1. Data Quality Check

```sql
--We are running an experiment at an item-level, which means all users
--who visit will see the same page, but the layout of different item
--pages may differ. Compare this table to the assignment events we
--captured for user_level_testing.

--Does this table have everything you need to compute metrics like
--30-day view-binary?
--  No, this table does not have all of the data required to
--  compute a 30-day view-binary. Specifically, we will need to
--  know the dates of the tests and the orders so we can create
--  a 30-day window.

SELECT
  *
FROM
  dsv1069.final_assignments_qa;

```

2. Reformat the Data

```sql
--Reformat the final_assignments_qa to look like the final_assignments
--table, filling in any missing values with a placeholder of the
--appropriate data type.

SELECT
  item_id,
  test_a AS test_assignment,
  (CASE
    WHEN test_a is NOT NULL
    THEN 'test_a'
    ELSE NULL
  END) AS test_number,
  (CASE
    WHEN test_a is NOT NULL
    THEN '2025-01-01 00:00:00'
    ELSE NULL
  END) AS test_start_date
FROM dsv1069.final_assignments_qa

UNION

SELECT
  item_id,
  test_b AS test_assignment,
  (CASE
    WHEN test_b is NOT NULL
    THEN 'test_b'
    ELSE NULL
  END) AS test_number,
  (CASE
    WHEN test_b is NOT NULL
    THEN '2025-01-01 00:00:00'
    ELSE NULL
  END) AS test_start_date
FROM dsv1069.final_assignments_qa

UNION

SELECT
  item_id,
  test_c AS test_assignment,
  (CASE
    WHEN test_c is NOT NULL
    THEN 'test_c'
    ELSE NULL
```

```
  END) AS test_number,
  (CASE
    WHEN test_c is NOT NULL
    THEN '2025-01-01 00:00:00'
    ELSE NULL
  END) AS test_start_date
FROM dsv1069.final_assignments_qa

UNION

SELECT
  item_id,
  test_d AS test_assignment,
  (CASE
    WHEN test_d is NOT NULL
    THEN 'test_d'
    ELSE NULL
  END) AS test_number,
  (CASE
    WHEN test_d is NOT NULL
    THEN '2025-01-01 00:00:00'
    ELSE NULL
  END) AS test_start_date
FROM dsv1069.final_assignments_qa

UNION

SELECT
  item_id,
  test_e AS test_assignment,
  (CASE
    WHEN test_e is NOT NULL
    THEN 'test_e'
    ELSE NULL
  END) AS test_number,
  (CASE
    WHEN test_e is NOT NULL
    THEN '2025-01-01 00:00:00'
    ELSE NULL
  END) AS test_start_date
FROM dsv1069.final_assignments_qa

UNION

SELECT
  item_id,
```

```
  test_f AS test_assignment,
  (CASE
    WHEN test_f is NOT NULL
    THEN 'test_f'
    ELSE NULL
  END) AS test_number,
  (CASE
    WHEN test_f is NOT NULL
    THEN '2025-01-01 00:00:00'
    ELSE NULL
  END) AS test_start_date
FROM dsv1069.final_assignments_qa;
```

3. Compute Order Binary

```sql
-- Use this table to compute order_binary for
-- the 30 day window after the test_start_date
-- for the test named item_test_2

SELECT
  test_assignment,
  SUM(order_binary) AS ordered_items,
  COUNT(DISTINCT item_id) AS number_of_items
FROM
  (SELECT
    item_test_2.item_id,
    item_test_2.test_assignment,
    item_test_2.test_number,
    item_test_2.test_start_date,
    item_test_2.created_at,
    MAX(CASE
            WHEN (created_at > test_start_date
            AND DATE_PART('day', created_at - test_start_date) <= 30)
            THEN 1
            ELSE 0
        END) AS order_binary
  FROM
    (
      SELECT
        final_assignments.*,
        DATE(orders.created_at) AS created_at
      FROM
        dsv1069.final_assignments AS final_assignments
      LEFT JOIN
        dsv1069.orders AS orders
      ON
        final_assignments.item_id = orders.item_id
      WHERE
        test_number = 'item_test_2'
    ) AS item_test_2
  GROUP BY
    item_test_2.item_id,
    item_test_2.test_assignment,
    item_test_2.test_number,
    item_test_2.test_start_date,
    item_test_2.created_at) AS order_binary
GROUP BY
  test_assignment;

```

4. Compute View Item Metrics

```sql
-- Use the final_assignments table to calculate the view binary, and
-- average views for the 30 day window after the test assignment for
-- item_test_2. (You may include the day the test started)

SELECT
  test_assignment,
  SUM(binary_view) AS viewed_items,
  COUNT(item_id) AS items_assignment,
  SUM(views) AS total_views,
  SUM(views)/COUNT(item_id) AS average_views
FROM
(
 SELECT
  final_assignments.test_assignment,
  final_assignments.item_id,
  MAX(
    CASE
      WHEN views.event_time > final_assignments.test_start_date
      THEN 1
      ELSE 0
    END)  AS binary_view,
  COUNT(views.event_id) AS views
 FROM
    dsv1069.final_assignments
 LEFT JOIN
  (
    SELECT
      event_time,
      event_id,
      CAST(parameter_value AS INT) AS item_id
    FROM
      dsv1069.events
    WHERE
      event_name = 'view_item'
    AND
      parameter_name = 'item_id'
  ) views
 ON
  final_assignments.item_id = views.item_id
 AND
  views.event_time >= final_assignments.test_start_date
 AND
  DATE_PART('day', views.event_time - final_assignments.test_start_date ) <= 30
```

```sql
 WHERE
  final_assignments.test_number = 'item_test_2'
 GROUP BY
  final_assignments.item_id,
  final_assignments.test_assignment
) view_metrics

GROUP BY
  view_metrics.test_assignment;
```

5 & 6. P-values and Report Builder

FINAL ASSIGNMENT

1. DATA QUALITY CHECK

	item_id	test_a	test_b	test_c	test_d	test_e	test_f
1	2512	1	0	1	1	0	1
2	482	0	1	1	1	0	0
3	2446	0	1	1	0	1	0
4	1312	0	0	0	0	0	1
5	3556	1	1	0	1	0	0
6	131	0	0	0	0	1	1
7	1178	1	0	1	0	1	1
8	110	0	1	1	1	1	0
9	47	0	0	1	0	1	1
10	1696	0	0	1	1	1	1
11	3196	0	0	0	1	0	1
12	1578	1	0	0	1	0	1

Does this table have everything you need to compute metrics like 30-day view-binary?

No, this table does not have all of the data required to compute a 30-day view-binary. Specifically, we will need to know the dates of the tests and the orders so we can create a 30-day window.

2. REFORMAT THE DATA

	item_id	test_assignment	test_number	test_start_date
1	3824	1	test_f	2025-01-01 00:00:00
2	2098	0	test_b	2025-01-01 00:00:00
3	556	0	test_d	2025-01-01 00:00:00
4	3033	1	test_f	2025-01-01 00:00:00
5	445	1	test_a	2025-01-01 00:00:00
6	3332	0	test_a	2025-01-01 00:00:00
7	805	1	test_c	2025-01-01 00:00:00
8	610	1	test_e	2025-01-01 00:00:00
9	3504	0	test_a	2025-01-01 00:00:00
10	1281	0	test_c	2025-01-01 00:00:00
11	3706	0	test_b	2025-01-01 00:00:00
12	259	1	test_e	2025-01-01 00:00:00

3. COMPUTE ORDER BINARY

	test_assignment	ordered_items	number_of_items
1	0	386	1130
2	1	363	1068

LIFT AND P-VALUE FOR ORDER METRICS:

Condition Success Rate
Control 31% - 37% (34%)
Treatment 31% - 37% (34%)

Improvement: -12% - 11% (-0.5%)
p-value: 0.93

When comparing orders in each condition, there is no observed difference. Further, with a p-value far above 0.05, we are unable to say whether there is any difference.

4. COMPUTE VIEW ITEM METRICS

	test_assignment	viewed_items	items_assignment	total_views	average_views
1	0	918	1130	1916	1.69557522124
2	1	890	1068	1862	1.74344569288

LIFT AND P-VALUE FOR VIEW METRICS:

Condition Success Rate
Control 79% - 83% (81%)
Treatment 81% - 85% (83%)

Improvement: -1.4% - 6.5% (2.6%)
p-value: 0.20

For views, the treatment condition does seem to have increased the success rate slightly. However, with a p-value of 0.2, we are unable to reject the null hypothesis and conclude that the treatment condition contributes to a higher view rate.