Q1 - A/B Testing & Product Monetization

Understanding the Current State Before A/B Testing

Before conducting an A/B test, it is crucial to evaluate and thoroughly understand the current situation. Here's an analysis of the pre-test findings:

Pre-Test Observations

1. **Data and User Scope:**
    - The dataset includes test variants and event logs for each user, along with their **first event dates**.
    - Users started between **October 2 and October 22, 2022**, and the data spans from **October 2 to December 30, 2022**.
    - Although this covers nearly 3 months, users who joined on October 22 had about **70 days (10 weeks)** of data for the test.
2. **Subscription Types:**
    - There are two product durations:
        - **1 Week**
        - **12 Months**
3. **Test Groups:**
    - There are two experiment groups: **A** and **B**.
    - Metrics for both **1 Week** and **12 Months** subscriptions should be analyzed separately for each group.
4. **Churn Analysis Using Events:**
    - Important events like `'subscribe'` and `'auto_renew_off'` can be used to perform churn analysis.
    - Each user's first subscription date is taken as **Day 0 (D0)**.
5. **Revenue and Retention:**
    - Users remain subscribers until they cancel their subscriptions (`auto_renew_off` event).
    - Subtracting the **first subscription date** from the cancellation date provides the **subscription duration** for each user.
    - Calculating total revenue from users during this period is feasible.

Cohort Table Approach

To analyze retention and revenue:

1. **Weekly Grouping:**
   - Group users into cohorts based on their **subscription start date** (e.g., D0, D7, D14, ... up to D70).
   - This allows for a clear view of how long users from each cohort remain active.
2. **Cohort Metrics:**
   - **Number of new subscribers per week.**
   - **Retention:** The number of users still active each week after subscription.
   - **Churn Rate:** The percentage of users who cancel their subscriptions within each cohort.
   - **Revenue:** Total revenue generated by each cohort over the analysis period.

How This Helps in A/B Testing

1. **Variant Comparison:**

   By analyzing **1 Week** and **12 Months** subscriptions separately for groups A and B, we can determine which variant performs better in terms of retention, revenue, and churn.
2. **Data-Driven Insights:**

   Retention and churn trends across cohorts highlight the strengths or weaknesses of each test variant.
3. **Strategic Decisions:**

   Understanding which variant drives longer subscription durations or higher revenue helps optimize subscription offerings.

```
WITH user_dates AS (
  SELECT
    user_id,
    DATE(TIMESTAMP_MILLIS(CAST(first_event_time AS INT64))) as first_date,
    DATE(TIMESTAMP_MILLIS(CAST(event_time AS INT64))) as event_date,
    event_name,
    experiment_variant,
    (SELECT value.string_value
     FROM UNNEST(properties)
     WHERE key = 'productDuration') as product_duration
  FROM `data-sciene-for-business-imp.app_analytics.dataset_experiment`
  WHERE experiment_variant = 'B'
),

paid_users AS (
  SELECT
    user_id,
    first_date,
    MIN(event_date) as first_subscribe_date
  FROM user_dates
  WHERE event_name = 'subscribe'
  AND product_duration = '1 Week'
  GROUP BY user_id, first_date
),

daily_new_users AS (
```

```
  SELECT
    first_date,
    COUNT(DISTINCT user_id) as d0_users
  FROM paid_users
  GROUP BY first_date
),

user_churn AS (
  SELECT
    pu.user_id,
    pu.first_date,
    MIN(CASE WHEN ud.event_name = 'auto_renew_off' THEN ud.event_date END) as churn_date
  FROM paid_users pu
  LEFT JOIN user_dates ud ON pu.user_id = ud.user_id
  WHERE ud.event_name = 'auto_renew_off'
  GROUP BY pu.user_id, pu.first_date
),

daily_active AS (
  SELECT
    d.first_date,
    d.d0_users,
    d.d0_users - COUNT(CASE WHEN DATE_DIFF(uc.churn_date, d.first_date, DAY) <= 7 THEN 1 END)
as d7_users,
    d.d0_users - COUNT(CASE WHEN DATE_DIFF(uc.churn_date, d.first_date, DAY) <= 14 THEN 1 END)
as d14_users,
    d.d0_users - COUNT(CASE WHEN DATE_DIFF(uc.churn_date, d.first_date, DAY) <= 21 THEN 1 END)
as d21_users,
    d.d0_users - COUNT(CASE WHEN DATE_DIFF(uc.churn_date, d.first_date, DAY) <= 28 THEN 1 END)
as d28_users,
    d.d0_users - COUNT(CASE WHEN DATE_DIFF(uc.churn_date, d.first_date, DAY) <= 35 THEN 1 END)
as d35_users,
    d.d0_users - COUNT(CASE WHEN DATE_DIFF(uc.churn_date, d.first_date, DAY) <= 42 THEN 1 END)
as d42_users,
    d.d0_users - COUNT(CASE WHEN DATE_DIFF(uc.churn_date, d.first_date, DAY) <= 49 THEN 1 END)
as d49_users,
    d.d0_users - COUNT(CASE WHEN DATE_DIFF(uc.churn_date, d.first_date, DAY) <= 56 THEN 1 END)
as d56_users,
    d.d0_users - COUNT(CASE WHEN DATE_DIFF(uc.churn_date, d.first_date, DAY) <= 63 THEN 1 END)
as d63_users,
    d.d0_users - COUNT(CASE WHEN DATE_DIFF(uc.churn_date, d.first_date, DAY) <= 70 THEN 1 END)
as d70_users
  FROM daily_new_users d
  LEFT JOIN user_churn uc ON d.first_date = uc.first_date
  GROUP BY d.first_date, d.d0_users
)

SELECT
  first_date,
  d0_users,
  d7_users,
  d14_users,
  d21_users,
  d28_users,
```

```
  d35_users,
  d42_users,
  d49_users,
  d56_users,
  d63_users,
  d70_users
FROM daily_active
WHERE d0_users > 0
ORDER BY first_date;
```

Above, we see the cohort table for the **1 Week product in Variant B**. We need to repeat this process for each variant and product. In this case, since we have **Variants A and B** and **1 Week and 12 Month products**, this query needs to be executed **4 times** for all combinations.

first_date	d0_users	d7_users	d14_users	d21_users	d28_users	d35_users	d42_users
2022-10-02	3	3	3	3	3	3	3
2022-10-03	7	5	3	2	2	2	2
2022-10-04	43	36	34	31	26	24	21
2022-10-05	64	53	48	42	40	40	36
2022-10-06	78	57	47	44	41	39	35
2022-10-07	57	42	31	26	24	23	19
2022-10-08	59	45	37	34	32	30	28
2022-10-09	54	42	38	34	29	28	26
2022-10-10	74	62	53	49	44	41	38
2022-10-11	99	73	61	50	47	45	39
2022-10-12	110	86	71	65	56	54	49
2022-10-13	58	46	35	32	31	30	30
2022-10-14	51	41	33	29	24	23	22
2022-10-15	55	42	36	32	26	23	21
2022-10-16	48	39	28	22	22	19	16
2022-10-17	76	60	49	44	43	41	36
2022-10-18	93	69	59	53	47	42	40
2022-10-19	68	54	48	45	38	32	29
2022-10-20	98	76	59	52	48	45	39
2022-10-21	72	54	45	41	38	34	33
2022-10-22	18	15	12	11	10	10	7
Sum	1285	1000	830	741	671	628	569
Retention by week	1	0.7782101167	0.83	0.8927710843	0.9055330634	0.9359165425	0.906050955
Retention by total	1	0.7782101167	0.6459143969	0.5766536965	0.5221789883	0.4887159533	0.442801556

Above, we observe the change in the total number of new subscribers for each day, tracked as a cohort, up to **D70**. In the **Retention by Week** row, we see retention rates between weeks, where each value is calculated based on the previous week's retention.

A/B-testing-weekly-retention Google Sheets

Key Points:

- **Week by Week Retention Rate:**
  - If we take the average of the **Retention by Week** values, we calculate the **Week by Week Retention Rate**.
    - For example:

      Avg. Week by Week Retention=0.9027051821
    - This value is for the **1 Week product in Variant B**.
- **Next Step for LTV Calculation:**
  - To proceed with **LTV calculation**, we need the **ARPPU (Average Revenue Per Paying User)** value.
  - Using the weekly retention rates and ARPPU, we can:
    1. Multiply weekly retention values by ARPPU.
    2. Sum these cumulatively to calculate the **weekly LTV**.

LTV Prediction:

- By extending this calculation to 52 or 53 weeks, we can project how LTV changes over time.
- This allows us to comment on metrics like **D365** and **D366**, as required in the scenario, providing insights into long-term user value and potential revenue growth trends.

```
WITH user_payments AS (
  SELECT
    user_id,
    experiment_variant,
    DATE(TIMESTAMP_MILLIS(CAST(event_time AS INT64))) AS event_date,
    (SELECT value.string_value
     FROM UNNEST(properties)
     WHERE key = 'productDuration') AS product_duration,
    (SELECT CAST(value.float_value AS FLOAT64)
     FROM UNNEST(properties)
     WHERE key = 'revenue') AS revenue,
    ROW_NUMBER() OVER(
      PARTITION BY user_id
      ORDER BY TIMESTAMP_MILLIS(CAST(event_time AS INT64))
    ) AS payment_order
  FROM `data-sciene-for-business-imp.app_analytics.dataset_experiment`
  WHERE event_name = 'subscribe'
),

revenue_metrics AS (
  SELECT
    experiment_variant,
    product_duration,
    AVG(revenue) AS arppu,
    SUM(revenue) AS total_revenue,
    COUNT(DISTINCT user_id) AS paying_users
  FROM user_payments
  WHERE payment_order = 1  -- for first subs
  GROUP BY experiment_variant, product_duration
),
```

```
total_users AS (
  SELECT
    experiment_variant,
    COUNT(DISTINCT user_id) AS total_users
  FROM `data-sciene-for-business-imp.app_analytics.dataset_experiment`
  GROUP BY experiment_variant
)

SELECT
  r.experiment_variant,
  r.product_duration,
  ROUND(r.arppu, 2) AS arppu,
  ROUND(r.total_revenue, 2) AS total_revenue,
  r.paying_users,
  t.total_users,
  ROUND(r.total_revenue / NULLIF(t.total_users, 0), 2) AS arpu,
  ROUND(CAST(r.paying_users AS FLOAT64) / NULLIF(t.total_users, 0) * 100, 2) AS conversion_rat
e
FROM revenue_metrics r
LEFT JOIN total_users t ON r.experiment_variant = t.experiment_variant
WHERE r.product_duration IS NOT NULL
ORDER BY
  r.experiment_variant,
  r.product_duration;
```

Using this query, I arrive at the following table. However, there is an important point I need to emphasize:

In my method, I considered **ARPPU** as **only the initial payments**. The reason for this is that I conducted a weekly analysis, so the cumulative accumulation should start based on the **Average Revenue Per Paid User (ARPPU)** in the **first week**.

This ensures that in the following weeks, the retention rate can be applied and summed weekly, providing a reliable estimate for the **LTV (Lifetime Value)** at any point during the year.

experiment_variant	product_duration	arppu	total_revenue first subs	paying_users	total_users	arpu	conversi
A	1 Week	$4.89	7,838.67	1603	4820	1.63	33.26
A	12 Month	$27.77	23,717.96	854	4820	4.92	17.72
B	1 Week	$4.89	6,283.65	1285	4911	1.28	26.17
B	12 Month	$19.72	33,299.62	1689	4911	6.78	34.39

Based on this table, the **ARPPU for the first payment of B 1 Week** is measured as **$4.89**. Using this value, we can calculate the **LTV for D365 and D366** as follows:

Steps for LTV Calculation

1. **Initial ARPPU:**
   - Use the first week's ARPPU as the starting point: **$4.89**.
2. **Weekly Retention Rates:**
   - Apply the weekly retention rate (e.g., **0.9027**) cumulatively to project user retention for each subsequent week.
3. **Cumulative Revenue Calculation:**
   - For each week, multiply the retention rate by the ARPPU and add the result to the previous weeks' revenue to calculate cumulative LTV.
4. **D365 vs. D366:**

- Extend the weekly retention calculation to **52 weeks** (D365) and then to **53 weeks** (D366) to observe the difference in LTV for one additional week.

Example Calculation

$$\text{LTV}_n = \sum_{i=1}^{n} (\text{ARPPU} \times \text{Retention Rate}_i)$$

Where:

- $n = \text{Number of weeks (e.g., 52 or 53)}$
- $\text{Retention Rate}_i = \text{Cumulative weekly retention rate for week } i$

Projection for D365 and D366

- **D365:** Calculate cumulative LTV after 52 weeks.
- **D366:** Add the retention-adjusted revenue for the 53rd week to the D365 LTV.

This method helps provide a clear comparison of how much incremental value is generated by retaining users for one extra week beyond a full year.

week	subs - retention	LTV
w0	1	$4.89
w1	0.7782101167	$8.70
w2	0.6459143969	$11.85
w3	0.5766536965	$14.67
w4	0.5221789883	$17.23
...	...	...
...	...	...
w50	0.00587533315	$45.10
w51	0.005303693681	$45.13
w52	0.00478767177	$45.15
w53	0.004321856117	$45.17

- For the **B group "1 Week" product**, the calculations result in:
  - **D365 LTV = $45.15**
  - **D366 LTV = $45.17**
- When the exact same process is applied to the **A group "1 Week" product**, the results are:
  - **D365 LTV = $44.94**
  - **D366 LTV = $44.96**

This indicates a slight difference in the lifetime value between the groups, which could reflect differences in retention patterns or user behavior for each variant.

LTV Calculation for the Annual Product

For the **12 Month product**, we follow the same approach as with the weekly product up to the point of LTV calculation, with one key adjustment:

Key Adjustments for Annual Product LTV Calculation

1. **Payment Period Consideration:**
   - For annual subscriptions, the LTV must reflect **only the first payment's Average Revenue Per Paid User (ARPPU)** until the **same day of the following year** when users are expected to renew their subscriptions.
   - Without renewal data for the subsequent year, we cannot calculate further cumulative revenue.
2. **Retention Projection:**
   - Using the **avg. week-by-week retention rate**, we can project how many users will still retain their subscription by the end of 52 weeks.
   - This provides an estimate of the number of users likely to renew their subscription for another year.

Sample Calculation for B Variant "12 Month" Product

1. **Base ARPPU:** Assume the initial ARPPU for the "12 Month" product in Variant B is calculated (e.g., **$X**).
2. **Retention at 52 Weeks:** Use the cumulative retention rate after 52 weeks to estimate the proportion of users renewing for another year.

Formula:

$$\text{LTV}_{12\text{ Month}} = \text{ARPPU} + (\text{ARPPU} \times \text{Retention}_{52})$$

Where:

- Retention_{52} is the cumulative retention rate at 52 weeks.

avg. week by week retention =	0.9641444118		
Avg. Revenue Per Paid User (First Subs) =	$19.72		
Expected yearly Retention Rate / d365 week 52	14.98%		
d366 week 53	14.44%		
d365 LTV =	**$19.72**		
d366 LTV =	AVG. Revenue per First Yearly Subs + (yearly Retention*ARPPU) = LTV for d366	19.72 + (19.72*0.144) =	**$22.57**

- Using the same approach as for Variant B, we calculate the **LTV** for the **A variant "12 Month" product**:

avg. week by week retention =	0.9625627312		
Avg. Revenue Per Paid User (First Subs) =	$27.77		
Expected yearly Retention Rate / d365 week 52	13.75%		
d366 week 53	13.24%		
Expected d365 LTV =	**$27.77**		
d366 LTV =	AVG. Revenue per First Yearly Subs + (yearly Retention*ARPPU) = LTV for d366	27.77 + (27.77*0.132) =	**$31.45**

Let's investigate why Product B's annual plan yields a lower LTV.

```
SELECT
  DISTINCT (
```

```
  SELECT
    value.float_value
  FROM
    UNNEST(properties)
  WHERE
    KEY = 'revenue') price,
  experiment_variant,
  CASE
    WHEN prop.key = 'productDuration' THEN prop.value.string_value
END
  AS product_duration,
FROM
  `data-sciene-for-business-imp.app_analytics.dataset_experiment`,
  UNNEST(properties) AS prop,
  UNNEST(properties) AS rev
WHERE
  prop.key = 'productDuration'
  AND rev.key = 'revenue'
  and  event_name = 'subscribe'
  order by 2,3
```

price	experiment_variant	product_duration
$0.00	A	1 Week
$4.89	A	1 Week
$5.94	A	1 Week
$33.99	A	12 Month
$27.99	A	12 Month
$10.49	A	12 Month
$13.99	A	12 Month
$4.89	B	1 Week
$0.00	B	12 Month
$27.99	B	12 Month
$10.49	B	12 Month
$13.99	B	12 Month

- I think the reason might be that it offers more discounted products.

Refund rates

```
WITH all_users AS (
  SELECT
    DISTINCT user_id,
    experiment_variant
  FROM
    `data-sciene-for-business-imp.app_analytics.dataset_experiment`
),

subscribers AS (
  SELECT
    DISTINCT user_id,
    experiment_variant
```

```
  FROM
    `data-sciene-for-business-imp.app_analytics.dataset_experiment`
  WHERE
    event_name = 'subscribe'
),

refunds AS (
  SELECT
    DISTINCT user_id,
    experiment_variant
  FROM
    `data-sciene-for-business-imp.app_analytics.dataset_experiment`
  WHERE
    event_name = 'refund'
)

SELECT
  au.experiment_variant,
  COUNT(DISTINCT au.user_id) AS total_users,
  COUNT(DISTINCT s.user_id) AS total_paying_users,
  COUNT(DISTINCT r.user_id) AS total_refund_users,
  COUNT(DISTINCT au.user_id) - COUNT(DISTINCT s.user_id) AS non_paying_users,
  ROUND(COUNT(DISTINCT s.user_id) / COUNT(DISTINCT au.user_id) * 100, 2) AS paying_user_ratio,
  ROUND(COUNT(DISTINCT r.user_id) / COUNT(DISTINCT au.user_id) * 100, 2) AS refund_user_ratio,
  ROUND((COUNT(DISTINCT au.user_id) - COUNT(DISTINCT s.user_id)) / COUNT(DISTINCT au.user_id)
* 100, 2) AS non_paying_user_ratio
FROM
  all_users au
LEFT JOIN
  subscribers s
ON
  au.user_id = s.user_id AND au.experiment_variant = s.experiment_variant
LEFT JOIN
  refunds r
ON
  au.user_id = r.user_id AND au.experiment_variant = r.experiment_variant
GROUP BY
  au.experiment_variant;
```

experiment_variant	total_users	total_paying_users	total_refund_users	non_paying_users	paying_user_ratio	refund_user_r
A	4820	2457	64	2363	50.98	1.33
B	4911	2974	51	1937	60.56	1.04

- **Variant B** is more successful in terms of the paying user ratio and has a lower refund rate. This indicates that users respond better to Variant B, and it has the potential to generate more revenue.

Findings

Retention Rates

1 Week Product:

- **Variant A**:
  - **Avg. week by week retention = 0.9016358301**

- Annual retention: 0.901635^{52}≈0.0045 (0.4**5%**)

- **Variant B**:
  - **Avg. week by week retention = 0.9027051821**
  - Annual retention: 0.902752^{52}≈0.0047 (0**.47%**)

12 Month Product:

- **Variant A**:
  - **Avg. week by week retention = 0.9625627312**
  - Annual retention: 0.962652^{52}≈0.1375 (**13.75%**)
- **Variant B**:
  - **Avg. week by week retention = 0.9641444118**
  - Annual retention: 0.964152^{52}≈0.1444 (**14.44%**)

ARPPU and LTV Comparison

1 Week Product:

- For Variant B:
  - **D365 LTV: $45.15**, **D366 LTV: $45.17**
- For Variant A:
  - **D365 LTV: $44.94**, **D366 LTV: $44.96**

12 Month Product:

- For Variant B:
  - **D365 LTV: $19.72**, **D366 LTV: $22.57**
- For Variant A:
  - **D365 LTV: $27.77**, **D366 LTV: $31.45**

Pricing Differences

- Variant B offers more **discounted pricing**, resulting in a lower ARPPU. However, this approach has positively impacted the overall revenue by increasing the payment conversion rate among users.

Refund Rates

- **Variant B's refund rate (1.04%)** is lower compared to **Variant A's (1.33%)**, indicating higher customer satisfaction with Variant B.

Higher Conversion Rate

- The paying user rate for Variant B is **60.56%**, compared to **50.98%** for Variant A.

Conclusion

Variant B emerges as the winning variant, demonstrating:

- **Higher retention rates**
- **Increased overall revenue through higher sales**
- **Lower refund rates**, indicating improved customer satisfaction

- **Better payment conversion rates**

Recommendations

To further optimize performance, especially for the **12 Month product**, the following steps are recommended:

1. **Reduce Excessive Discounts:**

   Variant B's **heavily discounted pricing** for the 12 Month product lowers its ARPPU. Moderating discounts and slightly increasing prices could balance user acquisition and revenue generation.

2. **Focus on the Weekly Product:**

   The **1 Week product** shows better long-term retention and LTV potential. Channeling more users into this plan could enhance both short-term revenue and long-term user engagement.

This strategy offers an opportunity to boost short-term revenue while ensuring long-term user retention and loyalty.