Example Query #1

Query Goal: This queries purpose is to return parameters based on requirements from the internal customer. This code is then automated and sent to the proper distribution groups. This query makes use of a number of window functions to return the `Cumulative` (Running) `eCPI`, `Cumulative` (Running) `Open Rate`, and `Cumulative` (Running) `Uninstall Rate`.

Code:

```
SELECT date(event_timestamp) as "Date",
carrier as "Carrier",
site_id,
site_name,
sum(preloads) as "Preloads",
sum(spend) as "Spend",
sum(installs) as "Opens",
sum(uninstalls) as "Uninstalls",
CASE WHEN sum(installs) > 0 THEN sum(spend)/sum(installs) ELSE 0 END AS "eCPI",

(CAST(sum(sum(spend)) OVER (PARTITION BY carrier ORDER BY date rows between unbounded preceding and current
row) as FLOAT) /
CAST(CASE WHEN sum(installs) > 0 THEN sum(sum(installs)) OVER (PARTITION BY carrier ORDER BY date rows
between unbounded preceding and current row) END as FLOAT)) AS "Cumulative eCPI",

CASE WHEN sum(preloads) > 0 THEN CAST(sum(installs) as FLOAT)/CAST(sum(preloads) as FLOAT) ELSE 0 END AS
"Open Rate",

(CAST(sum(sum(installs)) OVER (PARTITION BY carrier ORDER BY date rows between unbounded preceding and
current row) as FLOAT) /
CAST(CASE WHEN sum(preloads) > 0 THEN sum(sum(preloads)) OVER (PARTITION BY carrier ORDER BY date rows
between unbounded preceding and current row) END as FLOAT)) AS "Cumulative Open Rate",

CASE WHEN sum(preloads) > 0 THEN cast(sum(uninstalls) as FLOAT)/CAST(sum(preloads) as FLOAT) ELSE 0 END AS
"Uninstall Rate",

(CAST(sum(sum(uninstalls)) OVER (PARTITION BY carrier ORDER BY date rows between unbounded preceding and
current row) as FLOAT) /
CAST(CASE WHEN sum(preloads) > 0 THEN sum(sum(preloads)) OVER (PARTITION BY carrier ORDER BY date rows
between unbounded preceding and current row) END as FLOAT)) AS "Cumulative Uninstall Rate"

FROM

(SELECT date(ct.timestamp) as event_timestamp,
pi.partner_name AS carrier,
ct.site_id,
s.site_name,
count(distinct preloads) as preloads,
sum(price) as spend,
0 as Installs,
0 as Uninstalls

FROM campaign_tracking a

JOIN site s ON ct.[JOINID] = s.[JOINID]
JOIN partner_information pi ON s.[JOINID] = pi.[JOINID]

WHERE package_name = '[PACKAGENAME]'
AND DATE (ct.timestamp) >= date('11-01-2017')
AND DATE (ct.timestamp) != date(current_date)
and pi.partner_id IN ([PARTNERID1],[PARTNERID2])
```

```
GROUP BY 1,2,3,4

UNION

SELECT date(ct.timestamp) as event_timestamp, pi.partner_name AS carrier, ct.site_id, s.site_name, 0 as
preloads, 0 as spend,
sum(CASE when code = [EVENTCODE1] then 1 else 0 end) as installs,
sum(CASE when code = [EVENTCODE2] then 1 else 0 end) as uninstalls

FROM campaign_tracking ct
JOIN site s ON s.[JOINID] = ct.[JOINID]
JOIN partner_information pi ON s.[JOINID] = pi.[JOINID]
full outer JOIN (
       SELECT date(te.eventdate) as date, p.[JOINID]||'_'||t.[JOINID] AS systemid, e.code
       FROM process p
       JOIN transactions t ON p.processid = t.processid AND t.[JOINID] = p.[JOINID]
       JOIN transactions_events te ON t.transactionid = te.transactionid AND te.[JOINID] = p.[JOINID]
       JOIN events e ON e.eventId=te.eventId and e.[JOINID] = te.[JOINID]
       JOIN carrier c ON c.[JOINID] = p.[JOINID]
       JOIN carrier_app ca ON ca.carrieraid = t.carrierappid AND ca.[JOINID] = t.[JOINID]
       JOIN app a ON a.aid = ca.aid AND a.[JOINID] = ca.[JOINID]
       WHERE DATE (te.eventdate) >= date('01-08-2018')
       AND DATE (te.eventdate) != date(current_date)
       and e.code in ([EVENTCODE1],[EVENTCODE2])
       AND a.packagename = '[PACKAGENAME]'
       and ca.rs_carrier_id in ([CARRIERID1],[CARRIERID2])

) AS b ON b.systemid = ct.uniqueid
Where b.systemid IS NOT NULL
and package_name = '[PACKAGENAME]'
and pi.partner_id IN ([PARTNERID1],[PARTNERID2])
AND DATE (ct.timestamp) >= date('11-01-2017')
AND DATE (ct.timestamp) != date(current_date)
GROUP BY 1,2,3,4) a

GROUP BY 1,2,3,4
order by 2,1,3,4
```

Example Query #2

Query Goal: The purpose of this code is to track loyal users. The results need to include a count of users and those that are “loyal” (defined as having opened the app 3 times within the first 30 days). FROM there those numbers are used to provide KPI to both internal and external customers. A number of inner queries are required with this database to ensure accurate data, and remove the possibility of data duplication.

Code:

```
SELECT
a.preloadtime,
a.partner_name as "Carrier",
a.site_id,
a.site_name,
a.campaign_id,
a.campaign_name,
CASE WHEN a.campaign_name LIKE '%Tier 1%' THEN 'Tier 1'
       WHEN a.campaign_name LIKE '%Tier 2%' THEN 'Tier 2'
       WHEN a.campaign_name LIKE '%Tier 3%' THEN 'Tier 3'
       ELSE 'None' END AS "Tier",
sum(a.preloads) as preloads,
sum(a.installs) as installs,
sum(a.spend) as spend,
sum(b.app_opened) as "app_opened",
count(c.app_user) as "loyalusers",
CASE WHEN sum(a.preloads) > 0 THEN convert(float,sum(a.spend))/convert(float,sum(a.preloads)) END AS "Cost
Per Preload",
CASE WHEN sum(a.installs) > 0 THEN convert(float,sum(a.spend))/convert(float,sum(a.installs)) END AS "Cost
Per Install",
CASE WHEN sum(a.preloads) > 0 THEN convert(float,sum(a.installs))/convert(float,sum(a.preloads)) END AS
"Conversion Rate",
CASE WHEN count(c.app_user) > 0 THEN sum(a.spend)/count(distinct c.app_user) END AS "Cost Per Loyal User",
CASE WHEN sum(a.installs) > 0 THEN convert(float,count(c.app_user))/convert(float,sum(a.installs)) END AS
"Loyal Users per Install",
SUM(b.af_app_opened) AS "Sessions",
CASE WHEN sum(a.installs) > 0 THEN convert(float,sum(b.af_app_opened))/convert(float,sum(a.installs)) END AS
"Sessions per Install"


FROM (SELECT ct.preload_user, date(ct.timestamp) as preloadtime, ct2.timestamp as installtime,
ct.campaign_id, c.campaign_name,
ct.site_id, pi.partner_name, s.site_name,
1 as preloads, CASE WHEN ct2.timestamp IS NOT NULL THEN 1 ELSE 0 END as installs,
sum(ct.price) as spend
       FROM campaign_tracking ct
       left JOIN campaign_tracking2 ct2 ON ct.uniqueid = ct2.uniqueid and ct.[JOINID] = ct2.[JOINID] and
ct.[JOINID] = ct2.[JOINID]
       JOIN campaign c ON ct.[JOINID] = c.[JOINID]
       JOIN site s ON ct.[JOINID] = s.[JOINID]
       JOIN partner_information pi ON s.[JOINID] = pi.[JOINID]
       where ct.campaign_id IN ([CAMPAIGNID1],[CAMPAIGNID2],[CAMPAIGNID3])
       and date(ct.timestamp) >= date('2018-01-01')
       and date(ct.timestamp) != date(current_date)
       GROUP BY 1,2,3,4,5,6,7,8 ,9
) a

left JOIN (
              SELECT distinct pd.app_user, pd.campaign_id, pd.site_id,
              SUM(
              CASE
                     WHEN event = 'app_opened'
                     THEN 1
```

```
                    ELSE 0
                END) AS "app_opened"
                FROM postinstall_data pd
                WHERE pd.campaign_id IN ([CAMPAIGNID1],[CAMPAIGNID2],[CAMPAIGNID3])
                and date(event_timestamp) >= date('2018-01-01')
                AND date(event_timestamp) != date(current_date)
                GROUP BY 1,2,3
                ) b ON a.preload_user = b.app_user and a.[JOINID] = b.[JOINID] and a.[JOINID]=b.[JOINID]

left JOIN (
        SELECT app_user, campaign_id, site_id, sum(preload_user_id)
        FROM(
                SELECT distinct pd.app_user,
                pd.campaign_id,
                pd.site_id,
                ct2.timestamp,
                pd.event_timestamp,
                datediff(d, ct2.timestamp, CAST(pd.event_timestamp as TIMESTAMP)),
                count(preload_user_id) as preload_user_id

                FROM campaign_tracking2 ct2

                JOIN postinstall_data pd ON ct2.preload_user = pd.app_user and pd.[JOINID] = ct2.[JOINID] and
pd.[JOINID] = ct2.[JOINID]

                where campaign_id IN ([CAMPAIGNID1],[CAMPAIGNID2],[CAMPAIGNID3])
                and event = 'app_opened'
                GROUP BY 1,2,3,4,5
                having datediff(d, ct2.timestamp, CAST(pd.event_timestamp as TIMESTAMP)) < 31
                )
                GROUP BY 1,2,3
        having sum(app_user) >= 3
        ) c ON a.preload_user = c.app_user and a.[JOINID] = c.[JOINID] and a.[JOINID]=c.[JOINID]
GROUP BY 1,2,3,4,5,6
order by 1,2,3,4,5,6
```

*variable fields, table names, and joins obscured