0.0706	64.494949	4.554298
	2014	4	92	76373	5908	0.0774	63.793497	4.934885
	2015	1	78	64198	5420	0.0844	62.799917	5.301965

Result : Trend looks pretty good. session to order conversion rate has grown from 3.2% to 8.5%. Revenue per order has grown from roughly $50 to $62.8 and revenue generated per session has grown from $1.6 to $5.3

<u>Q.6 :</u> Show how each of our channels have grown over time. Evaluate session to order conversion rate for all the channels. Make note of the period where we have made major improvements/optimization.

Approach : Showcasing growth for following channels : gsearch non brand, bsearch non brand, brand sessions, organic sessions and direct type in sessions. Getting quarterly trends of orders volume using website_sessions table.

Left joining Orders table with website_sessions table to get session to order conversion rate for each of our channels. Observing for any seasonality.

```
SELECT
        YEAR(WS.created_at) Year,
        QUARTER(WS.created_at) quarter,
        COUNT(DISTINCT(Dayofyear(WS.created_at))) days_in_quarter,
        COUNT(DISTINCT(orders.order_id)) orders,
       COUNT(DISTINCT(CASE WHEN WS.utm_source = 'gsearch' AND utm_campaign =
'nonbrand' THEN orders.order_id ELSE NULL END)) gsearch_nonbrand_orders,
        COUNT(DISTINCT(CASE WHEN WS.utm_source = 'bsearch' AND utm_campaign =
'nonbrand' THEN orders.order_id ELSE NULL END)) bsearch_nonbrand_orders,
        COUNT(DISTINCT(CASE WHEN utm_campaign = 'brand' THEN orders.order_id ELSE
NULL END)) brand_orders,
```

```
        COUNT(DISTINCT(CASE WHEN utm_source IS NULL AND http_referer IN
('https://www.gsearch.com','https://www.bsearch.com') THEN orders.order_id ELSE NULL END))
organic_orders,
        COUNT(DISTINCT(CASE WHEN utm_source IS NULL AND http_referer IS NULL THEN
orders.order_id ELSE NULL END)) direct_typein_orders

FROM website_sessions WS
LEFT JOIN orders
ON WS.website_session_id = orders.website_session_id
GROUP BY 1,2;
```

Year	quarter	days_in_quarter	sessions	orders	gsearch_nonbrand_orders	bsearch_nonbrand_orders	brand_orders	organic_orders	direct_typein_orders
2012	1	13	1879	60	60	0	0	0	0
2012	2	91	11433	347	291	0	20	15	21
2012	3	92	16892	684	482	82	48	40	32
2012	4	92	32266	1495	913	311	88	94	89
2013	1	90	19833	1273	766	183	108	125	91
2013	2	91	24745	1718	1114	237	114	134	119
2013	3	92	27663	1840	1132	245	153	167	143
2013	4	92	40540	2616	1657	291	248	223	197
2014	1	90	46779	3069	1667	344	354	338	311
2014	2	91	53129	3848	2208	427	410	436	367
2014	3	92	57141	4035	2259	434	432	445	402
2014	4	92	76373	5908	3248	683	615	605	532
2015	1	78	64198	5420	3025	581	622	640	552

As can be seen from above, the count of orders for each of our channels have grown overtime with most orders coming through gsearch_nonbrand. Orders through organic and direct_type_in have also grown significantly which looks great for business.
Seasonality trend : In the 4th Quarter of each year, we can see a significant Spike in sessions & orders.

Now, getting sessions to order conversion rate for each channel.

```
SELECT
        Year(WS.created_at) yr,
        QUARTER(WS.created_at) quarter,
        COUNT(DISTINCT(Dayofyear(WS.created_at))) days_in_quarter,

      COUNT(DISTINCT(CASE WHEN WS.utm_source = 'gsearch' AND utm_campaign = 'nonbrand'
THEN O.order_id ELSE NULL END)) /
      COUNT(DISTINCT(CASE WHEN WS.utm_source = 'gsearch' AND utm_campaign = 'nonbrand'
THEN WS.website_session_id ELSE NULL END)) g_nonbrand_ssn_order_cnv_rt,
```

```
        COUNT(DISTINCT(CASE WHEN WS.utm_source = 'bsearch' AND utm_campaign = 'nonbrand'
THEN O.order_id ELSE NULL END)) /
        COUNT(DISTINCT(CASE WHEN WS.utm_source = 'bsearch' AND utm_campaign =
'nonbrand' THEN WS.website_session_id ELSE NULL END)) b_nonbrand_ssn_order_cnv_rt,

        COUNT(DISTINCT(CASE WHEN utm_campaign = 'brand' THEN O.order_id ELSE NULL
END)) /
        COUNT(DISTINCT(CASE WHEN utm_campaign = 'brand' THEN WS.website_session_id
ELSE NULL END)) brand_ssn_order_cnv_rt,

        COUNT(DISTINCT(CASE WHEN WS.utm_source IS NULL  AND http_referer IS NOT NULL
THEN O.order_id ELSE NULL END)) /
        COUNT(DISTINCT(CASE WHEN WS.utm_source IS NULL  AND http_referer IS NOT NULL
THEN WS.website_session_id ELSE NULL END)) organic_ssn_order_cnv_rt,

        COUNT(DISTINCT(CASE WHEN WS.utm_source IS NULL  AND http_referer IS NULL
THEN O.order_id ELSE NULL END)) /
        COUNT(DISTINCT(CASE WHEN WS.utm_source IS NULL  AND http_referer IS  NULL
THEN WS.website_session_id ELSE NULL END)) direct_typein_ssn_order_cnv_rt

FROM website_sessions WS
LEFT JOIN orders O
ON WS.website_session_id = O.website_session_id
GROUP BY 1,2;
```

yr	quarter	days_in_quarter	g_nonbrand_ssn_order_cnv_rt	b_nonbrand_ssn_order_cnv_rt	brand_ssn_order_cnv_rt	organic_ssn_order_cnv_rt	direct_typein_ssn_order_cnv
2012	1	13	0.0324	NULL	0.0000	0.0000	0.0000
2012	2	91	0.0284	NULL	0.0526	0.0359	0.0536
2012	3	92	0.0384	0.0408	0.0602	0.0498	0.0443
2012	4	92	0.0436	0.0497	0.0531	0.0539	0.0537
2013	1	90	0.0612	0.0693	0.0703	0.0753	0.0614
2013	2	91	0.0685	0.0690	0.0679	0.0760	0.0735
2013	3	92	0.0639	0.0697	0.0703	0.0734	0.0719
2013	4	92	0.0629	0.0601	0.0801	0.0694	0.0647
2014	1	90	0.0693	0.0704	0.0839	0.0756	0.0765
2014	2	91	0.0702	0.0695	0.0804	0.0797	0.0738
2014	3	92	0.0703	0.0698	0.0756	0.0733	0.0702
2014	4	92	0.0782	0.0841	0.0812	0.0784	0.0748
2015	1	78	0.0861	0.0850	0.0852	0.0821	0.0775

There is a sudden spike in session to order conversion rate in 1st quarter of 2013 for each channel type, This is due to the new product launched in Jan, 2013.

Q.7 : 25th Sept, 2013 onwards, users are allowed to add one more product in cart within the same order. Pull monthly trend for revenue and margin product wise along with total sales and total revenue, Notice Seasonality trend.

Approach : View product table to know when new products were launched. Using orders and order_items table to get monthly trend of revenue and profit margin for each product.

Find below the product table

product_id	created_at	product_name
1	2012-03-19 08:00:00	The Original Mr. Fuzzy
2	2013-01-06 13:00:00	The Forever Love Bear
3	2013-12-12 09:00:00	The Birthday Sugar Panda
4	2014-02-05 10:00:00	The Hudson River Mini bear

Getting product wise monthly trend :-

```
WITH cte1 AS(

SELECT
      YEAR(orders.created_at) Year,
      MONTH(orders.created_at) Mon,

     SUM(CASE WHEN product_id = 1 THEN order_items.price_USD ELSE NULL END)
product1_revenue,
     SUM(CASE WHEN product_id = 1 THEN order_items.price_USD ELSE NULL END) -
SUM(CASE WHEN product_id = 1 THEN order_items.cogs_usd ELSE NULL END) product1_margin,

     SUM(CASE WHEN product_id = 2 THEN order_items.price_USD ELSE NULL END)
product2_revenue,
     SUM(CASE WHEN product_id = 2 THEN order_items.price_USD ELSE NULL END) -
SUM(CASE WHEN product_id = 2 THEN order_items.cogs_usd ELSE NULL END) product2_margin,

     SUM(CASE WHEN product_id = 3 THEN order_items.price_USD ELSE NULL END)
product3_revenue,
     SUM(CASE WHEN product_id = 3 THEN order_items.price_USD ELSE NULL END) -
SUM(CASE WHEN product_id = 3 THEN order_items.cogs_usd ELSE NULL END) product3_margin,
```

```
    SUM(CASE WHEN product_id = 4 THEN order_items.price_USD ELSE NULL END)
product4_revenue,
    SUM(CASE WHEN product_id = 4 THEN order_items.price_USD ELSE NULL END) -
SUM(CASE WHEN product_id = 4 THEN order_items.cogs_usd ELSE NULL END) product4_margin

FROM orders
INNER JOIN order_items
ON order_items.order_id = orders.order_id
GROUP BY 1,2),

cte2 AS(
SELECT
            YEAR(orders.created_at) Year,
            MONTH(orders.created_at) Month,
          SUM(orders.items_purchased) total_sales,
          SUM(orders.price_usd) total_revenue
FROM orders
GROUP BY 1,2)

SELECT
          cte1.Year,
          cte1.Mon,
         cte2.total_sales,
         cte2.total_revenue,
         cte1.product1_revenue,
          cte1.product1_margin,
         cte1.product2_revenue,
          cte1.product2_margin,
        cte1.product3_revenue,
          cte1.product3_margin,
        cte1.product4_revenue,
          cte1.product4_margin

FROM cte1
INNER JOIN cte2
ON cte2.Month = cte1.Mon
AND cte2.Year = cte1.Year;
```

Year	Mon	total_sales	total_revenue	product1_revenue	product1_margin	product2_revenue	product2_margin	product3_revenue	product3_margin	product4_revenue	product4_r
2012	3	60	2999.40	2999.40	1830.00	NULL	NULL	NULL	NULL	NULL	NULL
2012	4	99	4949.01	4949.01	3019.50	NULL	NULL	NULL	NULL	NULL	NULL
2012	5	108	5398.92	5398.92	3294.00	NULL	NULL	NULL	NULL	NULL	NULL
2012	6	140	6998.60	6998.60	4270.00	NULL	NULL	NULL	NULL	NULL	NULL
2012	7	169	8448.31	8448.31	5154.50	NULL	NULL	NULL	NULL	NULL	NULL
2012	8	228	11397.72	11397.72	6954.00	NULL	NULL	NULL	NULL	NULL	NULL
2012	9	287	14347.13	14347.13	8753.50	NULL	NULL	NULL	NULL	NULL	NULL
2012	10	371	18546.29	18546.29	11315.50	NULL	NULL	NULL	NULL	NULL	NULL
2012	11	618	30893.82	30893.82	18849.00	NULL	NULL	NULL	NULL	NULL	NULL
2012	12	506	25294.94	25294.94	15433.00	NULL	NULL	NULL	NULL	NULL	NULL
2013	1	390	19966.10	17146.57	10461.50	2819.53	1762.50	NULL	NULL	NULL	NULL
2013	2	498	26515.02	16796.64	10248.00	9718.38	6075.00	NULL	NULL	NULL	NULL
2013	3	385	19896.15	15996.80	9760.00	3899.35	2437.50	NULL	NULL	NULL	NULL
2013	4	553	28584.47	22945.41	13999.50	5639.06	3525.00	NULL	NULL	NULL	NULL
2013	5	571	29364.29	24445.11	14914.50	4919.18	3075.00	NULL	NULL	NULL	NULL
2013	6	593	30544.07	25144.97	15341.50	5399.10	3375.00	NULL	NULL	NULL	NULL
2013	7	604	31143.96	25444.91	15524.50	5699.05	3562.50	NULL	NULL	NULL	NULL
2013	8	608	31373.92	25494.90	15555.00	5879.02	3675.00	NULL	NULL	NULL	NULL
2013	9	635	32723.65	26844.63	16378.50	5879.02	3675.00	NULL	NULL	NULL	NULL
2013	10	738	38242.62	30143.97	18391.50	8098.65	5062.50	NULL	NULL	NULL	NULL
2013	11	898	46631.02	36192.76	22082.00	10438.26	6525.00	NULL	NULL	NULL	NULL
2013	12	1140	58262.60	40891.82	24949.00	10978.17	6862.50	6392.61	4378.50	NULL	NULL
2014	1	1111	56568.89	36392.72	22204.00	10978.17	6862.50	9198.00	6300.00	NULL	NULL
2014	2	1348	66012.52	29194.16	17812.00	21056.49	13162.50	9703.89	6646.50	6057.98	4141.00
2014	3	1427	68189.73	39242.15	23942.50	11578.07	7237.50	11221.56	7686.00	6147.95	4202.50
2014	4	1657	78725.43	45840.83	27968.50	12837.86	8025.00	12279.33	8410.50	7767.41	5309.50
2014	5	1873	88935.27	51489.70	31415.00	14757.54	9225.00	13751.01	9418.50	8937.02	6109.00
2014	6	1675	80051.25	44641.07	27236.50	14697.55	9187.50	13245.12	9072.00	7467.51	5104.50
2014	7	1745	83288.55	48040.39	29310.50	14637.56	9150.00	12693.24	8694.00	7917.36	5412.00
2014	8	1792	84716.08	47890.42	29219.00	14217.63	8887.50	13521.06	9261.00	9086.97	6211.50
2014	9	1951	92232.49	52789.44	32208.00	15057.49	9412.50	14578.83	9985.50	9806.73	6703.50
2014	10	2202	103905.98	58638.27	35776.50	17037.16	10650.00	16924.32	11592.00	11306.23	7728.50
2014	11	2702	128162.98	72535.49	44255.50	22616.23	14137.50	19545.75	13387.50	13465.51	9204.50
2014	12	3098	144823.02	79184.16	48312.00	23216.13	14512.50	24788.61	16978.50	17634.12	12054.00
2015	1	2846	132211.54	69586.08	42456.00	23636.06	14775.00	20695.50	14175.00	18293.90	12505.00
2015	2	2706	129212.94	55638.87	33946.50	38633.56	24150.00	18625.95	12757.50	16314.56	11152.00
2015	3	1693	78951.07	43191.36	26352.00	13377.77	8362.50	12095.37	8284.50	10286.57	7031.50

seasonality trends were found for following months of years :
Nov, Dec 2012 - spike up , Mar, 2013 spike down, Nov, Dec 2013 - spike up, Nov, Dec 2014 spike up
need to dig deep into these specific months to get specific week of month and day of week we can see spike up/down and try looking at cause/effect.

Q.8 : Analyze the impact of introducing new products.

Approach : To analyze the impact of introducing new products, look at change in clickthrough rates over time by getting monthly trends of sessions which are landing on '/product' page and then clicking through product page to land on next page. finally getting '/product' page to order conversion rate

```
SELECT

YEAR(WP.created_at) Year,
 MONTH(WP.created_at) Mon,
COUNT(DISTINCT(WP.website_session_id)) sessions,

COUNT(DISTINCT(CASE WHEN WP.pageview_url = '/products' THEN WP.website_session_id ELSE
NULL END)) AS to_products,

 COUNT(DISTINCT(CASE WHEN WP.pageview_url IN
('/the-original-mr-fuzzy','/the-forever-love-bear','/the-birthday-sugar-panda','/the-hudson-river-mini-bear')
THEN WP.website_session_id ELSE NULL END)) to_next_page,

COUNT(DISTINCT(CASE WHEN WP.pageview_url IN
('/the-original-mr-fuzzy','/the-forever-love-bear','/the-birthday-sugar-panda','/the-hudson-river-mini-bear')
THEN WP.website_session_id ELSE NULL END))/

 COUNT(DISTINCT(CASE WHEN WP.pageview_url = '/products' THEN WP.website_session_id ELSE
NULL END)) clickthough_products,

COUNT(DISTINCT(CASE WHEN WP.pageview_url = '/products' THEN orders.order_id ELSE NULL
END))  orders_placed,

 COUNT(DISTINCT(CASE WHEN WP.pageview_url = '/products' THEN orders.order_id ELSE NULL
END))/COUNT(DISTINCT(CASE WHEN WP.pageview_url = '/products' THEN WP.website_session_id
ELSE NULL END)) product_to_order_cnv_rt

FROM website_pageviews WP
LEFT JOIN orders
ON orders.website_session_id = WP.website_session_id
GROUP BY 1,2
```

Year	Mon	sessions	to_products	to_next_page	clickthough_products	orders_placed	product_to_order_cnv_rt
2012	3	1879	743	530	0.7133	60	0.0808
2012	4	3735	1447	1029	0.7111	99	0.0684
2012	5	3736	1584	1135	0.7165	108	0.0682
2012	6	3965	1752	1247	0.7118	140	0.0799
2012	7	4249	2018	1438	0.7126	169	0.0837
2012	8	6098	3012	2198	0.7297	228	0.0757
2012	9	6546	3126	2258	0.7223	287	0.0918
2012	10	8183	4030	2948	0.7315	371	0.0921
2012	11	14011	6743	4849	0.7191	618	0.0917
2012	12	10073	5013	3620	0.7221	506	0.1009
2013	1	6401	3380	2595	0.7678	391	0.1157
2013	2	7169	3685	2803	0.7607	497	0.1349
2013	3	6264	3371	2576	0.7642	385	0.1142
2013	4	7972	4362	3356	0.7694	553	0.1268
2013	5	8450	4684	3609	0.7705	571	0.1219
2013	6	8325	4600	3536	0.7687	594	0.1291
2013	7	8904	5020	3890	0.7749	603	0.1201
2013	8	9180	5226	3951	0.7560	608	0.1163
2013	9	9580	5399	4072	0.7542	629	0.1165
2013	10	10774	6038	4564	0.7559	708	0.1173
2013	11	14033	7886	5900	0.7482	861	0.1092
2013	12	15735	8840	7026	0.7948	1047	0.1184
2014	1	14826	7790	6386	0.8198	983	0.1262
2014	2	16286	7960	6486	0.8148	1021	0.1283
2014	3	15671	8110	6669	0.8223	1065	0.1313
2014	4	17353	9744	7957	0.8166	1241	0.1274
2014	5	18063	10261	8466	0.8251	1368	0.1333
2014	6	17715	10011	8260	0.8251	1239	0.1238
2014	7	19039	10837	8958	0.8266	1287	0.1188
2014	8	18591	10768	8980	0.8340	1324	0.1230
2014	9	19515	11128	9154	0.8226	1424	0.1280
2014	10	21528	12335	10237	0.8299	1609	0.1304
2014	11	25125	14476	12020	0.8303	1985	0.1371
2014	12	29722	17240	14609	0.8474	2314	0.1342
2015	1	25337	15217	12992	0.8538	2099	0.1379
2015	2	23780	14373	12187	0.8479	2067	0.1438
2015	3	15084	9022	7723	0.8560	1254	0.1390

We see a gradual increase in Clickthrough rate of product page, product to order conversion rate throughout the period Click through rate has increased from roughly 71% to 85%. Conversion rate has also increased from 8% to 14%.

<u>Q.9 : Cross sell analysis</u> - 4th product is available to be purchased as primary product since 5th DEC 2014(previously only cross sold item). Since 5th dec 2014, show how well each product cross sells with one another

Approach :
Performing a join between orders and order_items table. For each order placed, getting primary product_id from orders table and cross sell product_id (products which are not primary item)from order_items table.

```
WIth cte AS(

SELECT
   O.order_id,
   O.items_purchased,
   O.primary_product_id,
   OI.product_id AS X_sell_product
FROM orders O
INNER JOIN order_items OI
ON OI.order_id = O.order_id
WHERE O.created_at > '2014-12-05'
AND OI.is_primary_item =0)

SELECT
 primary_product_id,
COUNT(DISTINCT(order_id)) total_orders,

COUNT(DISTINCT( CASE WHEN X_sell_product = 1 THEN order_id ELSE NULL END))
product1_x_sold,

COUNT(DISTINCT(CASE WHEN X_sell_product = 2 THEN order_id ELSE NULL END))
product2_x_sold,

COUNT(DISTINCT(CASE WHEN X_sell_product = 3 THEN order_id ELSE NULL END))
product3_x_sold,

COUNT(DISTINCT(CASE WHEN X_sell_product = 4 THEN order_id ELSE NULL END))
product4_x_sold

FROM cte
GROUP BY 1;
```

primary_product_id	total_orders	product1_x_sold	product2_x_sold	product3_x_sold	product4_x_sold
1	1724	0	238	553	933
2	325	25	0	40	260
3	332	84	40	0	208
4	47	16	9	22	0

Product 4 cross sells best with other products.