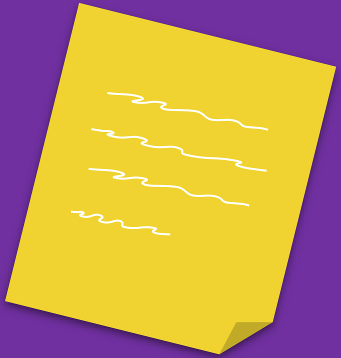
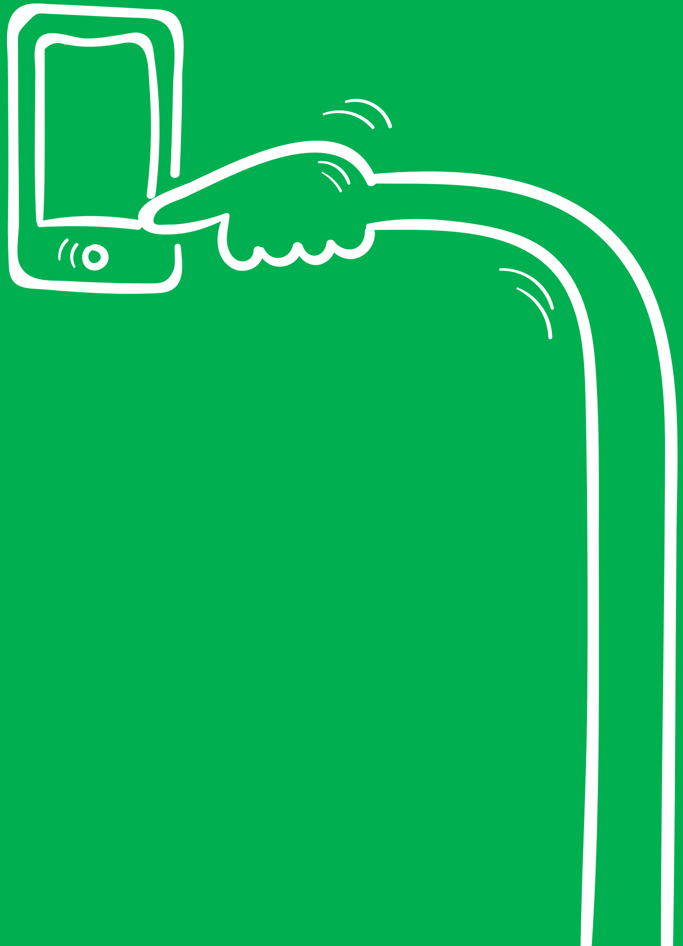
Operation Analytics & Investigating Metric Spike



Project Description

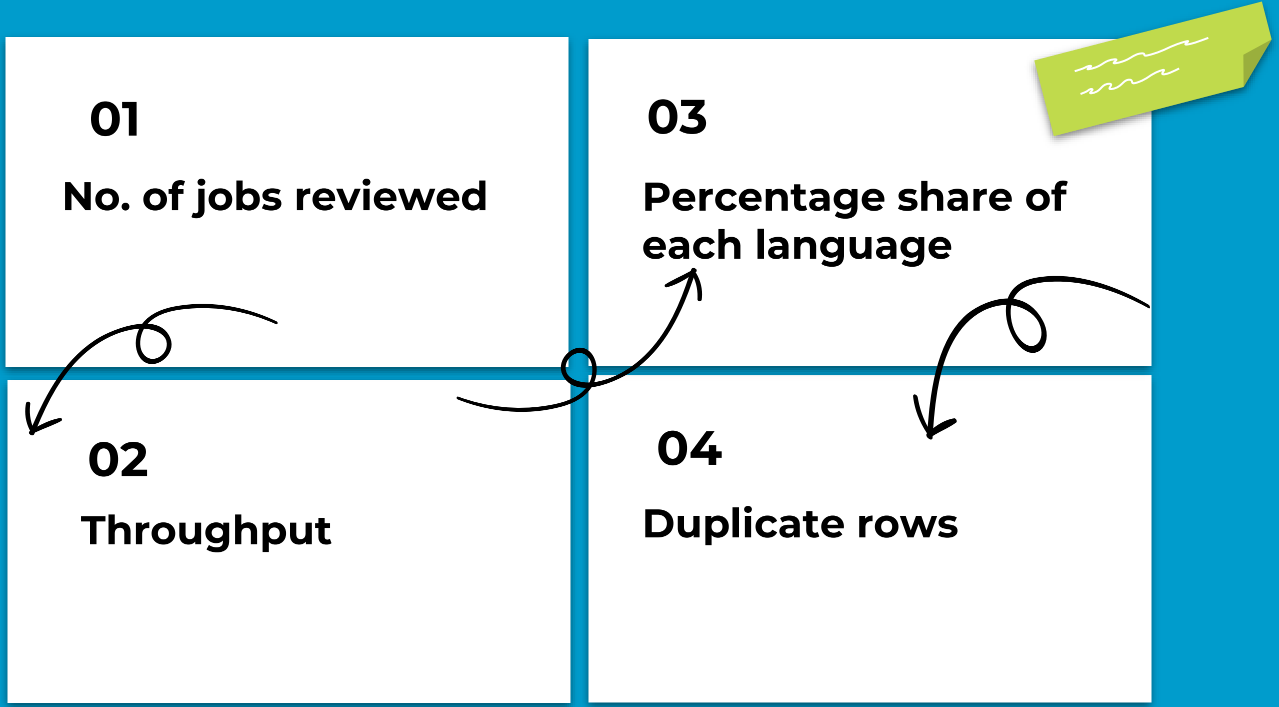
This project is about performing data analysis on the two given data set. This project mainly requires MySQL and Microsoft excel to fetch data about user, events, date e.t.c.

Tech-Stack



MySQL version 8.0.33
Microsoft Excel 2019

Case Study 1 (Job Data)



01

No. of jobs reviewed

02

Throughput

03

Percentage share of each language

04

Duplicate rows

Number of jobs reviewed

Amount of jobs reviewed over time. Calculate the number of jobs reviewed per hour per day for November 2020?

SQL Query

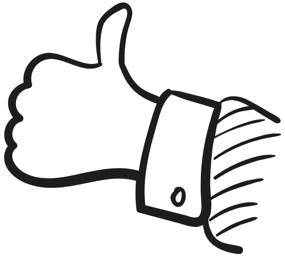
```
select
ds, sum(time_spent)/3600 as
total_time_in_hour
from project1
where
ds between '2020-11-01' and '2020-11-30’
group by ds order by ds;
```

Output

ds	total_time_in_hour
25-11-2020	0.0125
26-11-2020	0.0156
27-11-2020	0.0289
28-11-2020	0.0092
29-11-2020	0.0056
30-11-2020	0.0111

Throughput

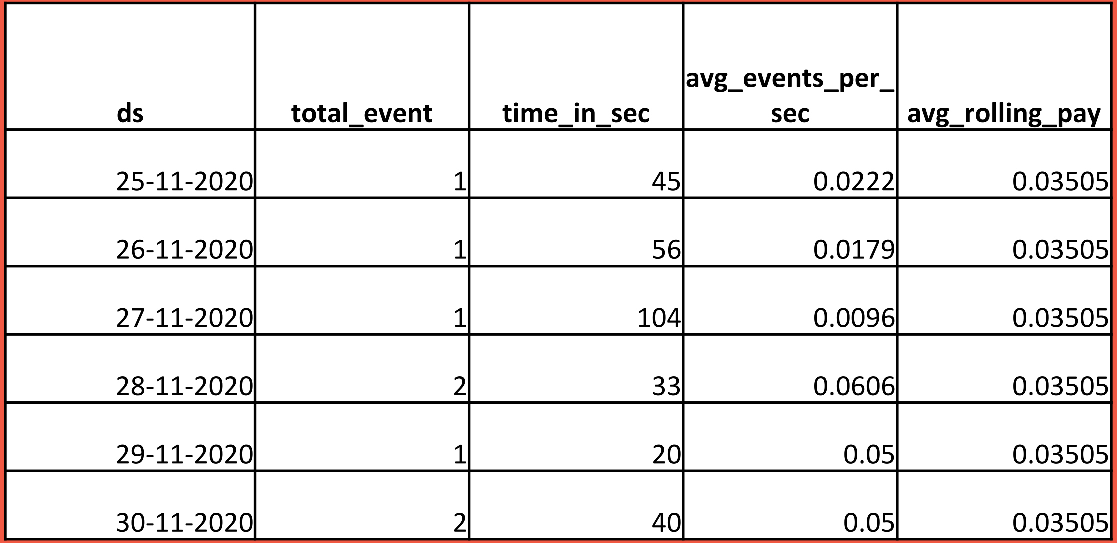
Throughput: It is the no. of events happening per second.
Let’s say the above metric is called throughput. Calculate 7 day rolling average of throughput? For throughput, do you prefer daily metric or 7-day rolling and why?



SQL Query

```
select *, round(avg(avg_events_per_sec)
over(), 5) as avg_rolling_pay
from
(select ds, count(`event`)
as total_event, sum(time_spent) as
time_in_sec,
count(`event`)/sum(time_spent) as
avg_events_per_sec from
project1 group by 1 order by 1)t1;
```

Output



ds	total_event	time_in_sec	avg_events_per_sec	avg_rolling_pay
25-11-2020	1	45	0.0222	0.03505
26-11-2020	1	56	0.0179	0.03505
27-11-2020	1	104	0.0096	0.03505
28-11-2020	2	33	0.0606	0.03505
29-11-2020	1	20	0.05	0.03505
30-11-2020	2	40	0.05	0.03505

Daily rolling pay is better because in this limited data of one week it is beneficial to know daily rolling pay rather than whole week average.

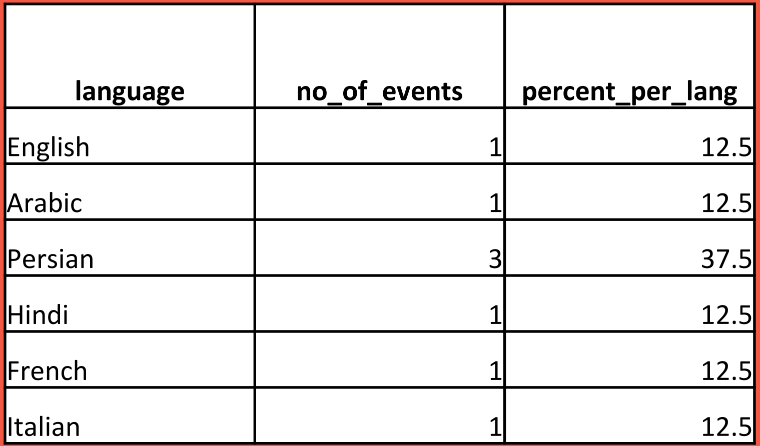
Percentage share by each language

Percentage share of each language: Share of each language for different contents.
Your task: Calculate the percentage share of each language in the last 30 days?

SQL Query

```
select *,
round(no_of_events/sum(t1.no_of_events)
over() *100, 1)
as percent_per_lang
from
(select `language`, count(`event`) as
no_of_events  from project1 group
by`language`)t1;
```

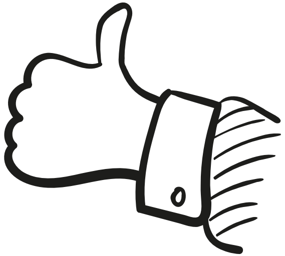
Output



language	no_of_events	percent_per_lang
English	1	12.5
Arabic	1	12.5
Persian	3	37.5
Hindi	1	12.5
French	1	12.5
Italian	1	12.5

Duplicate Rows

Duplicate rows: Rows that have the same value present in them.
Your task: Let’s say you see some duplicate rows in the data. How will you display duplicates from the table?



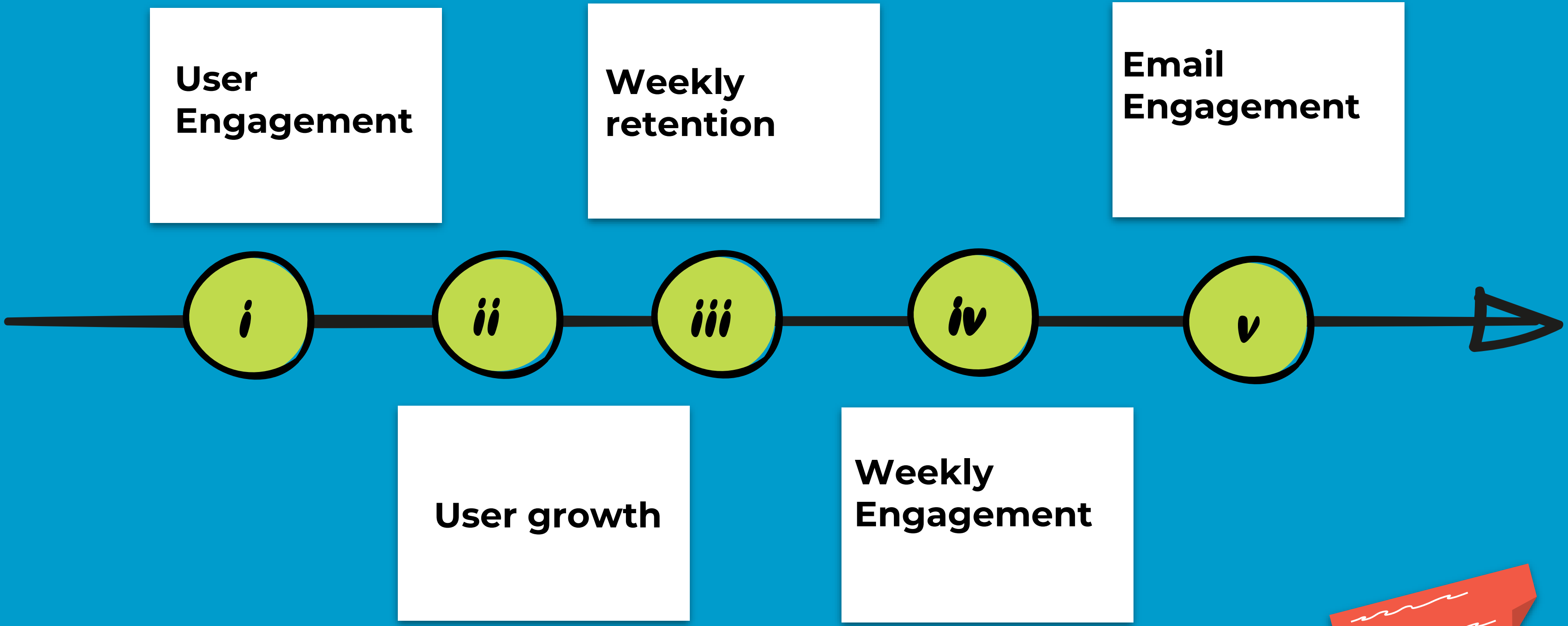
SQL Query

```
select *
from
(select *, count(job_id) over(partition by
job_id) as count_ from project1)t1
where count_ > 1;
```

Output

ds	job_id	actor_id	event	language	time_spent	org	count_
29-11-2020	23	1003	decision	Persian	20	C	3
28-11-2020	23	1005	transfer	Persian	22	D	3
26-11-2020	23	1004	skip	Persian	56	A	3

Case Study 2 – Investigating Metric Spike



User Engagement

User growth

Weekly retention

Weekly Engagement

Email Engagement

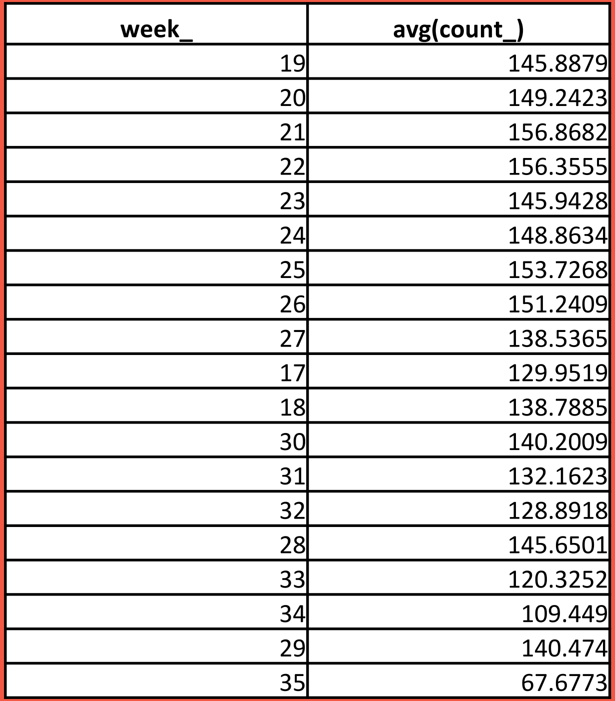
User Engagement

User Engagement: To measure the activeness of a user. Measuring if the user finds quality in a product/service.
Your task: Calculate the weekly user engagement?

SQL Query

```
select
week(occurred_at) as week_,
avg(count_)
from
(select occurred_at, user_id,
count(event_type) over(partition by
user_id)as count_
from
event_table where event_type =
'engagement’)t1
group by 1;
```

Output



week_	avg(count_)
19	145.8879
20	149.2423
21	156.8682
22	156.3555
23	145.9428
24	148.8634
25	153.7268
26	151.2409
27	138.5365
17	129.9519
18	138.7885
30	140.2009
31	132.1623
32	128.8918
28	145.6501
33	120.3252
34	109.449
29	140.474
35	67.6773

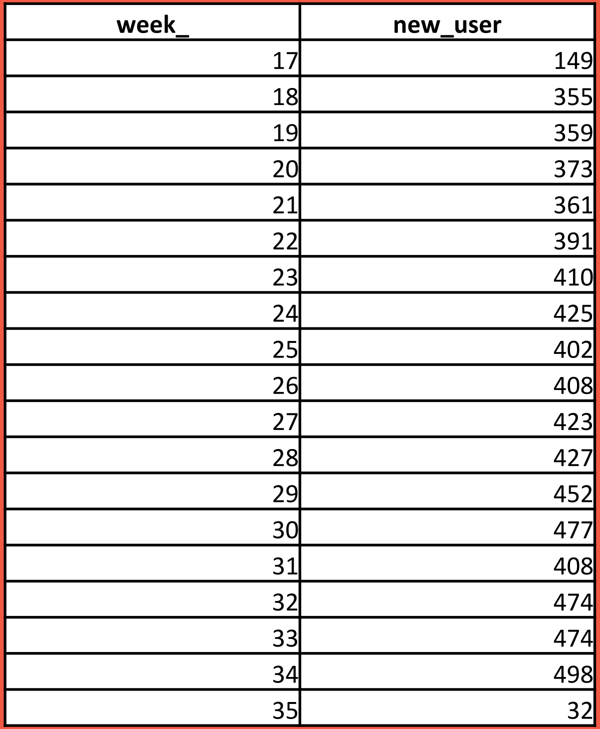
User growth

User Growth: Amount of users growing over time for a product.
Your task: Calculate the user growth for product?

SQL Query

```
select
week(occurred_at) as week_,
count(user_id) as new_user
from
event_table where  event_name =
'create_user' group by week_;
```

Output



week_	new_user
17	149
18	355
19	359
20	373
21	361
22	391
23	410
24	425
25	402
26	408
27	423
28	427
29	452
30	477
31	408
32	474
33	474
34	498
35	32

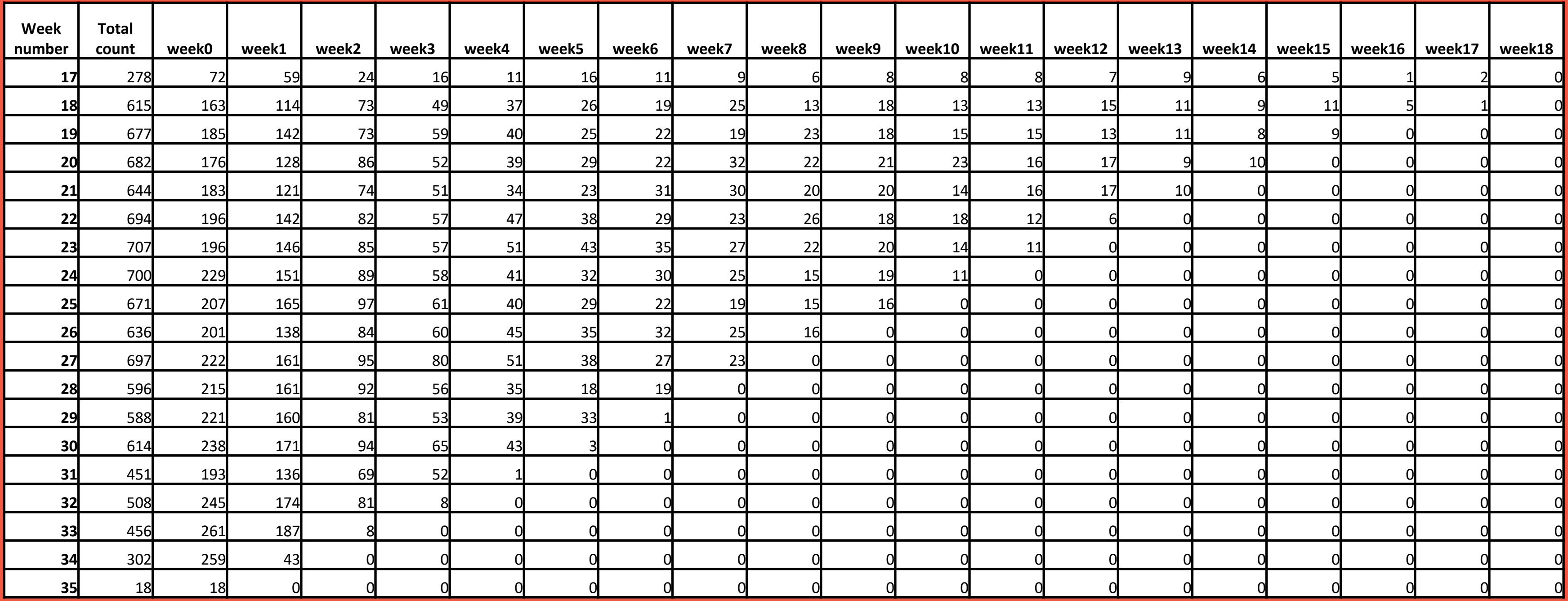
Weekly retention

Weekly Retention: Users getting retained weekly after signing-up for a product.
Your task: Calculate the weekly retention of users-sign up cohort?

SQL Query

```
select first_date as week_number,count(id) as total_count,
sum(case when week_ = 0 then 1 else 0 end) as 'week0’,
sum(case when week_ = 1 then 1 else 0 end) as 'week1’,
sum(case when week_ = 2 then 1 else 0 end) as 'week2’,
sum(case when week_ = 3 then 1 else 0 end) as 'week3’,
sum(case when week_ = 4 then 1 else 0 end) as 'week4’,
sum(case when week_ = 5 then 1 else 0 end) as 'week5’,
sum(case when week_ = 6 then 1 else 0 end) as 'week6’,
sum(case when week_ = 7 then 1 else 0 end) as 'week7’,
sum(case when week_ = 8 then 1 else 0 end) as 'week8’,
sum(case when week_ = 9 then 1 else 0 end) as 'week9’,
sum(case when week_ = 10 then 1 else 0 end) as 'week10’,
sum(case when week_ = 11 then 1 else 0 end) as 'week11’,
sum(case when week_ = 12 then 1 else 0 end) as 'week12’,
sum(case when week_ = 13 then 1 else 0 end) as 'week13’,
sum(case when week_ = 14 then 1 else 0 end) as 'week14’,
sum(case when week_ = 15 then 1 else 0 end) as 'week15’,
sum(case when week_ = 16 then 1 else 0 end) as 'week16’,
sum(case when week_ = 17 then 1 else 0 end) as 'week17’,
sum(case when week_ = 18 then 1 else 0 end) as 'week18’
from
(select a.id ,b.user_id, b.repeat_date, a.first_date, b.repeat_date -
first_date as week_ from
(select user_id as id, week(occurred_at) as first_date from event_
where event_name = 'complete_signup’ )a
left join
(select distinct(user_id), week(occurred_at) as repeat_date from
event_table where event_type = 'engagement')b on a.id = b.user_id) t1
group by first_date order by first_date;
```

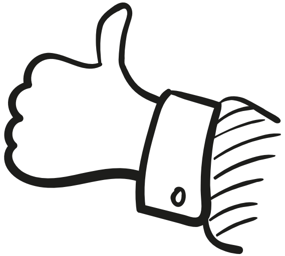
Output



Week number	Total count	week0	week1	week2	week3	week4	week5	week6	week7	week8	week9	week10	week11	week12	week13	week14	week15	week16	week17	week18
17	278	72	59	24	16	11	16	11	9	6	8	8	8	7	9	6	5	1	2	0
18	615	163	114	73	49	37	26	19	25	13	18	13	13	15	11	9	11	5	1	0
19	677	185	142	73	59	40	25	22	19	23	18	15	15	13	11	8	9	0	0	0
20	682	176	128	86	52	39	29	22	32	22	21	23	16	17	9	10	0	0	0	0
21	644	183	121	74	51	34	23	31	30	20	20	14	16	17	10	0	0	0	0	0
22	694	196	142	82	57	47	38	29	23	26	18	18	12	6	0	0	0	0	0	0
23	707	196	146	85	57	51	43	35	27	22	20	14	11	0	0	0	0	0	0	0
24	700	229	151	89	58	41	32	30	25	15	19	11	0	0	0	0	0	0	0	0
25	671	207	165	97	61	40	29	22	19	15	16	0	0	0	0	0	0	0	0	0
26	636	201	138	84	60	45	35	32	25	16	0	0	0	0	0	0	0	0	0	0
27	697	222	161	95	80	51	38	27	23	0	0	0	0	0	0	0	0	0	0	0
28	596	215	161	92	56	35	18	19	0	0	0	0	0	0	0	0	0	0	0	0
29	588	221	160	81	53	39	33	1	0	0	0	0	0	0	0	0	0	0	0	0
30	614	238	171	94	65	43	3	0	0	0	0	0	0	0	0	0	0	0	0	0
31	451	193	136	69	52	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	508	245	174	81	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	456	261	187	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	302	259	43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	18	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Weekly Engagement

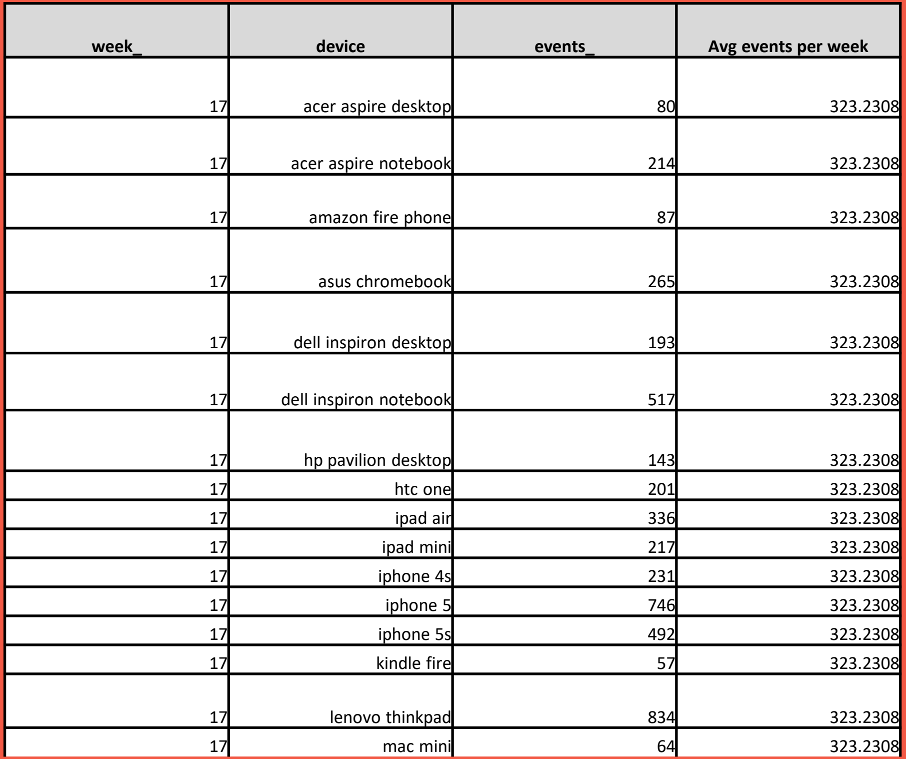
Weekly Engagement: To measure the activeness of a user. Measuring if the user finds quality in a product/service weekly.
Your task: Calculate the weekly engagement per device?



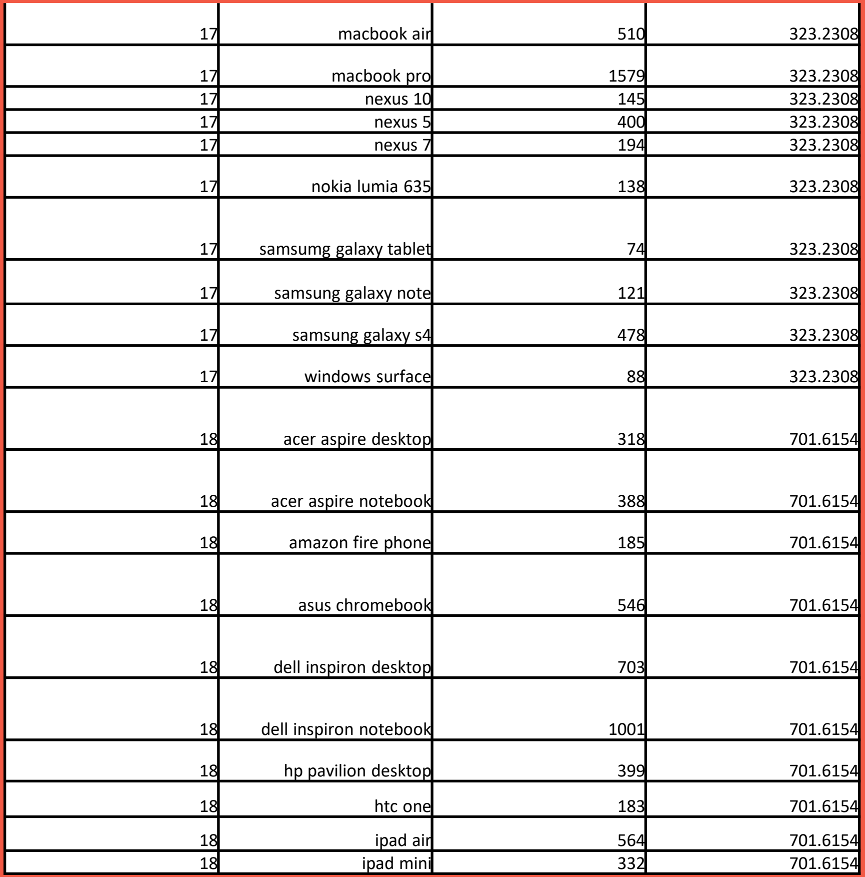
SQL Query

```
select
week_, device,
events_,
avg(events_) over(partition by week_num) as
avg_events_per_week
from
(select week(occurred_at) as week_,
device,
count(event_name) as events_
from event_table where event_type =
'engagement’ group by 1,2)t1 ;
```

Output



week_	device	events_	Avg events per week
17	acer aspire desktop	80	323.2308
17	acer aspire notebook	214	323.2308
17	amazon fire phone	87	323.2308
17	asus chromebook	265	323.2308
17	dell inspiron desktop	193	323.2308
17	dell inspiron notebook	517	323.2308
17	hp pavilion desktop	143	323.2308
17	htc one	201	323.2308
17	ipad air	336	323.2308
17	ipad mini	217	323.2308
17	iphone 4s	231	323.2308
17	iphone 5	746	323.2308
17	iphone 5s	492	323.2308
17	kindle fire	57	323.2308
17	lenovo thinkpad	834	323.2308
17	mac mini	64	323.2308



17	macbook air	510	323.2308
17	macbook pro	1579	323.2308
17	nexus 10	145	323.2308
17	nexus 5	400	323.2308
17	nexus 7	194	323.2308
17	nokia lumia 635	138	323.2308
17	samsumg galaxy tablet	74	323.2308
17	samsung galaxy note	121	323.2308
17	samsung galaxy s4	478	323.2308
17	windows surface	88	323.2308
18	acer aspire desktop	318	701.6154
18	acer aspire notebook	388	701.6154
18	amazon fire phone	185	701.6154
18	asus chromebook	546	701.6154
18	dell inspiron desktop	703	701.6154
18	dell inspiron notebook	1001	701.6154
18	hp pavilion desktop	399	701.6154
18	htc one	183	701.6154
18	ipad air	564	701.6154
18	ipad mini	332	701.6154

continues (491 rows)

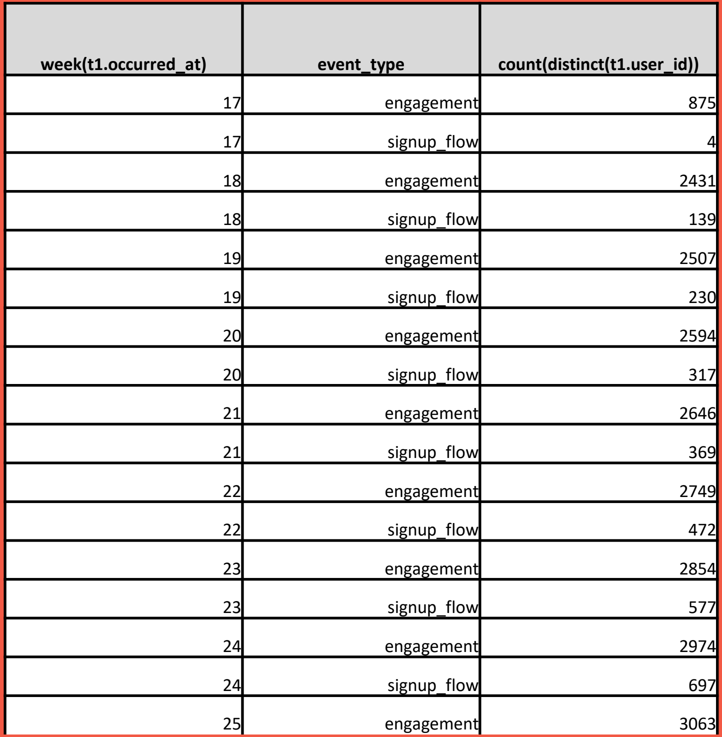
Email Engagement

Email Engagement: Users engaging with the email service.
Your task: Calculate the email engagement metrics?

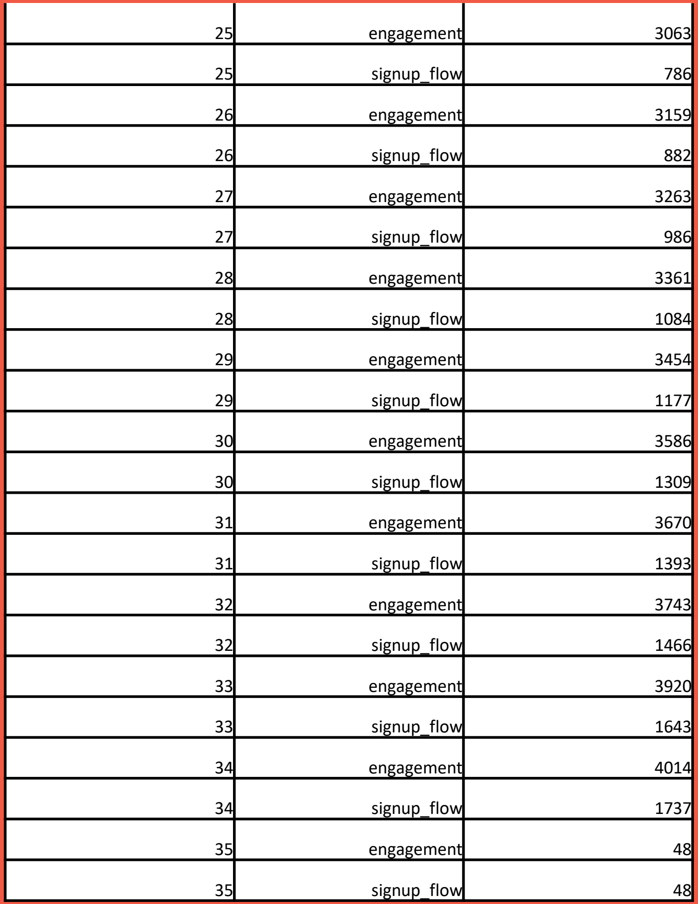
SQL Query

```
select week(t1.occurred_at),
event_type,
count(distinct(t1.user_id))
from
(select user_id, occurred_at, `action`from
email)t1 inner join(select user_id,
occurred_at, event_type from event_table)t2
on t1.user_id = t2.user_id  where event_type
is not null group by 1,2;
```

Output



week(t1.occurred_at)	event_type	count(distinct(t1.user_id))
17	engagement	875
17	signup_flow	4
18	engagement	2431
18	signup_flow	139
19	engagement	2507
19	signup_flow	230
20	engagement	2594
20	signup_flow	317
21	engagement	2646
21	signup_flow	369
22	engagement	2749
22	signup_flow	472
23	engagement	2854
23	signup_flow	577
24	engagement	2974
24	signup_flow	697
25	engagement	3063



25	engagement	3063
25	signup_flow	786
26	engagement	3159
26	signup_flow	882
27	engagement	3263
27	signup_flow	986
28	engagement	3361
28	signup_flow	1084
29	engagement	3454
29	signup_flow	1177
30	engagement	3586
30	signup_flow	1309
31	engagement	3670
31	signup_flow	1393
32	engagement	3743
32	signup_flow	1466
33	engagement	3920
33	signup_flow	1643
34	engagement	4014
34	signup_flow	1737
35	engagement	48
35	signup_flow	48

Approach

In this project got know about various functions of mySQL like –

- Windows function
- Case function
- Sub query
- Joins e.t.c.

I have also learned about how to clean the data using MS excel & insert, manipulate data in mySQL.

Insights & Results-

This project helped me understand different types of queries and functions of MySQL and MS excel.

Thanks