CLICK - STREAM BUSINESS CASE STUDY

INDEX

1. Our CEO wants to know how our website is doing in terms of customer acquisition?

1. How many users are added each month?

Query:

```
WITH new_users_added_per_month AS
    (SELECT
            EXTRACT(YEAR FROM created_at) AS year,
            EXTRACT(MONTH FROM created_at) AS month,
            COUNT(DISTINCT user_id) AS new_users_added
      FROM
            click_stream.users
      GROUP BY year, month
      ORDER BY year, month)

SELECT
      *
  FROM
          new_users_added_per_month;
```

Result:

Row	year	month	new_users_added
1	2013	2	9
2	2013	3	153
3	2013	4	288
4	2013	5	376
5	2013	6	481
6	2013	7	588
7	2013	8	648

Results per page: 50 1 – 50 of 65

2. How many users are deleted or merged (with parent user) each month?

Query:

```
WITH deleted_or_merged_users_per_month AS
    (SELECT
            del_merged_users_temp.year,
            del_merged_users_temp.month,
            SUM(del_merged_users_temp.users_del_or_merged) AS users_del_or_merged
    FROM
        (SELECT
                    EXTRACT(YEAR FROM deleted_at) AS year,
                    EXTRACT(MONTH FROM deleted_at) AS month,
                    COUNT(DISTINCT user_id) AS users_del_or_merged
            FROM
                    click_stream.users
            WHERE
                    deleted_at IS NOT NULL
            GROUP BY year,
                            month
            UNION ALL
            SELECT
                    EXTRACT(YEAR FROM merged_at) AS year,
                    EXTRACT(MONTH FROM merged_at) AS month,
                    COUNT(DISTINCT user_id) AS users_del_or_merged
            FROM
                    click_stream.users
            WHERE
                    merged_at IS NOT NULL
              AND user_id != parent_user_id
            GROUP BY year,
                    month) AS del_merged_users_temp
    GROUP BY del_merged_users_temp.year,
                del_merged_users_temp.month
    ORDER BY del_merged_users_temp.year,
                del_merged_users_temp.month)

SELECT
      *
  FROM
            deleted_or_merged_users_per_month;
```

Result:

Row	year	month	users_del_or_merged
1	2013	5	1
2	2013	7	2
3	2013	8	6
4	2013	9	5
5	2013	10	10
6	2013	11	11
7	2013	12	10

Results per page: 50 1 – 50 of 61

3. How many users are added effectively each month.

(Base tables **new_users_added_per_month** & **deleted_or_merged_users_per_month** on Pages 3 & 4)

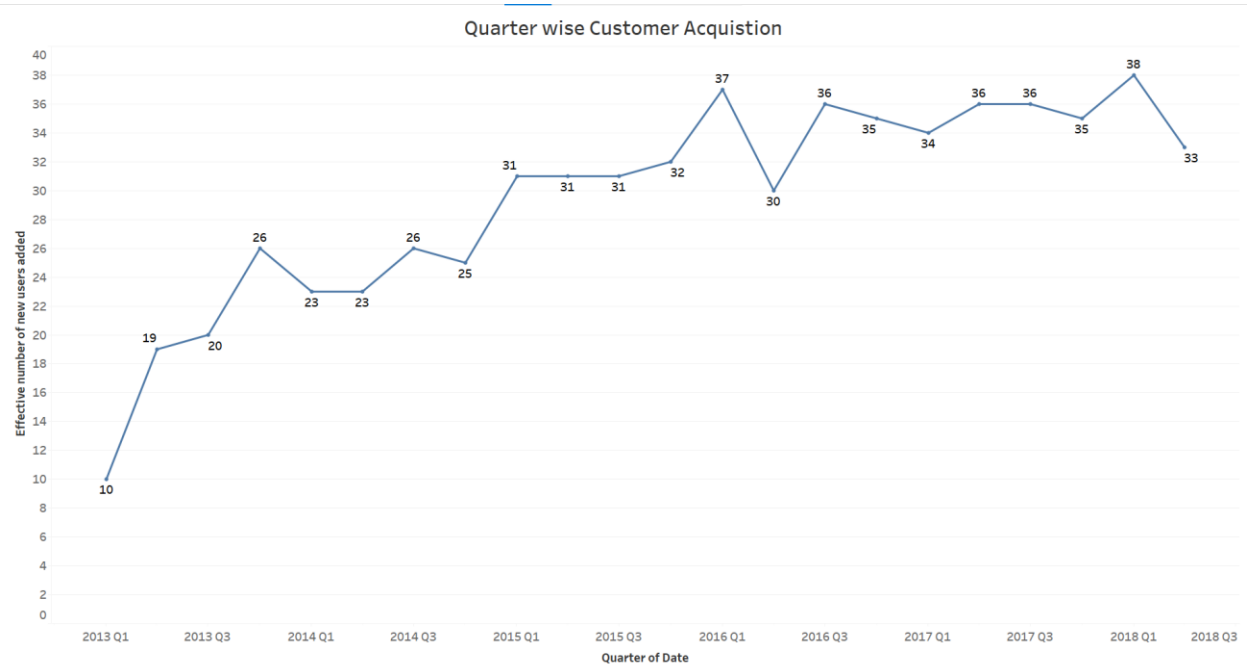
Query:

```
month_wise_user_data  AS
 (SELECT
        a.year,
        a.month,
        a.new_users_added,
        COALESCE(d.users_del_or_merged, 0) AS users_deleted_or_merged,
        a.new_users_added - COALESCE(d.users_del_or_merged, 0) AS eff_users_added
  FROM
      new_users_added_per_month AS a
            LEFT JOIN
      deleted_or_merged_users_per_month AS d ON a.year = d.year
                                                AND a.month = d.month
  ORDER BY a.year,
                a.month)

SELECT
    *
FROM
    month_wise_user_data;
```

Result:

Row	year	month	new_users_added	users_deleted_or_merged	eff_users_added
1	2013	2	9	0	9
2	2013	3	153	0	153
3	2013	4	288	0	288
4	2013	5	376	1	375
5	2013	6	481	0	481
6	2013	7	588	2	586
7	2013	8	648	6	642

Results per page: 50 1 – 50 of 65



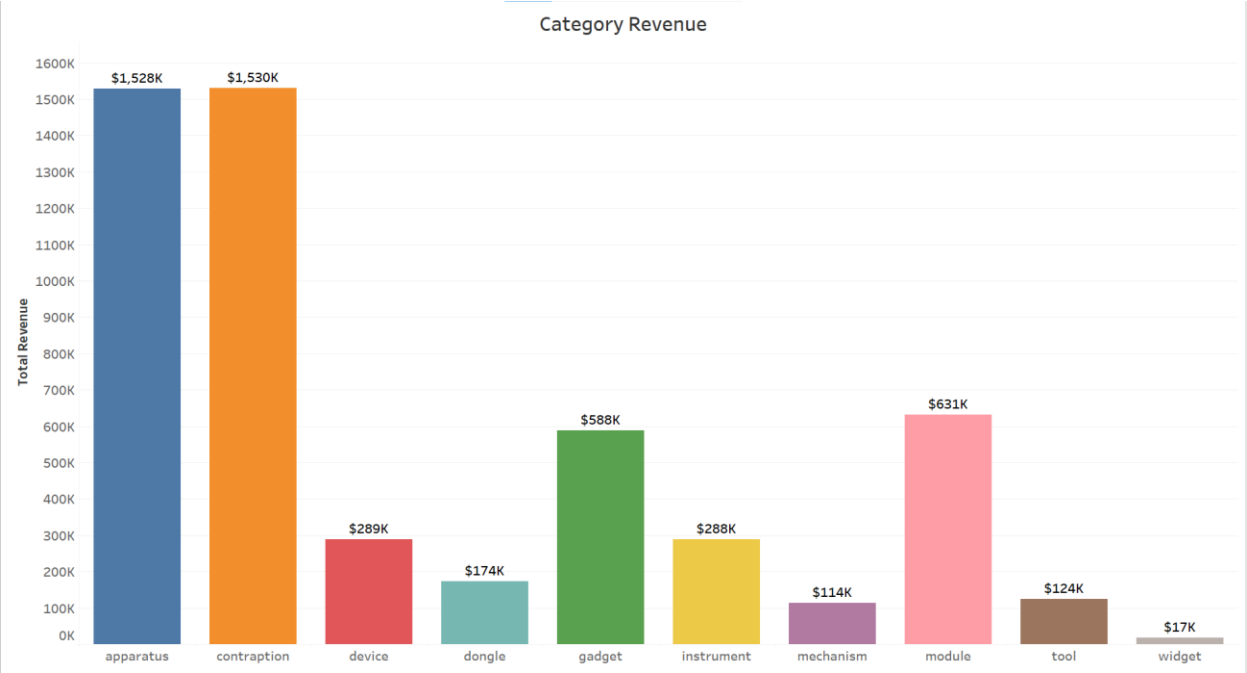
2. Investigate how each product category is doing and their popularities with respect to their revenues and how many times, any item from that category has been ordered.

Query:

```
WITH most_popular_categories AS
    (SELECT
            i.category,
            ROUND(SUM(COALESCE(o.price, 0))) AS total_revenue,
            COUNT(o.line_item_id) AS items_ordered
    FROM
            click_stream.items AS i
              LEFT JOIN
            click_stream.orders AS o ON i.item_id = o.item_id
    GROUP BY i.category
    ORDER BY SUM(o.price) DESC,
                COUNT(DISTINCT o.line_item_id) DESC)

SELECT
      *
FROM
      most_popular_categories;
```

Result:

Row	category	total_revenue	items_ordered
1	contraption	1529855.0	4700
2	apparatus	1527894.0	4892
3	module	631364.0	4800
4	gadget	588179.0	4695
5	device	288543.0	4735
6	instrument	288342.0	4767
7	dongle	174168.0	4665
8	tool	124022.0	4633



3. Our CEO wants to know the trend of how our company is doing in terms of revenue and number of orders.

1. Revenue and number of orders on a daily basis.

Query:

```
WITH daily_revenue_data AS
    (SELECT
                EXTRACT(DATE FROM created_at) AS day,
        COUNT(DISTINCT invoice_id) AS orders,
        ROUND(SUM(price), 1) AS revenue
    FROM
        click_stream.orders
    GROUP BY day
    ORDER BY day)

SELECT
    *
FROM
    daily_revenue_data;
```

Result:

Row	day	orders	revenue
1	2013-03-12	1	387.5
2	2013-03-26	1	48.4
3	2013-03-30	1	64.5
4	2013-04-05	1	41.1
5	2013-04-12	1	13.5
6	2013-04-15	1	118.8
7	2013-04-22	1	15.0

Results per page: 50 1 – 50 of 1820

2. Weekly running average of revenue and number of orders.

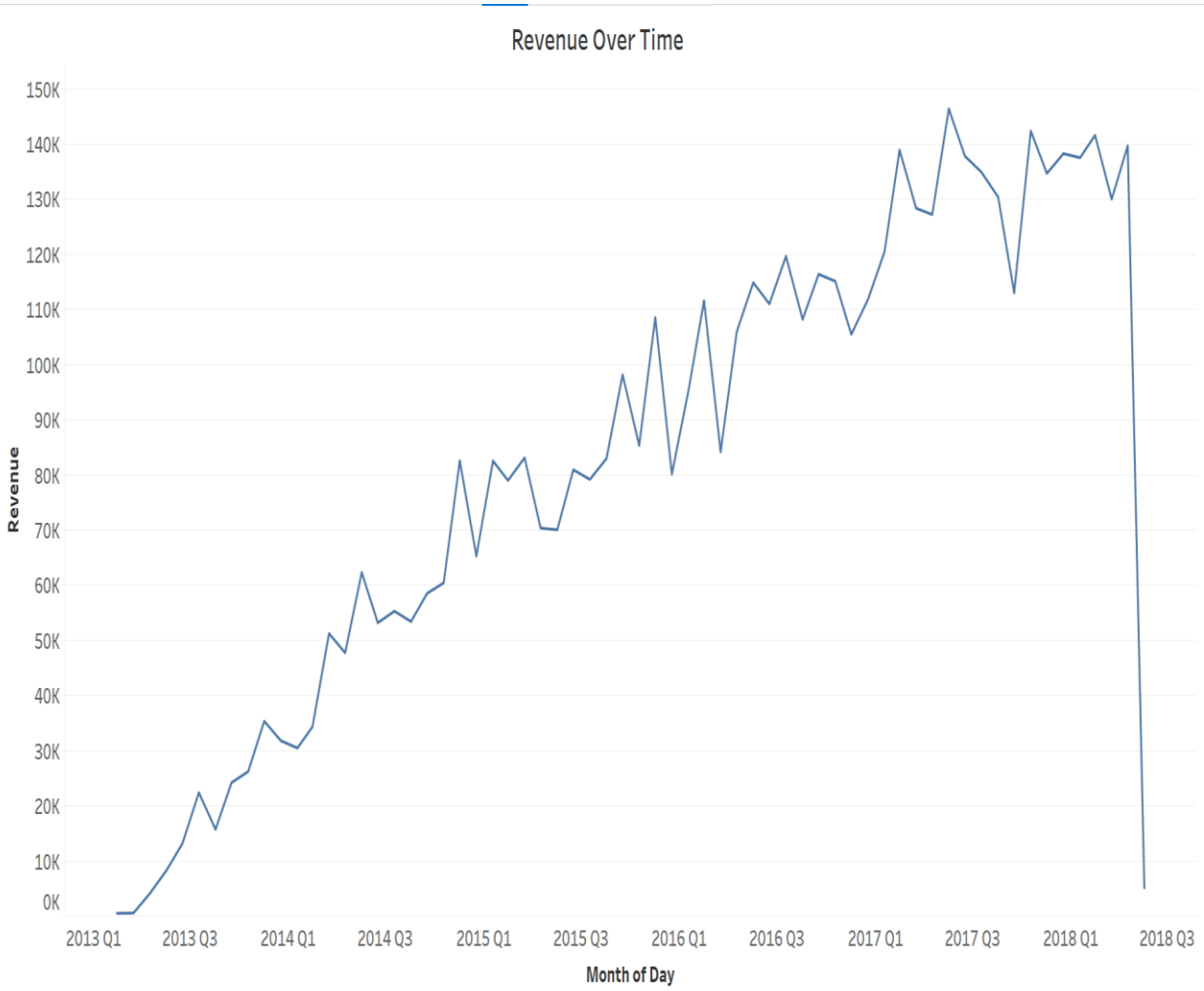
Query:

```
WITH daily_revenue_data AS
    (SELECT
            EXTRACT(DATE FROM created_at) AS day,
            COUNT(DISTINCT invoice_id) AS orders,
            ROUND(SUM(price), 1) AS revenue
    FROM
            click_stream.orders
    GROUP BY day
    ORDER BY day),

seven_day_rolling_avg AS
    (SELECT
          d1.day,
          ROUND(AVG(d2.orders), 2) AS sev_day_roll_avg_orders,
          ROUND(AVG(d2.revenue), 1) AS sev_day_roll_avg_revenue
    FROM
          daily_revenue_data AS d1
            JOIN

daily_revenue_data AS d2 ON DATETIME_DIFF(d1.day, d2.day, DAY) BETWEEN 0 AND 6
    GROUP BY d1.day
    ORDER BY d1.day)

SELECT
    *
FROM
    seven_day_rolling_avg;
```

Result:

Row	day	sev_day_roll_avg_orders	sev_day_roll_avg_revenue
1	2013-03-12	1.0	387.5
2	2013-03-26	1.0	48.4
3	2013-03-30	1.0	56.5
4	2013-04-05	1.0	52.8
5	2013-04-12	1.0	13.5
6	2013-04-15	1.0	66.2
7	2013-04-22	1.0	15.0

Results per page: 50 1 – 50 of 1820



4. The data science team at our organization wants to analyze the customer engagement of our website. Please analyze the data to find out the average time (in days) which a user takes to revisit our website and order an item.

Query:

```
WITH users_distinct_orders AS
    (SELECT     invoice_id,
                        user_id,
                        ROUND(SUM(price), 1) AS price,
                        created_at,
                paid_at
     FROM
                 click_stream.orders
    GROUP BY invoice_id,
                 user_id,
                 created_at,
                 paid_at
    ORDER BY user_id, created_at),

users_consecutive_orders AS
    (SELECT
            user_id,
            invoice_id AS cur_invoice_id,
            created_at AS cur_order_date,
            price AS cur_order_price,
            LAG(invoice_id) OVER(PARTITION BY user_id ORDER BY created_at) AS last_invoice_
id,
            LAG(created_at) OVER(PARTITION BY user_id ORDER BY created_at) AS last_order_d
ate,
            LAG(price) OVER(PARTITION BY user_id ORDER BY created_at) AS last_order_price,
            DATETIME_DIFF(created_at, LAG(created_at) OVER(PARTITION BY user_id ORDER B
Y created_at),
        DAY)  AS day_diff
    FROM
                    users_distinct_orders),

user_level_avg_reorder_time AS
    (SELECT
            user_id,
            ROUND(AVG(day_diff), 2) AS avg_reorder_time_in_days
    FROM
            users_consecutive_orders
    GROUP BY user_id
    HAVING AVG(day_diff) IS NOT NULL
    ORDER BY user_id),
```

```
avg_reorder_time_in_days AS
    (SELECT
        ROUND(AVG(avg_reorder_time_in_days), 1) AS average_reorder_in_days
     FROM
            user_level_avg_reorder_time)

SELECT
    *
FROM
    user_level_avg_reorder_time;
```

Result:

Row	user_id	avg_reorder_time_in_days
1	694	130.0
2	849	27.0
3	1004	62.0
4	1009	126.0
5	1053	41.0
6	1078	117.0
7	1317	67.0

Results per page: 50 1 – 50 of 1421

```
SELECT
    *
FROM
    avg_reorder_time_in_days;
```

Result:

Row	average_reorder_in_days
1	65.9

5. The data science team at our company wants to design a recommender system which recommends a subscription of products from a specific category to the customer if he has ordered items from that category in the past for two or more times. Please help analyze the data to give relevant insights.

Query:

```
WITH categories_ordered AS
    (SELECT
            COALESCE(u.parent_user_id, u.user_id) AS user_id,
            item_category,
            1 AS order_count
    FROM
            click_stream.orders AS o
             JOIN
            click_stream.users AS u ON o.user_id = COALESCE(u.parent_user_id, u.user_id)
    WHERE
            u.deleted_at IS NULL
    GROUP BY invoice_id, COALESCE(u.parent_user_id, u.user_id), item_category),

categories_reordered AS
    (SELECT
             user_id,
             item_category,
             SUM(order_count) AS orders
    FROM
             categories_ordered
    GROUP BY user_id, item_category
    HAVING SUM(order_count) >= 2),

recommended_categories AS
    (SELECT
            user_id,
            STRING_AGG(item_category, ',' ORDER BY orders DESC) AS category_recommendatio
ns
    FROM
            categories_reordered
    GROUP BY user_id
    ORDER BY LENGTH(category_recommendations) DESC)

SELECT
    *
FROM
    recommended_categories;
```

Result:

Row	user_id	category_recommendations
1	38278	mechanism,contraption,gadget
2	104899	mechanism,instrument,gadget
3	20865	instrument,device
4	60455	instrument,gadget
5	88724	apparatus,widget
6	92223	dongle,mechanism
7	219570	device,dongle

Results per page: 50 1 – 50 of 162

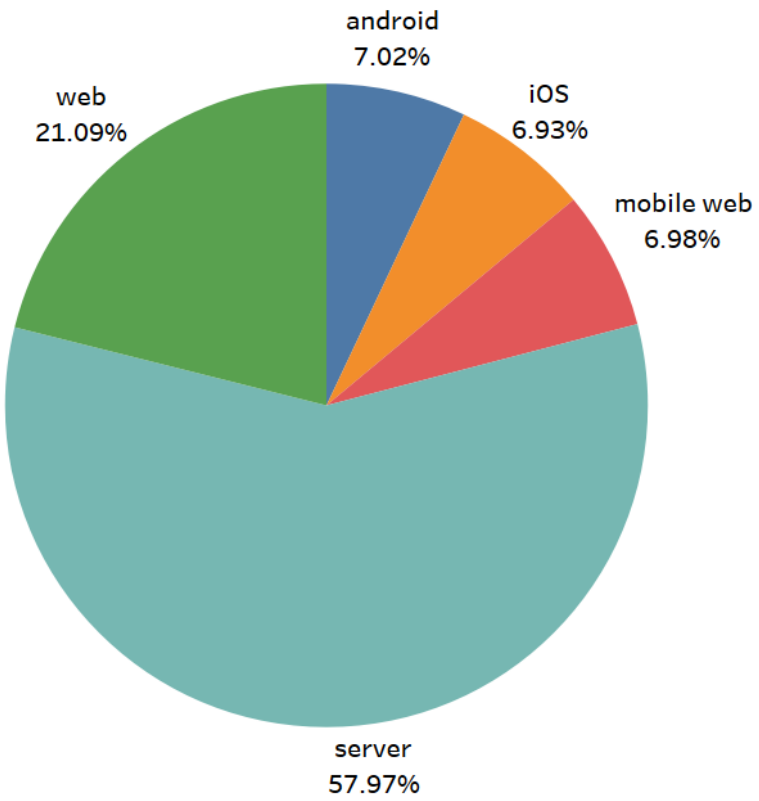
6. Our company want to run an advertisement campaign but for that, their data science team want you to analyze the customer engagement on various platforms like web browser, iOS etc. For that, they have provided you the user click stream. Please help them do the analysis so that they can plan in a better manner.

Query:

```
WITH user_platforms AS
    (SELECT DISTINCT
            COALESCE(parent_user_id, u.user_id) AS user_id,
            platform
    FROM
            click_stream.events AS e
            JOIN
            click_stream.users AS u ON e.user_id = u.user_id
    WHERE
            u.deleted_at IS NULL
    ORDER BY user_id),

platform_users AS
    (SELECT
            platform,
            COUNT(user_id) AS users
    FROM
          user_platforms
    GROUP BY platform),

platform_pct AS
    (SELECT
            platform,
            users,
            ROUND(users * 100
                    /(SELECT SUM(users) FROM platform_users), 1) AS users_pct
    FROM
          platform_users
    ORDER BY users_pct DESC)

SELECT
    *
FROM
    platform_pct;
```

Result:

Row	platform	users	users_pct
1	server	92906	58.0
2	web	33801	21.1
3	android	11255	7.0
4	mobile web	11192	7.0
5	iOS	11102	6.9



7. Customers Visiting Item Webpage.

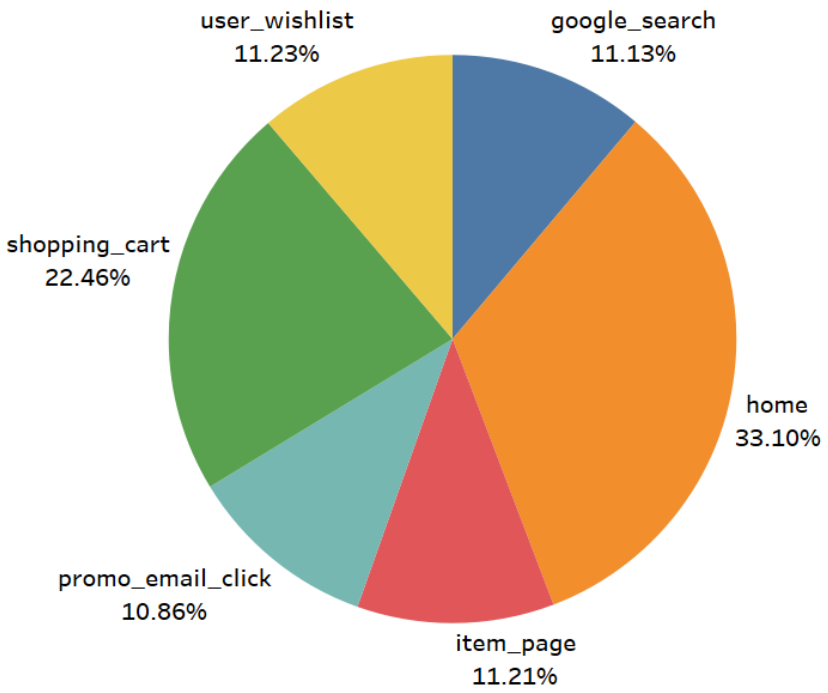
1. The data science team at our organization want to understand the relative contribution of various referrers such as google search, promo e-mail etc. which made customer visit the webpage and view a particular item.

Query:

```
WITH referrer_count AS
    (SELECT
            parameter_value AS referrer,
            COUNT(*) AS engagement
     FROM
            click_stream.events
    WHERE
            parameter_name = 'referrer'
    GROUP BY parameter_value),

referrer_pct AS
    (SELECT
             referrer,
             ROUND(engagement * 100

    / (SELECT SUM(engagement) FROM referrer_count), 1) AS referrer_engagement_pct
    FROM
           referrer_count
    ORDER BY referrer_engagement_pct DESC)

SELECT
    *
  FROM
    referrer_pct ;
```

Result:

Row	referrer	referrer_engagement_pct
1	home	33.1
2	shopping_cart	22.5
3	item_page	11.2
4	user_wishlist	11.2
5	google_search	11.1
6	promo_email_click	10.9



2. Conversion rate of view item event to an actual order in case of various referrers.

Query:

```
WITH view_item_events AS
    (SELECT DISTINCT
            event_id,
            event_time,
            user_id,
            MAX(CASE
                    WHEN parameter_name = 'item_id' THEN parameter_value
                END) AS item_id,
            MAX(CASE
                    WHEN parameter_name = 'referrer' THEN parameter_value
                END) AS referrer
    FROM
          click_stream.events
    WHERE
          event_name = 'view_item'
    GROUP BY event_id,
                event_time,
             user_id),

ordered_vs_not_ordered AS
    (SELECT
           v.referrer,
           CASE
              WHEN o.invoice_id IS NULL THEN 0
              ELSE 1
           END AS ordered_item_binary,
           COUNT(*) AS event_count
    FROM
          view_item_events AS v
             LEFT JOIN
          click_stream.orders AS o
                    ON CAST(v.user_id AS INT) = CAST(o.user_id AS INT)
                        AND CAST(v.item_id AS INT) = CAST(o.item_id AS INT)

AND DATETIME_DIFF(o.created_at, v.event_time, DAY) BETWEEN 0 AND 30
    GROUP BY v.referrer,
             ordered_item_binary
    ORDER BY v.referrer),
```

```
order_conversion_rate AS
    (SELECT
        referrer,
        ROUND(
            SUM(CASE
                WHEN ordered_item_binary = 1 THEN event_count
            END) / SUM(event_count), 2) AS order_conversion_rate
    FROM
        ordered_vs_not_ordered
    GROUP BY referrer)

SELECT
    *
FROM
    order_conversion_rate;
```

Result:

Row	referrer	order_conversion_rate
1	home	0.15
2	shopping_cart	0.15
3	google_search	0.14
4	promo_email_click	0.15
5	user_wishlist	0.14
6	item_page	0.15

8. The data science team at our company want you to analyze the click stream data of customer activities and send a reminder email to those customers who viewed an item recently on our website but haven't ordered yet.

Query:

```
WITH last_viewed_items AS
    (SELECT
                    temp_table.user_id,
                    MAX(temp_table.last_viewed_item_id) AS last_viewed_item_id,
                    MAX(temp_table.event_time) AS view_item_time
    FROM
                   (SELECT
            user_id,
            MAX(event_time) OVER(PARTITION BY user_id) AS event_time,

    LAST_VALUE(parameter_value) OVER(PARTITION BY user_id ORDER BY event_time)
            AS last_viewed_item_id
                  FROM
            click_stream.events
           WHERE
            event_name = 'view_item'
            AND parameter_name = 'item_id') AS temp_table
    GROUP BY temp_table.user_id
    ORDER BY temp_table.user_id),

user_reminder_email_data AS
    (SELECT
          v.user_id,
                   v.view_item_time,
            v.last_viewed_item_id,
            i.name AS item_name,
            u.email_address
    FROM
          last_viewed_items AS v
            LEFT JOIN
          click_stream.orders AS o ON v.user_id = o.user_id

AND CAST(v.last_viewed_item_id  AS INT) = o.item_id AND o.created_at >= v.view_item_time
                    JOIN
          click_stream.users AS u ON u.user_id = v.user_id
                  JOIN
          click_stream.items AS i ON i.item_id = CAST(v.last_viewed_item_id AS INT)
    WHERE
          o.invoice_id IS NULL)
```

```
SELECT
    *
FROM
    user_reminder_email_data;
```

Result:

Row	user_id	view_item_time	last_viewed_item_id	item_name	email_address
1	4	2013-09-04T04:47:43	3924	organic device how-to-manual	LMurphy1964@earthlink.com
2	8	2013-07-26T10:42:32	2430	gadget refill	Hanah_Schmidt1965@gmail.edu
3	9	2013-03-11T12:12:31	3953	organic widget wrapper	AanyaLee@gmail.info
4	10	2013-03-19T01:15:17	3964	reflective device storage_unit	SonyaChen@outlook.biz
5	21	2013-08-29T03:58:35	36	matte mechanism cleaner	Lucas_Murphy1957@gmail.org
6	23	2013-06-14T20:24:32	3445	prize-winning mechanism carry…	Mateo_Russell1969@inbox.com
7	24	2013-08-26T15:03:20	3348	rechargable gadget storage_unit	DiegoHuang@gmail.biz
8	34	2013-07-12T06:17:17	3605	aerodynamic module	Lisa_Petrov@gmail.org
9	41	2013-09-12T14:00:39	299	miniature widget how-to-manual	Mohamed_Gupta1970@inbox.i…
10	45	2013-09-02T03:06:24	3033	widget opener	IreneFourie@gmail.edu
11	50	2013-03-11T07:36:22	603	aerodynamic instrument carryi…	A_Martin2004@gmail.info

Results per page: 50 ▾ 1 – 50 of 62338

9. Imagine you are the part of the data science team of the company and the company wants to run some promotional campaigns. The CEO of the company asks you to determine the customer engagement at different times during the day so that campaign can be planned accordingly. Please share the insights.

Query:

```
WITH view_item_events_timeslots AS
    (SELECT
            CASE
              WHEN TIME (event_time) BETWEEN '00:00:00' AND '03:00:00' THEN '00:00 - 03:00'
              WHEN TIME (event_time) BETWEEN '03:00:01' AND '06:00:00' THEN '03:00 - 06:00'
              WHEN TIME (event_time) BETWEEN '06:00:01' AND '09:00:00' THEN '06:00 - 09:00'
              WHEN TIME (event_time) BETWEEN '09:00:01' AND '12:00:00' THEN '09:00 - 12:00'
              WHEN TIME (event_time) BETWEEN '12:00:01' AND '15:00:00' THEN '12:00 - 15:00'
              WHEN TIME (event_time) BETWEEN '15:00:01' AND '18:00:00' THEN '15:00 - 18:00'
              WHEN TIME (event_time) BETWEEN '18:00:01' AND '21:00:00' THEN '18:00 - 21:00'
              WHEN TIME (event_time) BETWEEN '21:00:01' AND '23:59:59' THEN '21:00 - 00:00'
          END AS timeslot,
          COUNT (DISTINCT event_id) AS view_item_events
    FROM
          click_stream.events
    WHERE
          event_name = 'view_item'
    GROUP BY timeslot),

timeslot_proportion AS
    (SELECT
                  timeslot,
            ROUND (view_item_events
                    / (SELECT SUM (view_item_events) FROM view_item_events_timeslots), 2)
        AS proportion_of_total
    FROM
            view_item_events_timeslots
    ORDER BY timeslot)

SELECT
    *
FROM
    timeslot_proportion;
```

Result:

Row	timeslot	proportion_of_total
1	00:00 - 03:00	0.13
2	03:00 - 06:00	0.12
3	06:00 - 09:00	0.13
4	09:00 - 12:00	0.12
5	12:00 - 15:00	0.13
6	15:00 - 18:00	0.13
7	18:00 - 21:00	0.12
8	21:00 - 00:00	0.13

10. We have conducted three A/B tests at different points of time and subjected few items to the control group and few to the treatment group. In these A/B tests, we ran a promotional scheme for the items in the treatment group and the items in the control group are as they were before. After that, we compared that how many of the items in the two groups (control & treatment) have been ordered.

1. For A/B Test 1 in the item_test_assignments table, conclude the results and tell whether the effect is statistically significant or not ?

Query:

```
WITH item_test_1_subjects AS
    (SELECT
            *
     FROM
            click_stream.item_test_assignments
      WHERE
          test_number = 'item_test_1'),

items_ordered_during_item_test_1 AS
    (SELECT DISTINCT
          item_id
     FROM
            click_stream.orders
    WHERE
            paid_at BETWEEN '2013-01-05 00:00:00' AND '2014-01-04 23:59:59'),

item_order_binary_test_1 AS
    (SELECT
            s.item_id,
            s.test_number,
            s.test_assignment,
            CASE
              WHEN o.item_id IS NOT NULL THEN 1
              ELSE 0
          END AS order_binary
     FROM
          item_test_1_subjects AS s
            LEFT JOIN
          items_ordered_during_item_test_1 AS o ON s.item_id = o.item_id),
```

```
a_b_test_1_results AS
    (SELECT
            test_number,
            test_assignment,
            CASE
              WHEN test_assignment = 0 THEN 'control'
              WHEN test_assignment = 1 THEN 'treatment'
            END AS assigned_group,
            COUNT(*) AS group_size,
            SUM(order_binary) AS ordered
    FROM
            item_order_binary_test_1
    GROUP BY test_number, test_assignment)

SELECT
    *
FROM
    a_b_test_1_results;
```

Result:

Row	test_number	test_assignment	assigned_group	group_size	ordered
1	item_test_1	0	control	1112	493
2	item_test_1	1	treatment	1086	515

	Successes	Total	Success Rate	p-value	Improvement
Control	493	1,112	41% – 47% (44%)	—	—
Treatment	515	1,086	44% – 50% (47%)	0.15	-2.4% – 16% (7%)

Conclusion:

A relative lift of just -0.5 % - 1.1 % in treatment group with reference to the control group and p-Value of 0.44 clearly indicates that the results are statistically insignificant.

2. For A/B Test 2 in the item_test_assignments table, conclude the results and tell whether the effect is statistically significant or not ?

Query:

```
WITH item_test_2_subjects AS
    (SELECT
            *
     FROM
            click_stream.item_test_assignments
     WHERE
         test_number = 'item_test_2'),

items_ordered_during_item_test_2 AS
    (SELECT DISTINCT
            item_id
      FROM
            click_stream.orders
      WHERE
            paid_at BETWEEN '2015-03-14 00:00:00' AND '2016-03-13 23:59:59'),

item_order_binary_test_2 AS
    (SELECT
            s.item_id,
            s.test_number,
            s.test_assignment,
            CASE
                WHEN o.item_id IS NOT NULL THEN 1
                ELSE 0
            END AS order_binary
     FROM
            item_test_2_subjects AS s
              LEFT JOIN
            items_ordered_during_item_test_2 AS o ON s.item_id = o.item_id),
```

```
a_b_test_2_results AS
    (SELECT
            test_number,
            test_assignment,
            CASE
              WHEN test_assignment = 0 THEN 'control'
              WHEN test_assignment = 1 THEN 'treatment'
            END AS assigned_group,
            COUNT(*) AS group_size,
            SUM(order_binary) AS ordered
     FROM
            item_order_binary_test_2
     GROUP BY test_number, test_assignment)

SELECT
    *
FROM
    a_b_test_2_results;
```

Result:

Row	test_number	test_assignment	assigned_group	group_size	ordered
1	item_test_2	0	control	1130	1115
2	item_test_2	1	treatment	1068	1056

	Successes	Total	Success Rate	p-value	Improvement
Control	1,115	1,130	98% – 99% (99%)	—	—
Treatment	1,056	1,068	98% – 99% (99%)	0.67	-0.8% – 1.2% (0.21%)

Conclusion:

A relative lift of just -0.5 % - 1.1 % in treatment group with reference to the control group and p-Value of 0.44 clearly indicates that the results are statistically insignificant.

3. For A/B Test 3 in the item_test_assignments table, conclude the results and tell whether the effect is statistically significant or not ?

Query:

```
WITH item_test_3_subjects AS
    (SELECT
            *
     FROM
            click_stream.item_test_assignments
     WHERE
        test_number = 'item_test_3'),

items_ordered_during_item_test_3 AS
    (SELECT DISTINCT
            item_id
     FROM
            click_stream.orders
    WHERE
            paid_at BETWEEN '2016-01-07 00:00:00' AND '2017-01-06 23:59:59'),

item_order_binary_test_3 AS
    (SELECT
            s.item_id,
            s.test_number,
            s.test_assignment,
            CASE
                WHEN o.item_id IS NOT NULL THEN 1
                ELSE 0
            END AS order_binary
     FROM
            item_test_3_subjects AS s
              LEFT JOIN
            items_ordered_during_item_test_3 AS o ON s.item_id = o.item_id),
```

```
a_b_test_3_results AS
    (SELECT
            test_number,
            test_assignment,
            CASE
              WHEN test_assignment = 0 THEN 'control'
              WHEN test_assignment = 1 THEN 'treatment'
            END AS assigned_group,
            COUNT(*) AS group_size,
            SUM(order_binary) AS ordered
  FROM
            item_order_binary_test_3
  GROUP BY test_number, test_assignment)

SELECT
    *
FROM
    a_b_test_3_results;
```

Result:

Row	test_number	test_assignment	assigned_group	group_size	ordered
1	item_test_3	0	control	1075	1065
2	item_test_3	1	treatment	1123	1116

	Successes	Total	Success Rate	p-value	Improvement
Control	1,065	1,075	98% – 100% (99%)	—	—
Treatment	1,116	1,123	99% – 100% (99%)	0.44	-0.5% – 1.1% (0.31%)

Conclusion:

A relative lift of just -0.5 % - 1.1 % in treatment group with reference to the control group and p-Value of 0.44 clearly indicates that the results are statistically insignificant.