A. BigQuery – Link to Google Analytics

How to set up BigQuery Export?

There are four main steps that we need to take to set up BigQuery Export. You may find details below:

- Step 1: Create a Google API Console Project and enable BigQuery

1. Log in to the Google APIs Console.
2. Create a Google APIs Console project.

   You can create a new project or select an existing project.
3. Navigate to the APIs table.

   Open the *Navigation* menu in the top-left corner, click **APIs & Services**, then click **Library**.
4. Activate BigQuery.

   Under *Google Cloud APIs*, click **BigQuery API**. On the following page, click **Enable**.
5. If prompted, review and agree to the Terms of Service.

- Step 2: Add a billing account using Google APIs Console

1. Ensure *Billing* is enabled for your project.

   If you do not have *Billing* enabled for your project, open the *Navigation* menu in the top-left corner, then click **Billing**.
2. If prompted, create a billing account.

   A billing account is necessary to apply billing to a project. A single billing account may be shared across multiple projects. Follow the steps in the API console to create your billing account.
3. Accept the free trial if it's available.

   If you are offered a free trial, it is safe to accept it; however, you must also enter billing details in order for BigQuery to continue receiving exported data once the free trial is over.
4. Validate *Billing* enablement.

   Open your project at https://console.cloud.google.com/bigquery, and try to create a data set in the project. Click the **blue arrow** next to project name, then click **Create data set**. If you can create the data set, billing is setup correctly. If there are any errors, make sure billing is enabled.
5. Add the service account to your project.

   Add analytics-processing-dev@system.gserviceaccount.com as a member of the project, and ensure that permission at the *project level* is set to **Editor** (as opposed to BigQuery Data Editor). The Editor role is required in order to export data from Analytics to BigQuery.

"BigQuery has a free tier which allows you to query 1 terabyte per month and store 10 gigabytes." Additional information regarding the payment may be accessed via the provided link.

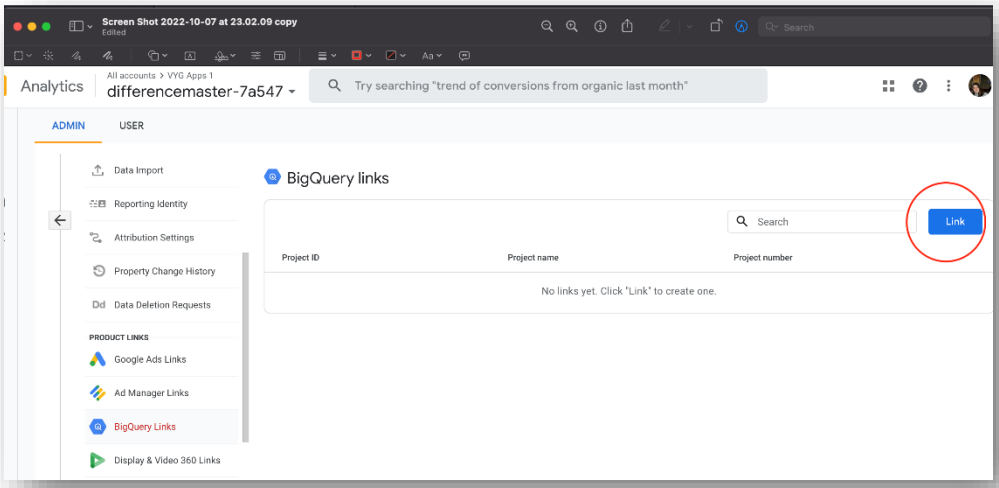
- Step 3: Link the Google Analytics account with the BigQuery

After you complete the first two steps, you can enable BigQuery Export from Analytics Admin.

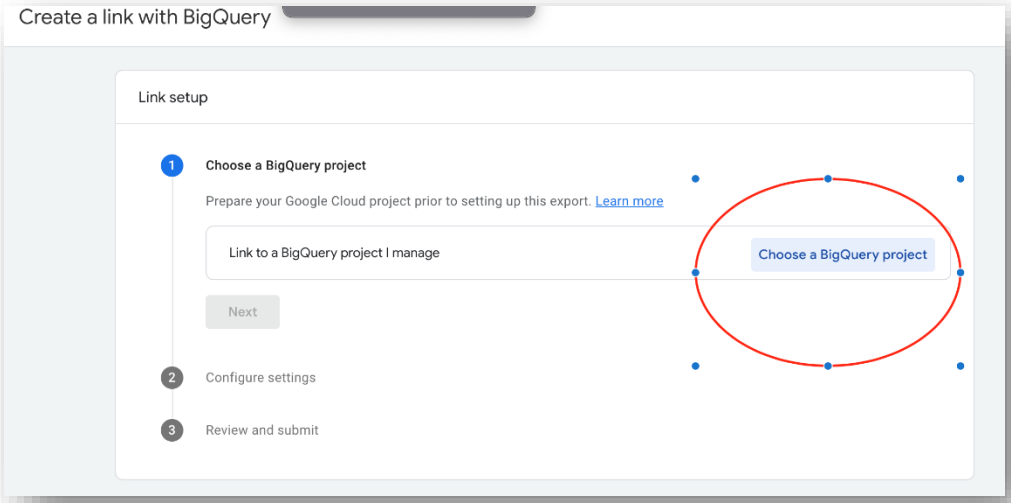
1. Sign in to Google Analytics. Use an email address that has *OWNER* access to the BigQuery project, and also has the *Editor role* for the Analytics property that includes the view you want to link.
2. Click **Admin**, and navigate to the Analytics 360 **property** that contains the view you want to link.
3. In the *PROPERTY* column, click **All Products**, then click **Link BigQuery**.
4. Enter your BigQuery project number or ID. (Learn more about how to locate your project number and ID.)
5. Select the view you want to link.
6. Optional: Select the email addresses at which you would like to receive daily success and/or failure notifications.
7. Optional: Select your current-day export preference. Note that the continuous export option uses the Cloud streaming service, which includes an additional $0.05 charge per GB sent.
8. Confirm that you have enabled billing and applied any relevant credits or coupons to your project.
9. Click **Save**.
10. If you need to stop the export, return to this page, and click **Adjust Link** in the *BigQuery* section.

Details of the STEP 3

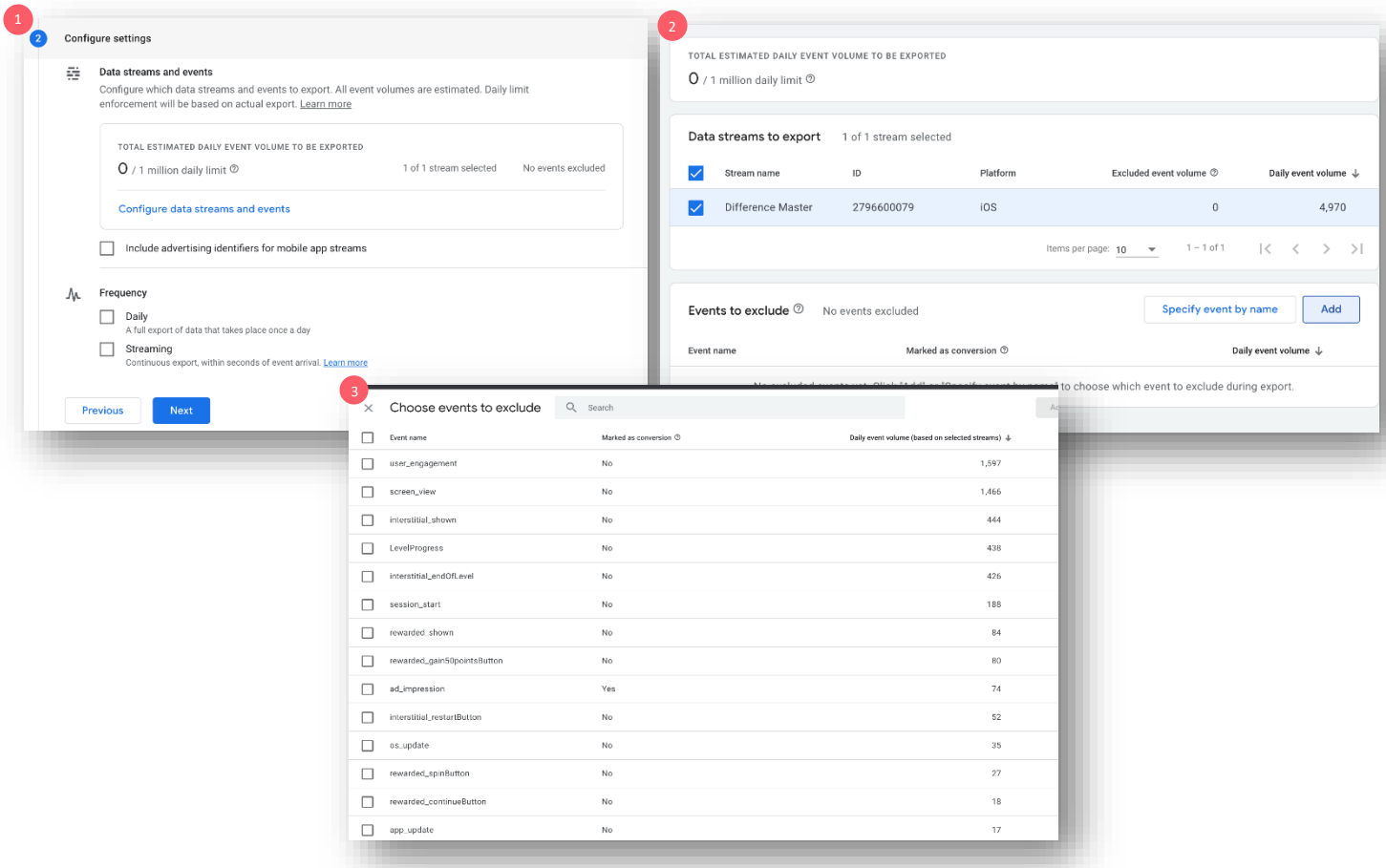
(1) Open Google Analytics → admin→ property → product links → BigQuery Links



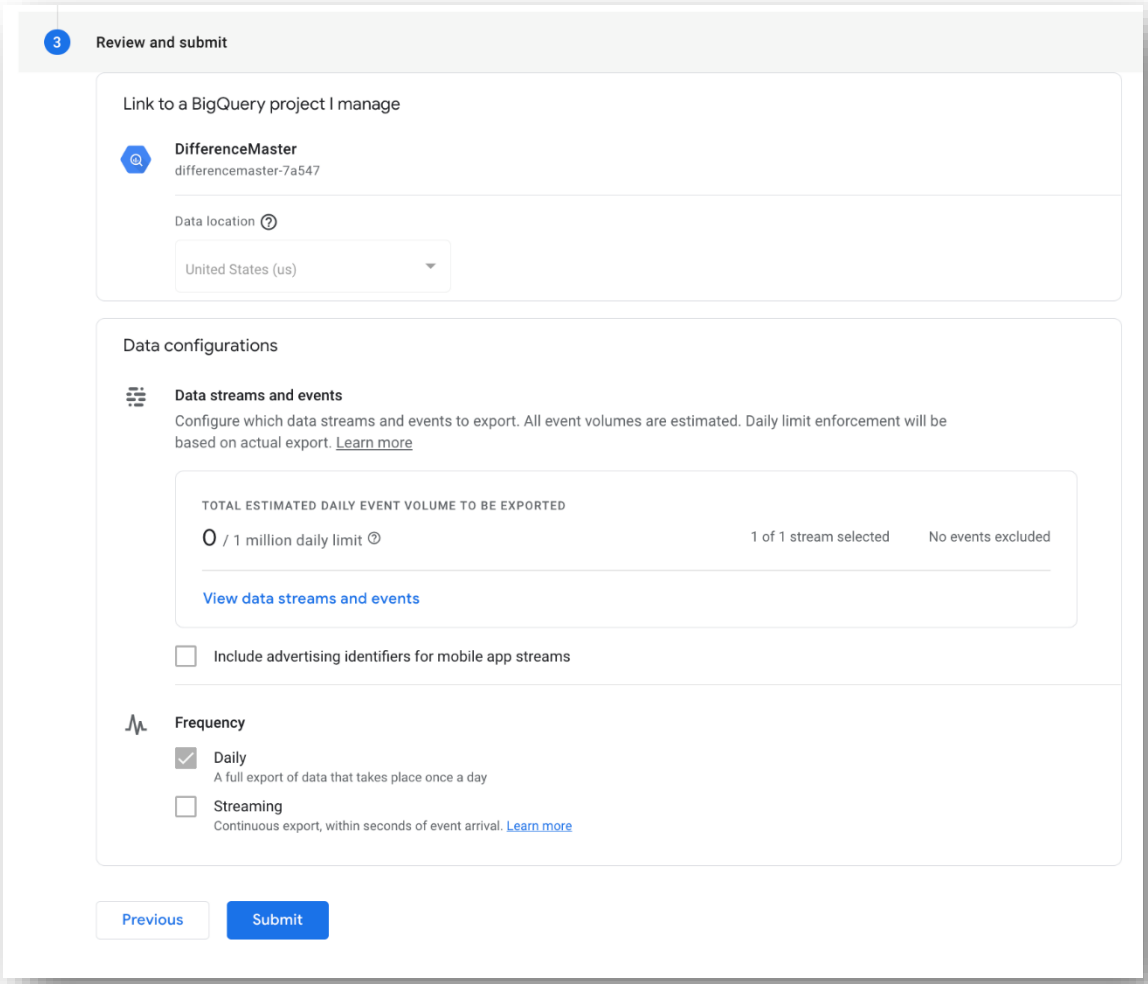
(2) Choose a BigQuery Project
- Choose data location



(3) Configure Settings
- Exclude events that you do not need to collect data (→ configure data streams and events)
- Click to Add—> Choose events that you want to exclude
- Choose frequency of data → daily or streaming

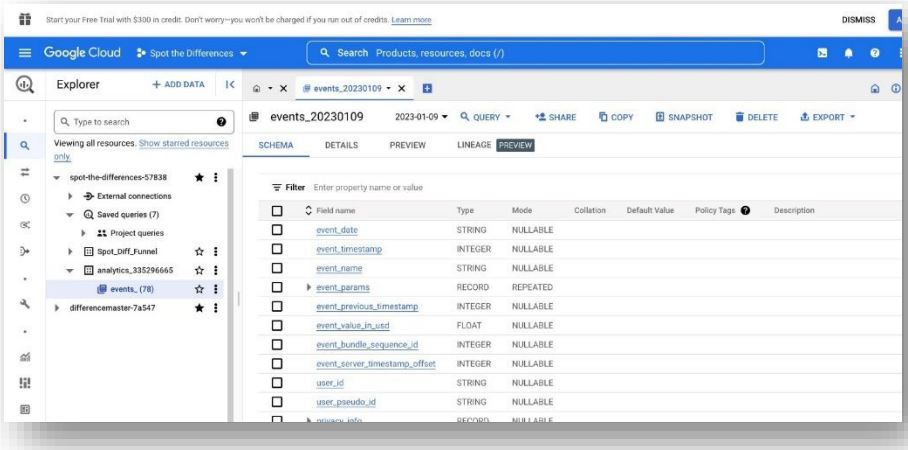


(4) Final step: Review your selections and submit



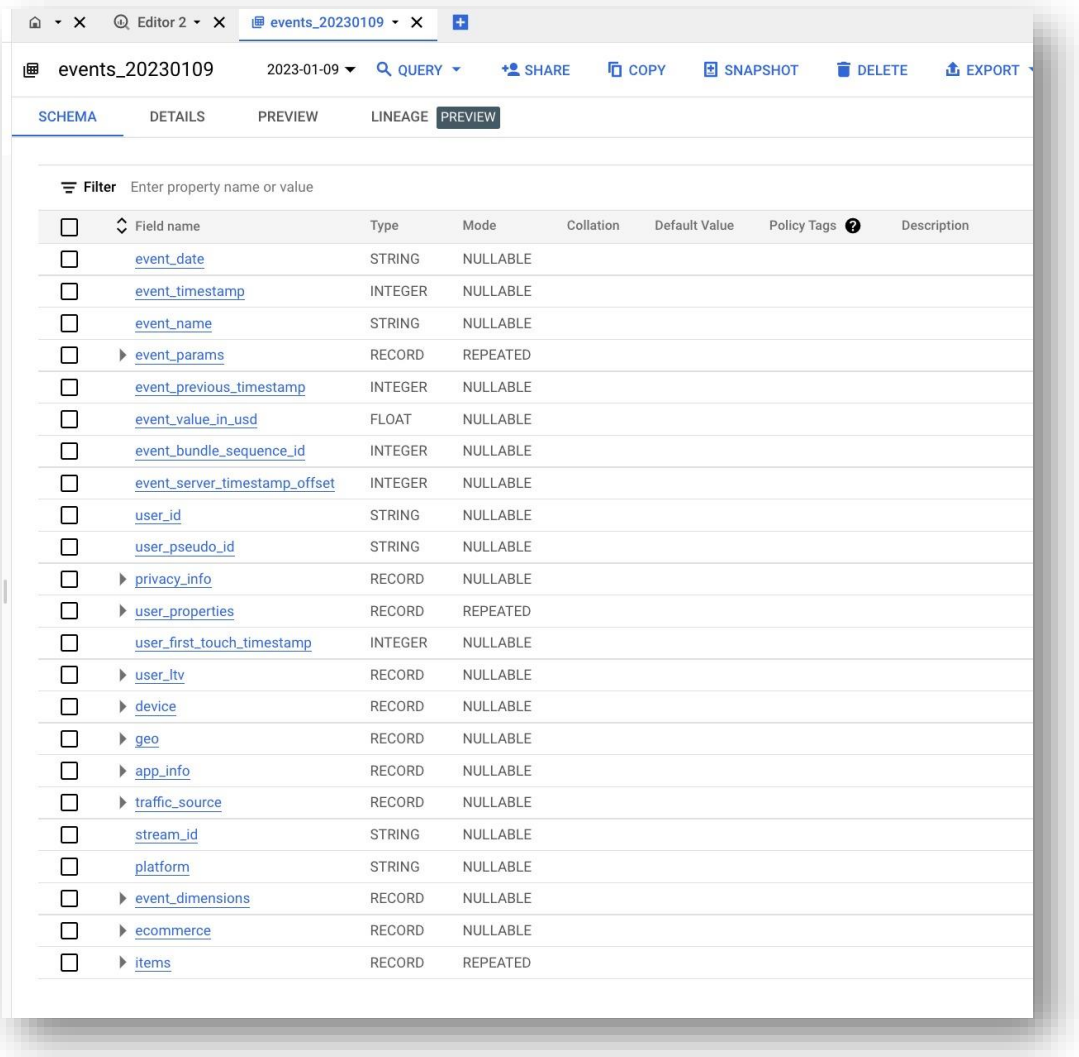
- Step 4: Start to write queries using BigQuery tool
  - Google Cloud Console → BigQuery → SQL workspace

If you choose daily data frequency at the previous step than BigQuery will send collected data once a day as a new event table. A table containing event data for January 9th, 2023, will be named "events_20230109" since the BigQuery naming convention includes the date in the format of "YYYYMMDD"



BigQuery Export Schema

- Daily tables will utilize the following data schema to store event data:



Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
event_date	STRING	NULLABLE				
event_timestamp	INTEGER	NULLABLE				
event_name	STRING	NULLABLE				
event_params	RECORD	REPEATED				
event_previous_timestamp	INTEGER	NULLABLE				
event_value_in_usd	FLOAT	NULLABLE				
event_bundle_sequence_id	INTEGER	NULLABLE				
event_server_timestamp_offset	INTEGER	NULLABLE				
user_id	STRING	NULLABLE				
user_pseudo_id	STRING	NULLABLE				
privacy_info	RECORD	NULLABLE				
user_properties	RECORD	REPEATED				
user_first_touch_timestamp	INTEGER	NULLABLE				
user_ltv	RECORD	NULLABLE				
device	RECORD	NULLABLE				
geo	RECORD	NULLABLE				
app_info	RECORD	NULLABLE				
traffic_source	RECORD	NULLABLE				
stream_id	STRING	NULLABLE				
platform	STRING	NULLABLE				
event_dimensions	RECORD	NULLABLE				
ecommerce	RECORD	NULLABLE				
items	RECORD	REPEATED				

- the most common metrics that are used in other stages

Variable name	Type	Definition
event_date	string	The date on which the event was logged (YYYYMMDD format in the registered timezone of your app).
event_timestamp	integer	The time (in microseconds, UTC) at which the event was logged on the client.
event_name	string	The name of the event.
event_params	record	A repeated record of the parameters associated with this event.
user_pseudo_id	string	The pseudonymous id (e.g., app instance ID) for the user.
geo.country	string	The country from which events were reported, based on IP address.
app_info.version	string	The app's versionName (Android) or short bundle version.

Additional information regarding the data schema may be accessed via the provided link.

B. Useful SQL queries to measure primary game metrics

Number of daily new user by country and app version

```
SELECT
     geo.country as country,
     app_info.version as app_version,
     Count(distinct user_pseudo_id) as number_user
FROM
      `dataset_id.events_*`,
       UNNEST(event_params) as p
WHERE
      event_name ='first_open'
      AND p.key='firebase_conversion'
      AND p.value.int_value = 1
      AND _TABLE_SUFFIX BETWEEN '20221024'
      AND '20221030'
 GROUP BY 1, 2
```

Utilized variables : event_name ; event_params.key ; event_params.value.int_value ; geo.country ; app_info.version ; user_pseudo_id

Level based churn rate by country and app version

```
CREATE OR REPLACE TABLE `dataset_id.Level`
OPTIONS(
  description="Detailed level progress info"
) AS
  SELECT
    Userid,
    CAST(t.FirstOpen_date AS DATE FORMAT 'YYYYMMDD') as open_date,
    t.FirstOpen_time,
    Event_date,
    Event_timestamp,
    Event_name,
    Key,
    Value,
    t.Country,
    t.App_Version
  FROM
    (SELECT
    DISTINCT(user_pseudo_id)AS Userid,
    event_date AS FirstOpen_date,
    event_timestamp AS FirstOpen_time,
    geo.country AS Country,
    app_info.version as App_Version
  FROM
    `dataset_id.events_*`,
    UNNEST(event_params) AS p
  WHERE
    event_name ='first_open'
    AND p.key='firebase_conversion'
    AND p.value.int_value = 1
    AND _TABLE_SUFFIX BETWEEN '20221119'
    AND '20230104' ) AS t
  LEFT JOIN (
    SELECT
      user_pseudo_id,
      Event_date,
      Event_timestamp,
      Event_name,
      p.key AS Key,
      p.value.string_value AS Value
    FROM
      `dataset_id.events_*`,
      UNNEST(event_params) AS p
    WHERE
      event_name ='LevelProgress'
      AND p.key='levelCompleted'
      AND _TABLE_SUFFIX BETWEEN '20221119'
      AND '20230104' ) AS l
  ON
    t.Userid = l.user_pseudo_id
  ORDER BY
    Userid,
    Value ;
```


melizturkileri@sabanciuniv.edu

```
INSERT `dataset_id.Level`(Userid, open_date, FirstOpen_time,Event_date,Event_time
stamp,Event_name,Key,Value,Country,App_Version)
SELECT
    DISTINCT(Userid)AS Userid,
    open_date,
    FirstOpen_time,
    "Same as Open Date" as Event_date,
    FirstOpen_time as Event_timestamp,
    "First Open" AS Event_name,
    "First Open" AS Key,
    "Level_0" AS Value,
    Country,
    App_Version
  FROM
    `dataset_id.Level`
```

Utilized variables : event_name ; event_params.key ; event_params.value.int_value ; geo.country ; app_info.version ; user_pseudo_id

Day0 impressions for interstitital and rewarded

```
  /*

-- run when you first created the table
CREATE OR REPLACE TABLE `spot-the-differences-57838.Spot_Diff_Funnel.Ads`
OPTIONS(
  description="Interstitials and Rewarded info"
) AS
*/

-- delete last five days included in the current table

DELETE `dataset_id.Ads`
WHERE cohort between '2022-12-24' and  '2023-01-02';
```

```
-- expand the cohort interval (include latest release)
INSERT `dataset_id.Ads` (country, app_Version, cohort, unique_user, interstitial_
shown, rewarded_shown)
SELECT
m.country,
m.app_version,
CAST(m.firstOpen_date AS DATE FORMAT 'YYYYMMDD') as cohort,
Count(distinct m.userid) as unique_user ,
SUM(n.intersitital) as interstitinal_shown,
SUM(n.rewarded) as rewarded_shown
FROM
(SELECT
    DISTINCT(user_pseudo_id)AS userid,
    event_date AS firstOpen_date,
    TIMESTAMP_MICROS(event_timestamp) AS firstOpen_time,
    geo.country AS country,
    app_info.version as app_version
  FROM
    `dataset_id.events_*`,
    UNNEST(event_params) AS p
  WHERE
    event_name ='first_open'
    AND p.key='firebase_conversion'
    AND p.value.int_value = 1
    AND _TABLE_SUFFIX BETWEEN '20221224'
    AND '20230104'
   ) AS m
  LEFT JOIN
(SELECT
   distinct (userid) AS userid,
   t.country,
   t.app_Version,
   CAST(t.firstOpen_date AS DATE FORMAT 'YYYYMMDD') as open_date,
   l.event_name,
   Count(Value),
   CASE When  l.event_name='interstitial_shown' then Count(value)
        ELSE 0
        END intersitital,
    CASE When  l.event_name='rewarded_shown' then Count(value)
        ELSE 0
        END rewarded,
```

```
FROM
    (SELECT
    DISTINCT(user_pseudo_id)AS userid,
    event_date AS firstOpen_date,
    TIMESTAMP_MICROS(event_timestamp) AS firstOpen_time,
    geo.country AS country,
    app_info.version as app_Version
  FROM
    `dataset_id.events_*`,
    UNNEST(event_params) AS p
  WHERE
    event_name ='first_open'
    AND p.key='firebase_conversion'
    AND p.value.int_value = 1
    AND _TABLE_SUFFIX BETWEEN '20221224'
    AND '20230104'
   ) AS t
  LEFT JOIN (
    SELECT
      user_pseudo_id,
      event_date,
      TIMESTAMP_MICROS(event_timestamp) AS event_timestamp,
      event_name,
      p.key AS key,
      p.value.int_value AS value
    FROM
      `dataset_id.events_*`,
      UNNEST(event_params) AS p
    WHERE
      (event_name ='interstitial_shown' or event_name ="rewarded_shown")
      AND p.key='engaged_session_event'
      AND p.value.int_value = 1
      AND _TABLE_SUFFIX BETWEEN '20221224'
      AND '20230104'  ) AS l
  ON
    t.Userid = l.user_pseudo_id
  WHERE TIMESTAMP_DIFF( l.event_timestamp,  t.firstOpen_time, HOUR)<=24
GROUP BY 1,2,3,4,5
ORDER BY 1,2,3,4,5) AS n
ON m.userid = n.userid
GROUP BY 1,2,3
ORDER BY 1,2,3
```


melizturkileri@sabanciuniv.edu

A/B Test Results – Level based churn rate by country and app version

```
/*
CREATE OR REPLACE TABLE ` dataset_id.AB`
OPTIONS(
  description="AB Test results"
) AS
*/

DELETE `dataset_id.AB`
WHERE Experiment_name = "LevelOrderABTest3";

INSERT `dataset_id.AB` (Userid, Experiment_name, Exper
iment_variant, open_date, FirstOpen_time, Event_date,E
vent_timestamp,Event_name,Key,Value,Country,App_Versio
n)
  SELECT
    Userid,
    l.experimentName AS Experiment_name,
    l.experimentVariant AS Experiment_variant,
    CAST(t.FirstOpen_date AS DATE FORMAT 'YYYYMMDD') a
s open_date,
    t.FirstOpen_time,
    Event_date,
    Event_timestamp,
    Event_name,
    Key,
    Value,
    t.Country,
    t.App_Version
  FROM
    (SELECT
    DISTINCT(user_pseudo_id)AS Userid,
    event_date AS FirstOpen_date,
    event_timestamp AS FirstOpen_time,
    geo.country AS Country,
    app_info.version as App_Version
  FROM
    ` dataset_id.events_*`,
    UNNEST(event_params) AS p
  WHERE
    event_name ='first_open'
    AND p.key='firebase_conversion'
    AND p.value.int_value = 1
    AND _TABLE_SUFFIX BETWEEN '20230102'
    AND '20230104' ) AS t
  INNER JOIN (
```

```
SELECT
      "LevelOrderABTest3" AS experimentName,
      user_pseudo_id,
      Event_date,
      Event_timestamp,
      Event_name,
      CASE userProperty.value.string_value WHEN "0" THEN
"Baseline"WHEN "1" THEN "Variant A" END AS experimentVari
ant,
      p.key AS Key,
      p.value.string_value AS Value
FROM
      ` dataset_id.events_*`,
      UNNEST(event_params) AS p,
      UNNEST(user_properties) AS userProperty
    WHERE
      event_name ='LevelProgress'
      AND p.key='levelCompleted'
      AND userProperty.key = "firebase_exp_4"
      AND _TABLE_SUFFIX BETWEEN '20230102'
      AND '20230104') AS l
  ON
    t.Userid = l.user_pseudo_id
  WHERE  l.experimentName  is not null
  ORDER BY
    Userid,
    Value
;
```


melizturkileri@sabanciuniv.edu

```
INSERT `dataset_id.AB`(Userid, Experiment_name, Experiment_variant, open_date, First
Open_time,Event_date,Event_timestamp,Event_name,Key,Value,Country,App_Version)
SELECT
    DISTINCT (t.Userid ) AS Userid,
    Experiment_name,
    Experiment_variant,
    CAST(event_date AS DATE FORMAT 'YYYYMMDD') AS open_date,
    event_timestamp AS FirstOpen_time,
    event_date AS Event_date,
    event_timestamp AS Event_timestamp,
    "First Open" AS Event_name,
    "First Open" AS Key,
    "Level_0" AS Value,
    Country,
    App_Version
  FROM (
    SELECT
    user_pseudo_id AS Userid,
    CAST(event_date AS DATE FORMAT 'YYYYMMDD') AS open_date,
    event_timestamp AS FirstOpen_time,
    event_date AS Event_date,
    event_timestamp AS Event_timestamp,
    geo.country AS Country,
    app_info.version as App_Version
    FROM
    ` dataset_id.events_*`,
    UNNEST(event_params) AS p
  WHERE
    event_name ='first_open'
    AND p.key='firebase_conversion'
    AND p.value.int_value = 1
    AND _TABLE_SUFFIX BETWEEN '20230102'
    AND '20230104') AS t
  INNER JOIN
  (SELECT
  DISTINCT(user_pseudo_id) AS Userid,
  "LevelOrderABTest3" AS Experiment_name,
  CASE userProperty.value.string_value WHEN "0" THEN "Baseline"WHEN "1" THEN "Varian
t A" END AS Experiment_variant,
  FROM ` dataset_id.events_*`,
  UNNEST(user_properties) AS userProperty
  WHERE
  (_TABLE_SUFFIX between '20230102' AND '20230104')
  AND userProperty.key = "firebase_exp_4" ) AS ab
  ON t.Userid = ab.Userid
```