# TRAINITY

PROJECT

**Operation Analytics and Investigating Metric Spike**

## PROJECT DESCRIPTION:

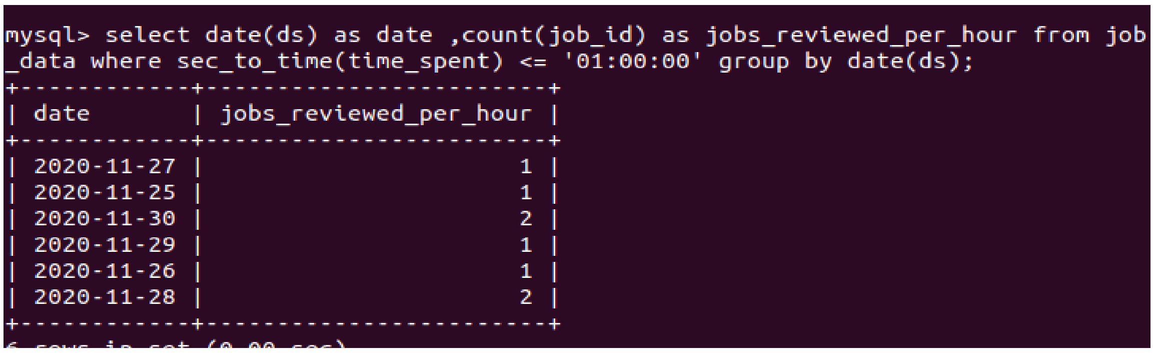
This project is about **Operation Analytics and Investigating Metric Spike** using Advanced SQL. SQL plays a crucial role in data analytics. SQL allows users to access, manipulate, analyse the data which is stored in the database. For a Data Analyst, SQL is the most powerful tool used for Data Retrieval, Data Manipulation, Data Cleaning and Transformation, Data Joining, Data Security and etc… This project involves keen observation,understanding and analysing to get the required output from the database.

## APPROACH :

### CASE - STUDY 1 : JOB DATA ANALYSIS

**A. Jobs Reviewed Over Time:**

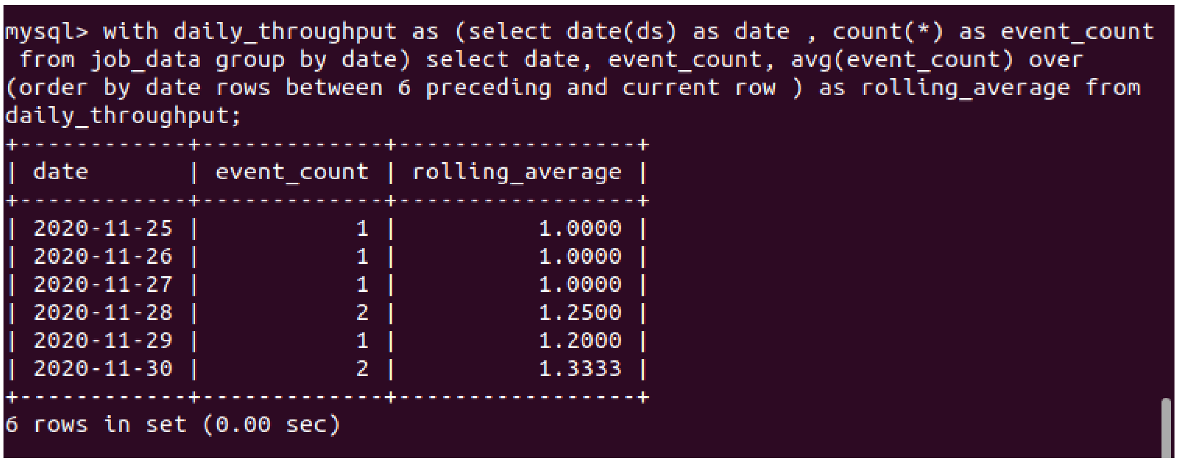
Your Task: Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.



```
mysql> select date(ds) as date ,count(job_id) as jobs_reviewed_per_hour from job
_data where sec_to_time(time_spent) <= '01:00:00' group by date(ds);
+------------+------------------------+
| date       | jobs_reviewed_per_hour |
+------------+------------------------+
| 2020-11-27 |                      1 |
| 2020-11-25 |                      1 |
| 2020-11-30 |                      2 |
| 2020-11-29 |                      1 |
| 2020-11-26 |                      1 |
| 2020-11-28 |                      2 |
+------------+------------------------+
6 rows in set (0.00 sec)
```

**B. Throughput Analysis:**

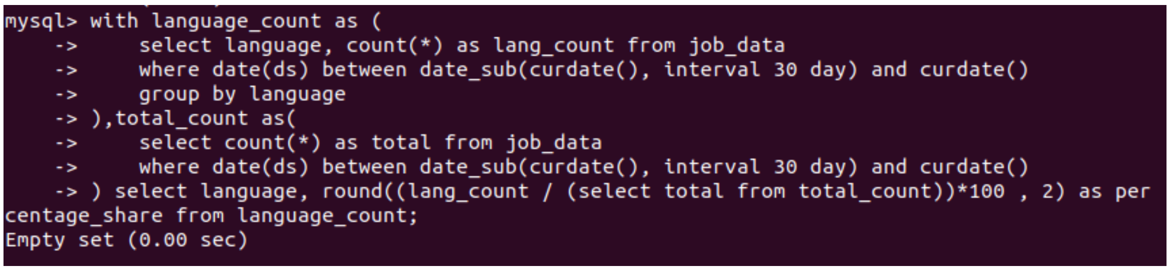
Your Task: Write an SQL query to calculate the 7-day rolling average of throughput. Additionally, explain whether you prefer using the daily metric or the 7-day rolling average for throughput, and why.



```
mysql> with daily_throughput as (select date(ds) as date , count(*) as event_count
 from job_data group by date) select date, event_count, avg(event_count) over
(order by date rows between 6 preceding and current row ) as rolling_average from
daily_throughput;
+------------+-------------+-----------------+
| date       | event_count | rolling_average |
+------------+-------------+-----------------+
| 2020-11-25 |           1 |          1.0000 |
| 2020-11-26 |           1 |          1.0000 |
| 2020-11-27 |           1 |          1.0000 |
| 2020-11-28 |           2 |          1.2500 |
| 2020-11-29 |           1 |          1.2000 |
| 2020-11-30 |           2 |          1.3333 |
+------------+-------------+-----------------+
6 rows in set (0.00 sec)
```

**C. Language Share Analysis:**

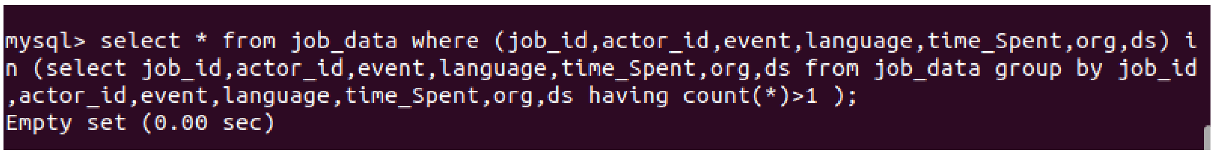
Your Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.



```
mysql> with language_count as (
    ->     select language, count(*) as lang_count from job_data
    ->     where date(ds) between date_sub(curdate(), interval 30 day) and curdate()
    ->     group by language
    -> ),total_count as(
    ->     select count(*) as total from job_data
    ->     where date(ds) between date_sub(curdate(), interval 30 day) and curdate()
    -> ) select language, round((lang_count / (select total from total_count))*100 , 2) as per
centage_share from language_count;
Empty set (0.00 sec)
```

**D. Duplicate Rows Detection:**

Your Task: Write an SQL query to display duplicate rows from the job_data table.



```
mysql> select * from job_data where (job_id,actor_id,event,language,time_Spent,org,ds) i
n (select job_id,actor_id,event,language,time_Spent,org,ds from job_data group by job_id
,actor_id,event,language,time_Spent,org,ds having count(*)>1 );
Empty set (0.00 sec)
```

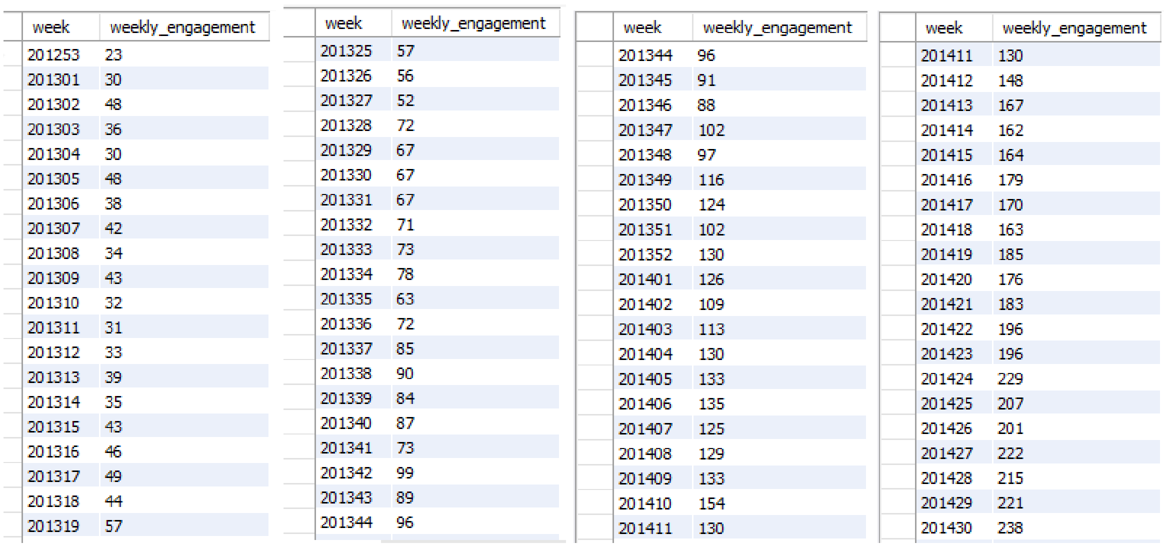
CASE - STUDY 2: INVESTIGATING  METRIC  SPIKE

**A. Weekly User Engagement:**

Your Task: Write an SQL query to calculate the weekly user engagement.

```
select yearweek(activated_at) as week,count(distinct user_id) as weekly_engagement from users
where activated_at is not null group by yearweek(activated_at);
```

**Output:**

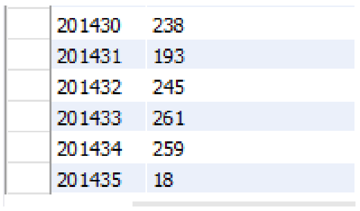


| week | weekly_engagement |
|---|---|
| 201253 | 23 |
| 201301 | 30 |
| 201302 | 48 |
| 201303 | 36 |
| 201304 | 30 |
| 201305 | 48 |
| 201306 | 38 |
| 201307 | 42 |
| 201308 | 34 |
| 201309 | 43 |
| 201310 | 32 |
| 201311 | 31 |
| 201312 | 33 |
| 201313 | 39 |
| 201314 | 35 |
| 201315 | 43 |
| 201316 | 46 |
| 201317 | 49 |
| 201318 | 44 |
| 201319 | 57 |

| week | weekly_engagement |
|---|---|
| 201325 | 57 |
| 201326 | 56 |
| 201327 | 52 |
| 201328 | 72 |
| 201329 | 67 |
| 201330 | 67 |
| 201331 | 67 |
| 201332 | 71 |
| 201333 | 73 |
| 201334 | 78 |
| 201335 | 63 |
| 201336 | 72 |
| 201337 | 85 |
| 201338 | 90 |
| 201339 | 84 |
| 201340 | 87 |
| 201341 | 73 |
| 201342 | 99 |
| 201343 | 89 |
| 201344 | 96 |

| week | weekly_engagement |
|---|---|
| 201344 | 96 |
| 201345 | 91 |
| 201346 | 88 |
| 201347 | 102 |
| 201348 | 97 |
| 201349 | 116 |
| 201350 | 124 |
| 201351 | 102 |
| 201352 | 130 |
| 201401 | 126 |
| 201402 | 109 |
| 201403 | 113 |
| 201404 | 130 |
| 201405 | 133 |
| 201406 | 135 |
| 201407 | 125 |
| 201408 | 129 |
| 201409 | 133 |
| 201410 | 154 |
| 201411 | 130 |

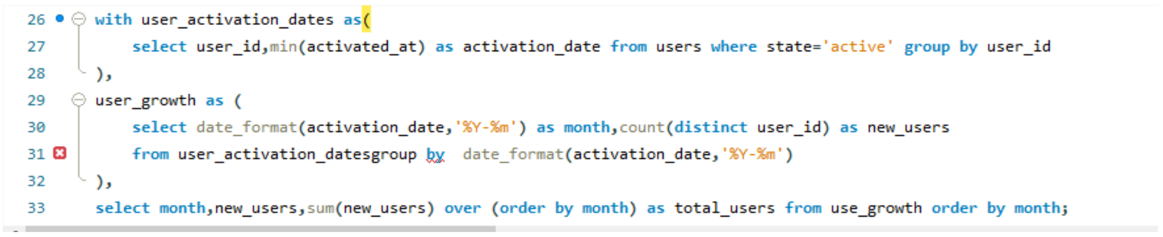
| week | weekly_engagement |
|---|---|
| 201411 | 130 |
| 201412 | 148 |
| 201413 | 167 |
| 201414 | 162 |
| 201415 | 164 |
| 201416 | 179 |
| 201417 | 170 |
| 201418 | 163 |
| 201419 | 185 |
| 201420 | 176 |
| 201421 | 183 |
| 201422 | 196 |
| 201423 | 196 |
| 201424 | 229 |
| 201425 | 207 |
| 201426 | 201 |
| 201427 | 222 |
| 201428 | 215 |
| 201429 | 221 |
| 201430 | 238 |



| | |
|---|---|
| 201430 | 238 |
| 201431 | 193 |
| 201432 | 245 |
| 201433 | 261 |
| 201434 | 259 |
| 201435 | 18 |

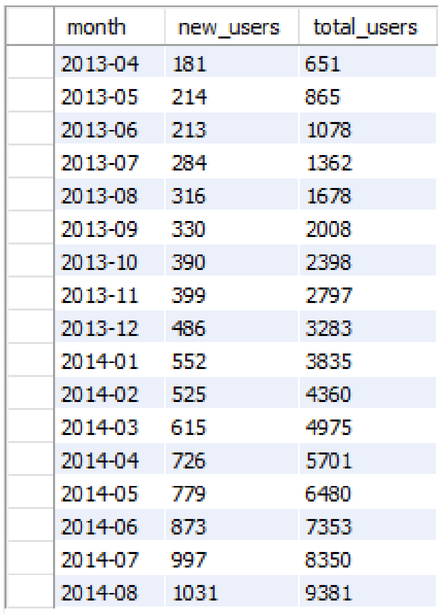
## B. User Growth Analysis:

Your Task: Write an SQL query to calculate the user growth for the product.



```sql
with user_activation_dates as(
    select user_id,min(activated_at) as activation_date from users where state='active' group by user_id
),
user_growth as (
    select date_format(activation_date,'%Y-%m') as month,count(distinct user_id) as new_users
    from user_activation_datesgroup by  date_format(activation_date,'%Y-%m')
),
select month,new_users,sum(new_users) over (order by month) as total_users from use_growth order by month;
```

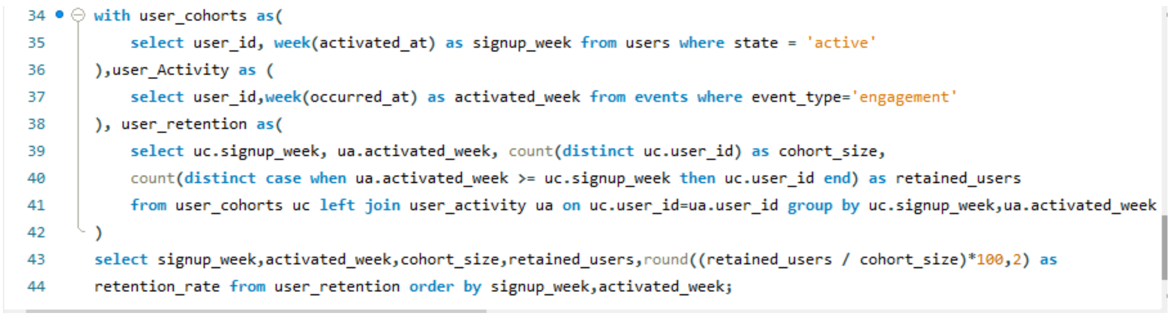
**Output:**



| month | new_users | total_users |
|---|---|---|
| 2013-04 | 181 | 651 |
| 2013-05 | 214 | 865 |
| 2013-06 | 213 | 1078 |
| 2013-07 | 284 | 1362 |
| 2013-08 | 316 | 1678 |
| 2013-09 | 330 | 2008 |
| 2013-10 | 390 | 2398 |
| 2013-11 | 399 | 2797 |
| 2013-12 | 486 | 3283 |
| 2014-01 | 552 | 3835 |
| 2014-02 | 525 | 4360 |
| 2014-03 | 615 | 4975 |
| 2014-04 | 726 | 5701 |
| 2014-05 | 779 | 6480 |
| 2014-06 | 873 | 7353 |
| 2014-07 | 997 | 8350 |
| 2014-08 | 1031 | 9381 |

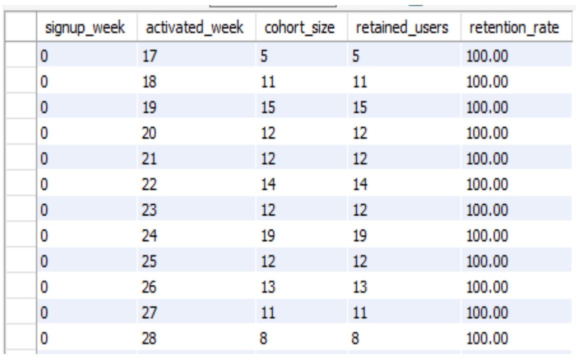
## C. Weekly Retention Analysis:

Your Task: Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

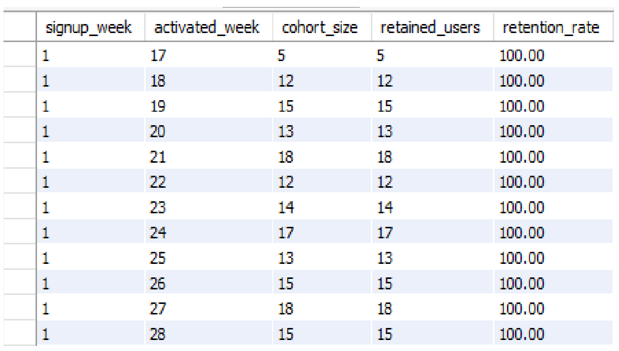


```sql
with user_cohorts as(
    select user_id, week(activated_at) as signup_week from users where state = 'active'
),user_Activity as (
    select user_id,week(occurred_at) as activated_week from events where event_type='engagement'
), user_retention as(
    select uc.signup_week, ua.activated_week, count(distinct uc.user_id) as cohort_size,
    count(distinct case when ua.activated_week >= uc.signup_week then uc.user_id end) as retained_users
    from user_cohorts uc left join user_activity ua on uc.user_id=ua.user_id group by uc.signup_week,ua.activated_week
)
select signup_week,activated_week,cohort_size,retained_users,round((retained_users / cohort_size)*100,2) as
retention_rate from user_retention order by signup_week,activated_week;
```

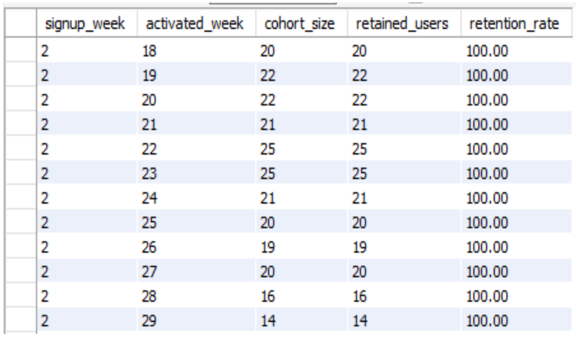
**Output:**



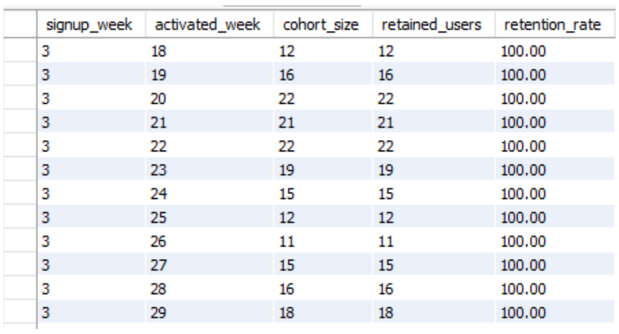
| signup_week | activated_week | cohort_size | retained_users | retention_rate |
|---|---|---|---|---|
| 0 | 17 | 5 | 5 | 100.00 |
| 0 | 18 | 11 | 11 | 100.00 |
| 0 | 19 | 15 | 15 | 100.00 |
| 0 | 20 | 12 | 12 | 100.00 |
| 0 | 21 | 12 | 12 | 100.00 |
| 0 | 22 | 14 | 14 | 100.00 |
| 0 | 23 | 12 | 12 | 100.00 |
| 0 | 24 | 19 | 19 | 100.00 |
| 0 | 25 | 12 | 12 | 100.00 |
| 0 | 26 | 13 | 13 | 100.00 |
| 0 | 27 | 11 | 11 | 100.00 |
| 0 | 28 | 8 | 8 | 100.00 |



| signup_week | activated_week | cohort_size | retained_users | retention_rate |
|---|---|---|---|---|
| 1 | 17 | 5 | 5 | 100.00 |
| 1 | 18 | 12 | 12 | 100.00 |
| 1 | 19 | 15 | 15 | 100.00 |
| 1 | 20 | 13 | 13 | 100.00 |
| 1 | 21 | 18 | 18 | 100.00 |
| 1 | 22 | 12 | 12 | 100.00 |
| 1 | 23 | 14 | 14 | 100.00 |
| 1 | 24 | 17 | 17 | 100.00 |
| 1 | 25 | 13 | 13 | 100.00 |
| 1 | 26 | 15 | 15 | 100.00 |
| 1 | 27 | 18 | 18 | 100.00 |
| 1 | 28 | 15 | 15 | 100.00 |



| signup_week | activated_week | cohort_size | retained_users | retention_rate |
|---|---|---|---|---|
| 2 | 18 | 20 | 20 | 100.00 |
| 2 | 19 | 22 | 22 | 100.00 |
| 2 | 20 | 22 | 22 | 100.00 |
| 2 | 21 | 21 | 21 | 100.00 |
| 2 | 22 | 25 | 25 | 100.00 |
| 2 | 23 | 25 | 25 | 100.00 |
| 2 | 24 | 21 | 21 | 100.00 |
| 2 | 25 | 20 | 20 | 100.00 |
| 2 | 26 | 19 | 19 | 100.00 |
| 2 | 27 | 20 | 20 | 100.00 |
| 2 | 28 | 16 | 16 | 100.00 |
| 2 | 29 | 14 | 14 | 100.00 |

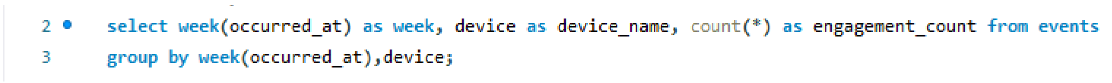


| signup_week | activated_week | cohort_size | retained_users | retention_rate |
|---|---|---|---|---|
| 3 | 18 | 12 | 12 | 100.00 |
| 3 | 19 | 16 | 16 | 100.00 |
| 3 | 20 | 22 | 22 | 100.00 |
| 3 | 21 | 21 | 21 | 100.00 |
| 3 | 22 | 22 | 22 | 100.00 |
| 3 | 23 | 19 | 19 | 100.00 |
| 3 | 24 | 15 | 15 | 100.00 |
| 3 | 25 | 12 | 12 | 100.00 |
| 3 | 26 | 11 | 11 | 100.00 |
| 3 | 27 | 15 | 15 | 100.00 |
| 3 | 28 | 16 | 16 | 100.00 |
| 3 | 29 | 18 | 18 | 100.00 |

And many more records upto 52nd signup_week.

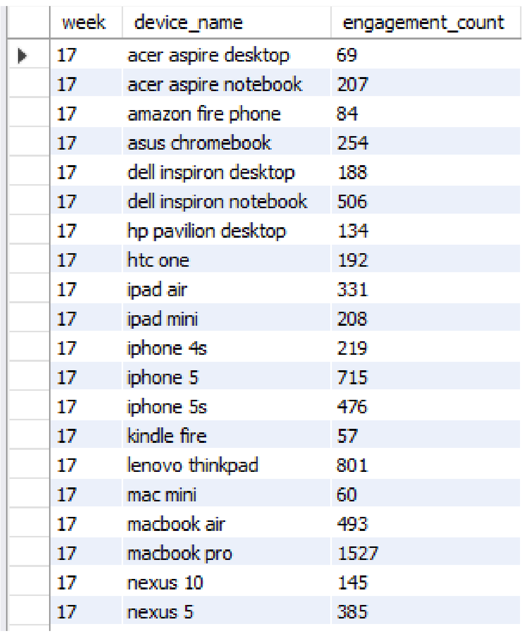
## D. Weekly Engagement Per Device:

Your Task: Write an SQL query to calculate the weekly engagement per device.

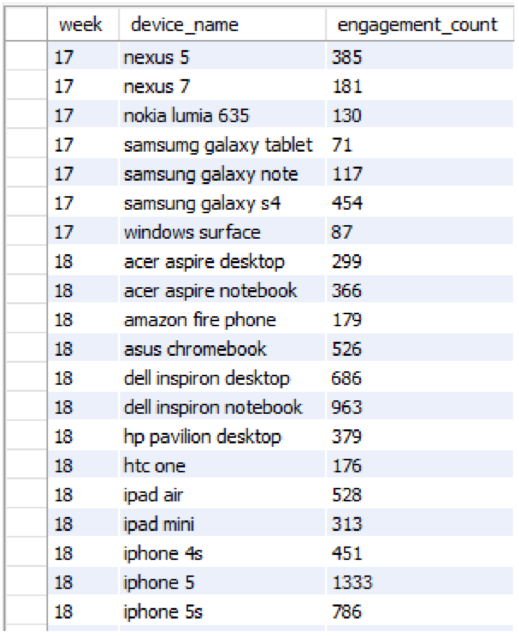


```sql
select week(occurred_at) as week, device as device_name, count(*) as engagement_count from events
group by week(occurred_at),device;
```

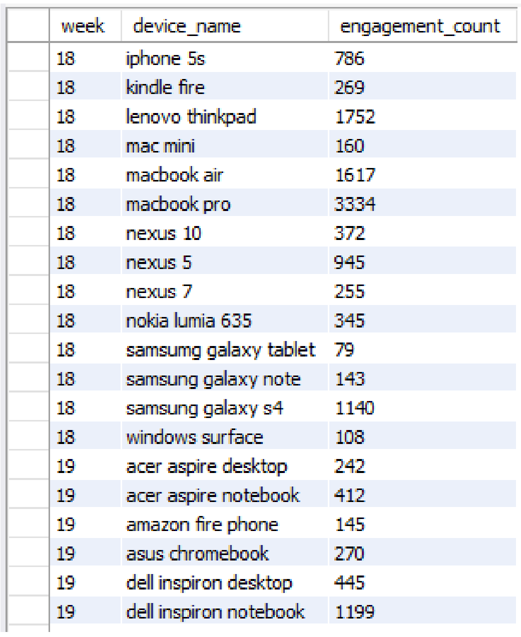
**Output:**



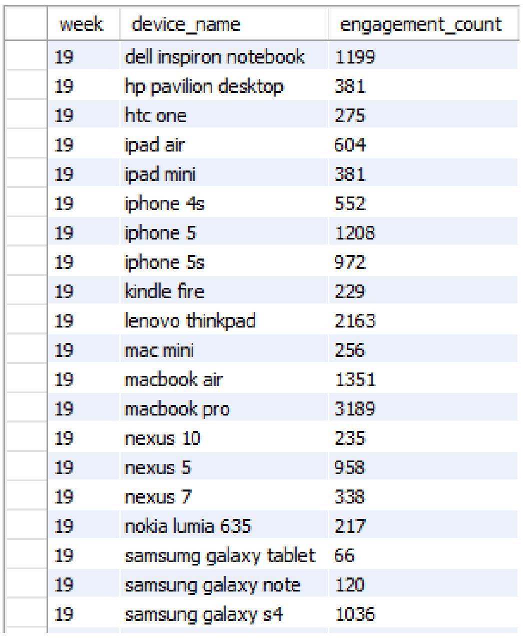
| week | device_name | engagement_count |
|---|---|---|
| 17 | acer aspire desktop | 69 |
| 17 | acer aspire notebook | 207 |
| 17 | amazon fire phone | 84 |
| 17 | asus chromebook | 254 |
| 17 | dell inspiron desktop | 188 |
| 17 | dell inspiron notebook | 506 |
| 17 | hp pavilion desktop | 134 |
| 17 | htc one | 192 |
| 17 | ipad air | 331 |
| 17 | ipad mini | 208 |
| 17 | iphone 4s | 219 |
| 17 | iphone 5 | 715 |
| 17 | iphone 5s | 476 |
| 17 | kindle fire | 57 |
| 17 | lenovo thinkpad | 801 |
| 17 | mac mini | 60 |
| 17 | macbook air | 493 |
| 17 | macbook pro | 1527 |
| 17 | nexus 10 | 145 |
| 17 | nexus 5 | 385 |



| week | device_name | engagement_count |
|---|---|---|
| 17 | nexus 5 | 385 |
| 17 | nexus 7 | 181 |
| 17 | nokia lumia 635 | 130 |
| 17 | samsumg galaxy tablet | 71 |
| 17 | samsung galaxy note | 117 |
| 17 | samsung galaxy s4 | 454 |
| 17 | windows surface | 87 |
| 18 | acer aspire desktop | 299 |
| 18 | acer aspire notebook | 366 |
| 18 | amazon fire phone | 179 |
| 18 | asus chromebook | 526 |
| 18 | dell inspiron desktop | 686 |
| 18 | dell inspiron notebook | 963 |
| 18 | hp pavilion desktop | 379 |
| 18 | htc one | 176 |
| 18 | ipad air | 528 |
| 18 | ipad mini | 313 |
| 18 | iphone 4s | 451 |
| 18 | iphone 5 | 1333 |
| 18 | iphone 5s | 786 |



| week | device_name | engagement_count |
|---|---|---|
| 18 | iphone 5s | 786 |
| 18 | kindle fire | 269 |
| 18 | lenovo thinkpad | 1752 |
| 18 | mac mini | 160 |
| 18 | macbook air | 1617 |
| 18 | macbook pro | 3334 |
| 18 | nexus 10 | 372 |
| 18 | nexus 5 | 945 |
| 18 | nexus 7 | 255 |
| 18 | nokia lumia 635 | 345 |
| 18 | samsumg galaxy tablet | 79 |
| 18 | samsung galaxy note | 143 |
| 18 | samsung galaxy s4 | 1140 |
| 18 | windows surface | 108 |
| 19 | acer aspire desktop | 242 |
| 19 | acer aspire notebook | 412 |
| 19 | amazon fire phone | 145 |
| 19 | asus chromebook | 270 |
| 19 | dell inspiron desktop | 445 |
| 19 | dell inspiron notebook | 1199 |

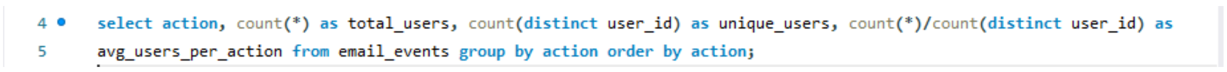


| week | device_name | engagement_count |
|---|---|---|
| 19 | dell inspiron notebook | 1199 |
| 19 | hp pavilion desktop | 381 |
| 19 | htc one | 275 |
| 19 | ipad air | 604 |
| 19 | ipad mini | 381 |
| 19 | iphone 4s | 552 |
| 19 | iphone 5 | 1208 |
| 19 | iphone 5s | 972 |
| 19 | kindle fire | 229 |
| 19 | lenovo thinkpad | 2163 |
| 19 | mac mini | 256 |
| 19 | macbook air | 1351 |
| 19 | macbook pro | 3189 |
| 19 | nexus 10 | 235 |
| 19 | nexus 5 | 958 |
| 19 | nexus 7 | 338 |
| 19 | nokia lumia 635 | 217 |
| 19 | samsumg galaxy tablet | 66 |
| 19 | samsung galaxy note | 120 |
| 19 | samsung galaxy s4 | 1036 |

And many more records upto week 35

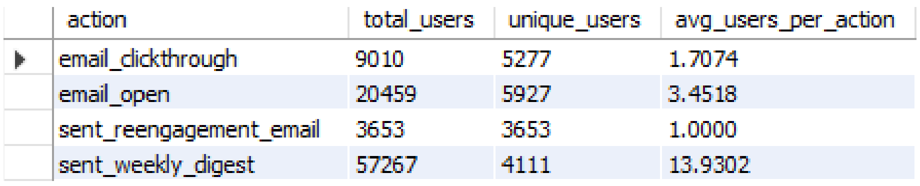
**E. Email Engagement Analysis:**

Your Task: Write an SQL query to calculate the email engagement metrics.



```
select action, count(*) as total_users, count(distinct user_id) as unique_users, count(*)/count(distinct user_id) as
avg_users_per_action from email_events group by action order by action;
```

**Output:**



| action | total_users | unique_users | avg_users_per_action |
|---|---|---|---|
| email_clickthrough | 9010 | 5277 | 1.7074 |
| email_open | 20459 | 5927 | 3.4518 |
| sent_reengagement_email | 3653 | 3653 | 1.0000 |
| sent_weekly_digest | 57267 | 4111 | 13.9302 |

## TECH-STACK USED:

I’ve used mysql linux version
**mysql  Ver 8.0.35-0ubuntu0.20.04.1 for Linux on x86_64**
And also Mysql Workbench in windows to explore the difference between these two tech-stacks.

## INSIGHTS :

As I am already aware of SQL, It became easy to do this project. But I learnt some new concepts like types of SUBQUERY, CASE statements and CORE WINDOW FUNCTIONS . By this project I came to know the practical implementation of the concepts mentioned above.

## RESULT :

I have learnt many new concepts from this project. Through this project, I came to know that as compared to MS Excel, mysql extracts the data in a more efficient way. MySQL not only helps users to extract the data, but also Data Filtering, Data Joining , Keen Understanding and Analysing the data.