Introduction

I've recently joined Maven Fuzzy Factory as an eCommerce Database Analyst, where I'm part of the startup team working closely with the CEO, Head of Marketing, and Website Manager. Our primary focus is to navigate the online retail space, analyze marketing channels, optimize website performance, and measure the impact of our new product launches.

Data Information

Our database plays a pivotal role in our analysis and decision-making processes. It consists of six key tables:

1. **website_sessions**: This table holds essential information about website sessions, including details like session duration, user source, campaign, device type, and referral sources.
2. **website_pageviews**: Here, we store data related to website page views, tracking user activity across our web pages.
3. **products**: This table is dedicated to our product catalog, containing valuable data about our products, including their names and creation dates.
4. **orders**: We keep track of all customer orders in this table, capturing data on order IDs, website sessions, user IDs, primary products, items purchased, prices, and costs.
5. **order_items**: Each order can consist of multiple items, and this table stores the details of these items, such as product IDs, prices, and costs.
6. **order_item_refunds**: In case of any refunds, we meticulously record refund data here, including refund amounts and their associations with orders and items.

As we dive deeper into the data, my goal is to become the data expert for Maven Fuzzy Factory, providing mission-critical analyses that help drive our business forward and achieve our goals.

Traffic Source Analysis

In the realm of eCommerce, understanding where your customers are coming from and which channels are driving the highest quality traffic is paramount. This analysis focuses on leveraging the UTM parameters stored in our database to identify paid website sessions and determine the revenue generated by these paid campaigns.

1. Identifying Traffic Sources

We commence by exploring the size of various traffic sources, primarily examining the UTM content that drives the most sessions within the specified dataset range.

```
SELECT utm_content,
       Count(DISTINCT website_session_id) AS sessions
FROM   website_sessions
WHERE  website_session_id BETWEEN 1000 AND 2000
GROUP  BY utm_content
ORDER  BY sessions DESC;
```

Result:

utm_content	sessions
g_ad_1	975
null	18
g_ad_2	6
b_ad_2	2

In this context, 'g_ad_1' stands out as the dominant traffic source, driving 975 sessions. Null entries are also present, representing traffic without specific UTM content, totaling 18 sessions. Additionally, 'g_ad_2' and 'b_ad_2' contribute smaller session counts.

2. Linking Traffic Sources to Orders

Next, we expand our analysis to link these traffic sources to orders, gaining insights into how effectively these sources convert sessions into actual orders.

```
SELECT a.utm_content,
       Count(DISTINCT a.website_session_id) AS sessions,
       Count(DISTINCT b.order_id)           AS orders
FROM   website_sessions a
       LEFT JOIN orders b
              ON b.website_session_id = a.website_session_id
WHERE  a.website_session_id BETWEEN 1000 AND 2000
GROUP  BY a.utm_content
ORDER  BY sessions DESC;
```

Result:

utm_content	sessions	orders
g_ad_1	975	35
null	18	0
g_ad_2	6	0
b_ad_2	2	0

Unsurprisingly, 'g_ad_1' still leads in sessions with 975, contributing 35 orders. However, it's notable that 'null,' 'g_ad_2,' and 'b_ad_2' did not result in any orders during this period.

3. Session-to-Order Conversion Rate

To provide a comprehensive view, we calculate the session-to-order conversion rate for each traffic source.

```sql
SELECT a.utm_content,
       Count(DISTINCT a.website_session_id)    AS sessions,
       Count(DISTINCT b.order_id)              AS orders,
       Round(( Count(DISTINCT b.order_id) / Count(DISTINCT
             a.website_session_id) ) * 100, 2) AS sess_ord_conv_rate
FROM   website_sessions a
       LEFT JOIN orders b
              ON b.website_session_id = a.website_session_id
WHERE  a.website_session_id BETWEEN 1000 AND 2000
GROUP  BY a.utm_content
ORDER  BY sessions DESC;
```

Result:

utm_content	sessions	orders	sess_ord_conv_rate
g_ad_1	975	35	3.59
null	18	0	0.00
g_ad_2	6	0	0.00
b_ad_2	2	0	0.00

When assessing the session-to-order conversion rates, 'g_ad_1' shows a rate of 3.59%. The other sources, including 'null,' 'g_ad_2,' and 'b_ad_2,' did not yield any conversions during this analyzed period.

In summary, this analysis provides insights into the effectiveness of various traffic sources in driving sessions and orders, with 'g_ad_1' being the most significant contributor to both sessions and orders. Understanding these conversion rates is crucial for optimizing our marketing strategies and improving the overall business performance.

Analysis of Website Sessions – UTM Source Breakdown

Request Explanation: Cindy, the date requested is April 12, 2012. The objective is to gain insights into the bulk of website sessions leading up to that date, specifically focusing on the breakdown by UTM source, campaign, and referring domain if available.

SQL Query:

```
select utm_content, utm_campaign, http_referer, count(DISTINCT website_session_id) as sessions
from website_sessions
where created_at < '2012-04-12'
GROUP BY utm_content, utm_campaign, http_referer
ORDER BY sessions DESC;
```

SQL Query Explanation:

1. `select utm_content, utm_campaign, http_referer, count(DISTINCT website_session_id) as sessions` : This part of the query selects the relevant columns from the `website_sessions` table, including UTM content, UTM campaign, HTTP referer, and calculates the count of distinct website session IDs, which represents the number of sessions.
2. `from website_sessions` : It specifies the source table from which the data is retrieved, which is the `website_sessions` table.
3. `where created_at < '2012-04-12'` : This clause filters the data based on the `created_at` column, ensuring that only records with a creation date before April 12, 2012, are included in the analysis.
4. `GROUP BY utm_content, utm_campaign, http_referer` : This clause groups the data by UTM content, UTM campaign, and HTTP referer, allowing for a breakdown of sessions by these parameters.
5. `ORDER BY sessions DESC` : The results are then ordered in descending order based on the number of sessions, with the most significant number of sessions appearing first in the output.

Result:

utm_content	utm_campaign	http_referer	sessions
g_ad_1	nonbrand	https://www.gsearch.com	3613

utm_content	utm_campaign	http_referer	sessions
			28
		https://www.gsearch.com	27
g_ad_2	brand	https://www.gsearch.com	26
	7	https://www.bsearch.com	
b_ad_2	brand	https://www.bsearch.com	7

Interpretation:

The provided data showcases a comprehensive breakdown of website sessions, sorted by UTM source, campaign, and referring domain. It's apparent that 'g_ad_1' nonbrand sessions from 'https://www.gsearch.com' dominate with 3613 sessions. Notably, there are several entries with null values for UTM content and campaign, which collectively contribute 28 sessions.

Follow-up:

As we observe the dominance of 'g_ad_1' nonbrand sessions, it is advisable to conduct a more detailed analysis of this particular source to identify optimization opportunities. To gain further insights and establish next steps, involving Tom for his expertise is a practical course of action.

Request 1:

Tom wants to determine the conversion rate (CVR) from website sessions to orders, specifically focusing on sessions generated through the "gsearch" source with a "nonbrand" campaign. The goal is to calculate whether the CVR meets the minimum threshold of 4% needed to make the financial numbers work. If the CVR is below this threshold, adjustments to reduce search bids might be necessary. Tom's request aims to evaluate the economic viability of their paid search campaigns.

SQL Query:

```sql
SELECT Count(DISTINCT a.website_session_id)    AS sessions,
       Count(DISTINCT b.order_id)              AS orders,
       Round(( Count(DISTINCT b.order_id) / Count(DISTINCT
             a.website_session_id) ) * 100, 2) AS CVR
FROM   website_sessions a
       LEFT JOIN orders b
              ON b.website_session_id = a.website_session_id
WHERE  a.created_at < '2012-04-14'
       AND utm_source = "gsearch"
       AND utm_campaign = "nonbrand"
ORDER  BY 3 DESC;
```

SQL Query Explanation:

- The query calculates the CVR by counting the number of distinct website sessions and orders.
- It computes the CVR by dividing the number of orders by the number of sessions and then multiplying by 100 to express it as a percentage.
- The query filters data based on the "gsearch" source and "nonbrand" campaign to focus on sessions from this specific marketing channel.
- Results are sorted in descending order based on the CVR.

Answer:

# sessions	orders	CVR
3895	112	2.88

Answer Interpretation: The analysis shows that the CVR for sessions generated through the "gsearch" source with a "nonbrand" campaign is approximately 2.88%. This CVR falls short of the 4% threshold needed to make the economics work for the paid search campaigns. Consequently, there is a need to reduce search bids since the current conversion rate indicates that they are over-spending.

Comment by Tom: Tom acknowledges the analysis and mentions that, based on the findings, they need to dial down their search bids. The reason for this is that they are currently over-spending on their campaigns due to the lower-than-desired conversion rate. The analysis has helped them make more cost-effective decisions and saved them money.

Request 2: The request is to analyze traffic sources and bid optimization based on website sessions and orders data. The goal is to understand the value of different segments of paid traffic for bid optimization.

Query 1: Trended Sessions This query calculates the trended sessions by year and week. It focuses on sessions falling within the specified `website_session_id` range. The results include the year, week, the start date of the week, and the number of sessions for each week.

```
select year(created_at), week(created_at),
min(date(created_at)) as week_start,
count(DISTINCT website_session_id) as sessions
from website_sessions
where website_session_id BETWEEN 100000 AND 115000
GROUP BY 1,2;
```

Explanation 1: The query extracts trended sessions, which are useful for understanding website traffic patterns over time. It filters sessions within the defined `website_session_id` range and groups them by year and week. The results show the number of sessions for each week, helping identify trends.

Answer 1: The first answer provides trended sessions data, including the year, week, week start date, and the number of sessions for each week within the specified `website_session_id` range.

Query 2: Pivot Table in SQL using Orders Data This query constructs a pivot table to analyze primary product categories' performance in terms of single-item and two-item orders. It counts the number of single-item and two-item orders for each primary product category within the specified `order_id` range.

```
select primary_product_id,
count(DISTINCT case when items_purchased = 1 then order_id else null end) as count_single_item_orders,
count(DISTINCT case when items_purchased = 2 then order_id else null end) as count_two_item_orders
from orders
```

```
where order_id between 31000 AND 32000
GROUP BY 1;
```

Explanation 2: This query creates a pivot table to show how different primary product categories perform in terms of order types. It categorizes orders as single-item or two-item orders and counts them for each primary product category within the specified `order_id` range. This analysis helps understand how products perform with different order types.

Answer 2: The second answer offers a pivot table displaying the count of single-item and two-item orders for each primary product category in the specified `order_id` range.

Request 3 (May 10, 2012): The request is to analyze the trended session volume for the "gsearch nonbrand" traffic source by week. This analysis aims to determine if the bid changes made on 2012-04-15 have affected the volume of sessions for this traffic source.

Query: This SQL query retrieves the week's starting date and the number of distinct website sessions for the "gsearch nonbrand" traffic source. It filters the data based on the created date and specific UTM source and campaign conditions.

```
SELECT Min(Date(created_at))              AS week_start,
       Count(DISTINCT website_session_id) AS sessions
FROM   website_sessions
WHERE  created_at < '2012-05-10'
       AND utm_source = "gsearch"
       AND utm_campaign = "nonbrand"
GROUP  BY Yearweek(created_at);
```

Explanation: The query extracts data on the number of website sessions for "gsearch nonbrand" by weeks. It calculates the starting date of each week and counts the sessions within the specified date range. The data is grouped by the year and week of the created date.

Answer:

# week_start	sessions
2012-03-19	896

# week_start	sessions
2012-03-25	956
2012-04-01	1152
2012-04-08	983
2012-04-15	621
2012-04-22	594
2012-04-29	681
2012-05-06	399

The answer table presents the week's starting date and the corresponding number of sessions for "gsearch nonbrand." This data allows for the assessment of changes in session volume over time, particularly concerning the bid adjustments.

Comment by the Requester (Tom - May 10, 2012): The requester acknowledges the analysis and expresses that it appears that bid changes have a significant impact on the sensitivity of "gsearch nonbrand" traffic. While the goal is to maximize volume, it's crucial not to overspend on ads, and they plan to contemplate potential strategies based on this insight.

Request 4 (May 11, 2012): The request is to calculate conversion rates from website session to order, categorized by device type (desktop and mobile). This analysis aims to compare the performance of desktop and mobile devices. The goal is to identify whether desktop performance is superior to mobile performance, as this insight could inform bid adjustments for desktop to potentially increase sales volume.

Query: This SQL query retrieves data related to website sessions and orders. It calculates the number of sessions, orders, and the conversion rate (CVR) for each device type. The data is filtered based on various conditions, including the created date and specific UTM source and campaign criteria.

```
SELECT a.device_type,
       Count(DISTINCT a.website_session_id)    AS sessions,
       Count(DISTINCT b.order_id)              AS orders,
```

```
       Round(( Count(DISTINCT b.order_id) / Count(DISTINCT a.website_session_id) ) * 100, 2) AS CVR
FROM   website_sessions a
       LEFT JOIN orders b
              ON b.website_session_id = a.website_session_id
WHERE  a.created_at < '2012-05-11'
       AND utm_source = "gsearch"
       AND utm_campaign = "nonbrand"
GROUP BY 1
ORDER  BY 3 DESC;
```

Explanation: The query fetches data related to the number of website sessions and orders, and it calculates the conversion rate (CVR) for both desktop and mobile devices. It joins data from the "website_sessions" and "orders" tables based on the website session ID. The data is filtered by specific conditions, including the created date, UTM source, and campaign criteria. The result is grouped by device type and sorted by the order count in descending order.

Answer:

# device_type	sessions	orders	CVR
desktop	3911	146	3.73
mobile	2492	24	0.96

The answer table presents the device types (desktop and mobile), along with the number of sessions, orders, and the corresponding conversion rates (CVR). This data provides insights into the performance of each device type, with desktop showing a significantly higher CVR compared to mobile.

Comment by the Requester (Tom - May 11, 2012): The requester acknowledges the results and confirms their decision to increase bids specifically for desktop devices. By bidding higher, they aim to improve their ranking in auctions, which, based on the provided insights, is expected to boost sales. The requester praises the analysis and expresses gratitude for the insights.

Request 5 (June 09, 2012): In this request, the requester asks for an analysis of weekly trends for both desktop and mobile devices following recent bid changes for "gsearch nonbrand" campaigns. The analysis aims to understand the impact on website session volumes

for desktop and mobile devices after the bid changes. The specified date range for the analysis is from April 15, 2012, to June 9, 2012. This period is considered the baseline for comparison.

Query: This SQL query is designed to retrieve data related to website sessions. It calculates the count of sessions for both desktop and mobile devices and categorizes them by week. The data is filtered based on specific conditions, including the created date and UTM source and campaign criteria.

```
SELECT
min(date(created_at)) as week_start,
Count(DISTINCT case when device_type = "desktop" then website_session_id else null end) AS dtop_sessions,
Count(DISTINCT case when device_type = "mobile" then website_session_id else null end) AS mob_sessions
FROM   website_sessions
WHERE  created_at >= '2012-04-15' AND created_at < '2012-06-9'
and utm_source = "gsearch"
and utm_campaign = "nonbrand"
GROUP BY yearweek(created_at);
```

Explanation: The query retrieves website session data for both desktop and mobile devices, categorizing sessions by week. It uses conditional statements to count sessions specifically for each device type. The data is filtered based on specific date ranges, UTM source, and campaign criteria, allowing the comparison of trends for desktop and mobile devices.

Answer:

# week_start	dtop_sessions	mob_sessions
2012-04-15	383	238
2012-04-22	360	234
2012-04-29	425	256
2012-05-06	430	282
2012-05-13	403	214

# week_start	dtop_sessions	mob_sessions
2012-05-20	661	190
2012-05-27	585	183
2012-06-03	582	157

The answer table presents the week start date, along with the number of sessions for desktop and mobile devices in each corresponding week. The data shows how session volumes have changed during the specified period, following the bid changes. This information provides insights into the impact of bid adjustments on website session volumes for different devices.

Extra Analysis: An additional SQL query is provided, showing the device type, week start date, session count, order count, and conversion rate (CVR) for both desktop and mobile devices during the same date range. This extra analysis allows for a more detailed understanding of the performance of each device type.

```
SELECT a.device_type,
min(date(a.created_at)) as week_start,
Count(DISTINCT a.website_session_id)    AS sessions,
       Count(DISTINCT b.order_id)              AS orders,
       Round(( Count(DISTINCT b.order_id) / Count(DISTINCT
             a.website_session_id) ) * 100, 2) AS CVR
FROM   website_sessions a
       LEFT JOIN orders b
              ON b.website_session_id = a.website_session_id
WHERE  a.created_at>='2012-04-15' AND a.created_at < '2012-06-9'
and utm_source = "gsearch"
and utm_campaign = "nonbrand"
GROUP BY 1, yearweek(a.created_at);
```

ANSWER:

# device_type	week_start	sessions	orders	CVR
desktop	2012-04-15	383	11	2.87
desktop	2012-04-22	360	13	3.61
desktop	2012-04-29	425	12	2.82
desktop	2012-05-06	430	14	3.26
desktop	2012-05-13	403	15	3.72
desktop	2012-05-20	661	25	3.78
desktop	2012-05-27	585	27	4.62
desktop	2012-06-03	582	25	4.30
mobile	2012-04-15	238	6	2.52
mobile	2012-04-22	234	1	0.43
mobile	2012-04-29	256	2	0.78
mobile	2012-05-06	282	1	0.35
mobile	2012-05-13	214	3	1.40
mobile	2012-05-20	190	0	0.00
mobile	2012-05-27	183	2	1.09
mobile	2012-06-03	157	2	1.27

Comment by the Requester (Tom - June 09, 2012): The requester acknowledges the analysis results and expresses satisfaction with the insights. They note that the mobile performance appears to have been relatively flat or slightly declining, while desktop performance has

improved due to the bid changes made based on the previous conversion rate analysis. The requester appreciates the positive direction in which things are moving.

ANALYZING TOP WEBSITE CONTENT:

The goal of this analysis is to understand which pages on a website receive the most traffic. This information can help identify the most popular content and, in turn, guide improvements to enhance the user experience or business strategies.

SQL Query:

```
-- Retrieve the first 100 website_pageviews
select * from website_pageviews
where website_pageview_id < 1000;
```

Explanation:

This SQL query retrieves the first 100 rows from the `website_pageviews` table based on the `website_pageview_id` column. It provides a glimpse of the data, showing the pageview details, including the `website_pageview_id`, `created_at` timestamp, `website_session_id`, and the `pageview_url`.

Answer:

# website_pageview_id	created_at	website_session_id	pageview_url
1	2012-03-19 08:04:16	1	/home
2	2012-03-19 08:16:49	2	/home
3	2012-03-19 08:26:55	3	/home
4	2012-03-19 08:37:33	4	/home
5	2012-03-19 09:00:55	5	/home

# website_pageview_id	created_at	website_session_id	pageview_url
6	2012-03-19 09:05:46	6	/home
7	2012-03-19 09:06:27	7	/home
8	2012-03-19 09:10:08	6	/products
9	2012-03-19 09:10:52	6	/the-original-mr-fuzzy
10	2012-03-19 09:14:02	6	/cart
...	...	...	...

The answer shows a portion of the `website_pageviews` table, indicating the website pages visited during specific sessions.

SQL Query:

```
-- Top-Content analysis [pages with most views]
select pageview_url, count(DISTINCT website_pageview_id) as pageviews
from website_pageviews
where website_pageview_id < 1000
GROUP BY 1
ORDER BY 2 DESC;
```

Explanation:

This SQL query aims to identify the most viewed website pages. It counts the distinct `website_pageview_id` values for each `pageview_url`, filtering only the records where `website_pageview_id` is less than 1000. The results are then grouped by `pageview_url` and ordered by the count of pageviews in descending order.

Answer:

# pageview_url	pageviews
/home	523
/products	195
/the-original-mr-fuzzy	134
/cart	56
/shipping	39
/billing	34
/thank-you-for-your-order	18

The answer table displays the website pages with the most views (pageview_url) and the respective count of pageviews. The "/home" page is the most visited, with 523 pageviews.

SQL Query:

```
-- Top ENTRY Pages
with
cte1 as(select website_session_id, min(website_pageview_id) as first_entry
from website_pageviews
where website_pageview_id < 1000
GROUP BY 1)
SELECT b.pageview_url, count(DISTINCT a.website_session_id ) as number_of_times_first_entered from cte1 a
left join website_pageviews b on a.first_entry = b.website_pageview_id
GROUP BY 1;
```

Explanation:

This SQL query focuses on identifying the top entry pages. It utilizes a Common Table Expression (CTE) named `cte1` to find the first pageview (`first_entry`) for each session (`website_session_id`) from the `website_pageviews` table where `website_pageview_id` is less

than 1000. It then joins the CTE with the `website_pageviews` table to find the pageview_url of the first entry. The results are grouped by `pageview_url`, and the count of distinct sessions is calculated to show how many times each page served as the entry point.

Answer:

# pageview_url	number_of_times_first_entered
/home	523

The answer table presents the pageview_url that served as the entry point the most, which is "/home," with 523 occurrences.

In summary, this analysis provides insights into the most popular pages on the website, with "/home" being the top-viewed page and the most frequently used entry point. This information can be valuable for optimizing and tailoring content and strategies to meet user preferences and business objectives.

Request 6 - June 09, 2012:

Hi there! I'm Morgan, the new Website Manager. Could you help me get my head around the site by pulling the most-viewed website pages, ranked by session volume? Thanks! -Morgan

SQL Query:

```
select pageview_url,
count(DISTINCT website_pageview_id) as sessions
from website_pageviews
where created_at < '2012-06-09'
GROUP BY 1
ORDER BY 2 DESC;
```

Explanation:

In this request, Morgan, the Website Manager, asks for assistance in understanding the most-viewed website pages, ranked by session volume. The SQL query retrieves data from the `website_pageviews` table, counting the number of distinct session IDs for each

`pageview_url` before June 9, 2012. The results are grouped by `pageview_url` and ordered by the count of sessions in descending order.

Answer:

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

The answer table provides a list of the most-viewed website pages, with session counts. It reveals that the homepage ("/home") received the highest number of sessions, followed by the products page and the Mr. Fuzzy page.

Comment Thank you! It definitely seems like the homepage, the products page, and the Mr. Fuzzy page get the bulk of our traffic. I would like to understand traffic patterns more. I'll follow up soon with a request to look at entry pages. Thanks! -Morgan

Request 7 - June 12, 2012:

Hi there! Would you be able to pull a list of the top entry pages? I want to confirm where our users are hitting the site. If you could pull all entry pages and rank them on entry volume, that would be great. Thanks! -Morgan

SQL Query:

```
with
cte1 as(select website_session_id, min(website_pageview_id) as first_entry
from website_pageviews
```

```
where created_at < '2012-06-12'
GROUP BY 1)
SELECT b.pageview_url, count(DISTINCT a.website_session_id ) as number_of_times_first_entered from cte1 a
left join website_pageviews b on a.first_entry = b.website_pageview_id
GROUP BY 1;
```

Explanation:

In the second request, Morgan seeks to understand the entry points of website users. The SQL query first creates a Common Table Expression (CTE) named `cte1`, which identifies the first entry page (`first_entry`) for each session before June 12, 2012. It groups the results by `website_session_id`. The main query then joins this CTE with the `website_pageviews` table to determine which pages users first enter and calculates the count of distinct sessions for each entry page.

Answer:

# pageview_url	number_of_times_first_entered
/home	10714

The answer table displays the top entry pages ranked by entry volume. In this case, the homepage ("/home") is the primary entry point, with 10,714 instances.

Comment: In the comment following the second request, Morgan acknowledges that all the website traffic comes in through the homepage. They find it quite obvious that this is the area where they should focus on making improvements, and they hint at the possibility of follow-up requests to examine the performance of the homepage. It shows an understanding of the importance of the homepage as the initial point of interaction with users and the potential for optimization in that area. Morgan's request and comments suggest a proactive approach to website management and improvement.

LANDING PAGE PERFORMANCE & TESTING:

The goal of this SQL analysis is to evaluate the performance of landing pages during a specific time period. The analysis involves multiple steps:

1. Finding the first website pageview for relevant sessions.
2. Identifying the landing page for each session.
3. Counting pageviews for each session to identify "bounces" (sessions where users viewed only one page).
4. Summarizing the total sessions and bounced sessions, categorized by landing pages.

SQL Query:

```sql
WITH
-- First website_pageview_id for relevant sessions
cte1 AS (
    SELECT website_session_id, MIN(website_pageview_id) AS first_entry
    FROM website_pageviews
    WHERE created_at BETWEEN '2014-01-01' AND '2014-02-01'
    GROUP BY 1
),
-- Identifying the landing page of each session
cte2 AS (
    SELECT a.website_session_id, b.pageview_url AS landing_page
    FROM cte1 a
    LEFT JOIN website_pageviews b ON a.first_entry = b.website_pageview_id
),
-- Counting pageviews for each session to identify "bounces"
cte3 AS (
    SELECT a.website_session_id, a.landing_page, COUNT(b.website_pageview_id) AS count_of_pages_viewed
    FROM cte2 a
    LEFT JOIN website_pageviews b ON b.website_session_id = a.website_session_id
    GROUP BY 1, 2
    HAVING COUNT(b.website_pageview_id) = 1
),
-- Merging all sessions and bounced sessions in a single table
cte4 AS (
    SELECT a.landing_page, a.website_session_id, b.website_session_id AS bounced_website_session_id
    FROM cte2 a
    LEFT JOIN cte3 b ON a.website_session_id = b.website_session_id
    ORDER BY a.website_session_id
```

```
)
-- Summarizing the total sessions and bounced sessions, by Landing Pages
SELECT landing_page,
       COUNT(DISTINCT website_session_id) AS sessions,
       COUNT(DISTINCT bounced_website_session_id) AS bounced_sessions,
       ROUND((COUNT(DISTINCT bounced_website_session_id) / COUNT(DISTINCT website_session_id)) * 100, 2) AS bounced_cvr
FROM cte4
GROUP BY 1
ORDER BY bounced_cvr DESC;
```

Explanation:

1. **cte1:** The first Common Table Expression (CTE) `cte1` identifies the first pageview (`first_entry`) for each relevant session that occurred between January 1, 2014, and February 1, 2014.
2. **cte2:** The second CTE `cte2` associates each session with its landing page by joining the first pageview URL with the session ID.
3. **cte3:** The third CTE `cte3` counts the number of pageviews for each session, and only includes sessions where users viewed just one page, indicating a "bounce."
4. **cte4:** The fourth CTE `cte4` merges all sessions with their bounced sessions by joining `cte2` and `cte3` .
5. The final query summarizes the data. It groups the results by landing pages and calculates the total sessions, bounced sessions, and the bounce rate (bounced_cvr) as a percentage of total sessions.

Answer:

# landing_page	sessions	bounced_sessions	bounced_cvr
/lander-3	4232	2606	61.58
/lander-2	6500	2855	43.92
/home	4093	1575	38.48

# landing_page	sessions	bounced_sessions	bounced_cvr
/products	1	0	0.00

The answer table provides landing page performance data for the specified time period. It includes the landing page URLs, the number of sessions, the number of bounced sessions, and the bounce rate. The landing page "/lander-3" has the highest bounce rate at 61.58%, followed by "/lander-2" at 43.92%. The homepage "/home" and "/products" have lower bounce rates.

This analysis helps in understanding how well landing pages are retaining users, and it highlights areas that may need improvement to reduce bounce rates and increase user engagement.

Request 8 (June 14, 2012): Morgan requested information about the performance of the landing page, specifically the homepage. They wanted to know the sessions, bounced sessions, and the bounce rate for traffic landing on the homepage.

SQL Query:

```
WITH
-- First website_pageview_id for relevant sessions
cte1 AS (
    SELECT website_session_id, MIN(website_pageview_id) AS first_entry
    FROM website_pageviews
    WHERE created_at < '2012-06-14'
    GROUP BY 1
),
-- Identifying the landing page of each session
cte2 AS (
    SELECT a.website_session_id, b.pageview_url AS landing_page
    FROM cte1 a
    LEFT JOIN website_pageviews b ON a.first_entry = b.website_pageview_id
),
-- Counting pageviews for each session to identify "bounces"
cte3 AS (
    SELECT a.website_session_id, a.landing_page, COUNT(b.website_pageview_id) AS count_of_pages_viewed
    FROM cte2 a
    LEFT JOIN website_pageviews b ON b.website_session_id = a.website_session_id
```

```
    GROUP BY 1, 2
    HAVING COUNT(b.website_pageview_id) = 1
),
-- Merging all sessions and bounced sessions in a single table
cte4 AS (
    SELECT a.landing_page, a.website_session_id, b.website_session_id AS bounced_website_session_id
    FROM cte2 a
    LEFT JOIN cte3 b ON a.website_session_id = b.website_session_id
    ORDER BY a.website_session_id
)
-- Summarizing the total sessions and bounced sessions, by Landing Pages
SELECT landing_page,
       COUNT(DISTINCT website_session_id) AS sessions,
       COUNT(DISTINCT bounced_website_session_id) AS bounced_sessions,
       ROUND((COUNT(DISTINCT bounced_website_session_id) / COUNT(DISTINCT website_session_id)) * 100, 2) AS bounced_cvr
FROM cte4
GROUP BY 1
ORDER BY bounced_cvr DESC;
```

Explanation: The SQL query begins by identifying the first website pageview for relevant sessions within a specified time period. It then associates each session with its landing page and counts pageviews to identify bounced sessions (sessions with only one pageview). The data is summarized for the homepage, showing the number of sessions, bounced sessions, and the bounce rate.

Answer:

# landing_page	sessions	bounced_sessions	bounced_cvr
/home	11048	6538	59.18

The answer table provides the requested information for the homepage. The homepage "/home" had a total of 11,048 sessions, out of which 6,538 sessions resulted in bounces, resulting in a bounce rate of 59.18%.

Comment by the Requester: Morgan acknowledges the high bounce rate of almost 60% and expresses concern about the quality of paid search traffic. They plan to create a custom landing page for search and set up an experiment to analyze its performance with the help of

more data. The comment indicates that Morgan intends to take actions to improve the bounce rate on the homepage.

Request 9 (July 28, 2012): Morgan requested bounce rate information for two specific landing pages, "/home" and "/lander-1." These pages were part of a 50/50 test for gsearch nonbrand traffic. Morgan wanted to evaluate the new page, "/lander-1," by comparing the bounce rates for the two pages during the time period when "/lander-1" was receiving traffic.

SQL Query:

```
-- Finding out the /lander-1 first view date
SELECT created_at, Min(website_pageview_id)
FROM website_pageviews
WHERE pageview_url = "/lander-1"
GROUP BY 1
LIMIT 1;

WITH
-- First website_pageview_id for relevant sessions
cte1 AS (
    SELECT a.website_session_id, Min(a.website_pageview_id) AS first_entry
    FROM website_pageviews a
    INNER JOIN website_sessions b
    ON b.website_session_id = a.website_session_id
    AND b.created_at > '2012-06-19'
    AND b.created_at < '2012-07-28'
    AND b.utm_source = "gsearch"
    AND b.utm_campaign = "nonbrand"
    GROUP BY 1
),
-- Identifying the landing page of each session
cte2 AS (
    SELECT a.website_session_id, b.pageview_url AS landing_page
    FROM cte1 a
    LEFT JOIN website_pageviews b ON a.first_entry = b.website_pageview_id
),
-- Counting pageviews for each session to identify "bounces"
cte3 AS (
```

```
    SELECT a.website_session_id, a.landing_page, COUNT(b.website_pageview_id) AS count_of_pages_viewed
    FROM cte2 a
    LEFT JOIN website_pageviews b ON b.website_session_id = a.website_session_id
    GROUP BY 1, 2
    HAVING COUNT(b.website_pageview_id) = 1
),
-- Merging all sessions and bounced sessions in a single table
cte4 AS (
    SELECT a.landing_page, a.website_session_id, b.website_session_id AS bounced_website_session_id
    FROM cte2 a
    LEFT JOIN cte3 b ON a.website_session_id = b.website_session_id
    ORDER BY a.website_session_id
)
-- Summarizing the total sessions and bounced sessions, by Landing Pages
SELECT landing_page,
       COUNT(DISTINCT website_session_id) AS sessions,
       COUNT(DISTINCT bounced_website_session_id) AS bounced_sessions,
       ROUND((COUNT(DISTINCT bounced_website_session_id) / COUNT(DISTINCT website_session_id)) * 100, 2) AS bounce_rate
FROM cte4
GROUP BY 1
ORDER BY bounce_rate DESC;
```

Explanation: The SQL query begins by identifying the first view date of the "/lander-1" landing page. It then calculates the bounce rates for both "/home" and "/lander-1" landing pages during the specified time period and for gsearch nonbrand traffic. The bounce rate is calculated as the percentage of bounced sessions out of the total sessions.

Answer:

# landing_page	sessions	bounced_sessions	bounce_rate
/home	2261	1319	58.34
/lander-1	2316	1233	53.24

The answer table provides the requested information for the two landing pages. For "/home," there were 2,261 sessions with 1,319 bounced sessions, resulting in a bounce rate of 58.34%. For "/lander-1," there were 2,316 sessions with 1,233 bounced sessions, resulting in a lower bounce rate of 53.24%.

Comment by the Requester: Morgan expresses satisfaction with the results, noting that the custom landing page "/lander-1" has a lower bounce rate, indicating success. Morgan plans to update campaigns to direct all nonbrand paid traffic to the new page. They also express the intention to have trends analyzed in a few weeks to ensure that things have moved in the right direction.

Request 10 (August 31, 2012): Morgan requested data related to paid search nonbrand traffic landing on two pages, "/home" and "/lander-1," trended weekly since June 1st. The goal was to confirm that the traffic routing was correct and to assess the impact of the change on the overall paid search bounce rate.

SQL Query:

```
WITH
-- First website_pageview_id for relevant sessions
cte1 AS (
    SELECT a.website_session_id, Min(a.website_pageview_id) AS first_entry, a.created_at
    FROM website_pageviews a
    INNER JOIN website_sessions b
    ON b.website_session_id = a.website_session_id
    AND a.created_at > '2012-06-01'
    AND a.created_at < '2012-08-31'
    AND b.utm_source = "gsearch"
    AND b.utm_campaign = "nonbrand"
    GROUP BY 1, 3
),
-- Identifying the landing page of each session
cte2 AS (
    SELECT a.website_session_id, a.first_entry, b.pageview_url AS landing_page, a.created_at
    FROM cte1 a
    LEFT JOIN website_pageviews b ON a.first_entry = b.website_pageview_id
),
-- Counting pageviews for each session to identify "bounces"
cte3 AS (
```

```
    SELECT a.website_session_id, a.first_entry, a.landing_page, Count(b.website_pageview_id) AS count_of_pages_viewed, a.created_
    FROM cte2 a
    LEFT JOIN website_pageviews b ON b.website_session_id = a.website_session_id
    GROUP BY 1, 2, 5
)
-- Summarizing the total sessions and bounced sessions, by Landing Pages
SELECT Min(Date(created_at)) AS week_start,
       Round(( Count(DISTINCT CASE WHEN count_of_pages_viewed = 1 THEN website_session_id ELSE NULL END) / Count(DISTINCT website
       Count(DISTINCT CASE WHEN landing_page = "/home" THEN website_session_id ELSE NULL END) AS home_sessions,
       Count(DISTINCT CASE WHEN landing_page = "/lander-1" THEN website_session_id ELSE NULL END) AS lander1_sessions
FROM cte3
GROUP BY Yearweek(created_at);
```

Explanation: The SQL query retrieves data for two landing pages, "/home" and "/lander-1," and trends it weekly since June 1st for paid search nonbrand traffic. The query calculates the bounce rate, home sessions, and "/lander-1" sessions. Data is grouped by week using the `Yearweek` function.

Answer:

# week_start	bounce_rate	home_sessions	lander1_sessions
2012-06-01	60.23	175	0
2012-06-03	58.71	792	0
2012-06-10	61.60	875	0
2012-06-17	55.82	492	350
2012-06-24	58.28	369	386
2012-07-01	58.21	392	388
2012-07-08	56.68	390	411
2012-07-15	54.24	429	421

# week_start	bounce_rate	home_sessions	lander1_sessions
2012-07-22	51.38	402	394
2012-07-29	49.71	33	995
2012-08-05	53.82	0	1087
2012-08-12	51.40	0	998
2012-08-19	50.05	0	1012
2012-08-26	53.78	0	833

The answer table provides the requested data, showing the trend for each week. It includes the weekly start date, bounce rate, home sessions, and "/lander-1" sessions.

Comment by the Requester: Morgan expresses gratitude for the provided data, indicating that both pages received traffic for a while and the full switch to the custom lander was successful. They also note that the overall bounce rate has decreased over time, which is considered positive. Morgan mentions plans for a deep dive into the site and hints at future requests.

ANALYZING & TESTING CONVERSION FUNNELS:

Conversion funnel analysis is a crucial aspect of understanding and optimizing the user journey towards making a purchase or completing a specific action on a website. It involves examining the steps in the conversion flow to evaluate how many users progress through each step and how many drop off at each stage. This analysis helps businesses identify bottlenecks and opportunities for improvement in their conversion process.

Business Context:

The request seeks to perform a mini conversion funnel analysis from the "/lander-2" page to the "/cart" page. The primary objectives are to determine the number of users who reach each step in the funnel and calculate drop-off rates. This analysis is specifically focused on the "/lander-2" traffic and customers who have an interest in a product named "Mr. Fuzzy."

SQL Query 1 (Counting Users at Each Funnel Step):

```sql
WITH
-- Step 1: Select All Pageviews for Relevant Sessions (CTE1)
cte1 AS (
    SELECT a.website_session_id, b.pageview_url, b.created_at as pageview_created_at,
    CASE WHEN pageview_url = '/products' THEN 1 ELSE 0 END AS product_page,
    CASE WHEN pageview_url = '/the-original-mr-fuzzy' THEN 1 ELSE 0 END AS mrfuzzy_page,
    CASE WHEN pageview_url = '/cart' THEN 1 ELSE 0 END AS cart_page
    FROM website_sessions a
    LEFT JOIN website_pageviews b ON a.website_session_id = b.website_session_id
    WHERE a.created_at BETWEEN '2014-01-01' AND '2014-02-01'
    AND b.pageview_url IN ('/lander-2', '/products', '/the-original-mr-fuzzy', '/cart')
    ORDER BY 1, 3
),
-- Step 2: Identify Each Relevant Pageview as a Funnel Step (CTE2)
cte2 AS (
    SELECT website_session_id, MAX(product_page) AS product_made_it, MAX(mrfuzzy_page) AS mrfuzzy_made_it, MAX(cart_page) AS cart
    FROM cte1
    GROUP BY 1
)
-- Step 3: Create the Session-Level Conversion Funnel View (Main Query)
SELECT
    COUNT(DISTINCT website_session_id) AS sessions,
    COUNT(DISTINCT CASE WHEN product_made_it = 1 THEN website_session_id ELSE NULL END) AS to_product,
    COUNT(DISTINCT CASE WHEN mrfuzzy_made_it = 1 THEN website_session_id ELSE NULL END) AS to_mrfuzzy,
    COUNT(DISTINCT CASE WHEN cart_made_it = 1 THEN website_session_id ELSE NULL END) AS to_cart
FROM cte2;
```

Explanation of Query 1:

1. **Step 1 (CTE1)** selects relevant pageviews for sessions within the specified date range and matching specific pageview URLs that correspond to the conversion funnel steps. It assigns binary values to each pageview URL to track if a session reached a specific step.

2. **Step 2 (CTE2)** summarizes the data by grouping it based on the website session. It calculates binary values for each session, indicating whether they reached the '/products,' '/the-original-mr-fuzzy,' and '/cart' pages.

3. **Step 3 (Main Query)** calculates the following metrics:

    - `sessions` : The total number of distinct sessions that meet the criteria.
    - `to_product` : The number of sessions that reached the '/products' page (the first step of the funnel).
    - `to_mrfuzzy` : The number of sessions that reached the '/the-original-mr-fuzzy' page (the second step of the funnel).
    - `to_cart` : The number of sessions that reached the '/cart' page (the final step of the funnel).

Answer Table for Query 1:

# sessions	to_product	to_mrfuzzy	to_cart
10644	7789	4777	2889

- `sessions` : 10,644 sessions were considered in the analysis.
- `to_product` : 7,789 sessions progressed to the '/products' page.
- `to_mrfuzzy` : 4,777 sessions reached the '/the-original-mr-fuzzy' page.
- `to_cart` : 2,889 sessions advanced to the '/cart' page.

SQL Query 2 (Calculating Click-Through Rates):

```
WITH
-- Reusing the CTE1 and CTE2 from Query 1
-- Step 3: Create the Session-Level Conversion Funnel View (Main Query)
SELECT
    COUNT(DISTINCT website_session_id) AS sessions,
    ROUND((COUNT(DISTINCT CASE WHEN product_made_it = 1 THEN website_session_id ELSE NULL END) / COUNT(DISTINCT website_session_i
    ROUND((COUNT(DISTINCT CASE WHEN mrfuzzy_made_it = 1 THEN website_session_id ELSE NULL END) / COUNT(DISTINCT CASE WHEN product
```

```
    ROUND((COUNT(DISTINCT CASE WHEN cart_made_it = 1 THEN website_session_id ELSE NULL END) / COUNT(DISTINCT CASE WHEN mrfuzzy_ma
FROM cte2;
```

Explanation of Query 2:

This query is a continuation of the analysis, using the same CTEs (CTE1 and CTE2) established in Query 1. It calculates click-through rates for each step of the conversion funnel.

- `lander-clickthrough_rate` : The click-through rate from '/lander-2' to '/products.'
- `product-clickthrough_rate` : The click-through rate from '/products' to '/the-original-mr-fuzzy.'
- `mrfuzzy-clickthrough_rate` : The click-through rate from '/the-original-mr-fuzzy' to '/cart.'

Answer Table for Query 2:

# sessions	lander_clickthrough_rate	product_clickthrough_rate	mrfuzzy_clickthrough_rate
10644	73.18	61.33	60.48

- `sessions` : 10,644 sessions are part of the analysis.
- `lander-clickthrough_rate` : The click-through rate from '/lander-2' to '/products' is 73.18%.
- `product-clickthrough_rate` : The click-through rate from '/products' to '/the-original-mr-fuzzy' is 61.33%.
- `mrfuzzy-clickthrough-rate` : The click-through rate from '/the-original-mr-fuzzy' to '/cart' is 60.48%.

Interpretation:

- The analysis in Query 1 provides insights into the number of users at each step of the conversion funnel. It identifies the drop-off rates as users progress through the funnel.
- The analysis in Query 2 calculates click-through rates, which are essential for understanding the effectiveness of guiding users through each step of the funnel.

In summary, these SQL queries and the resulting tables help assess and optimize the conversion funnel's performance for "/lander-2" traffic, particularly for users interested in "Mr. Fuzzy." The click-through rates indicate how well users are transitioning from one step to the next in the funnel.

Request 11 (September 5, 2012):

Morgan, the requester, is interested in understanding the user journey of "gsearch" visitors on their website, specifically, how visitors navigate from the "/lander-1" page to placing an order. The objective is to create a full conversion funnel analysis, breaking down how many customers successfully complete each step of the process. The analysis should start at the "/lander-1" page and encompass the entire journey to the "thank you" page. The data used for this analysis spans from August 5th to September 5th, 2012.

SQL Query 1 (Counting Users at Each Funnel Step):

```
WITH
-- Step 1: Select All Pageviews for Relevant Sessions (CTE1)
cte1 AS (
    SELECT a.website_session_id, b.pageview_url, b.created_at as pageview_created_at,
    CASE WHEN pageview_url = '/products' THEN 1 ELSE 0 END AS product_page,
    CASE WHEN pageview_url = '/the-original-mr-fuzzy' THEN 1 ELSE 0 END AS mrfuzzy_page,
    CASE WHEN pageview_url = '/cart' THEN 1 ELSE 0 END AS cart_page,
    CASE WHEN pageview_url = '/shipping' THEN 1 ELSE 0 END AS shipping_page,
    CASE WHEN pageview_url = '/billing' THEN 1 ELSE 0 END AS billing_page,
    CASE WHEN pageview_url = '/thank-you-for-your-order' THEN 1 ELSE 0 END AS thanks_page
    FROM website_sessions a
    LEFT JOIN website_pageviews b ON a.website_session_id = b.website_session_id
    WHERE a.created_at BETWEEN '2012-08-05' AND '2012-09-05'
    AND utm_source = "gsearch"
    AND utm_campaign = "nonbrand"
    AND b.pageview_url IN ('/lander-1', '/products', '/the-original-mr-fuzzy', '/cart', '/shipping', '/billing', '/thank-you-for-
    ORDER BY 1, 3
),
-- Step 2: Identify Each Relevant Pageview as a Funnel Step (CTE2)
cte2 AS (
    SELECT website_session_id,
    MAX(product_page) AS product_made_it,
```

```
    MAX(mrfuzzy_page) AS mrfuzzy_made_it,
    MAX(cart_page) AS cart_made_it,
    MAX(shipping_page) AS shipping_made_it,
    MAX(billing_page) AS billing_made_it,
    MAX(thanks_page) AS thanks_made_it
    FROM cte1
    GROUP BY 1
)
-- Step 3: Create the Session-Level Conversion Funnel View (Main Query)
SELECT
    COUNT(DISTINCT website_session_id) AS sessions,
    COUNT(DISTINCT CASE WHEN product_made_it = 1 THEN website_session_id ELSE NULL END) AS to_product,
    COUNT(DISTINCT CASE WHEN mrfuzzy_made_it = 1 THEN website_session_id ELSE NULL END) AS to_mrfuzzy,
    COUNT(DISTINCT CASE WHEN cart_made_it = 1 THEN website_session_id ELSE NULL END) AS to_cart,
    COUNT(DISTINCT CASE WHEN shipping_made_it = 1 THEN website_session_id ELSE NULL END) AS to_shipping,
    COUNT(DISTINCT CASE WHEN billing_made_it = 1 THEN website_session_id ELSE NULL END) AS to_billing,
    COUNT(DISTINCT CASE WHEN thanks_made_it = 1 THEN website_session_id ELSE NULL END) AS to_thanks
FROM cte2;
```

Explanation of Query 1:

1. **Step 1 (CTE1)** selects relevant pageviews for sessions within the specified date range, focusing on "gsearch" visitors who arrived via the "nonbrand" campaign. It matches specific pageview URLs that correspond to the conversion funnel steps and assigns binary values to track if a session reached a particular step.

2. **Step 2 (CTE2)** summarizes the data by grouping it based on the website session. It calculates binary values for each session, indicating whether they reached the '/products,' '/the-original-mr-fuzzy,' '/cart,' '/shipping,' '/billing,' and '/thank-you-for-your-order' pages.

3. **Step 3 (Main Query)** calculates the following metrics:

    - `sessions` : The total number of distinct sessions that meet the criteria.
    - `to_product` : The number of sessions that reached the '/products' page (the first step of the funnel).
    - `to_mrfuzzy` : The number of sessions that reached the '/the-original-mr-fuzzy' page.
    - `to_cart` : The number of sessions that reached the '/cart' page.

- `to_shipping` : The number of sessions that reached the '/shipping' page.
- `to_billing` : The number of sessions that reached the '/billing' page.
- `to_thanks` : The number of sessions that reached the '/thank-you-for-your-order' page (the final step of the funnel).

ANSWER:

# sessions	to_product	to_mrfuzzy	to_cart	to_shipping	to_billing	to_thanks
4493	2115	1567	683	455	361	158

SQL Query 2 (Calculating Click-Through Rates):

```
WITH
-- Reusing the CTE1 and CTE2 from Query 1
-- Step 3: Create the Session-Level Conversion Funnel View (Main Query)
SELECT
    COUNT(DISTINCT website_session_id) AS sessions,
    ROUND((COUNT(DISTINCT CASE WHEN product_made_it = 1 THEN website_session_id ELSE NULL END) / COUNT(DISTINCT website_session_i
    ROUND((COUNT(DISTINCT CASE WHEN mrfuzzy_made_it = 1 THEN website_session_id ELSE NULL END) / COUNT(DISTINCT CASE WHEN product
    ROUND((COUNT(DISTINCT CASE WHEN cart_made_it = 1 THEN website_session_id ELSE NULL END) / COUNT(DISTINCT CASE WHEN mrfuzzy_ma
    ROUND((COUNT(DISTINCT CASE WHEN shipping_made_it = 1 THEN website_session_id ELSE NULL END) / COUNT(DISTINCT CASE WHEN cart_m
    ROUND((COUNT(DISTINCT CASE WHEN billing_made_it = 1 THEN website_session_id ELSE NULL END) / COUNT(DISTINCT CASE WHEN shippin
    ROUND((COUNT(DISTINCT CASE WHEN thanks_made_it = 1 THEN website_session_id ELSE NULL END) / COUNT(DISTINCT CASE WHEN billing_

ade_it = 1 THEN website_session_id ELSE NULL END) * 100), 2) AS billing_clickthrough_rate
FROM cte2;
```

Explanation of Query 2:

This query calculates the click-through rates between each step of the conversion funnel. It builds upon the CTEs created in Query 1.

- `lander_clickthrough_rate` : The percentage of sessions that proceed from "/lander-1" to "/products."

- `product_clickthrough_rate` : The percentage of sessions that progress from "/products" to "/the-original-mr-fuzzy," relative to those who reached "/products."
- `mrfuzzy_clickthrough_rate` : The percentage of sessions that advance from "/the-original-mr-fuzzy" to "/cart," relative to those who reached "/the-original-mr-fuzzy."
- `cart_clickthrough_rate` : The percentage of sessions that continue from "/cart" to "/shipping," relative to those who reached "/cart."
- `shipping_clickthrough_rate` : The percentage of sessions that move from "/shipping" to "/billing," relative to those who reached "/shipping."
- `billing_clickthrough_rate` : The percentage of sessions that reach the "/thank-you-for-your-order" page after "/billing," relative to those who reached "/billing."

ANSWER:

# sessions	lander_clickthrough_rate	product_clickthrough_rate	mrfuzzy_clickthrough_rate	cart_clickthrough_rate	shippin
4493	47.07	74.09	43.59	66.62	79.34

Interpretation of the Answers:

- Out of 4,493 sessions initiated at "/lander-1," 2,115 sessions proceeded to the "/products" page (to_product).
- Subsequently, 1,567 sessions reached the "/the-original-mr-fuzzy" page (to_mrfuzzy).
- Of those, 683 sessions made it to the "/cart" page (to_cart).
- The funnel continues, with 455 sessions reaching the "/shipping" page (to_shipping).
- 361 sessions progressed to the "/billing" page (to_billing).
- Finally, 158 sessions successfully completed the funnel, landing on the "/thank-you-for-your-order" page (to_thanks).

Comment by the Requester:

The requester's comment in the initial email expressed the desire to understand where visitors arriving from "gsearch" lose interest in their website. The request specifically asked for a full conversion funnel analysis to track the number of customers at each step of the process.

The data used for analysis spans from August 5th to September 5th, 2012.

The analysis enables the requester to pinpoint which step in the conversion process has the highest drop-off rate, providing insights for optimization and potential improvements.

Request 12 (November 10, 2012):

Morgan, the requester, inquired about the effectiveness of an updated billing page, denoted as "/billing-2," compared to the original "/billing" page. The objective was to determine what percentage of sessions on these pages ultimately led to a successful order placement. It's worth noting that this test encompassed all website traffic, not just "gsearch" visitors. The request was made on November 10, 2012.

SQL Query:

```sql
-- First, finding the starting point to frame the analysis
SELECT MIN(website_pageview_id) AS first_entry
FROM website_pageviews
WHERE pageview_url = '/billing-2';
-- 53550 is the first entry pageview id

WITH
-- CTE1: Joining Website Pageviews and Orders
cte1 AS (
    SELECT a.website_session_id, a.pageview_url AS billing_version_seen, b.order_id
    FROM website_pageviews a
    LEFT JOIN orders b ON b.website_session_id = a.website_session_id
    WHERE a.created_at < '2012-11-10' -- Time of assignment
    AND a.website_pageview_id >= 53550 -- First pageview id
    AND a.pageview_url IN ('/billing', '/billing-2')
)
-- Main Query: Calculate Billing Page Conversion Metrics
SELECT billing_version_seen,
    COUNT(DISTINCT website_session_id) AS sessions,
    COUNT(DISTINCT order_id) AS orders,
    ROUND((COUNT(DISTINCT order_id) / COUNT(DISTINCT website_session_id)) * 100, 2) AS billing_order_rt
```

```
FROM cte1
GROUP BY billing_version_seen;
```

Explanation of the Query:

The SQL query's purpose is to compare the performance of two billing pages, "/billing" and "/billing-2," in terms of their ability to convert visitors into customers.

1. The initial part of the query determines the starting point for the analysis. It identifies the earliest pageview ID for the "/billing-2" page. In this case, the first entry pageview ID is found to be 53550.

2. The query uses a Common Table Expression (CTE), denoted as `cte1`, to join website pageviews and order data. It selects relevant sessions where pageviews occurred before November 10, 2012 (the time of assignment) and had a pageview ID greater than or equal to 53550. It filters the data for both "/billing" and "/billing-2" pages.

3. In the main part of the query, it calculates the following metrics for each billing page version:

    - `billing_version_seen`: Indicates the version of the billing page (either "/billing" or "/billing-2").
    - `sessions`: The count of distinct website sessions that viewed the respective billing page.
    - `orders`: The count of distinct orders placed by users who viewed the billing page.
    - `billing_order_rt`: The percentage of sessions that resulted in orders on the respective billing page.

ANSWER:

# billing_version_seen	sessions	orders	billing_order_rt
/billing	657	300	45.66
/billing-2	654	410	62.69

Interpretation of the Answer:

The query yields the following results for the two billing page versions:

- For "/billing," there were 657 sessions, resulting in 300 orders, with a billing page order conversion rate of 45.66%.
- For "/billing-2," there were 654 sessions, resulting in 410 orders, with a billing page order conversion rate of 62.69%.

Comment by the Requester (November 10, 2012):

Morgan expresses enthusiasm upon receiving the results, as it indicates that the new billing page version ("/billing-2") is significantly outperforming the original billing page ("/billing") in converting customers. The requester plans to have the engineering team roll out the new billing page version to all customers immediately. Morgan acknowledges that the insights provided through this analysis have contributed to a substantial increase in revenue.

This comment reflects the positive impact of data-driven decision-making on the company's bottom line.

Channel Portfolio Optimization

The concept here revolves around the optimization of a marketing channel portfolio. Analyzing a portfolio of marketing channels involves evaluating the performance of different channels, which can include various online advertising sources, social media platforms, or other marketing avenues. The goal is to use data-driven insights to allocate the marketing budget efficiently and maximize the effectiveness of marketing campaigns.

Request Explanation: The request is to analyze the performance of various marketing channels based on data related to website sessions and orders. The analysis focuses on a specific date range, from January 1, 2014, to February 1, 2014. The key metrics of interest are the number of sessions generated by each marketing channel, the number of orders placed through these sessions, and the conversion rate (CVR) for each channel.

SQL Query:

```
SELECT a.utm_content, COUNT(DISTINCT a.website_session_id) AS sessions,
    COUNT(DISTINCT b.order_id) AS orders,
    ROUND((COUNT(DISTINCT b.order_id) / COUNT(DISTINCT a.website_session_id)) * 100, 2) AS cvr
FROM website_sessions a
LEFT JOIN orders b ON b.website_session_id = a.website_session_id
WHERE a.created_at BETWEEN '2014-01-01' AND '2014-02-01'
```

```
GROUP BY 1
ORDER BY 2 DESC;
```

Query Explanation:

The SQL query retrieves and calculates metrics related to marketing channels. Here's how it works:

1. The `SELECT` statement extracts the `utm_content` (a parameter that can identify different marketing channels), the count of distinct website sessions, the count of distinct orders, and the conversion rate (CVR) for each channel.
2. The `FROM` clause specifies the source tables for the query, which are `website_sessions` (aliased as 'a') and `orders` (aliased as 'b'). The query performs a left join to connect sessions with orders based on the `website_session_id`.
3. The `WHERE` clause filters the data, restricting it to sessions created between January 1, 2014, and February 1, 2014.
4. The `GROUP BY` clause groups the results by `utm_content`, enabling the calculation of metrics for each marketing channel.
5. The `ORDER BY` clause arranges the results in descending order of the number of sessions to see which channels generated the most sessions.

Answer Table:

The query yields a table with the following columns and data:

# utm_content	sessions	orders	cvr
g_ad_1	7500	543	7.24
null	2724	194	7.12
social_ad_1	1618	17	1.05
b_ad_1	1614	109	6.75
g_ad_2	1107	91	8.22

# utm_content	sessions	orders	cvr
b_ad_2	262	29	11.07

Interpretation of the Answer:

The answer table presents data for different marketing channels, focusing on the number of sessions, the number of orders, and the conversion rate (CVR) for each channel. Here's the interpretation of the results:

- **"g_ad_1"** generated the highest number of sessions, with 7,500 sessions, resulting in 543 orders and a CVR of 7.24%.
- An unidentified channel (blank entry) generated 2,724 sessions, with 194 orders and a CVR of 7.12%.
- **"social_ad_1"** resulted in 1,618 sessions but had a lower CVR of 1.05%, resulting in only 17 orders.
- **"b_ad_1"** contributed 1,614 sessions, 109 orders, and a CVR of 6.75%.
- **"g_ad_2"** generated 1,107 sessions with 91 orders, achieving a CVR of 8.22%.
- **"b_ad_2"** resulted in 262 sessions with 29 orders and the highest CVR at 11.07%.

The analysis provides insights into the performance of different marketing channels, helping marketing professionals allocate their budget and resources more effectively to achieve the desired conversion rates and optimize their channel portfolio. It also highlights the importance of tracking and analyzing CVR to assess channel effectiveness.

Request 13 (November 29, 2012):

Hi there,

With gsearch doing well and the site performing better, we launched a second paid search channel, bsearch, around August 22. Can you pull weekly trended session volume since then and compare to gsearch nonbrand so I can get a sense for how important this will be for the business?

Thanks, Tom

Request Explanation:

The request from Tom is to analyze the performance of a new paid search channel called "bsearch," which was launched on August 22. Specifically, he wants to understand how the session volume for this new channel compares to the existing "gsearch" nonbrand channel on a weekly basis. This comparison will help Tom gauge the significance of the new channel's impact on the business.

SQL Query:

```sql
SELECT
    MIN(DATE(created_at)) AS week_start_date,
    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
FROM website_sessions
WHERE created_at BETWEEN '2012-08-22' AND '2012-11-29'
    AND utm_campaign = 'nonbrand'
GROUP BY YEARWEEK(created_at);
```

Query Explanation:

The SQL query retrieves weekly trended session volumes for "gsearch" and "bsearch" channels. Here's how it works:

1. The `SELECT` statement selects the minimum date within each week as the `week_start_date`. It also counts the number of distinct sessions for "gsearch" and "bsearch" channels. These session counts are calculated using conditional aggregation.
2. The `FROM` clause specifies the source table as "website_sessions."
3. The `WHERE` clause filters the data to the specified date range, from August 22, 2012, to November 29, 2012, and selects only the records where the campaign is "nonbrand."
4. The results are grouped by the week using the `YEARWEEK` function, which groups sessions by the year and week.

Answer Table:

The query provides a table with the following columns and data:

# week_start_date	gsearch_sessions	bsearch_sessions
2012-08-22	590	197
2012-08-26	1056	343
2012-09-02	925	290
2012-09-09	951	329
2012-09-16	1151	365
2012-09-23	1050	321
2012-09-30	999	316
2012-10-07	1002	330
2012-10-14	1257	420
2012-10-21	1302	431
2012-10-28	1211	384
2012-11-04	1350	429
2012-11-11	1246	438
2012-11-18	3508	1093
2012-11-25	2286	774

Interpretation of the Answer:

The answer table presents a comparison of weekly session volumes between the "gsearch" nonbrand channel and the newly launched "bsearch" channel. Here's the interpretation:

- The "gsearch" channel consistently had a higher number of sessions compared to "bsearch" throughout the period.

- In the early weeks after the launch of "bsearch," "gsearch" had approximately three times as many sessions.
- While "bsearch" experienced growth in sessions over time, it did not surpass "gsearch."

Tom's follow-up comment suggests that "bsearch" tends to receive about one-third of the traffic of "gsearch." This insight is essential for assessing the importance of the new channel and planning future strategies.

Tom also mentions that he will follow up with requests to understand channel characteristics and conversion performance, indicating a deeper analysis of the "bsearch" channel to determine its impact on the business.

The analysis helps provide a clear picture of how the new marketing channel, "bsearch," is performing in terms of session volume compared to the existing "gsearch" channel.

Request 14 (November 30, 2012):

Hi there,

I'd like to learn more about the bsearch nonbrand campaign. Could you please pull the percentage of traffic coming on Mobile, and compare that to gsearch? Feel free to dig around and share anything else you find interesting. Aggregate data since August 22nd is great, no need to show trending at this point.

Thanks, Tom

Request Explanation:

Tom's request is to gain a deeper understanding of the "bsearch" nonbrand campaign. Specifically, he wants to know the percentage of website traffic that is coming from mobile devices for this campaign. Additionally, he wants to compare this data to the "gsearch" campaign. Tom encourages the analyst to explore and share any other interesting findings. The analysis should consider data since August 22nd.

SQL Query:

```
SELECT
    utm_source,
```

```
    COUNT(DISTINCT website_session_id) AS sessions,
    COUNT(DISTINCT CASE WHEN device_type = 'mobile' THEN website_session_id ELSE NULL END) AS mobile_sessions,
    ROUND((COUNT(DISTINCT CASE WHEN device_type = 'mobile' THEN website_session_id ELSE NULL END) / COUNT(DISTINCT website_sessio
    COUNT(DISTINCT website_session_id) - COUNT(DISTINCT CASE WHEN device_type = 'mobile' THEN website_session_id ELSE NULL END) A
    100 - ROUND((COUNT(DISTINCT CASE WHEN device_type = 'mobile' THEN website_session_id ELSE NULL END) / COUNT(DISTINCT website_
FROM website_sessions
WHERE created_at BETWEEN '2012-08-22' AND '2012-11-30'
    AND utm_campaign = 'nonbrand'
GROUP BY utm_source;
```

Query Explanation:

The SQL query is designed to analyze and compare the percentage of mobile and desktop traffic for the "bsearch" and "gsearch" nonbrand campaigns. Here's how the query works:

1. The `SELECT` statement includes the `utm_source` (campaign source), total sessions, mobile sessions, the percentage of mobile sessions (`pct_mobile`), desktop sessions, and the percentage of desktop sessions (`pct_desktop`).
2. In the `FROM` clause, data is selected from the "website_sessions" table.
3. The `WHERE` clause filters the data to the specified date range, from August 22, 2012, to November 30, 2012, and selects only the records where the campaign is "nonbrand."
4. The results are grouped by `utm_source` to provide separate statistics for the "bsearch" and "gsearch" campaigns.

Answer Table:

The query yields a table with the following columns and data:

# utm_source	sessions	mobile_sessions	pct_mobile	desktop_sessions	pct_desktop
bsearch	6522	562	8.62	5960	91.38
gsearch	20073	4921	24.52	15152	75.48

Interpretation of the Answer:

The answer table provides insights into the device-specific distribution of traffic for the "bsearch" and "gsearch" nonbrand campaigns. Here's the interpretation:

- **bsearch Campaign:**
  - Total sessions: 6,522
  - Mobile sessions: 562 (8.62% of total sessions)
  - Desktop sessions: 5,960 (91.38% of total sessions)
- **gsearch Campaign:**
  - Total sessions: 20,073
  - Mobile sessions: 4,921 (24.52% of total sessions)
  - Desktop sessions: 15,152 (75.48% of total sessions)

The analysis reveals significant differences in device distribution between the two campaigns. The "gsearch" campaign has a notably higher percentage of mobile sessions compared to the "bsearch" campaign. This insight can be valuable for optimizing marketing strategies and adjusting bids for each campaign based on device-specific performance.

Tom's follow-up comment acknowledges the interesting insights regarding the desktop-to-mobile splits. He emphasizes the importance of recognizing these differences between the channels and suggests that further analysis is needed to optimize bids effectively. Tom expresses his appreciation for the work and encourages continued efforts in this direction.

Request 15 (December 01, 2012):

Hi there,

I’m wondering if bsearch nonbrand should have the same bids as gsearch. Could you pull nonbrand conversion rates from session to order for gsearch and bsearch, and slice the data by device type? Please analyze data from August 22 to September 18; we ran a special pre-holiday campaign for gsearch starting on September 19th, so the data after that isn’t fair game.

Thanks, Tom

Request Explanation:

Tom's request is to compare the conversion rates from sessions to orders for the "bsearch" and "gsearch" nonbrand campaigns. He wants this comparison to be segmented by device type. The purpose is to determine if both campaigns should have the same bidding strategy. Tom specifies that the analysis should cover the period from August 22 to September 18, excluding data after September 19 due to a special pre-holiday campaign for "gsearch."

SQL Query:

```
select a.utm_source, a.device_type,
count(DISTINCT a.website_session_id) as sessions,
count(DISTINCT b.order_id) as orders,
round((count(DISTINCT b.order_id)/count(DISTINCT a.website_session_id))*100,2) as cvr
from website_sessions a
left join orders b on b.website_session_id = a.website_session_id
where a.created_at between '2012-08-22' AND '2012-09-18'
AND utm_campaign = 'nonbrand'
GROUP BY 1,2;
```

Query Explanation:

The SQL query is designed to calculate and compare conversion rates (CVR) from sessions to orders for the "bsearch" and "gsearch" nonbrand campaigns, while segmenting the data by device type. Here's how the query works:

1. The `SELECT` statement includes the `utm_source` (campaign source), `device_type`, total sessions, total orders, and the conversion rate (`cvr`).
2. In the `FROM` clause, data is selected from the "website_sessions" table, and an outer join with the "orders" table is performed based on the `website_session_id`.
3. The `WHERE` clause filters the data for the specified date range, from August 22, 2012, to September 18, 2012, and ensures that the campaign is "nonbrand."

4. The results are grouped by both `utm_source` and `device_type` to provide detailed conversion rate data for each combination.

Answer Table:

The query yields a table with the following columns and data:

# utm_source	device_type	sessions	orders	cvr
bsearch	desktop	1118	43	3.85
bsearch	mobile	125	1	0.80
gsearch	desktop	2850	130	4.56
gsearch	mobile	962	11	1.14

Interpretation of the Answer:

The answer table provides conversion rate data for the "bsearch" and "gsearch" nonbrand campaigns, segmented by device type. Here's the interpretation:

- **bsearch Campaign:**
  - Desktop sessions: 1,118
  - Desktop orders: 43
  - Desktop conversion rate (CVR): 3.85%
  - Mobile sessions: 125
  - Mobile orders: 1
  - Mobile conversion rate (CVR): 0.80%
- **gsearch Campaign:**
  - Desktop sessions: 2,850
  - Desktop orders: 130

- Desktop conversion rate (CVR): 4.56%
- Mobile sessions: 962
- Mobile orders: 11
- Mobile conversion rate (CVR): 1.14%

The analysis clearly shows that the two campaigns have different conversion rates, particularly when considering desktop and mobile devices. The "gsearch" campaign generally outperforms the "bsearch" campaign in terms of conversion rates.

In response to the analysis, Tom acknowledges the differences in performance between the two channels and expresses his intention to adjust the bidding strategy. He plans to bid down the "bsearch" campaign based on its underperformance. This demonstrates the practical implications of the analysis on optimizing the overall paid marketing budget. Tom appreciates the work and encourages further efforts in this direction.

Request 16 (December 22, 2012):

Hi there,

Based on your last analysis, we bid down bsearch nonbrand on December 2nd. Can you pull weekly session volume for gsearch and bsearch nonbrand, broken down by device, since November 4th? If you can include a comparison metric to show bsearch as a percent of gsearch for each device, that would be great too.

Thanks, Tom

Request Explanation:

Tom's request is to analyze the weekly session volumes for the "gsearch" and "bsearch" nonbrand campaigns, categorized by device type. This analysis covers the period from November 4th to December 22nd. Additionally, Tom requests a comparison metric that shows "bsearch" as a percentage of "gsearch" for each device type. The purpose is to understand how session volumes changed after they bid down the "bsearch" campaign and assess the impact on the two channels.

SQL Query:

```sql
select
min(date(created_at)) as week_start_date,
count(DISTINCT case when device_type = 'desktop' and utm_source = 'gsearch' then website_session_id else null end) as g_dtop_sess
count(DISTINCT case when device_type = 'desktop' and utm_source = 'bsearch' then website_session_id else null end) as b_dtop_sess
round((count(DISTINCT case when device_type = 'desktop' and utm_source = 'bsearch' then website_session_id else null end)/count(D
case when device_type = 'desktop' and utm_source = 'gsearch' then website_session_id else null end))*100,2) as b_pct_g_dtop,
count(DISTINCT case when device_type = 'mobile' and utm_source = 'gsearch' then website_session_id else null end) as g_mob_sessio
count(DISTINCT case when device_type = 'mobile' and utm_source = 'bsearch' then website_session_id else null end) as b_mob_sessio
round((count(DISTINCT case when device_type = 'mobile' and utm_source = 'bsearch' then website_session_id else null end)/count(DI
case when device_type = 'mobile' and utm_source = 'gsearch' then website_session_id else null end))*100,2) as b_pct_g_mob
from website_sessions
where created_at BETWEEN '2012-11-04' AND '2012-12-22'
AND utm_campaign = 'nonbrand'
GROUP BY yearweek(created_at);
```

Query Explanation:

The SQL query aims to calculate and compare weekly session volumes for the "gsearch" and "bsearch" nonbrand campaigns, divided by device type. It also includes a metric showing "bsearch" as a percentage of "gsearch" for each device type. Here's how the query works:

1. The `SELECT` statement includes several calculated fields:
    - `week_start_date`: The earliest date within each week.
    - `g_dtop_sessions`: Count of desktop sessions for "gsearch."
    - `b_dtop_sessions`: Count of desktop sessions for "bsearch."
    - `b_pct_g_dtop`: The percentage of "bsearch" sessions compared to "gsearch" for desktop.
    - `g_mob_sessions`: Count of mobile sessions for "gsearch."
    - `b_mob_sessions`: Count of mobile sessions for "bsearch."
    - `b_pct_g_mob`: The percentage of "bsearch" sessions compared to "gsearch" for mobile.
2. The data is selected from the "website_sessions" table, and conditions are set to filter sessions between November 4, 2012, and December 22, 2012, for the "nonbrand" campaign.

3. The results are grouped by the week's starting date using the `yearweek` function.

Answer Table:

The query provides a table with the following columns and data:

# week_start_date	g_dtop_sessions	b_dtop_sessions	b_pct_g_dtop	g_mob_sessions	b_mob_sessions	b_pct_g_mob
2012-11-04	1027	400	38.95	323	29	8.98
2012-11-11	956	401	41.95	290	37	12.76
2012-11-18	2655	1008	37.97	853	85	9.96
2012-11-25	2058	843	40.96	692	62	8.96
2012-12-02	1326	517	38.99	396	31	7.83
2012-12-09	1277	293	22.94	424	46	10.85
2012-12-16	1270	348	27.40	376	41	10.90

Interpretation of the Answer:

The answer table provides weekly session volume data for "gsearch" and "bsearch" nonbrand campaigns, broken down by device type, along with the percentage of "bsearch" as compared to "gsearch" for each device type. Here's the interpretation:

- **Desktop Sessions:**
    - For the week starting on November 4th, there were 1,027 "gsearch" desktop sessions and 400 "bsearch" desktop sessions, making "bsearch" 38.95% of "gsearch" for desktop.
    - The percentage fluctuates over the weeks but tends to be lower for "bsearch."
- **Mobile Sessions:**

- For mobile devices, "bsearch" has a much smaller share of sessions compared to "gsearch." The percentage of "bsearch" sessions relative to "gsearch" sessions is considerably lower on mobile.

Tom's comment acknowledges that traffic for both "gsearch" and "bsearch" declined after Black Friday and Cyber Monday. However, he notes that "bsearch" experienced a more substantial drop. He considers this to be acceptable, especially given the lower conversion rate for "bsearch." This insight reflects the impact of bid adjustments and provides valuable information for optimizing the marketing strategy. Tom appreciates the work that has been done in this regard.

Analyzing Direct Traffic

This analysis focuses on evaluating direct traffic sources to gain insights into how well a brand is performing and how effectively it drives business. Direct traffic typically consists of users who directly type in a website's URL, visit from bookmarks, or do not have referral sources. The goal is to categorize and quantify different types of direct traffic, such as organic traffic from specific search engines and other sources.

Request Explanation:

The SQL query is designed to categorize and count different types of direct traffic sources and the number of sessions associated with each source. It analyzes website sessions within a specific range of website_session_id values (between 100,000 and 115,000). The traffic sources are categorized into four groups:

1. 'direct_type_in': Users who directly type in the website URL or visit via bookmarks.
2. 'gsearch_organic': Users arriving from 'https://www.gsearch.com' with no specific utm_source, indicating organic traffic from Gsearch.
3. 'bsearch_organic': Users arriving from 'https://www.bsearch.com' with no specific utm_source, indicating organic traffic from Bsearch.
4. 'other': All other sources of direct traffic not falling into the above categories.

SQL Query:

```
select
    case
        when http_referer is null then 'direct_type_in'
        when http_referer = 'https://www.gsearch.com' AND utm_source is null then 'gsearch_organic'
        when http_referer = 'https://www.bsearch.com' AND utm_source is null then 'bsearch_organic'
```

```
        else 'other'
    end as Traffics,
    count(DISTINCT website_session_id) as sessions
from website_sessions
where website_session_id between 100000 and 115000
GROUP BY 1
ORDER BY 2 desc;
```

Query Explanation:

The SQL query uses a `CASE` statement to categorize the direct traffic based on the criteria explained in the request. Here's how the query works:

- It categorizes the traffic into four types: 'direct_type_in,' 'gsearch_organic,' 'bsearch_organic,' and 'other' based on the conditions defined.
- The data is selected from the "website_sessions" table, and the session IDs are limited to those falling within the specified range (between 100,000 and 115,000).
- The result is grouped by the 'Traffics' categories.
- The final output is ordered by the number of sessions in descending order.

Answer Table:

The answer table provides the following data:

# Traffics	sessions
other	12,760
direct_type_in	1,055
gsearch_organic	966
bsearch_organic	220

Interpretation of the Answer:

The answer table shows the count of sessions for different types of direct traffic sources within the specified session ID range. Here's the interpretation:

- 'other': This category has the highest number of sessions, with 12,760 sessions. These are likely visitors who accessed the website through various unspecified direct methods.
- 'direct_type_in': Users who directly typed in the website URL or used bookmarks accounted for 1,055 sessions.
- 'gsearch_organic': There were 966 sessions from users arriving through 'https://www.gsearch.com' with no specific utm_source, indicating organic traffic from Gsearch.
- 'bsearch_organic': Users arriving from 'https://www.bsearch.com' with no specific utm_source contributed 220 sessions, indicating organic traffic from Bsearch.

The analysis provides a breakdown of different types of direct traffic sources, allowing for a better understanding of how users access the website. This information can be valuable for assessing the brand's performance and the effectiveness of various traffic sources.

Request Explanation:

Cindy is requesting an analysis to understand the trend of different traffic sources by month (organic search, direct type-in, and paid brand search) and their relative percentage concerning paid search nonbrand. The goal is to ascertain if the brand is gaining momentum, particularly in organic, direct, and paid brand searches, and to demonstrate these growth trends as a percentage of the total paid nonbrand traffic.

SQL Queries:

1. Initial Query for Data Collection:

```
SELECT DISTINCT utm_source, utm_campaign, http_referer
FROM website_sessions
WHERE created_at < '2012-12-23';
```

This query fetches distinct values of 'utm_source', 'utm_campaign', and 'http_referer' from 'website_sessions' created before December 23, 2012, providing insight into various traffic sources.

ANSWER:

# utm_source	utm_campaign	http_referer
gsearch	nonbrand	https://www.gsearch.com
null	null	null
gsearch	brand	https://www.gsearch.com
null	null	https://www.gsearch.com
bsearch	brand	https://www.bsearch.com
null	null	https://www.bsearch.com
bsearch	nonbrand	https://www.bsearch.com

2. Channel Categorization Query:

```
SELECT DISTINCT
    CASE
        WHEN utm_source IS NULL AND http_referer IN ('https://www.gsearch.com', 'https://www.bsearch.com') THEN 'organic_search'
        WHEN utm_campaign = 'nonbrand' THEN 'paid_nonbrand'
        WHEN utm_campaign = 'brand' THEN 'paid_brand'
        WHEN utm_source IS NULL AND http_referer IS NULL THEN 'direct_type_in'
    END AS channel_group,
    utm_source,
    utm_campaign,
    http_referer
```

```
FROM website_sessions
WHERE created_at < '2012-12-23';
```

This query categorizes the traffic into different groups ('organic_search', 'paid_nonbrand', 'paid_brand', and 'direct_type_in') based on specific conditions related to 'utm_source', 'utm_campaign', and 'http_referer'.

ANSWER:

# channel_group	utm_source	utm_campaign	http_referer
paid_nonbrand	gsearch	nonbrand	https://www.gsearch.com
null	null	null	direct_type_in
paid_brand	gsearch	brand	https://www.gsearch.com
organic_search	null	null	https://www.gsearch.com
paid_brand	bsearch	brand	https://www.bsearch.com
organic_search	null	null	https://www.bsearch.com
paid_nonbrand	bsearch	nonbrand	https://www.bsearch.com

3. Analysis Query for Traffic Sessions by Month:

```
WITH cte1 AS (
    SELECT
        website_session_id,
        created_at,
        CASE
            WHEN utm_source IS NULL AND http_referer IN ('https://www.gsearch.com', 'https://www.bsearch.com') THEN 'organic_sear
            WHEN utm_campaign = 'nonbrand' THEN 'paid_nonbrand'
            WHEN utm_campaign = 'brand' THEN 'paid_brand'
```

```
            WHEN utm_source IS NULL AND http_referer IS NULL THEN 'direct_type_in'
        END AS channel_group
    FROM website_sessions
    WHERE created_at < '2012-12-23'
)
SELECT
    year(created_at) AS yr,
    month(created_at) AS month,
    COUNT(DISTINCT CASE WHEN channel_group = 'paid_nonbrand' THEN website_session_id ELSE NULL END) AS nonbrand,
    COUNT(DISTINCT CASE WHEN channel_group = 'paid_brand' THEN website_session_id ELSE NULL END) AS nonbrand,
    COUNT(DISTINCT CASE WHEN channel_group = 'paid_brand' THEN website_session_id ELSE NULL END) / COUNT(DISTINCT CASE WHEN chann
    COUNT(DISTINCT CASE WHEN channel_group = 'direct_type_in' THEN website_session_id ELSE NULL END) AS direct,
    COUNT(DISTINCT CASE WHEN channel_group = 'direct_type_in' THEN website_session_id ELSE NULL END) / COUNT(DISTINCT CASE WHEN c
    COUNT(DISTINCT CASE WHEN channel_group = 'organic_search' THEN website_session_id ELSE NULL END) AS organic,
    COUNT(DISTINCT CASE WHEN channel_group = 'organic_search' THEN website_session_id ELSE NULL END) / COUNT(DISTINCT CASE WHEN c
FROM cte1
GROUP BY 1,2;
```

ANSWER:

# yr	month	nonbrand	nonbrand	brand_pct_of_nonbrand	direct	direct_pct_of_nonbrand	organic	organic_pct_
2012	3	1852	10	0.0054	9	0.0049	8	0.0043
2012	4	3509	76	0.0217	71	0.0202	78	0.0222
2012	5	3295	140	0.0425	151	0.0458	150	0.0455
2012	6	3439	164	0.0477	170	0.0494	190	0.0552
2012	7	3660	195	0.0533	187	0.0511	207	0.0566
2012	8	5318	264	0.0496	250	0.0470	265	0.0498
2012	9	5591	339	0.0606	285	0.0510	331	0.0592
2012	10	6883	432	0.0628	440	0.0639	428	0.0622

# yr	month	nonbrand	nonbrand	brand_pct_of_nonbrand	direct	direct_pct_of_nonbrand	organic	organic_pct_
2012	11	12260	556	0.0454	571	0.0466	624	0.0509
2012	12	6643	464	0.0698	482	0.0726	492	0.0741

Interpretation of the Answer:

The answer table displays the breakdown of various traffic sources by month, illustrating the sessions for each category and their respective percentages concerning the total paid nonbrand sessions.

- For each month of the year 2012, the table includes counts for 'nonbrand,' 'paid_brand,' 'direct,' and 'organic' sessions, along with their proportions in relation to the total 'paid_nonbrand' sessions.
- These columns highlight the absolute growth of different types of sessions over time and demonstrate how each channel contributes as a proportion of the paid nonbrand traffic.
- The growing counts of 'brand,' 'direct,' and 'organic' sessions are depicted in both absolute numbers and their increasing shares in the total 'paid_nonbrand' sessions.

Comment Explanation by Cindy:

Cindy expresses satisfaction with the analysis results, indicating that the brand, direct, and organic traffic volumes are not only increasing but also growing as a percentage of the paid traffic volume. She mentions this data as an encouraging narrative to present to an investor, suggesting that it portrays positive momentum for the brand's organic and direct traffic, showcasing potential growth beyond reliance solely on paid traffic.

ANALYZING SEASONALITY & BUSINESS PATTERNS

The SQL query is designed to analyze business patterns, specifically focusing on the time and date of website sessions. It extracts information such as the session ID, the date and time of the session, the hour of the day, the day of the week, the quarter, the month, the date, and the week number for a range of website session IDs. This data can help in understanding the patterns and trends in website traffic, which is valuable for optimizing efficiency and anticipating future business trends.

SQL Query:

```sql
select
    website_session_id,
    created_at,
    hour(created_at) as hr,
    weekday(created_at) as wkday,
    case
        when weekday(created_at) = 0 then 'Monday'
        when weekday(created_at) = 1 then 'Tuesday'
        when weekday(created_at) = 2 then 'Wednesday'
        when weekday(created_at) = 3 then 'Thursday'
        when weekday(created_at) = 4 then 'Friday'
        when weekday(created_at) = 5 then 'Saturday'
        when weekday(created_at) = 6 then 'Sunday'
        else 'other_day'
    end as clean_weekday,
    quarter(created_at) as qtr,
    month(created_at) as month,
    date(created_at) as date,
    week(created_at) as wk
from website_sessions
where website_session_id between 150000 and 155000;
```

Explanation:

- The query selects specific columns from the `website_sessions` table and calculates additional attributes based on the `created_at` timestamp.
- `hour(created_at) as hr` extracts the hour of the day from the timestamp.
- `weekday(created_at) as wkday` calculates the numeric day of the week (0 for Sunday, 6 for Saturday).
- The `case` statement transforms the numeric day of the week into a human-readable format, such as 'Monday,' 'Tuesday,' etc., and assigns it to the column `clean_weekday`.
- `quarter(created_at) as qtr` extracts the quarter from the timestamp.

- `month(created_at) as month` gets the month.
- `date(created_at) as date` retrieves the date.
- `week(created_at) as wk` calculates the week number of the year.

This query is applied to a specific range of `website_session_id` values between 150000 and 155000, presumably to focus the analysis on a specific subset of website sessions.

Answer Table:

# website_session_id	created_at	hr	wkday	clean_weekday	qtr	month	date	wk
150000	2013-11-14 10:46:56	10	3	Thursday	4	11	2013-11-14	45
150001	2013-11-14 10:50:42	10	3	Thursday	4	11	2013-11-14	45
150002	2013-11-14 10:53:39	10	3	Thursday	4	11	2013-11-14	45
150003	2013-11-14 10:59:46	10	3	Thursday	4	11	2013-11-14	45
150004	2013-11-14 11:01:11	11	3	Thursday	4	11	2013-11-14	45
150005	2013-11-14 11:02:39	11	3	Thursday	4	11	2013-11-14	45
150006	2013-11-14 11:12:59	11	3	Thursday	4	11	2013-11-14	45
150007	2013-11-14 11:16:06	11	3	Thursday	4	11	2013-11-14	45
150008	2013-11-14 11:17:09	11	3	Thursday	4	11	2013-11-14	45
150009	2013-11-14 11:18:09	11	3	Thursday	4	11	2013-11-14	45

P.S: ONLY A FRACTION OF THE OUTPUT HAS BEEN SHOWN!

The answer table contains the extracted data for website sessions within the specified range of `website_session_id` values. It provides information on the session ID, the timestamp, the hour of the day, the day of the week, the day of the week in a more readable format, the quarter, the month, the date, and the week number for each session.

Interpretation: The query generates a dataset that can be used for analyzing business patterns. It helps in understanding how website traffic varies by different time-related attributes, such as hour, day of the week, and more. This information can be valuable for identifying trends and optimizing business operations based on the patterns observed in the data.

Request 17 Date: January 02, 2013

Request by Cindy: Good morning, 2012 was a great year for us. As we continue to grow, we should take a look at 2012's monthly and weekly volume patterns, to see if we can find any seasonal trends we should plan for in 2013. If you can pull session volume and order volume, that would be excellent. Thanks, -Cindy

SQL Query:

```
select year(a.created_at) as yr,
month(a.created_at) as month,
count(DISTINCT a.website_session_id) as sessions,
count(DISTINCT b.order_id) as orders
from website_sessions a
left join orders b on b.website_session_id = a.website_session_id
where a.created_at < '2013-01-01'
GROUP BY 1,2;
```

Explanation:

- The query is designed to provide an analysis of monthly volume patterns for the year 2012 by counting the number of website sessions and orders.
- It selects the year and month from the `created_at` timestamp, calculates the number of unique website sessions (sessions), and the number of unique orders (orders).
- The data is obtained from the `website_sessions` table and the `orders` table, with a LEFT JOIN based on the common key `website_session_id`.
- The data is filtered to include only records with a `created_at` date before January 1, 2013.
- The results are grouped by year and month, providing a summary of monthly session and order volumes for 2012.

Answer Table:

# yr	month	sessions	orders
2012	3	1879	60
2012	4	3734	99
2012	5	3736	108
2012	6	3963	140
2012	7	4249	169
2012	8	6097	228
2012	9	6546	287
2012	10	8183	371
2012	11	14011	618
2012	12	10072	506

The answer table displays the analysis of monthly volume patterns for the year 2012. It includes columns for the year, month, the number of website sessions (sessions), and the number of orders (orders) for each month.

Additionally, there is another SQL query related to weekly volume patterns.

SQL Query for Weekly Volume Patterns:

```
select min(date(a.created_at)) as week_start_date,
count(DISTINCT a.website_session_id) as sessions,
count(DISTINCT b.order_id) as orders
from website_sessions a
left join orders b on b.website_session_id = a.website_session_id
```

```
where a.created_at < '2013-01-01'
GROUP BY yearweek(a.created_at);
```

Explanation (Weekly Query):

- This query is similar to the monthly query but focuses on weekly volume patterns for the year 2012.
- It calculates the number of sessions and orders on a weekly basis.
- The `yearweek()` function is used to group the data by year and week.
- The results provide insights into the weekly session and order volumes for 2012.

Answer Table:

# week_start_date	sessions	orders
2012-03-19	896	25
2012-03-25	983	35
2012-04-01	1193	29
2012-04-08	1029	28
2012-04-15	679	22
2012-04-22	655	18
2012-04-29	770	19
2012-05-06	798	17
2012-05-13	706	23
2012-05-20	965	28

Interpretation: The results show that 2012 saw steady growth throughout the year, with significant volume increases during holiday months, particularly during the weeks of Black Friday and Cyber Monday. This information is crucial for planning customer support and inventory management strategies in 2013 to accommodate these seasonal trends.

Comment by Cindy: Cindy's comment expresses appreciation for the provided analysis, noting the steady growth observed throughout the year and the significance of holiday-related volume increases. She acknowledges the importance of considering these patterns when planning for customer support and inventory management in 2013. Overall, Cindy finds the analysis great and valuable for future business planning.

Request 18 Date: January 05, 2013

Request by Cindy: Good morning, We're considering adding live chat support to the website to improve our customer experience. Could you analyze the average website session volume, by hour of day and by day week, so that we can staff appropriately? Let's avoid the holiday time period and use a date range of Sep 15 - Nov 15, 2013. Thanks, Cindy

SQL Query:

```
with
cte1 as (
    select date(created_at) as created_date,
    weekday(created_at) as wkday,
    hour(created_at) as hr,
    count(DISTINCT website_session_id) as website_sessions

    from website_sessions
    where created_at between '2012-09-15' AND '2012-11-15'
    group by 1,2,3
)
select hr,
round(avg(case when wkday = 0 then website_sessions else null end),1) as mon,
round(avg(case when wkday = 1 then website_sessions else null end),1) as tue,
round(avg(case when wkday = 2 then website_sessions else null end),1) as wed,
round(avg(case when wkday = 3 then website_sessions else null end),1) as thu,
round(avg(case when wkday = 4 then website_sessions else null end),1) as fri,
round(avg(case when wkday = 5 then website_sessions else null end),1) as sat,
```

```
round(avg(case when wkday = 6 then website_sessions else null end),1) as sun
from cte1
GROUP BY 1
order by 1;
```

Explanation:

- Cindy's request aims to analyze the average website session volume to help determine staffing needs for adding live chat support to the website.
- The SQL query calculates the average number of website sessions per hour of the day and by the day of the week for the specified date range (September 15 - November 15, 2013).
- It starts by creating a Common Table Expression (CTE) named `cte1` to aggregate website session data by date, weekday, and hour, focusing on the specified date range.
- The main query then calculates the average session volume for each hour of the day on different days of the week (Monday to Sunday) based on the `cte1` data.
- The results are rounded to one decimal place and presented in a tabular format.

Answer Table:

# hr	mon	tue	wed	thu	fri	sat	sun
0	8.7	7.7	6.3	7.4	6.8	5.0	5.0
1	6.6	6.7	5.3	4.9	7.1	5.0	3.0
2	6.1	4.4	4.4	6.1	4.6	3.7	3.0
3	5.7	4.0	4.7	4.6	3.6	3.9	3.4
4	5.9	6.3	6.0	4.0	6.1	2.8	2.4
5	5.0	5.4	5.1	5.4	4.6	4.3	3.9
6	5.4	5.6	4.8	6.0	6.8	4.0	2.6

# hr	mon	tue	wed	thu	fri	sat	sun
7	7.3	7.8	7.4	10.6	7.0	5.7	4.8
8	12.3	12.2	13.0	16.5	10.5	4.3	4.1
9	17.6	15.7	19.6	19.3	17.5	7.6	6.0
10	18.4	17.7	21.0	18.4	19.0	8.3	6.3
11	18.0	19.1	24.9	21.6	20.9	7.2	7.7
12	21.1	23.3	22.8	24.1	19.0	8.6	6.1
13	17.8	23.0	20.8	20.6	21.6	8.1	8.4
14	17.9	21.6	22.3	18.5	19.5	8.7	6.7
15	21.6	17.1	25.3	23.5	21.3	6.9	7.1
16	21.1	23.7	23.7	19.6	20.9	7.6	6.6
17	19.4	15.9	20.2	19.8	12.9	6.4	7.6
18	12.7	15.0	14.8	15.3	10.9	5.3	6.8
19	12.4	14.1	13.3	11.6	14.3	7.1	6.4
20	12.1	12.4	14.2	10.6	10.3	5.7	8.4
21	9.1	12.6	11.4	9.4	7.3	5.7	10.2
22	9.1	10.0	9.8	12.1	6.0	5.7	10.2
23	8.8	8.6	9.6	10.6	7.6	5.3	8.3

The answer table provides the average website session volume by hour of the day and day of the week, allowing for a detailed view of website activity patterns during the specified date range.

Interpretation: Cindy's interpretation of the results suggests that ~10 sessions per hour per employee staffed is an appropriate benchmark. She plans to have one support staff available around the clock and then doubles the staffing to two support members from 8 am to 5 pm, Monday through Friday. This staffing approach is designed to align with the observed website session volume trends, ensuring that there are adequate support staff members available during peak activity hours.

Comment by Cindy: Cindy expresses her appreciation, stating that the provided analysis is really helpful for planning live chat support staffing. She mentions that she has been in discussions with support companies and notes that approximately 10 sessions per hour per employee is the right staffing ratio. She outlines her staffing plan based on the analysis, which involves 24/7 support with additional staffing during regular business hours to meet customer needs efficiently.

Product Sales Analysis

This SQL query is designed for product sales analysis. Analyzing product sales is crucial for businesses to understand the performance of each product, how they contribute to overall revenue, and how product launches affect the product portfolio. In this specific case, the analysis focuses on sales of primary products within a range of order IDs (between 10000 and 11000). The query retrieves key metrics, including the number of orders, total revenue, margin, and average revenue for each primary product.

Request Explanation: The request for this SQL query is to obtain insights into product sales within the specified order ID range. It aims to identify the performance of each primary product by analyzing order counts, total revenue, margin, and the average revenue per order for products within this range.

SQL Query:

```
select
 primary_product_id,
 count(order_id) as orders,
 sum(price_usd) as revenue,
 sum(price_usd - cogs_usd) as margin,
 avg(price_usd) as average_revenue
 from orders
 where order_id between 10000 and 11000
```

```
group by 1
order by 4 desc;
```

Explanation:

- The SQL query begins with a `SELECT` statement that retrieves specific information for analysis.
- It selects the `primary_product_id` to identify the product, and then aggregates data for each product by using several aggregate functions.
- The `COUNT` function counts the number of orders for each product, `SUM` calculates the total revenue for each product, and `SUM(price_usd - cogs_usd)` computes the margin by subtracting the cost of goods sold (COGS) from the revenue.
- The `AVG` function calculates the average revenue per order for each product.
- The data used for analysis is filtered using a `WHERE` clause, which restricts it to orders with order IDs between 10000 and 11000.
- The results are grouped by `primary_product_id` and ordered in descending order based on the margin.

Answer Table:

# primary_product_id	orders	revenue	margin	average_revenue
1	731	42009.62	25900.00	57.468700
2	144	9118.46	5710.00	63.322639
3	126	6474.61	4386.50	51.385794

The answer table provides insights into the sales performance of primary products within the specified order ID range. It includes columns for the primary product ID, the number of orders, total revenue, margin, and average revenue per order for each product.

Interpretation: The analysis reveals key information about the products sold within the specified range of order IDs:

- Product 1 had the highest number of orders (731), generating $42,009.62 in revenue, with a margin of $25,900.00. The average revenue per order for this product is approximately $57.47.

- Product 2 had 144 orders, resulting in $9,118.46 in revenue, a margin of $5,710.00, and an average revenue per order of approximately $63.32.
- Product 3 had 126 orders, leading to $6,474.61 in revenue, a margin of $4,386.50, and an average revenue per order of approximately $51.39.

This analysis provides valuable insights into the performance of these products, enabling data-driven decisions and strategies for the product portfolio. It can help businesses understand which products are top performers and may indicate the need for additional marketing efforts or adjustments to product offerings.

Request 19 (January 04, 2013):
Cindy has requested a deep dive analysis into the company's current flagship product to prepare for the upcoming launch of a new product. Specifically, she asks for monthly trends in sales, total revenue, and total margin generated by the business. The goal is to establish baseline data that will help in tracking the evolution of revenue and margin as the new product is introduced. This analysis will also provide insights into the company's overall growth pattern.

SQL Query:

```
select year(created_at) as year,
month(created_at) as month,
count(DISTINCT order_id) as number_of_sales,
sum(price_usd) as total_revenue,
sum(price_usd - cogs_usd) as total_margin
from orders
where created_at < '2013-01-04'
GROUP BY 1,2;
```

Explanation:

- The SQL query begins with a `SELECT` statement to extract relevant data for analysis.
- It selects the `year` and `month` from the `created_at` timestamp to organize the data into monthly intervals.
- The query calculates key metrics for each month:

    - `number_of_sales`: Counts the number of distinct order IDs to determine the total sales for the flagship product during each month.
    - `total_revenue`: Sums the total revenue generated by the product for each month.
    - `total_margin`: Computes the total margin for each month by subtracting the cost of goods sold (COGS) from the total revenue.
- The data used for analysis is filtered using a `WHERE` clause. It includes only orders created before January 4, 2013, providing historical data up to that point.
- The results are then grouped by the year and month using `GROUP BY`, allowing for the calculation of monthly trends.

Answer Table:

# year	month	number_of_sales	total_revenue	total_margin
2012	3	60	2999.40	1830.00
2012	4	99	4949.01	3019.50
2012	5	108	5398.92	3294.00
2012	6	140	6998.60	4270.00
2012	7	169	8448.31	5154.50
2012	8	228	11397.72	6954.00
2012	9	287	14347.13	8753.50
2012	10	371	18546.29	11315.50
2012	11	618	30893.82	18849.00
2012	12	506	25294.94	15433.00

The answer table displays the historical data regarding the flagship product's performance in terms of number of sales, total revenue, and total margin, organized on a monthly basis.

Interpretation:
The analysis presents monthly trends for the flagship product up to January 4, 2013. It shows a consistent pattern of growth throughout the year 2012, with the number of sales, total revenue, and total margin increasing over time. This historical data serves as valuable baseline information, enabling the company to monitor how revenue and margin evolve following the launch of the new product.

Cindy's comment expresses her satisfaction with the provided data, emphasizing its importance as a baseline for tracking the impact of the new product. She also appreciates the insight into the company's overall growth pattern, which can inform future business strategies and decision-making.

Request (April 05, 2013):
Cindy has requested a trended analysis for the period since April 1, 2013, following the launch of the company's second product on January 6th. She seeks insights into several key performance indicators to assess the impact of this new product. The analysis includes monthly order volume, overall conversion rates, revenue per session, and a breakdown of sales by product.

SQL Query:

```
select
    year(a.created_at) as year,
    month(a.created_at) as month,
    count(DISTINCT b.order_id) as orders,
    count(distinct b.order_id)/count(DISTINCT a.website_session_id) as cvr,
    round(sum(b.price_usd)/count(DISTINCT a.website_session_id),2) as revenue_sessions,
    count(distinct case when b.primary_product_id = 1 then order_id else null end) as product_one_orders,
    count(distinct case when b.primary_product_id = 2 then order_id else null end) as product_two_orders

from website_sessions a
left join orders b on b.website_session_id = a.website_session_id
where a.created_at between '2012-04-01' AND '2013-04-05'
GROUP BY 1,2;
```

Explanation:

- The SQL query begins with a `SELECT` statement to extract and calculate relevant metrics for analysis.

- It selects the `year` and `month` from the `created_at` timestamp to group the data into monthly intervals.
- The query counts the number of distinct order IDs as `orders`, which represents the monthly order volume.
- It calculates the conversion rate (`cvr`) by dividing the count of distinct order IDs by the count of distinct website sessions, providing an overall conversion rate for each month.
- The revenue per session (`revenue_sessions`) is calculated by dividing the sum of `price_usd` by the count of distinct website sessions. The result is rounded to two decimal places.
- Two additional columns provide a breakdown of sales by product. The count of distinct order IDs is calculated for each product ID, representing `product_one_orders` and `product_two_orders`.
- The data used for analysis is filtered using a `WHERE` clause. It includes records created between April 1, 2012, and April 5, 2013, aligning with Cindy's request for data since April 1, 2013.
- The results are then grouped by the year and month, allowing for the calculation of monthly trends.

Answer Table:

# year	month	orders	cvr	revenue_sessions	product_one_orders	product_two_orders
2012	4	99	0.0265	1.33	99	0
2012	5	108	0.0289	1.45	108	0
2012	6	140	0.0353	1.77	140	0
2012	7	169	0.0398	1.99	169	0
2012	8	228	0.0374	1.87	228	0
2012	9	287	0.0438	2.19	287	0
2012	10	371	0.0453	2.27	371	0

# year	month	orders	cvr	revenue_sessions	product_one_orders	product_two_orders
2012	11	618	0.0441	2.20	618	0
2012	12	506	0.0502	2.51	506	0
2013	1	391	0.0611	3.13	344	47

The answer table presents the trended analysis for monthly order volume, conversion rates, revenue per session, and sales by product, spanning from April 2012 to April 2013.

Interpretation:
The analysis reveals the following insights:

- Conversion rates (`cvr`) have been steadily improving over time, from approximately 2.65% in April 2012 to around 6.11% in January 2013.
- Revenue per session (`revenue_sessions`) has also shown a positive trend, increasing from $1.33 per session in April 2012 to $3.13 per session in January 2013.
- The breakdown of sales by product shows that product one dominated the sales until January 2013, after which product two started contributing significantly.

Cindy's comment expresses her satisfaction with the analysis results, particularly regarding the improvements in conversion rates and revenue per session over time. However, she raises a valid question about whether the growth observed since January is primarily due to the new product's launch or part of the company's ongoing business improvements. To address this question, she plans to connect with Tom to conduct a more detailed analysis.

Product Level Website Analysis:

The concept presented here involves a product-level website analysis aimed at understanding how customers interact with specific products and how well each product converts customers. This analysis can provide insights into the performance of individual products on the company's website and their ability to attract and convert visitors into customers.

Request 20:
Cindy has requested a product-level website analysis for two specific products: "the-original-mr-fuzzy" and "the-forever-love-bear." The analysis focuses on the time period between February 1, 2013, and March 1, 2013. Specifically, she wants to know the following key metrics for each of the two products:

1. The number of website sessions (`sessions`).
2. The number of orders placed for each product (`orders`).
3. The conversion rate (`cvr`) calculated as the ratio of orders to sessions.

SQL Query:

```
select
    a.pageview_url,
    count(DISTINCT a.website_session_id) as sessions,
    count(DISTINCT b.order_id) as orders,
    count(DISTINCT b.order_id)/count(DISTINCT a.website_session_id) as cvr
from website_pageviews a
left join orders b on b.website_session_id = a.website_session_id
where a.created_at BETWEEN '2013-02-01' AND '2013-03-01'
and a.pageview_url in ('/the-original-mr-fuzzy', '/the-forever-love-bear')
GROUP BY 1;
```

Explanation:

- The SQL query selects data from the `website_pageviews` table, focusing on the specified time frame (February 1, 2013, to March 1, 2013).
- The `pageview_url` column is used to filter and group data by the two specific product pages: "/the-original-mr-fuzzy" and "/the-forever-love-bear."
- The query calculates the following metrics for each product:
  - `sessions` : The number of distinct website sessions, indicating how many users visited the product pages.
  - `orders` : The number of distinct orders placed for each product.

- `cvr` (Conversion Rate): Calculated by dividing the number of orders by the number of sessions, providing insight into how well each product converts website visitors into customers.

Answer Table:

# pageview_url	sessions	orders	cvr
/the-forever-love-bear	815	162	0.1988
/the-original-mr-fuzzy	1988	335	0.1685

The answer table displays the results of the analysis for the two specified products: "the-original-mr-fuzzy" and "the-forever-love-bear." It includes columns for `pageview_url`, `sessions`, `orders`, and `cvr` (conversion rate).

Interpretation:
The analysis reveals the following insights for the specified products during the given time period:

- "the-forever-love-bear" had 815 website sessions and 162 orders, resulting in a conversion rate (cvr) of approximately 19.88%.
- "the-original-mr-fuzzy" had 1988 website sessions and 335 orders, resulting in a conversion rate of approximately 16.85%.

This information allows the company to evaluate how effectively each product's webpage converts visitors into customers. It also helps in identifying potential areas for improvement in the customer journey or product presentation.

Request 21 (April 06, 2013):
Morgan has requested an analysis of user paths and conversion funnels on the company's website. Specifically, she is interested in understanding the clickthrough rates from the "/products" page to other product pages, comparing data before and after the launch of a new product on January 6, 2013. The analysis should include a breakdown of clickthrough rates by product.

SQL Query:

```
-- assignment: product pathing analysis
/*
1: find the relevant /products pageviews with website_session_id
2: find the next pageview id that occurs after the product overview
```

```
3: find the pageview_url associated with any applicable next pageview id
4: summarize the data and analyze the pre vs post periods
*/
-- step 1: finding the /products pageviews we care about
with
cte1 as (
    select website_session_id,
    website_pageview_id,
    created_at,
    case when created_at < '2013-01-06' then 'A. Pre_product_2'
         when created_at >= '2013-01-06' then 'B. Post_product_2'
         else 'error 404! not found'
    end as time_period
    from website_pageviews
    where created_at < '2013-04-06'
    and created_at > '2012-10-06' -- start date; of 3 months before product 2 launch
    and pageview_url = '/products'
),
-- 2: find the next pageview id that occurs after the product overview
cte2 as (
    select a.time_period, a.website_session_id, min(b.website_pageview_id) as min_next_pageview_id
    from cte1 a
    left join website_pageviews b
    on b.website_session_id = a.website_session_id
    and b.website_pageview_id > a.website_pageview_id
    GROUP BY 1,2
),
-- 3: find the pageview_url associated with any applicable next pageview id
cte3 as (
    select a.time_period, a.website_session_id,
    b.pageview_url as next_pageview_url
    from cte2 a
    left join website_pageviews b
    on b.website_pageview_id = a.min_next_pageview_id
)
-- 4: summarize the data and analyze the pre vs post periods
select
```

```
    time_period, count(DISTINCT website_session_id) as sessions,
    count(distinct case when next_pageview_url is not null then website_session_id else null end) as w_next_pg,
    count(distinct case when next_pageview_url is not null then website_session_id else null end)/count(DISTINCT website_session_
    count(distinct case when next_pageview_url = '/the-original-mr-fuzzy' then website_session_id else null end) as w_mrfuzzy,
    count(distinct case when next_pageview_url = '/the-original-mr-fuzzy' then website_session_id else null end)/count(DISTINCT w
    count(distinct case when next_pageview_url = '/the-forever-love-bear' then website_session_id else null end) as w_lovebear,
    count(distinct case when next_pageview_url = '/the-forever-love-bear' then website_session_id else null end)/count(DISTINCT w
from cte3
GROUP BY 1;
```

Explanation:
This SQL query performs the following steps:

Step 1: It selects relevant pageviews from the "/products" page within specific time periods:

- It assigns each record to a `time_period` category based on the date (pre or post product 2 launch).

Step 2: It identifies the next pageview that occurs after the "/products" pageview for each session and associates it with the corresponding `time_period`.

Step 3: It determines the pageview URL associated with the identified next pageview for each session and `time_period`.

Step 4: It summarizes the data and calculates various metrics, including the number of sessions, clickthrough rates to the next page, and clickthrough rates for specific product pages. These metrics are analyzed for the pre and post product launch periods.

Answer Table:

# time_period	sessions	w_next_pg	pct_w_next_pg	w_mrfuzzy	pct_w_mrfuzzy	w_lovebear	pct_w_lovebear
A. Pre_product_2	15696	11347	0.7229	11347	0.7229	0	0.0000
B. Post_product_2	10709	8200	0.7657	6654	0.6213	1546	0.1444

The answer table displays the results, showing the `time_period` (pre or post product 2 launch) and metrics such as the number of sessions, clickthrough rates to the next page, and clickthrough rates for specific product pages.

Interpretation:
The analysis provides a comparison of user paths and clickthrough rates before and after the launch of the new product. Some key findings include:

- In the "Pre_product_2" period, 72.29% of sessions that viewed the "/products" page proceeded to another page.
- In the "Post_product_2" period, the clickthrough rate increased to 76.57%.
- The clickthrough rate to the "/the-original-mr-fuzzy" product page decreased slightly in the "Post_product_2" period, while the clickthrough rate to the "/the-forever-love-bear" product page increased significantly.

Morgan acknowledges the analysis and notes that while the percentage of sessions clicking through to "Mr. Fuzzy" has decreased, the overall clickthrough rate has improved, suggesting increased interest in the product lineup. Further analysis is suggested to examine the conversion funnels for each product individually.

Request 22 (April 10, 2013):
Morgan is interested in analyzing the conversion funnels for the company's two products since January 6th. He wants to compare the conversion funnels for both products, focusing on all website traffic. The analysis should provide insights into the clickthrough rates at various stages of the conversion funnel for each product.

SQL Query:

```
-- step 1: select all pageviews for relevant sessions
-- step 2: figure out which pageview urls to look for
-- step 3: pull all pageviews and identify the funnel steps
-- step 4: create the session-level conversion funnel view
-- step 5: aggregate the data to assess funnel performance

-- step 1: select all pageviews for relevant sessions
create temporary table funnel
with
cte1 as (
    select website_session_id,
    website_pageview_id, pageview_url as product_page_seen
    from website_pageviews
    where created_at < '2013-04-10' -- date of assignment
```

```
    and created_at > '2013-01-06' -- product 2 launch
    and pageview_url in ('/the-original-mr-fuzzy', '/the-forever-love-bear')
),
-- step 2: figure out which pageview urls to look for
cte2 as (
    select distinct pageview_url
    from cte1 a
    left join website_pageviews b
    on b.website_session_id = a.website_session_id
    and b.website_pageview_id > a.website_pageview_id
),
-- step 3: pull all pageviews and identify the funnel steps
cte3 as (
    select a.website_session_id,
    a.product_page_seen,
    case when b.pageview_url = '/cart' then 1 else 0 end as cart_page,
    case when b.pageview_url = '/shipping' then 1 else 0 end as shipping_page,
    case when b.pageview_url = '/billing-2' then 1 else 0 end as billing_page,
    case when b.pageview_url = '/thank-you-for-your-order' then 1 else 0 end as thankyou_page
    from cte1 a
    left join website_pageviews b
    on b.website_session_id = a.website_session_id
    and b.website_pageview_id > a.website_pageview_id
    order by 1, b.created_at
)
-- step 4: create the session-level conversion funnel view
select website_session_id,
case when product_page_seen = '/the-original-mr-fuzzy' then 'mrfuzzy'
when product_page_seen = '/the-forever-love-bear' then 'lovebear'
else 'error 404!'
end as product_seen,
max(cart_page) as cart_made_it,
max(shipping_page) as shipping_made_it,
max(billing_page) as billing_made_it,
max(thankyou_page) as thankyou_made_it
from cte3
GROUP BY 1,2;
```

```sql
-- step 5: aggregate the data to assess funnel performance
-- overall sessions per products
select product_seen,
count(DISTINCT case when cart_made_it = 1 then website_session_id else null end) as to_cart,
count(DISTINCT case when shipping_made_it = 1 then website_session_id else null end) as to_shipping,
count(DISTINCT case when billing_made_it = 1 then website_session_id else null end) as to_billing,
count(DISTINCT case when thankyou_made_it = 1 then website_session_id else null end) as to_thankyou
from funnel
GROUP BY 1;
```

Explanation:
This SQL query is divided into several steps:

Step 1: It creates a temporary table named "funnel" and selects relevant pageviews for sessions within a specific time frame. The focus is on pageviews of two products (/the-original-mr-fuzzy and /the-forever-love-bear) after the launch of the second product on January 6th, 2013.

Step 2: It identifies the distinct pageview URLs to look for within the selected data.

Step 3: It pulls all pageviews and assigns funnel steps for each session. The funnel steps are identified based on specific pageview URLs (e.g., /cart, /shipping, /billing-2, /thank-you-for-your-order).

Step 4: It creates a session-level conversion funnel view, associating sessions with the products seen and recording whether each funnel step was completed.

Step 5: It aggregates the data to assess funnel performance. The query calculates various metrics related to conversion funnels for both products, including the number of sessions that reached different funnel steps.

Answer Tables:

# product_seen	to_cart	to_shipping	to_billing	to_thankyou
lovebear	877	603	488	301

# product_seen	to_cart	to_shipping	to_billing	to_thankyou
mrfuzzy	3038	2084	1710	1088

# product_seen	sessions	product_page_rt	cart_page_rt	shipping_page_rt	billing_page_rt
lovebear	1599	0.5485	0.6876	0.8093	0.6168
mrfuzzy	6985	0.4349	0.6860	0.8205	0.6363

The query provides two answer tables:

1. The first table shows the number of sessions that progressed through different funnel steps for both products, such as moving to the cart,

shipping, billing, and the thank-you page.

2. The second table displays metrics, including product page clickthrough rates and clickthrough rates between funnel steps, for each product.

Interpretation:
The analysis indicates the performance of conversion funnels for the two products, "Mr. Fuzzy" and "Love Bear," since the launch of the second product on January 6th, 2013.

- The "Love Bear" product had 877 sessions that progressed to the cart, while "Mr. Fuzzy" had 3,038 sessions.
- The clickthrough rate (CTR) from the product page to the cart page is higher for "Love Bear" (54.85%) compared to "Mr. Fuzzy" (43.49%).
- The CTR for subsequent funnel steps, including shipping, billing, and the thank-you page, is also provided for both products.

Morgan acknowledges the analysis and notes that the addition of the second product has positively impacted the overall clickthrough rate, especially in the context of the cart page. He raises the question of whether the company should consider adding a third product, suggesting that the second product has been a valuable addition to the business.

Cross-Selling Products:

Cross-selling analysis involves understanding which products are frequently purchased together, enabling businesses to offer intelligent product recommendations. By identifying product pairs that have a high correlation in customer orders, companies can enhance their marketing and sales strategies to boost revenue.

Request: In this analysis, the goal is to identify products that are often purchased together. The SQL query retrieves data related to cross-selling products and calculates the number of orders for each product pair. It looks for orders falling within a specific range (order_id between 10000 and 11000). The query also computes the percentage of each product's contribution to the total orders for its primary product.

SQL Query:

```
select
a.primary_product_id,
b.product_id as cross_sell_product,
count(distinct a.order_id) as orders
from orders a
left join order_items b
on b.order_id = a.order_id
and b.is_primary_item = 0
where a.order_id between 10000 and 11000
group by 1,2;
```

Explanation:
The SQL query consists of a `SELECT` statement that retrieves data from the `orders` and `order_items` tables. It joins these tables using the `order_id` and filters the data to include only orders within the specified order_id range (between 10000 and 11000). The query groups the results by the primary product's ID and the cross-sell product's ID.

Answer Table:

# primary_product_id	cross_sell_product	orders
1	null	624
1	2	39
1	3	68
2	null	134
2	1	5
2	3	5
3	null	113
3	1	10
3	2	3

The answer table shows the primary product's ID, the cross-sell product's ID, and the number of orders where these products were purchased together.

Interpretation:
The interpretation of the answer table reveals the number of orders in which specific products were cross-sold together. For example, product 1 was purchased together with product 2 in 39 orders and with product 3 in 68 orders. Similar cross-selling patterns are observed for products 2 and 3.

A subsequent SQL query builds on this analysis and calculates the percentage contribution of each cross-sell product to the total orders for its primary product. The answer table presents these percentages.

SQL Query (Part 2):

```
select
a.primary_product_id,
```

```
count(distinct a.order_id) as orders,
count(DISTINCT case when b.product_id = 1 then a.order_id else null end) as x_sell_prod1,
count(DISTINCT case when b.product_id = 2 then a.order_id else null end) as x_sell_prod2,
count(DISTINCT case when b.product_id = 3 then a.order_id else null end) as x_sell_prod3,
round(count(DISTINCT case when b.product_id = 1 then a.order_id else null end)/count(distinct a.order_id)*100,2) as x_sell_prod1_
round(count(DISTINCT case when b.product_id = 2 then a.order_id else null end)/count(distinct a.order_id)*100,2) as x_sell_prod2_
round(count(DISTINCT case when b.product_id = 3 then a.order_id else null end)/count(distinct a.order_id)*100,2) as x_sell_prod3_
from orders a
left join order_items b
on b.order_id = a.order_id
and b.is_primary_item = 0
where a.order_id between 10000 and 11000
group by 1;
```

Explanation (Part 2):

This SQL query extends the analysis by calculating the cross-sell percentages. It calculates the number of orders in which each cross-sell product was purchased with its primary product and the percentage that each cross-sell product contributes to the total orders for its primary product.

Answer Table (Part 2):

# primary_product_id	orders	x_sell_prod1	x_sell_prod2	x_sell_prod3	x_sell_prod1_rt	x_sell_prod2_rt	x_sell_prod3_rt
1	731	0	39	68	0.00	5.34	9.30
2	144	5	0	5	3.47	0.00	3.47
3	126	10	3	0	7.94	2.38	0.00

The answer table includes the primary product's ID, the total number of orders for the primary product, and the counts and percentages of orders in which each cross-sell product was purchased with the primary product.

Interpretation (Part 2):
The second answer table provides insights into the cross-selling patterns for each primary product. It shows the number of orders and the cross-sell rates for each cross-sell product associated with the primary product.

In this specific example, product 1 has a cross-sell rate of 5.34% for product 2 and 9.30% for product 3, indicating that these products are frequently purchased together with product 1. Similar cross-sell rates are calculated for products 2 and 3.

The analysis aids in understanding which products are often bought together, enabling businesses to make informed decisions about product bundling and recommendations to boost sales.

Request 23 (November 22, 2013): Cindy requested an analysis of the impact of a recent change introduced on September 25th. The change allowed customers to add a second product while on the /cart page. Cindy wanted to compare the month before and the month after the change. The specific metrics of interest were Click-Through Rate (CTR) from the /cart page, Average Products per Order, Average Order Value (AOV), and overall revenue per /cart page view.

SQL Query:

```
-- assignment cross-sell analysis
/*
1: Identify the relevant /cart page views and their sessions
2: See which of those /cart sessions clicked through to the shipping page
3: Find the orders associated with the /cart sessions. Analyze products purchased, AOV
4: Aggregate and analyze a summary of our findings
*/

-- Identify the relevant /cart page views and their sessions
with
cte1 as (select case when created_at < '2013-09-25' then 'pre_cross_sell'
        when created_at >= '2013-09-25' then 'post_cross_sell'
        else 'others' end as time_period,
website_session_id as cart_session_id,
website_pageview_id as cart_pageview_id
from website_pageviews
where created_at between '2013-08-25' AND '2013-10-25'
    and pageview_url = '/cart'),
```

```sql
-- See which of those /cart sessions clicked through to the shipping page
cte2 as (select time_period, cart_session_id, min(b.website_pageview_id) as pv_id_after_cart
    from cte1 a
    left join website_pageviews b
    on b.website_session_id = a.cart_session_id
    and b.website_pageview_id > a.cart_pageview_id
group by 1,2
having min(b.website_pageview_id) is not null),

-- Find the orders associated with the /cart sessions. Analyze products purchased, AOV
cte3 as (select time_period, cart_session_id, order_id, items_purchased, price_usd
from cte1 a
inner join orders b
    on a.cart_session_id = b.website_session_id),

cte4 as (select a.time_period, a.cart_session_id,
case when b.cart_session_id is null then 0 else 1 end as clicked_to_another_page,
case when c.order_id is null then 0 else 1 end as placed_order,
c.items_purchased,
c.price_usd
from cte1 a
    left join cte2 b
    on a.cart_session_id = b.cart_session_id
    left join cte3 c
    on a.cart_session_id = c.cart_session_id
ORDER BY 2),

-- Aggregate and analyze a summary of our findings
select time_period,
count(DISTINCT cart_session_id) as cart_sessions,
sum(clicked_to_another_page) as clickthroughs,
sum(clicked_to_another_page)/count(DISTINCT cart_session_id) as cart_ctr,
sum(items_purchased)/sum(placed_order) as products_per_order,
sum(price_usd)/sum(placed_order) as aov,
sum(price_usd)/count(DISTINCT cart_session_id) as rev_per_cart_session
```

```
from cte4
group by 1;
```

Explanation: This SQL query follows a multi-step process to analyze the effect of the cross-sell feature introduced in September. The steps include identifying relevant /cart page views, determining which of those sessions clicked through to another page (shipping page), finding the associated orders, and then aggregating and summarizing the findings.

Answer Table:

# time_period	cart_sessions	clickthroughs	cart_ctr	products_per_order	aov	rev_per_cart_session
post_cross_sell	1975	1351	0.6841	1.0447	54.251848	18.431894
pre_cross_sell	1830	1229	0.6716	1.0000	51.416380	18.318842

The answer table displays data divided into two time periods: 'pre_cross_sell' and 'post_cross_sell'. It includes metrics such as the number of /cart sessions, clickthroughs, CTR from the /cart page, products per order, AOV, and revenue per /cart session for both time periods.

Interpretation: The analysis compares the month before and the month after the cross-sell feature was implemented. The CTR from the /cart page didn't decrease, which was a concern. Instead, it remained consistent, and there were slight improvements in the number of products per order, AOV, and revenue per /cart session after the feature was introduced. While not a game changer, these positive trends suggest that the cross-sell feature has had a favorable impact on user behavior and revenue.

Request 24 (January 12, 2014): Cindy requested a pre-post analysis to compare the month before and the month after the launch of a third product, "Birthday Bear," which was introduced on December 12, 2013. The analysis should focus on several key metrics, including session-to-order conversion rate (CVR), average order value (AOV), products per order, and revenue per session.

SQL Query:

```
select
case when a.created_at < '2013-12-12' then 'Pre_Birthday_Bear'
when a.created_at >= '2013-12-12' then 'Post_Birthday_Bear'
else 'others' end as time_period,
```

```
count(DISTINCT b.order_id)/count(DISTINCT a.website_session_id) as cvr,
sum(b.price_usd)/count(DISTINCT b.order_id) as average_order_value,
sum(b.items_purchased)/count(DISTINCT b.order_id) as products_per_order,
sum(b.price_usd)/count(DISTINCT a.website_session_id) as revenue_per_session
from website_sessions a
left join orders b on b.website_session_id = a.website_session_id
where a.created_at between '2013-11-12' and '2014-01-12'
GROUP BY 1;
```

Explanation: This SQL query conducts a pre-post analysis of key performance metrics after launching the "Birthday Bear" product. It categorizes data into two time periods: "Pre_Birthday_Bear" and "Post_Birthday_Bear," which correspond to the month before and after the product launch. The metrics analyzed include CVR, AOV, products per order, and revenue per session.

Answer Table:

# time_period	cvr	average_order_value	products_per_order	revenue_per_session
Post_Birthday_Bear	0.0702	56.931319	1.1234	3.998763
Pre_Birthday_Bear	0.0608	54.226502	1.0464	3.298677

The answer table shows the results of the pre-post analysis, presenting metrics for the "Pre_Birthday_Bear" and "Post_Birthday_Bear" time periods.

Interpretation: The analysis reveals that all critical metrics have improved since the launch of the third product, "Birthday Bear." The post-launch period ("Post_Birthday_Bear") has a higher CVR, AOV, products per order, and revenue per session. These improvements indicate the positive impact of the new product on the website's performance and revenue.

Comment by the Requester: Cindy expressed her excitement about the positive results of the analysis. She noted that all critical metrics have improved since the introduction of the third product. She mentioned plans to meet with Tom to discuss increasing ad spend, considering the higher revenue per session, and the possibility of adding a fourth product.

Product Refund Analysis:

Product refund analysis involves examining the refund rates for specific products to control for quality and identify potential issues that may need addressing. This analysis helps in understanding which products have a higher likelihood of being refunded and the associated refund amounts.

Request: The SQL query retrieves data related to product refunds. It selects information about orders, order items, and refunds for specific order IDs (3489, 27061, and 32049). The data includes details such as order item IDs, prices, creation dates, order item refund IDs, refund amounts, and refund creation dates.

SQL Query:

```
select a.order_id, a.order_item_id, a.price_usd, a.created_at,
b.order_item_refund_id, b.refund_amount_usd, b.created_at
from order_items a
left join order_item_refunds b on b.order_item_id = a.order_item_id
where a.order_id in (3489, 32049, 27061);
```

Explanation: The SQL query utilizes a `LEFT JOIN` between the `order_items` and `order_item_refunds` tables. It links these tables based on the `order_item_id` to associate order items with their corresponding refunds. The query selects data from order items with order IDs 3489, 27061, and 32049. The resulting dataset provides detailed information about each order item, its price, creation date, associated refunds, refund amounts, and refund creation dates.

Answer Table:

# order_id	order_item_id	price_usd	created_at	order_item_refund_id	refund_amount_usd	created_at
3489	3489	49.99	2013-03-03 09:51:10	null	null	null
27061	33000	49.99	2015-01-03 16:47:12	1505	49.99	2015-01-12 11:47:12

# order_id	order_item_id	price_usd	created_at	order_item_refund_id	refund_amount_usd	created_at
27061	33001	45.99	2015-01-03 16:47:12	1526	45.99	2015-01-19 13:47:12
32049	39671	49.99	2015-03-15 15:33:51	1728	49.99	2015-03-30 21:33:51
32049	39672	45.99	2015-03-15 15:33:51	null	null	null

The answer table displays the extracted data, showing order IDs, order item IDs, prices, creation dates, order item refund IDs, refund amounts, and refund creation dates.

Interpretation: The table illustrates the relationships between order items and refunds for specific orders. It shows which order items have corresponding refunds and provides information about refund amounts and creation dates. For example, for order 27061, two order items (33000 and 33001) were refunded with corresponding refund amounts and dates. Order 32049 also had refunded items, while order 3489 did not have any associated refunds.

This data can be valuable for assessing product refund rates and identifying patterns or issues related to specific products or orders. It allows businesses to monitor product quality and take appropriate actions to minimize refund rates.

Request 25 (October 15, 2014):
Cindy requested a monthly product refund rate analysis to determine whether the quality issues with the "Mr. Fuzzy" product have been resolved. She specifically wanted to assess the product refund rates for different months, focusing on each product category.

SQL Query:

```
select year(a.created_at) as year,
month(a.created_at) as month,
count(DISTINCT case when product_id = 1 then a.order_item_id else null end) as p1_orders,
count(DISTINCT case when product_id = 1 then b.order_item_id else null end)/count(DISTINCT case when product_id = 1 then a.order_
```

```
count(DISTINCT case when product_id = 2 then a.order_item_id else null end) as p2_orders,
count(DISTINCT case when product_id = 2 then b.order_item_id else null end)/count(DISTINCT case when product_id = 2 then a.order_
count(DISTINCT case when product_id = 3 then a.order_item_id else null end) as p3_orders,
count(DISTINCT case when product_id = 3 then b.order_item_id else null end)/count(DISTINCT case when product_id = 3 then a.order_
count(DISTINCT case when product_id = 4 then a.order_item_id else null end) as p4_orders,
count(DISTINCT case when product_id = 4 then b.order_item_id else null end)/count(DISTINCT case when product_id = 4 then a.order_
from order_items a
left join order_item_refunds b on a.order_item_id = b.order_item_id
where a.created_at < '2014-10-15'
GROUP BY 1,2;
```

Explanation: This SQL query retrieves monthly product refund rates for different products (identified by product_id) up to October 15, 2014. It calculates the number of orders and refund rates for each product category for each month.

The query extracts the year and month from the created_at column to group the data by months. For each product category (p1, p2, p3, and p4), it calculates the total number of orders, the number of refunded orders, and the refund rate (the ratio of refunded orders to total orders).

Answer Table:

# year	month	p1_orders	p1_refund_rt	p2_orders	p2_refund_rt	p3_orders	p3_refund_rt	p4_orders	p4_refund_r
2012	3	60	0.0167	0	null	0	null	0	null
2012	4	99	0.0505	0	null	0	null	0	null
2012	5	108	0.0370	0	null	0	null	0	null
2012	6	140	0.0571	0	null	0	null	0	null
2012	7	169	0.0828	0	null	0	null	0	null
2012	8	228	0.0746	0	null	0	null	0	null

# year	month	p1_orders	p1_refund_rt	p2_orders	p2_refund_rt	p3_orders	p3_refund_rt	p4_orders	p4_refund_r
2012	9	287	0.0906	0	null	0	null	0	null
2012	10	371	0.0728	0	null	0	null	0	null
2012	11	618	0.0744	0	null	0	null	0	null
2012	12	506	0.0593	0	null	0	null	0	null
2013	1	343	0.0496	47	0.0213	0	null	0	null
2013	2	336	0.0714	162	0.0123	0	null	0	null
2013	3	320	0.0563	65	0.0462	0	null	0	null
2013	4	459	0.0414	94	0.0106	0	null	0	null
2013	5	489	0.0634	82	0.0244	0	null	0	null
2013	6	503	0.0775	90	0.0556	0	null	0	null
2013	7	509	0.0727	95	0.0316	0	null	0	null
2013	8	510	0.0549	98	0.0102	0	null	0	null
2013	9	537	0.0428	98	0.0102	0	null	0	null
2013	10	603	0.0282	135	0.0148	0	null	0	null
2013	11	724	0.0345	174	0.0230	0	null	0	null
2013	12	818	0.0232	183	0.0219	139	0.0719	0	null
2014	1	728	0.0426	183	0.0219	200	0.0650	0	0.0099
2014	2	584	0.0394	351	0.0171	211	0.0664	202	0.0099

# year	month	p1_orders	p1_refund_rt	p2_orders	p2_refund_rt	p3_orders	p3_refund_rt	p4_orders	p4_refund_r
2014	3	785	0.0306	193	0.0155	244	0.0697	205	0.0049
2014	4	917	0.0349	214	0.0187	267	0.0674	259	0.0154
2014	5	1030	0.0291	246	0.0163	299	0.0569	298	0.0067
2014	6	893	0.0571	245	0.0367	288	0.0556	249	0.0241
2014	7	961	0.0437	244	0.0369	276	0.0399	264	0.0152
2014	8	958	0.1378	237	0.0169	294	0.0680	303	0.0066
2014	9	1056	0.1326	251	0.0319	317	0.0662	327	0.0122
2014	10	513	0.0273	135	0.0074	165	0.0485	155	0.0323

SHOWING ONLY A FRACTION OF THE OUTPUT! The answer table presents a summary of the monthly product refund rates for different product categories.

Interpretation: The table provides a historical overview of product refund rates for each product category (p1, p2, p3, and p4) from 2012 to 2014. The refund rates are calculated for each month, with a focus on the period before October 15, 2014.

This data can help Cindy assess the impact of the quality issues and determine whether the replacement of the supplier in September 2014 has resulted in lower refund rates. By comparing refund rates before and after addressing the quality issues, she can confirm if the problems with the "Mr. Fuzzy" product have been resolved.

Comment by the Requester (Cindy): The requester is seeking confirmation that the quality issues have been fixed by analyzing the product refund rates. The comment is not provided in the request, so we can't offer a direct interpretation. However, Cindy likely used the analysis results to evaluate whether the quality issues have improved and made informed decisions based on the findings.

Analysing Repeat Behavior:

This analysis focuses on understanding repeat customer behavior, specifically tracking the time it takes for customers to request refunds after making a purchase. By calculating the time gap between the order and the refund request, businesses can gain insights into customer satisfaction and identify valuable customers.

Request: The SQL query aims to analyze repeat customer behavior by examining specific order items and their associated refunds. It retrieves details of order items, including their order IDs, item IDs, prices, creation timestamps, and any corresponding refunds. The key metric of interest is the number of days it takes for customers to request a refund after making a purchase. The query focuses on order IDs '3489,' '32049,' and '27061.'

SQL Query:

```
select a.order_id, a.order_item_id, a.price_usd, a.created_at,
b.order_item_refund_id, b.refund_amount_usd, b.created_at,
datediff(b.created_at, a.created_at) as days_order_to_refund
from order_items a
left join order_item_refunds b
on b.order_item_id = a.order_item_id
where a.order_id in ('3489', '32049', '27061');
```

Explanation: The SQL query retrieves information about order items and associated refunds for the specified order IDs ('3489,' '32049,' '27061'). It uses a left join to combine data from the 'order_items' table (a) and the 'order_item_refunds' table (b) based on the order item ID.

The columns selected include order ID, order item ID, item price, item creation timestamp, refund information (refund ID, refund amount, and refund creation timestamp), and the calculated number of days between the order and the refund request. If there's no associated refund for an order item, the corresponding columns show 'null.'

Answer Table:

# order_id	order_item_id	price_usd	created_at	order_item_refund_id	refund_amount_usd	created_at	days_order_to_
3489	3489	49.99	2013-03-03 09:51:10	null	null	null	null
27061	33000	49.99	2015-01-03 16:47:12	1505	49.99	2015-01-12 11:47:12	9
27061	33001	45.99	2015-01-03 16:47:12	1526	45.99	2015-01-19 13:47:12	16
32049	39671	49.99	2015-03-15 15:33:51	1728	49.99	2015-03-30 21:33:51	15
32049	39672	45.99	2015-03-15 15:33:51	null	null	null	null

The answer table presents the extracted data, showing order item details, refund details (if applicable), and the time gap in days between the order and refund request.

Interpretation: The answer table provides information on specific order items (identified by order IDs '3489,' '32049,' '27061') and any associated refunds. It includes details such as order item price, creation timestamps, and refund amounts. The 'days_order_to_refund' column represents the number of days it took for a customer to request a refund after the order.

For example, in the case of order '27061,' two order items had associated refunds. One item was refunded 9 days after the order, and the other item was refunded 16 days after the order.

This analysis can help businesses understand customer behavior related to refunds and potentially identify patterns or trends related to repeat customer behavior or product quality issues.

Request 26 (November 01, 2014): Tom is interested in understanding the value of customers who have repeat sessions on the website. The company has been primarily evaluating customer value based on their first session conversion and revenue. However, if customers return for multiple sessions, they may be more valuable than initially thought. To assess this, Tom requests data on how many website visitors return for another session during the year 2014 to date.

SQL Query: The SQL query is designed to identify and count visitors who return for multiple sessions on the website in 2014. It does this in several steps:

1. **cte1:** It initially selects user IDs and website session IDs for visitors in 2014 who have not had a repeat session.
2. **cte2:** This step identifies users who had at least one repeat session in 2014, distinguishing their new and repeat session IDs.
3. **cte3:** Here, it aggregates the data, counting the number of new sessions and repeat sessions for each user.

The final part of the query groups and counts users by the number of repeat sessions they have.

SQL Query:

```
with
cte1 as (select user_id, website_session_id
from website_sessions
where created_at between '2014-01-01' and '2014-11-01'
and is_repeat_session = 0),
cte2 as (select a.user_id, a.website_session_id as new_session_id,
b.website_session_id as repeat_session_id
from cte1 a
left join website_sessions b
on b.user_id = a.user_id
and b.is_repeat_session = 1
and b.website_session_id > a.website_session_id
and b.created_at BETWEEN '2014-01-01' and '2014-11-01'),
cte3 as(select user_id, count(DISTINCT new_session_id) as new_sessions,
```

```
count(DISTINCT repeat_session_id) as repeat_sessions
from cte2
GROUP BY 1
ORDER BY 3 desc)
select repeat_sessions,
count(DISTINCT user_id) as users
from cte3
GROUP BY 1;
```

Explanation: This SQL query uses common table expressions (CTEs) to first identify users with repeat sessions in 2014 and then aggregates the data to provide insights into customer behavior. The key components are:

- **cte1**: This CTE selects users who visited the website in 2014 without having a repeat session (is_repeat_session = 0).
- **cte2**: It links the first sessions (cte1) with any repeat sessions that occurred after them. This CTE also filters for sessions that fall within the specified date range.
- **cte3**: Here, the query aggregates the data, counting new sessions and repeat sessions for each user. It also calculates the number of repeat sessions and sorts the results by the number of repeat sessions in descending order.

The final part of the query counts the number of users based on the count of their repeat sessions and presents the results in a tabular format.

Answer Table:

# repeat_sessions	users
0	126813
1	14086
2	315
3	4686

The answer table displays the number of users who had 0, 1, 2, or 3 repeat sessions in 2014.

Interpretation: The table provides valuable insights into user behavior on the website during 2014. It shows that a significant number of users (approximately 126,813) had no repeat sessions. However, a noteworthy portion of users did return for multiple sessions, with 14,086 having one repeat session, 315 with two repeat sessions, and 4,686 with three repeat sessions.

Tom acknowledges the breakdown and expresses an interest in learning more about this pattern, suggesting future steps to explore the value of customers with repeat sessions further.

Request 27 (November 03, 2014): Tom is curious to understand the behavior of repeat customers who return to the website. Specifically, he wants to know the minimum, maximum, and average time between a customer's first session and their second session for those who do come back. The analysis should cover the period from January 1, 2014, to November 3, 2014.

SQL Query: The SQL query is designed to calculate the minimum, maximum, and average time between the first and second sessions for customers who return to the website. The query follows these steps:

1. **cte1:** It identifies users and their first sessions, filtering for visitors in 2014 who have not had a repeat session.
2. **cte2:** This step links the first sessions (cte1) with any repeat sessions that occurred after them. It also captures the timestamps for both sessions, focusing on sessions in 2014.
3. **cte3:** It identifies users, their new sessions, and related details. This step filters out cases where no repeat session exists.
4. The main query calculates the difference in days between the first and second sessions for each user, based on the data from cte3.
5. Finally, the query aggregates the user-level data to find the average, minimum, and maximum differences in days between first and second sessions.

SQL Query:

```
create temporary table first_to_second_session
with
cte1 as (select user_id, website_session_id, created_at
from website_sessions
```

```
where created_at between '2014-01-01' and '2014-11-03'
and is_repeat_session = 0),
cte2 as (select a.user_id, a.website_session_id as new_session_id, a.created_at as new_session_created_at,
b.website_session_id as repeat_session_id, b.created_at as repeat_session_created_at
from cte1 a
left join website_sessions b
on b.user_id = a.user_id
and b.is_repeat_session = 1
and b.website_session_id > a.website_session_id
and b.created_at BETWEEN '2014-01-01' and '2014-11-03'),
cte3 as (select user_id, new_session_id, new_session_created_at,
min(repeat_session_id) as second_session_id,
min(repeat_session_created_at) as second_session_created_at
from cte2
where repeat_session_id is not null
GROUP BY 1,2,3)
select user_id,
datediff(second_session_created_at, new_session_created_at) as days_first_to_second_session
from cte3;

select avg(days_first_to_second_session) as avg_days_first_to_second,
min(days_first_to_second_session) as min_days_first_to_second,
max(days_first_to_second_session) as max_days_first_to_second
from first_to_second_session;
```

Explanation: The SQL query is divided into several parts, as previously explained. It calculates the time difference in days between the first and second sessions for customers who return. The key components include identifying these sessions, linking first and second sessions, and calculating the differences.

Answer Table:

# avg_days_first_to_second	min_days_first_to_second	max_days_first_to_second
33.2622	1	69

The answer table provides statistics related to the time between the first and second sessions for returning customers. It includes the average, minimum, and maximum differences in days.

Interpretation: The analysis reveals that, on average, customers who return to the website for a second session do so approximately a month later, with an average of 33.26 days between their first and second sessions. The fastest return was within just 1 day, while the longest period before returning was 69 days.

Tom finds this data interesting and suggests investigating the channels that these returning visitors are using, indicating potential plans for further analysis.

Request 28 (November 05, 2014): Tom wants to explore the behavior of repeat customers and understand the channels through which they return to the website. He's particularly interested in whether these customers are mainly coming back through direct type-ins or if paid search ads play a role. Tom suggests comparing new and repeat sessions by channel, with a focus on data from 2014 to the current date.

SQL Query: The SQL query is designed to categorize sessions into different channel groups and count the number of new and repeat sessions for each channel. It classifies sessions into the following channel groups:

- **Organic Search:** Sessions with no UTM source and coming from specific search referers.
- **Paid Nonbrand:** Sessions with a specific UTM campaign.
- **Paid Brand:** Sessions with a different UTM campaign.
- **Direct Type-In:** Sessions with no UTM source and no HTTP referer.
- **Paid Social:** Sessions with a specific UTM source.

The query counts new sessions (is_repeat_session = 0) and repeat sessions (is_repeat_session = 1) within each channel group.

SQL Query:

```
SELECT case
    when utm_source is null and http_referer in ('https://www.gsearch.com', 'https://www.bsearch.com') then 'organic_search'
    when utm_campaign = 'nonbrand' then 'paid_nonbrand'
    when utm_campaign = 'brand' then 'paid_brand'
    when utm_source is null and http_referer is null then 'direct_type_in'
```

```
    when utm_source = 'socialbook' then 'paid_social'
    end as channel_group,
    count(case when is_repeat_session = 0 then website_session_id else null end) as new_sessions,
    count(case when is_repeat_session = 1 then website_session_id else null end) as repeat_sessions
from website_sessions
where created_at BETWEEN '2014-01-01' and '2014-11-05'
GROUP BY 1
order by 3 desc;
```

Explanation: The SQL query categorizes sessions into different channel groups based on various criteria such as UTM parameters, HTTP referer, and UTM source. It then counts new and repeat sessions for each channel group, focusing on the specified date range of 2014 to the current date.

Answer Table:

# channel_group	new_sessions	repeat_sessions
organic_search	7139	11507
paid_brand	6432	11027
direct_type_in	6591	10564
paid_nonbrand	119950	0
paid_social	7652	0

The answer table provides a breakdown of the channel groups, showing the number of new and repeat sessions for each group.

Interpretation: The analysis reveals how customers return to the website through different channels. The major findings are as follows:

- **Organic Search:** Both new and repeat sessions are significant, with repeat sessions outnumbering new ones. This suggests that organic search attracts repeat visitors.
- **Paid Brand:** Similar to organic search, paid brand campaigns also attract a substantial number of repeat sessions.

- **Direct Type-In:** Direct type-in sessions, where users enter the website URL directly, account for a significant portion of both new and repeat sessions.
- **Paid Nonbrand and Paid Social:** Interestingly, the data shows that no repeat sessions were attributed to paid nonbrand and paid social channels during the specified period.

Tom notes that most repeat visitors are returning through organic search, direct type-ins, and paid brand channels. Only about one-third of repeat visitors come through paid channels, and since brand clicks are cheaper than nonbrand clicks, the cost for subsequent visits is relatively low. He expresses interest in whether these repeat sessions convert into orders, hinting at potential further investigation.

Comment by the Requester: Tom's comment reflects his interest in the analysis results. He finds it valuable to discover that many repeat customers return through channels that do not involve significant ad spend. He's now considering whether these repeat sessions lead to order conversions, indicating his intent to explore this aspect further.

Request 29 (November 08, 2014): Morgan is interested in conducting a comparison of conversion rates and revenue per session for repeat sessions and new sessions. The analysis will focus on data from 2014 year-to-date.

SQL Query: The SQL query is designed to compare conversion rates (CVR) and revenue per session for repeat sessions (is_repeat_session = 1) and new sessions (is_repeat_session = 0). It counts the number of sessions, calculates the CVR, and computes the revenue per session for both groups.

SQL Query:

```
select a.is_repeat_session,
    count(DISTINCT a.website_session_id) as sessions,
    count(distinct b.order_id)/count(DISTINCT a.website_session_id) as cvr,
    sum(b.price_usd)/count(DISTINCT a.website_session_id) as rev_per_session
from website_sessions a
left join orders b
on a.website_session_id = b.website_session_id
where a.created_at between '2014-01-01' and '2014-11-08'
GROUP BY 1;
```

Explanation: The SQL query categorizes sessions into repeat sessions and new sessions and calculates the count of sessions, CVR, and revenue per session for both groups. The analysis is limited to data from January 1, 2014, to November 8, 2014.

Answer Table:

# is_repeat_session	sessions	cvr	rev_per_session
0	149787	0.0680	4.343754
1	33577	0.0811	5.168828

The answer table presents a comparison between repeat sessions and new sessions, showing the number of sessions, CVR, and revenue per session for each group.

Interpretation: The analysis reveals the following insights:

- **New Sessions:** There are 149,787 new sessions with a CVR of 6.80% and revenue per session of approximately $4.34.
- **Repeat Sessions:** There are 33,577 repeat sessions with a higher CVR of 8.11% and a greater revenue per session of about $5.17.

Morgan finds this comparison interesting. Repeat sessions have a higher CVR and generate more revenue per session compared to new sessions. This discovery suggests that considering repeat sessions in paid traffic bidding strategies could be beneficial.

Comment by the Requester: Morgan expresses her excitement about the findings. She acknowledges that repeat sessions are more likely to convert and produce higher revenue per session. She plans to discuss these insights with Tom and emphasizes that, given the low cost of repeat sessions, they should be factored into their paid traffic bidding decisions.

The Ending:

Overall, these are the analysis we did -

1. Traffic Source Analysis:

- Explored the sources of website traffic, helping to understand user acquisition channels.

- Accommodated ad hoc requests to investigate specific data points, providing on-demand insights.

2. Bid Optimization:

- Improved the efficiency of marketing spend by optimizing bids based on conversion rates and cost per click.

3. Analyzing Top Website Content:

- Identified high-performing website content to tailor content strategy and enhance user engagement.

4. Landing Page Performance & Testing:

- Evaluated landing page performance and conducted A/B tests to improve conversion rates.

5. Analyzing & Testing Conversion Funnels:

- Analyzed the conversion funnel to identify drop-off points and tested changes to enhance conversions.

6. Channel Portfolio Optimization:

- Optimized the marketing channel mix based on channel performance and cost-effectiveness.

7. Analyzing Direct Traffic:

- Explored direct traffic sources to gain insights into user behavior and direct entry patterns.

8. Analyzing Seasonality & Business Patterns:

- Investigated seasonal trends and business patterns to prepare for demand fluctuations and align strategies accordingly.

9. Product Sales Analysis:

- Analyzed product sales data, offering insights into product performance and trends.

10. Product Level Website Analysis: - Evaluated the performance of individual products, identifying best-sellers and underperforming products.

11. Cross-Selling Products: - Analyzed the impact of cross-selling, providing data on customer behavior and revenue generation.

12. Product Refund Analysis: - Examined product refund rates to control for quality issues and identify areas for improvement.

13. Analyze Repeat Behavior: - Investigated the behavior of repeat visitors, including their return intervals, to better understand and engage this valuable segment of customers.

Collectively, these analyses have contributed to data-driven decision-making, allowing the company to optimize marketing strategies, improve customer experiences, enhance product quality, and tailor offerings to customer preferences. This data-driven approach has empowered the company to adapt to changing market conditions, boost profitability, and maximize customer lifetime value. It has added significant value by providing insights that drive growth, efficiency, and competitiveness in the marketplace.

Through the various SQL analyses and daily requests, my efforts have added significant value to the company in several ways:

1. **Informed Decision-Making:** By providing data-driven insights, you have enabled the company to make informed decisions. Whether it's understanding the impact of new product launches, identifying the behavior of repeat customers, or evaluating the performance of marketing channels, your analyses have provided a strong foundation for decision-making.

2. **Optimized Marketing Spend:** The analyses related to marketing channels have allowed the company to optimize its marketing spend. By identifying the most effective and cost-efficient channels, the company can allocate resources more effectively, ensuring a better return on investment.

3. **Customer Segmentation:** Your analyses have helped the company segment its customers based on various factors, including repeat behavior and product preferences. This segmentation allows for targeted marketing and personalized customer experiences, which can lead to higher customer retention and increased sales.

4. **Quality Control:** Analyses related to product refunds have provided insights into quality control. By tracking and addressing issues with specific products, the company can improve product quality and reduce the likelihood of refunds, ultimately saving costs and preserving customer trust.

5. **Conversion Rate Optimization:** Comparing conversion rates before and after specific changes has helped the company understand the impact of these changes on customer behavior. This, in turn, has led to strategies that optimize conversion rates and improve revenue generation.

6. **Customer Lifetime Value:** By analyzing repeat behavior, you've helped the company understand the value of repeat customers. This knowledge is essential for long-term planning, as it allows the company to recognize the significance of customer retention and invest resources accordingly.

In summary, your SQL analyses have empowered the company to make data-driven decisions, optimize marketing strategies, enhance customer experiences, and improve overall business performance. By providing valuable insights into customer behavior and business processes, you've added significant value by contributing to the company's growth, profitability, and sustainability. Your work has enabled the company to stay competitive, adapt to changing market conditions, and ultimately thrive in its industry.

Conclusion:

In conclusion, the series of data analyses conducted across various aspects of the company's online operations has provided valuable insights and strategic direction. These analyses have not only addressed specific business questions but also contributed to an overarching understanding of customer behavior, market trends, and overall performance. Here are the key takeaways:

1. **Data-Driven Decision-Making:** The company has transitioned toward data-driven decision-making, leveraging SQL analyses to inform strategies and investments. This approach has improved the effectiveness of marketing efforts and operational efficiency.

2. **Optimized Marketing Spend:** Bid optimization, channel portfolio optimization, and traffic source analysis have collectively enhanced the company's marketing efficiency. Bids are now aligned with conversion rates and cost-effectiveness, leading to better return on investment.

3. **User-Centric Website Strategy:** The analysis of top website content, landing page performance, and direct traffic has guided the company in tailoring its website content and design to meet user needs and expectations.

4. **Conversion Funnel Enhancement:** The examination of conversion funnels has pinpointed areas for improvement, which has led to better conversion rates and revenue growth.

5. **Product and Cross-Sell Insights:** Analyses on product sales, refund rates, individual product performance, and cross-selling have deepened the company's understanding of product quality and customer preferences, ultimately driving product strategies.

6. **Customer Behavior Understanding:** The analysis of repeat customer behavior has shed light on the value of repeat sessions and their influence on revenue. This has led to the recognition of the need to adapt paid advertising strategies accordingly.

7. **Seasonal Adaptation:** Insights into seasonal trends and business patterns have equipped the company to proactively manage demand fluctuations and take advantage of peak seasons.

The culmination of these analyses has delivered a substantial value addition to the company. It has provided a comprehensive and holistic view of the business, aligning it with market dynamics, customer expectations, and the evolving competitive landscape. As the company continues to refine its strategies and invest in data-driven decision-making, it is better positioned for sustained growth, improved customer satisfaction, and long-term success in its industry.