"A/B TESTING INSIGHTS: SQL-BASED ANALYSIS OF MARKETING STRATEGIES"

PRESENTED BY ASHUTOSH SAHOO

A/B Testing Intro

A/B testing is a simple yet powerful way to compare two versions of something—like marketing campaigns or product features—to see which performs better. By dividing users into two groups (A and B) and analyzing how they respond, we can make data-backed decisions that improve performance, engagement, and ultimately, business outcomes.

Dataset Used

We sourced our dataset from Kaggle: A/B Testing Marketing Campaign Dataset. The dataset includes two CSV files, each containing 30 rows of data. Our primary analysis was conducted on the file, having two campaign groups , control_group.csv and test_group.csv.

Dataset Overview

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

Campaign Name	Date	Spend [USD]	# of Impressions	Reach	# of Website Clicks	# of Searches	# of View Content	# of Add to Cart	# of Purchase
Control Campaign	1.08.2019	2280	82702	56930	7016	2290	2159	1819	618
Control Campaign	2.08.2019	1757	121040	102513	8110	2033	1841	1219	511
Control Campaign	3.08.2019	2343	131711	110862	6508	1737	1549	1134	372
Control Campaign	4.08.2019	1940	72878	61235	3065	1042	982	1183	340
Control Campaign	5.08.2019	1835							
Control Campaign	6.08.2019	3083	109076	87998	4028	1709	1249	784	764
Control Campaign	7.08.2019	2544	142123	127852	2640	1388	1106	1166	499
Control Campaign	8.08.2019	1900	90939	65217	7260	3047	2746	930	462
Control Campaign	9.08.2019	2813	121332	94896	6198	2487	2179	645	501
Control Campaign	10.08.2019	2149	117624	91257	2277	2475	1984	1639	734

Data Cleaning

Combining two campaign data to one by UNION ALL.

```
CREATE TABLE ab_campaign_data AS
SELECT
    'control' AS group_type,
    `Campaign Name`,
    `Date`,
    `Spend [USD]`,
    `# of Impressions`,
    `Reach`,
    `# of Website Clicks`,
    `# of Searches`,
    `# of View Content`,
    `# of Add to Cart`,
    `# of Purchase`
FROM control_group
UNION ALL
SELECT
    'test' AS group_type,
    `Campaign Name`,
    `Date`,
    `Spend [USD]`,
    `# of Impressions`,
    `Reach`,
    `# of Website Clicks`,
    `# of Searches`,
    `# of View Content`,
    `# of Add to Cart`,
    `# of Purchase`
FROM test_group;
```

Replacing Blank Spaces to 0 , for better calculation.

```
SELECT DISTINCT `# of Purchase`
FROM ab_campaign_data;

UPDATE ab_campaign_data
SET `# of Purchase` = 0
WHERE TRIM(`# of Purchase`) = '' OR `# of Purchase` IS NULL;

UPDATE ab_campaign_data
SET `# of Impressions` = 0
WHERE TRIM(`# of Impressions`) = '' OR `# of Impressions` IS NULL;

UPDATE ab_campaign_data
SET `# of Website Clicks` = 0
WHERE TRIM(`# of Website Clicks`) = '' OR `# of Website Clicks` IS NULL;

UPDATE ab_campaign_data
SET `# of Searches` = 0
WHERE TRIM(`# of Searches`) = '' OR `# of Searches` IS NULL;

UPDATE ab_campaign_data
SET `# of View Content` = 0
WHERE TRIM(`# of View Content`) = '' OR `# of View Content` IS NULL;

UPDATE ab_campaign_data
SET `# of Add to Cart` = 0
WHERE TRIM(`# of Add to Cart`) = '' OR `# of Add to Cart` IS NULL;

UPDATE ab_campaign_data
SET `Reach` = 0
WHERE TRIM(`Reach`) = '' OR `Reach` IS NULL;
```

Redefining datatypes for all columns.

```
ALTER TABLE ab_campaign_data
MODIFY `# of Purchase` INT,
MODIFY `# of Impressions` INT,
MODIFY `# of Website Clicks` INT,
MODIFY `# of Searches` INT,
MODIFY `# of View Content` INT,
MODIFY `# of Add to Cart` INT;

ALTER TABLE ab_campaign_data
MODIFY `Reach` INT;

ALTER TABLE ab_campaign_data
MODIFY `Spend [USD]` FLOAT;
```

Rename columns to proper names.

```
ALTER TABLE ab_campaign_data
CHANGE `Campaign Name` campaign_name VARCHAR(255),
CHANGE `Date` campaign_date DATE,
CHANGE `Spend [USD]` spend_usd FLOAT,
CHANGE `# of Impressions` impressions INT,
CHANGE `Reach` reach INT,
CHANGE `# of Website Clicks` website_clicks INT,
CHANGE `# of Searches` searches INT,
CHANGE `# of View Content` view_content INT,
CHANGE `# of Add to Cart` add_to_cart INT,
CHANGE `# of Purchase` purchase INT;
```

Final Cleaned Dataset Overview

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

group_type	campaign_name	campaign_date	spend_usd	impressions	reach	website_clicks	searches	view_content	add_to_cart	purchase
control	Control Campaign	2001-08-20	2280	82702	56930	7016	2290	2159	1819	618
control	Control Campaign	2002-08-20	1757	121040	102513	8110	2033	1841	1219	511
control	Control Campaign	2003-08-20	2343	131711	110862	6508	1737	1549	1134	372
control	Control Campaign	2004-08-20	1940	72878	61235	3065	1042	982	1183	340
control	Control Campaign	2005-08-20	1835	0	0	0	0	0	0	0
control	Control Campaign	2006-08-20	3083	109076	87998	4028	1709	1249	784	764
control	Control Campaign	2007-08-20	2544	142123	127852	2640	1388	1106	1166	499
control	Control Campaign	2008-08-20	1900	90939	65217	7260	3047	2746	930	462
control	Control Campaign	2009-08-20	2813	121332	94896	6198	2487	2179	645	501
control	Control Campaign	2010-08-20	2149	117624	91257	2277	2475	1984	1629	734

Calculating Metrics / Performing Action on Data

Campaign Metrics Aggregation: Summarize total spend, reach, clicks, and purchases by group.

```
--  Common Table Expression to prepare aggregated data
WITH campaign_metrics AS (
   SELECT
      group_type,
                    COUNT(DISTINCT   campaign_name)   AS
total_campaigns,
      COUNT(*) AS total_days,
      SUM(spend_usd) AS total_spend,
      SUM(impressions) AS total_impressions,
      SUM(reach) AS total_reach,
      SUM(website_clicks) AS total_clicks,
      SUM(searches) AS total_searches,
      SUM(view_content) AS total_views,
      SUM(add_to_cart) AS total_carts,
      SUM(purchase) AS total_purchases,
        SUM(purchase) / NULLIF(SUM(website_clicks), 0) AS
purchase_per_click,
         SUM(add_to_cart) / NULLIF(SUM(view_content), 0)
AS cart_per_view
   FROM ab_campaign_data
   GROUP BY group_type
),
```

Conversion Rate:
Percentage of clicks that led to purchases.

```
-- Metric 1: Conversion Rate
conversion_rate AS (
    SELECT
        group_type,
        ROUND(100 * SUM(purchase) / NULLIF(SUM(website_clicks), 0), 2) AS conversion_rate_pct
    FROM ab_campaign_data
    GROUP BY group_type
),
```

Click-Through Rate (CTR): Percentage of impressions that turned into clicks.

```
--  Metric 2: Click-Through Rate (CTR)
ctr AS (
   SELECT
      group_type,
                        ROUND(100   *   SUM(website_clicks)   /
NULLIF(SUM(impressions), 0), 2) AS ctr_pct
   FROM ab_campaign_data
   GROUP BY group_type
),
```

Return on Ad Spend (ROAS): Total purchase value per dollar spent.

```
--  Metric 3: Return on Ad Spend (ROAS)
roas AS (
    SELECT
        group_type,
         ROUND(SUM(purchase) / NULLIF(SUM(spend_usd), 0),
2) AS roas
    FROM ab_campaign_data
    GROUP BY group_type
),
```

Cost Per Purchase (CPP):
Avg. amount spent per purchase.

```
-- Metric 4: Cost per Purchase (CPP)
cpp AS (
    SELECT
        group_type,
         ROUND(SUM(spend_usd) / NULLIF(SUM(purchase), 0),
2) AS cost_per_purchase
    FROM ab_campaign_data
    GROUP BY group_type
),
```

Engagement Rate:
Percentage of users engaging after seeing the ad.

```
-- Metric 5: Engagement Rate
engagement AS (
    SELECT
        group_type,
        ROUND(100 * (
                SUM(view_content) + SUM(add_to_cart) +
SUM(purchase)
                ) / NULLIF(SUM(impressions), 0), 2) AS
engagement_rate_pct
    FROM ab_campaign_data
    GROUP BY group_type
),
```

Average Daily Spend:
Total spend divided by campaign days.

```
--  Metric 6: Daily Average Spend
daily_spend AS (
    SELECT
        group_type,
                ROUND(SUM(spend_usd)  /  COUNT(DISTINCT
campaign_date), 2) AS avg_daily_spend
    FROM ab_campaign_data
    GROUP BY group_type
),
```

Combined Output:
Merge all metrics for comparison between A/B groups using Inner Join.

```
-- Final Output: Combine All Metrics
SELECT
    cm.group_type,
    cm.total_campaigns,
    cm.total_days,
    cm.total_spend,
    cm.total_impressions,
    cm.total_clicks,
    cm.total_purchases,
    conv.conversion_rate_pct,
    ctr.ctr_pct,
    roas.roas,
    cpp.cost_per_purchase,
    engagement.engagement_rate_pct,
    daily_spend.avg_daily_spend
FROM campaign_metrics cm
JOIN conversion_rate conv ON cm.group_type = conv.group_type
JOIN ctr ON cm.group_type = ctr.group_type
JOIN roas ON cm.group_type = roas.group_type
JOIN cpp ON cm.group_type = cpp.group_type
JOIN engagement ON cm.group_type = engagement.group_type
JOIN daily_spend ON cm.group_type = daily_spend.group_type;
```

Final Cleaned Output

Filter Rows: | Export: | Wrap Cell Content:

total_spend	total_impressions	total_clicks	total_purchases	conversion_rate_pct	ctr_pct	roas	cost_per_purchase	engagement_rate_pct	avg_daily_spend
68653	3177233	154303	15161	9.83	4.86	0.22	4.53	3.44	2288.43
76892	2237544	180970	15637	8.64	8.09	0.2	4.92	4.37	2563.07

Conclusion

After analyzing both campaigns, we observed that the Control Campaign outperformed the Test Campaign in several key areas. It recorded a higher conversion rate of 9.83% compared to 8.64%, and delivered a slightly better Return on Ad Spend (ROAS) of 0.22 against 0.20. Additionally, the cost per purchase was lower at $4.53 versus $4.92, making it more cost-efficient. On the other hand, the Test Campaign showed better engagement, with a Click-Through Rate (CTR) of 8.09% (vs 4.86%) and an engagement rate of 4.37% (vs 3.44%), along with a higher number of total clicks and purchases despite having fewer impressions. These findings suggest that while the Test Campaign succeeded in grabbing user attention, the Control Campaign was more effective in converting those interactions into meaningful purchases with better budget utilization. So, our final verdict is that the Control Campaign remains the more efficient and conversion-optimized strategy, making it the better choice for driving cost-effective results.

Thank you very much!