Advanced SQL Project

Operation Analytics and Investigating Metric Spike

Created BY- Mainak Mukherjee

Email- subha.mainak@gmail.com

| Approach |

I thoroughly examined the dataset after importing the database into MySQL Workbench. I made an ER Diagram of the whole dataset by carefully examining each table, its columns, rows, and the connections between them.

I need the information to comprehend the given database and have business knowledge before I can discover the answers to the questions. I constructed a data model that included the number of rows and columns in each table, the data type, the key, the linkages, and other details after doing data profiling.

Following all of this, I began looking up the answers to the questions that were given to me.

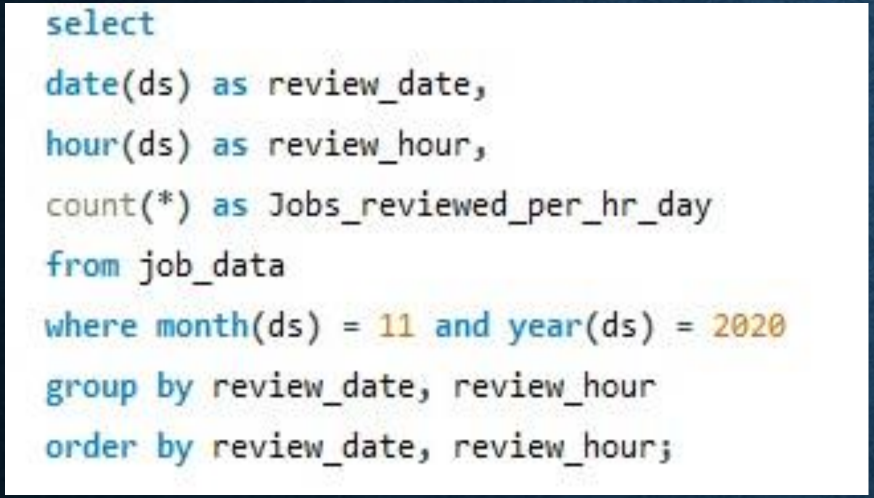
| Tech-Stack Used |

To query the database, I utilised Oracle's MySQL Workbench version 8.0.31 for project execution. It was a useful tool for project execution because of its graphical user interface, troubleshooting support, and simplicity of access and setup.

| Case Study 1: Job Data Analysis |

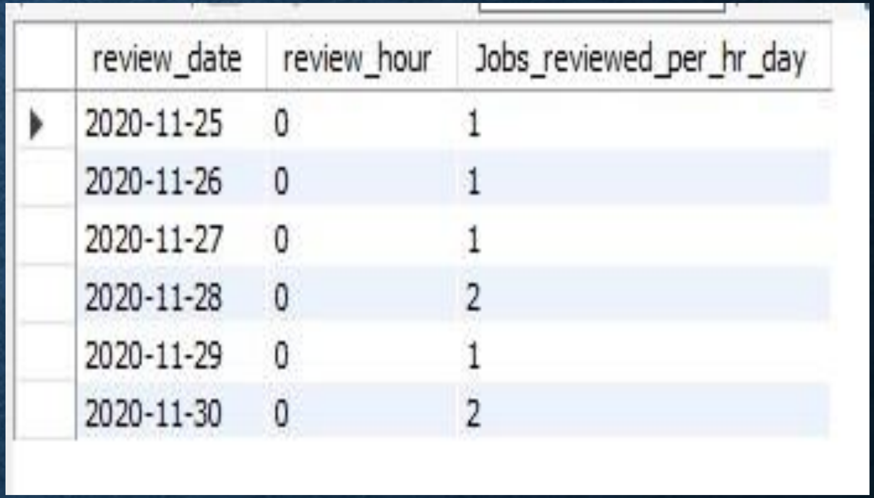
1. Jobs Reviewed Over Time: Calculate the number of jobs reviewed per hour for each day in November 2020.

Input



```
select
date(ds) as review_date,
hour(ds) as review_hour,
count(*) as Jobs_reviewed_per_hr_day
from job_data
where month(ds) = 11 and year(ds) = 2020
group by review_date, review_hour
order by review_date, review_hour;
```

Output

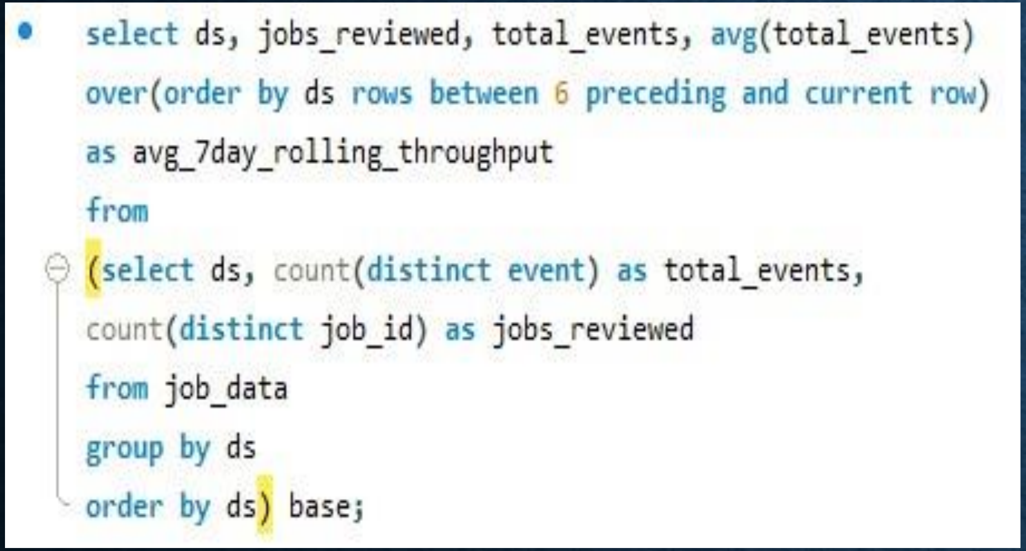


review_date	review_hour	Jobs_reviewed_per_hr_day
2020-11-25	0	1
2020-11-26	0	1
2020-11-27	0	1
2020-11-28	0	2
2020-11-29	0	1
2020-11-30	0	2

| Case Study 1: Job Data Analysis |

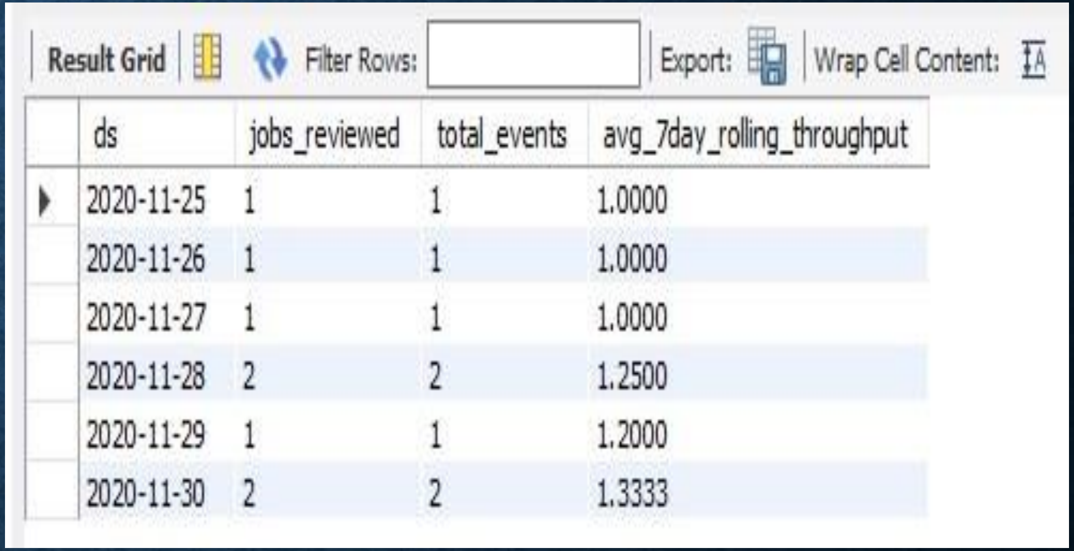
2. Throughput Analysis: Calculate the 7-day rolling average of throughput.

Input



```sql
select ds, jobs_reviewed, total_events, avg(total_events)
over(order by ds rows between 6 preceding and current row)
as avg_7day_rolling_throughput
from
(select ds, count(distinct event) as total_events,
count(distinct job_id) as jobs_reviewed
from job_data
group by ds
order by ds) base;
```

Output

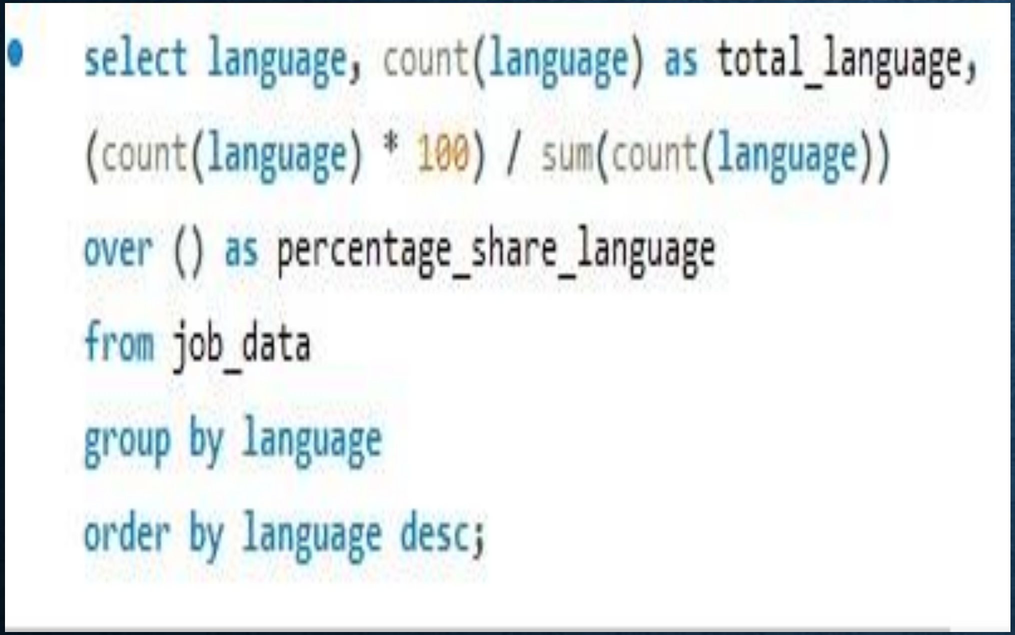


ds	jobs_reviewed	total_events	avg_7day_rolling_throughput
2020-11-25	1	1	1.0000
2020-11-26	1	1	1.0000
2020-11-27	1	1	1.0000
2020-11-28	2	2	1.2500
2020-11-29	1	1	1.2000
2020-11-30	2	2	1.3333

| Case Study 1: Job Data Analysis |

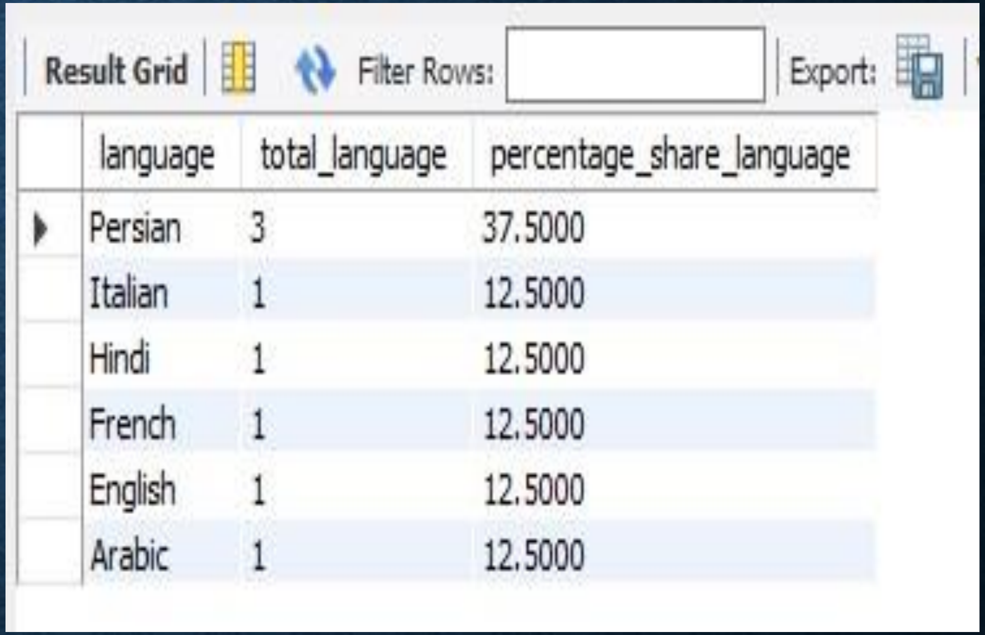
3. Language Share Analysis: Calculate the percentage share of each language over the last 30 days.

Input



```
select language, count(language) as total_language,
(count(language) * 100) / sum(count(language))
over () as percentage_share_language
from job_data
group by language
order by language desc;
```

Output



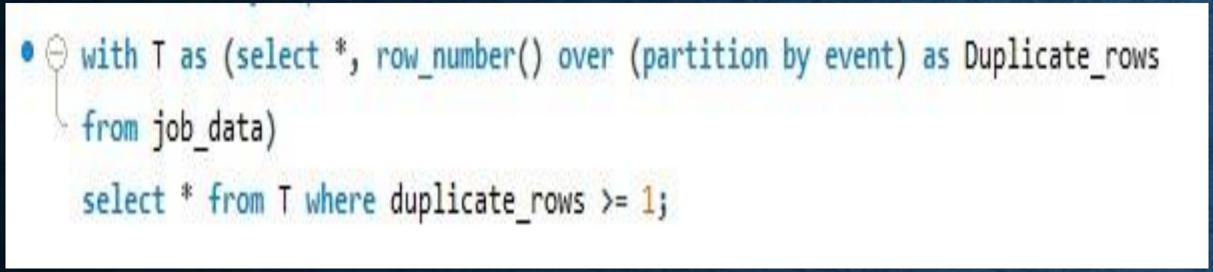
language	total_language	percentage_share_language
Persian	3	37.5000
Italian	1	12.5000
Hindi	1	12.5000
French	1	12.5000
English	1	12.5000
Arabic	1	12.5000

Insight:- For Persian language the share is 37.5% and for rest of the languages the share is 12.5% each.

| Case Study 1: Job Data Analysis |

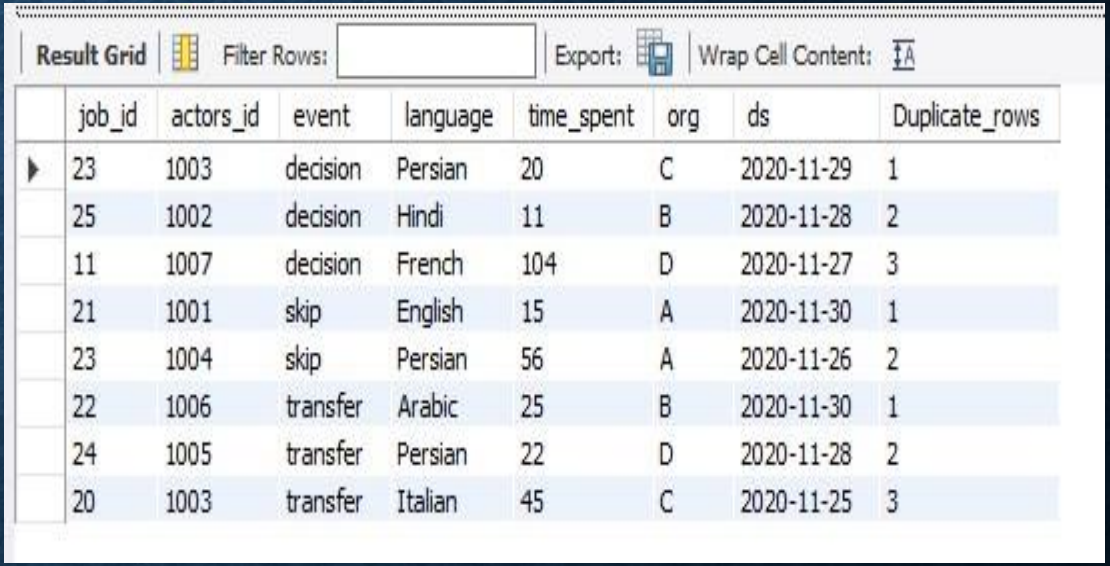
4. Duplicate Rows Detection: Display duplicate rows from the job_data table.

Input



```
with T as (select *, row_number() over (partition by event) as Duplicate_rows
from job_data)
select * from T where duplicate_rows >= 1;
```

Output



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

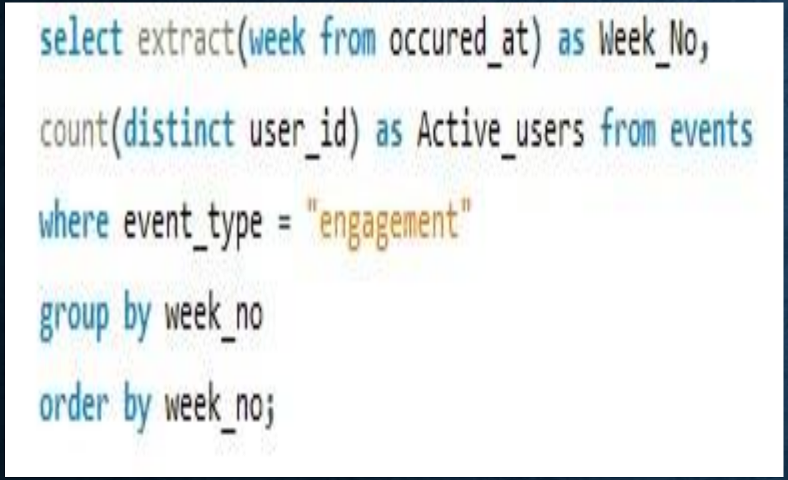
job_id	actors_id	event	language	time_spent	org	ds	Duplicate_rows
23	1003	decision	Persian	20	C	2020-11-29	1
25	1002	decision	Hindi	11	B	2020-11-28	2
11	1007	decision	French	104	D	2020-11-27	3
21	1001	skip	English	15	A	2020-11-30	1
23	1004	skip	Persian	56	A	2020-11-26	2
22	1006	transfer	Arabic	25	B	2020-11-30	1
24	1005	transfer	Persian	22	D	2020-11-28	2
20	1003	transfer	Italian	45	C	2020-11-25	3

Insights:- Job-id 23(1st) has 1 duplicate row, Job-id 25 has 2 duplicate rows, Job-id 11 has 3 duplicate rows, Job-id 21 has 1 duplicate row, Job-id 23(2nd) has 2 duplicate rows, Job-id 22 has 1 duplicate row, Job-id 24 has 2 duplicate rows, Job-id 20 has 3 duplicate row.

| Case Study 2: Investigating Metric Spike |

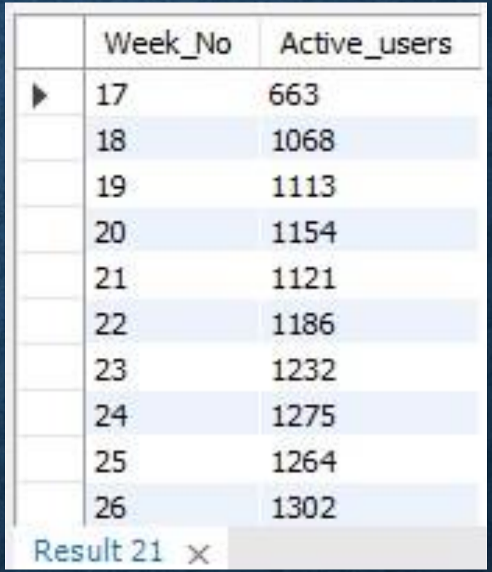
1. Weekly User Engagement: Calculate the weekly user engagement.

Input



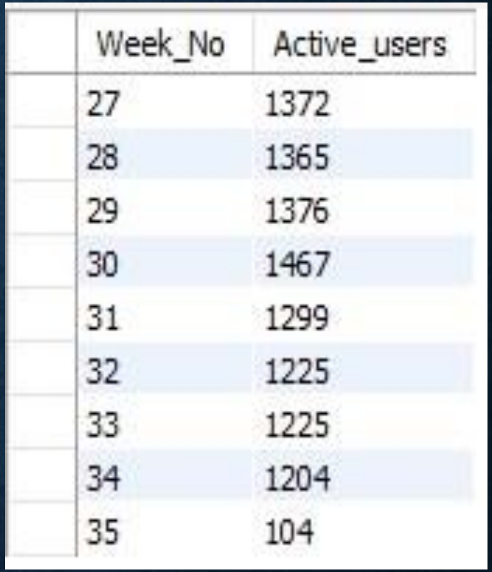
```
select extract(week from occured_at) as Week_No,
count(distinct user_id) as Active_users from events
where event_type = "engagement"
group by week_no
order by week_no;
```

Output



Week_No	Active_users
17	663
18	1068
19	1113
20	1154
21	1121
22	1186
23	1232
24	1275
25	1264
26	1302

Result 21 ×

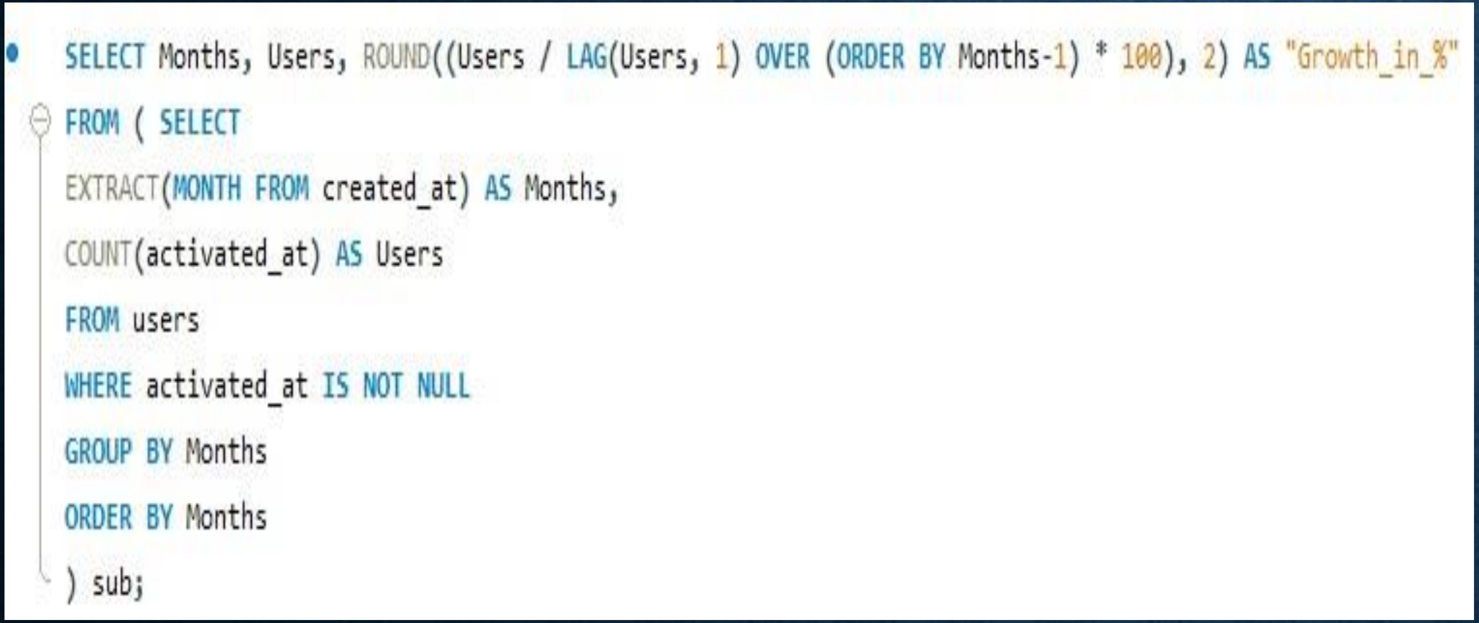


Week_No	Active_users
27	1372
28	1365
29	1376
30	1467
31	1299
32	1225
33	1225
34	1204
35	104

| Case Study 2: Investigating Metric Spike |

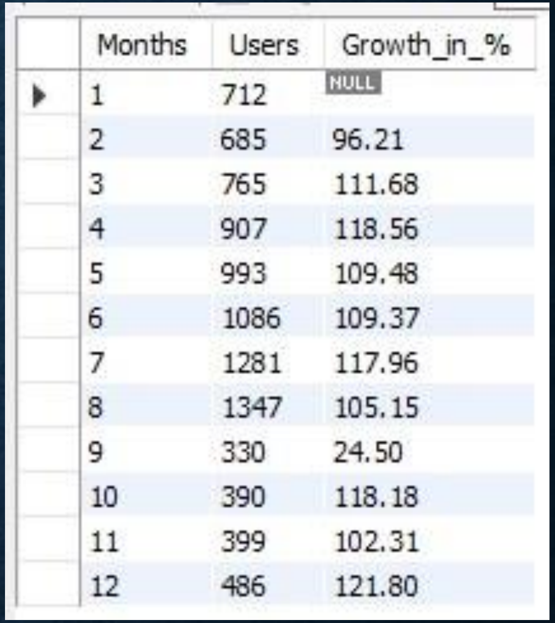
2. User Growth Analysis: Calculate the user growth for the product.

Input



```
SELECT Months, Users, ROUND((Users / LAG(Users, 1) OVER (ORDER BY Months-1) * 100), 2) AS "Growth_in_%"
FROM ( SELECT
EXTRACT(MONTH FROM created_at) AS Months,
COUNT(activated_at) AS Users
FROM users
WHERE activated_at IS NOT NULL
GROUP BY Months
ORDER BY Months
) sub;
```

Output

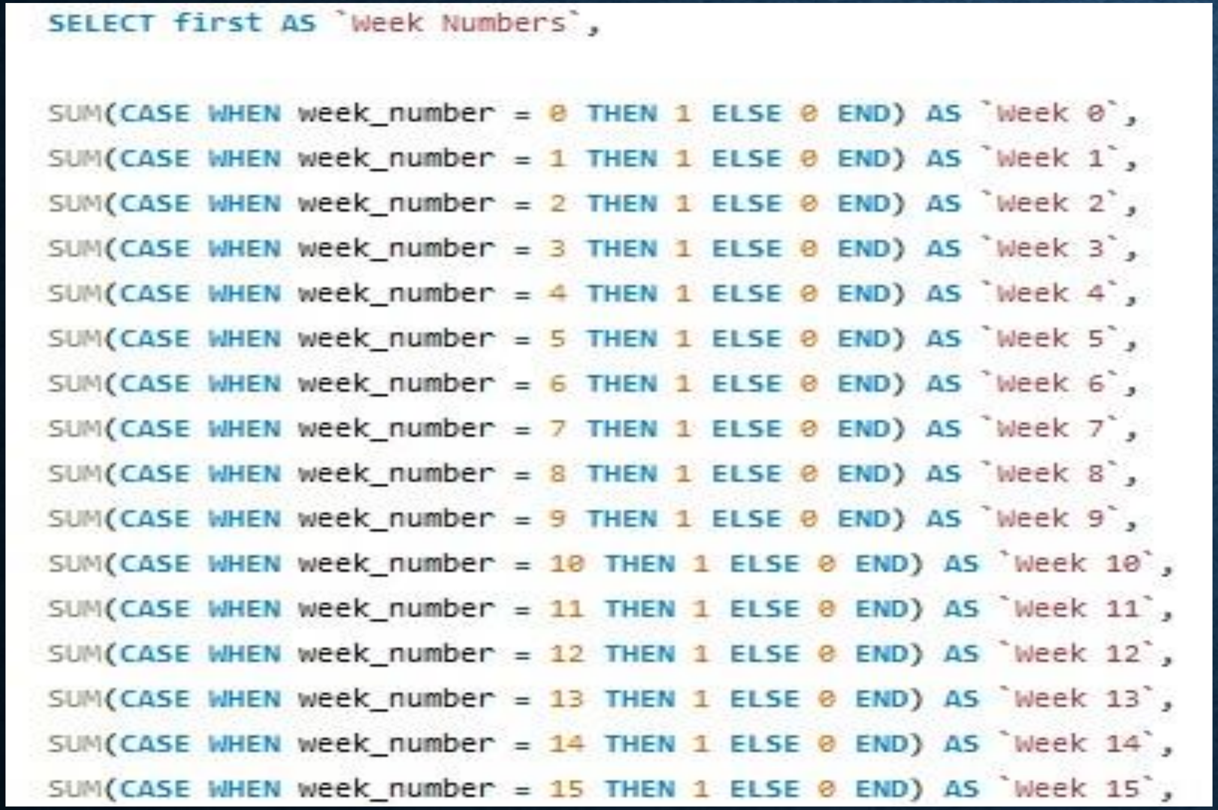


Months	Users	Growth_in_%
1	712	NULL
2	685	96.21
3	765	111.68
4	907	118.56
5	993	109.48
6	1086	109.37
7	1281	117.96
8	1347	105.15
9	330	24.50
10	390	118.18
11	399	102.31
12	486	121.80

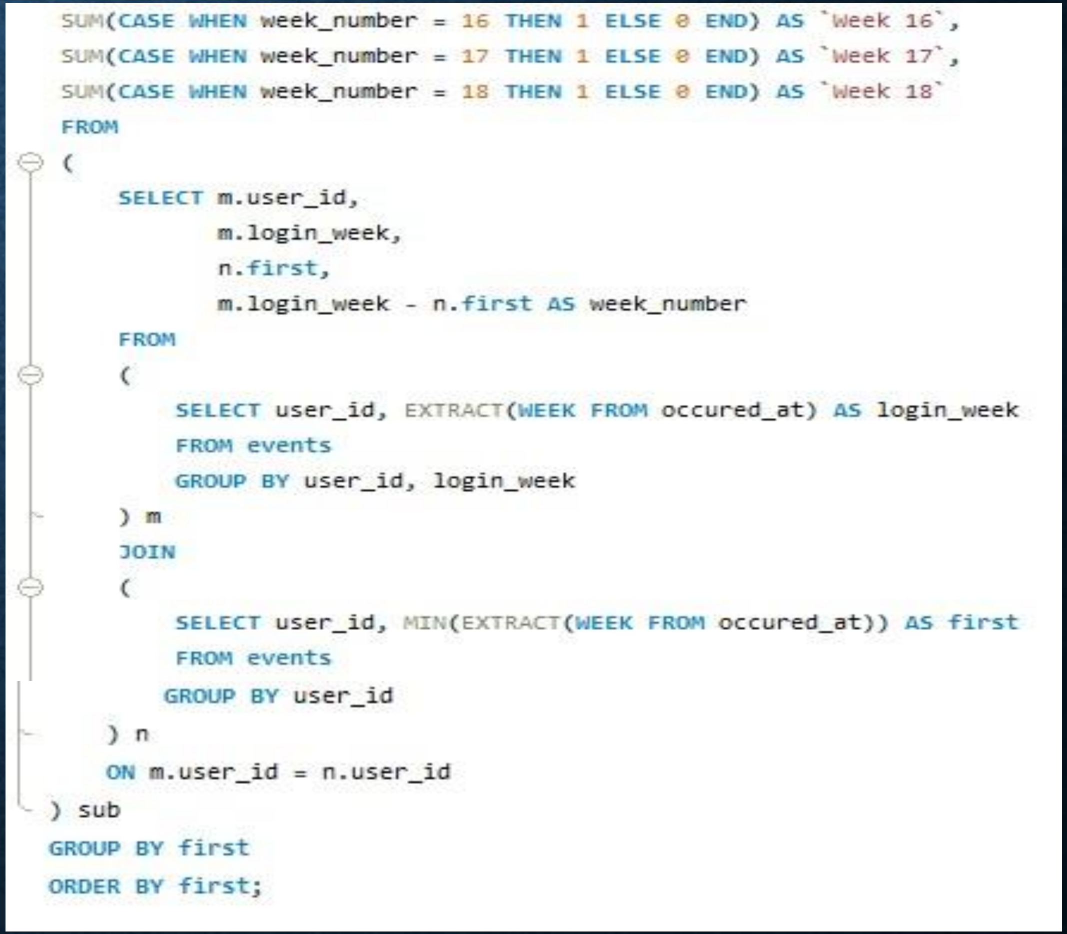
| Case Study 2: Investigating Metric Spike |

3. Weekly Retention Analysis: Calculate the weekly retention of users based on their sign-up cohort.

Input



```sql
SELECT first AS `Week Numbers`,

SUM(CASE WHEN week_number = 0 THEN 1 ELSE 0 END) AS `Week 0`,
SUM(CASE WHEN week_number = 1 THEN 1 ELSE 0 END) AS `Week 1`,
SUM(CASE WHEN week_number = 2 THEN 1 ELSE 0 END) AS `Week 2`,
SUM(CASE WHEN week_number = 3 THEN 1 ELSE 0 END) AS `Week 3`,
SUM(CASE WHEN week_number = 4 THEN 1 ELSE 0 END) AS `Week 4`,
SUM(CASE WHEN week_number = 5 THEN 1 ELSE 0 END) AS `Week 5`,
SUM(CASE WHEN week_number = 6 THEN 1 ELSE 0 END) AS `Week 6`,
SUM(CASE WHEN week_number = 7 THEN 1 ELSE 0 END) AS `Week 7`,
SUM(CASE WHEN week_number = 8 THEN 1 ELSE 0 END) AS `Week 8`,
SUM(CASE WHEN week_number = 9 THEN 1 ELSE 0 END) AS `Week 9`,
SUM(CASE WHEN week_number = 10 THEN 1 ELSE 0 END) AS `Week 10`,
SUM(CASE WHEN week_number = 11 THEN 1 ELSE 0 END) AS `Week 11`,
SUM(CASE WHEN week_number = 12 THEN 1 ELSE 0 END) AS `Week 12`,
SUM(CASE WHEN week_number = 13 THEN 1 ELSE 0 END) AS `Week 13`,
SUM(CASE WHEN week_number = 14 THEN 1 ELSE 0 END) AS `Week 14`,
SUM(CASE WHEN week_number = 15 THEN 1 ELSE 0 END) AS `Week 15`,
```

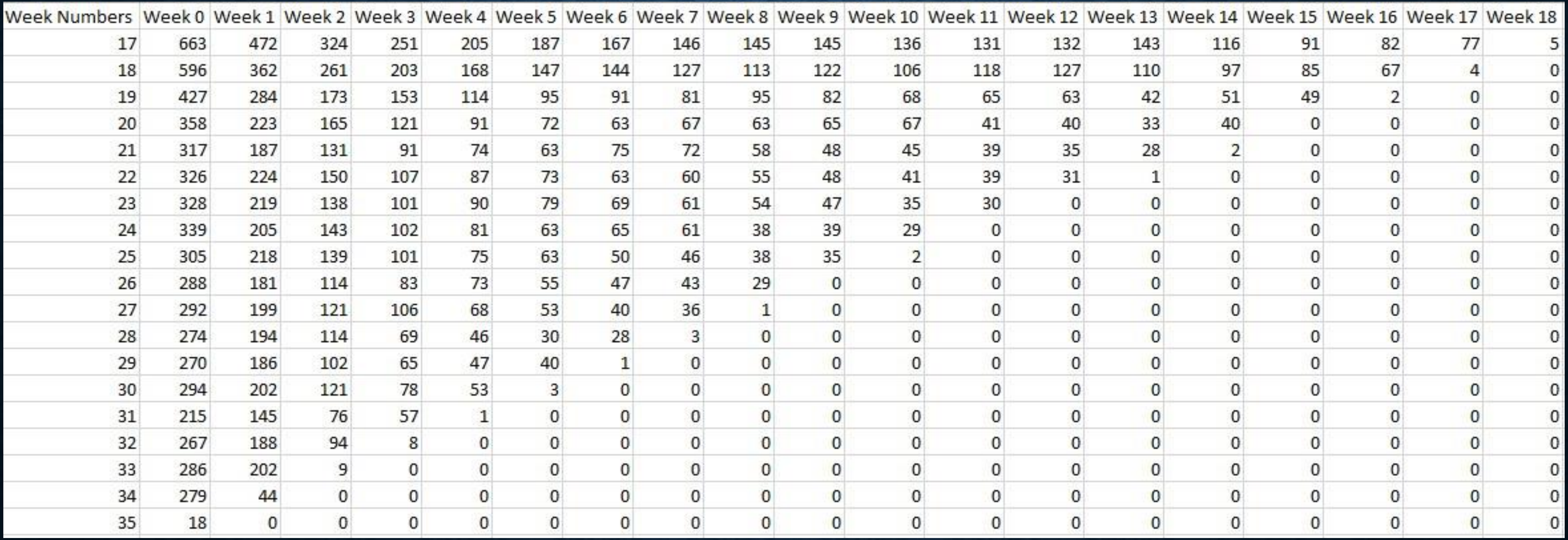


```sql
SUM(CASE WHEN week_number = 16 THEN 1 ELSE 0 END) AS `Week 16`,
SUM(CASE WHEN week_number = 17 THEN 1 ELSE 0 END) AS `Week 17`,
SUM(CASE WHEN week_number = 18 THEN 1 ELSE 0 END) AS `Week 18`
FROM
(
    SELECT m.user_id,
           m.login_week,
           n.first,
           m.login_week - n.first AS week_number
    FROM
    (
        SELECT user_id, EXTRACT(WEEK FROM occured_at) AS login_week
        FROM events
        GROUP BY user_id, login_week
    ) m
    JOIN
    (
        SELECT user_id, MIN(EXTRACT(WEEK FROM occured_at)) AS first
        FROM events
       GROUP BY user_id
   ) n
   ON m.user_id = n.user_id
) sub
GROUP BY first
ORDER BY first;
```

| Case Study 2: Investigating Metric Spike |

3. Weekly Retention Analysis: Calculate the weekly retention of users based on their sign-up cohort.

Output



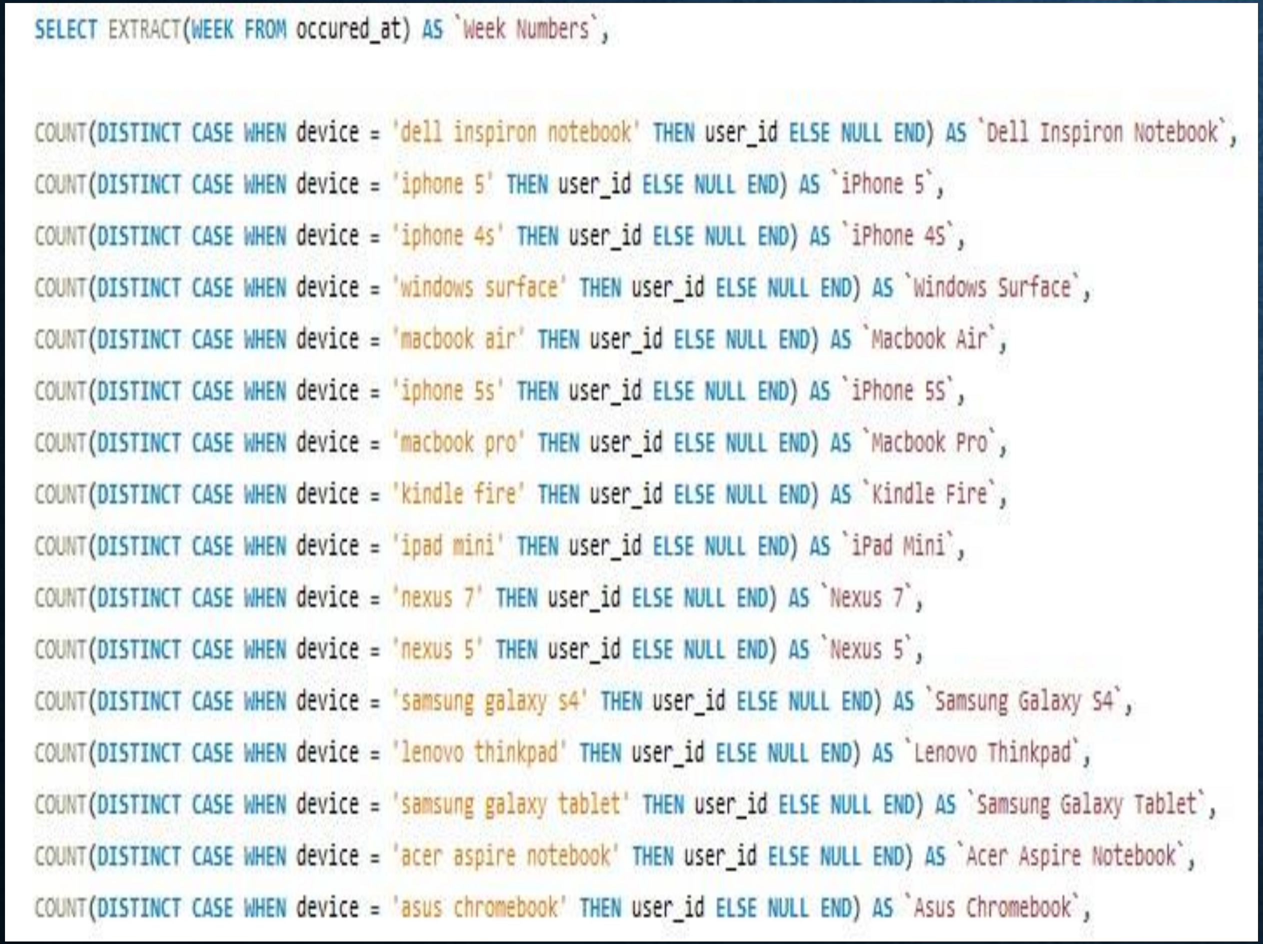
Week Numbers	Week 0	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16	Week 17	Week 18
17	663	472	324	251	205	187	167	146	145	145	136	131	132	143	116	91	82	77	5
18	596	362	261	203	168	147	144	127	113	122	106	118	127	110	97	85	67	4	0
19	427	284	173	153	114	95	91	81	95	82	68	65	63	42	51	49	2	0	0
20	358	223	165	121	91	72	63	67	63	65	67	41	40	33	40	0	0	0	0
21	317	187	131	91	74	63	75	72	58	48	45	39	35	28	2	0	0	0	0
22	326	224	150	107	87	73	63	60	55	48	41	39	31	1	0	0	0	0	0
23	328	219	138	101	90	79	69	61	54	47	35	30	0	0	0	0	0	0	0
24	339	205	143	102	81	63	65	61	38	39	29	0	0	0	0	0	0	0	0
25	305	218	139	101	75	63	50	46	38	35	2	0	0	0	0	0	0	0	0
26	288	181	114	83	73	55	47	43	29	0	0	0	0	0	0	0	0	0	0
27	292	199	121	106	68	53	40	36	1	0	0	0	0	0	0	0	0	0	0
28	274	194	114	69	46	30	28	3	0	0	0	0	0	0	0	0	0	0	0
29	270	186	102	65	47	40	1	0	0	0	0	0	0	0	0	0	0	0	0
30	294	202	121	78	53	3	0	0	0	0	0	0	0	0	0	0	0	0	0
31	215	145	76	57	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	267	188	94	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	286	202	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	279	44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* Due to Size and fitting issue the sql result table is exported in .csv format then the screen shot is taken.

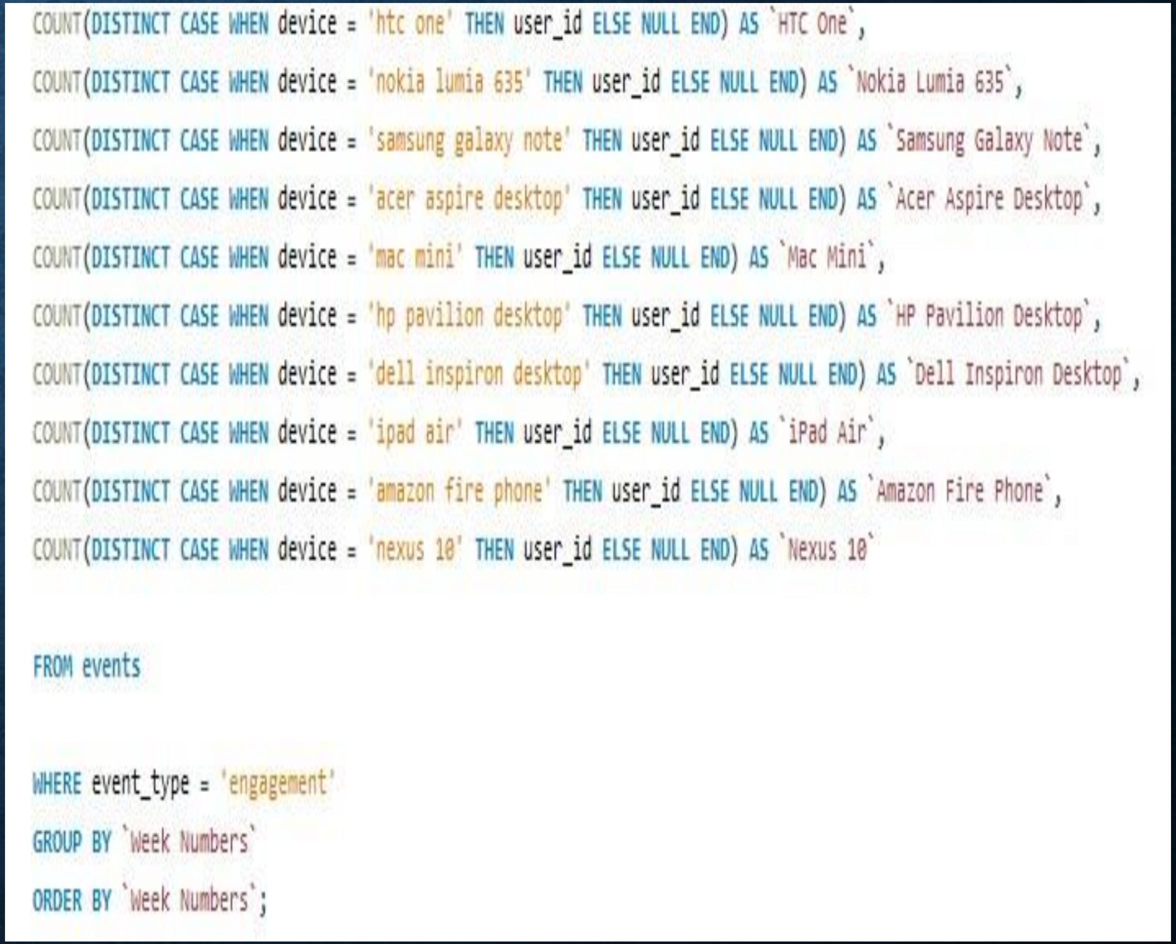
| Case Study 2: Investigating Metric Spike |

4. Weekly Engagement Per Device: Calculate the weekly engagement per device.

Input



```sql
SELECT EXTRACT(WEEK FROM occured_at) AS `Week Numbers`,

COUNT(DISTINCT CASE WHEN device = 'dell inspiron notebook' THEN user_id ELSE NULL END) AS `Dell Inspiron Notebook`,
COUNT(DISTINCT CASE WHEN device = 'iphone 5' THEN user_id ELSE NULL END) AS `iPhone 5`,
COUNT(DISTINCT CASE WHEN device = 'iphone 4s' THEN user_id ELSE NULL END) AS `iPhone 4S`,
COUNT(DISTINCT CASE WHEN device = 'windows surface' THEN user_id ELSE NULL END) AS `Windows Surface`,
COUNT(DISTINCT CASE WHEN device = 'macbook air' THEN user_id ELSE NULL END) AS `Macbook Air`,
COUNT(DISTINCT CASE WHEN device = 'iphone 5s' THEN user_id ELSE NULL END) AS `iPhone 5S`,
COUNT(DISTINCT CASE WHEN device = 'macbook pro' THEN user_id ELSE NULL END) AS `Macbook Pro`,
COUNT(DISTINCT CASE WHEN device = 'kindle fire' THEN user_id ELSE NULL END) AS `Kindle Fire`,
COUNT(DISTINCT CASE WHEN device = 'ipad mini' THEN user_id ELSE NULL END) AS `iPad Mini`,
COUNT(DISTINCT CASE WHEN device = 'nexus 7' THEN user_id ELSE NULL END) AS `Nexus 7`,
COUNT(DISTINCT CASE WHEN device = 'nexus 5' THEN user_id ELSE NULL END) AS `Nexus 5`,
COUNT(DISTINCT CASE WHEN device = 'samsung galaxy s4' THEN user_id ELSE NULL END) AS `Samsung Galaxy S4`,
COUNT(DISTINCT CASE WHEN device = 'lenovo thinkpad' THEN user_id ELSE NULL END) AS `Lenovo Thinkpad`,
COUNT(DISTINCT CASE WHEN device = 'samsung galaxy tablet' THEN user_id ELSE NULL END) AS `Samsung Galaxy Tablet`,
COUNT(DISTINCT CASE WHEN device = 'acer aspire notebook' THEN user_id ELSE NULL END) AS `Acer Aspire Notebook`,
COUNT(DISTINCT CASE WHEN device = 'asus chromebook' THEN user_id ELSE NULL END) AS `Asus Chromebook`,
```

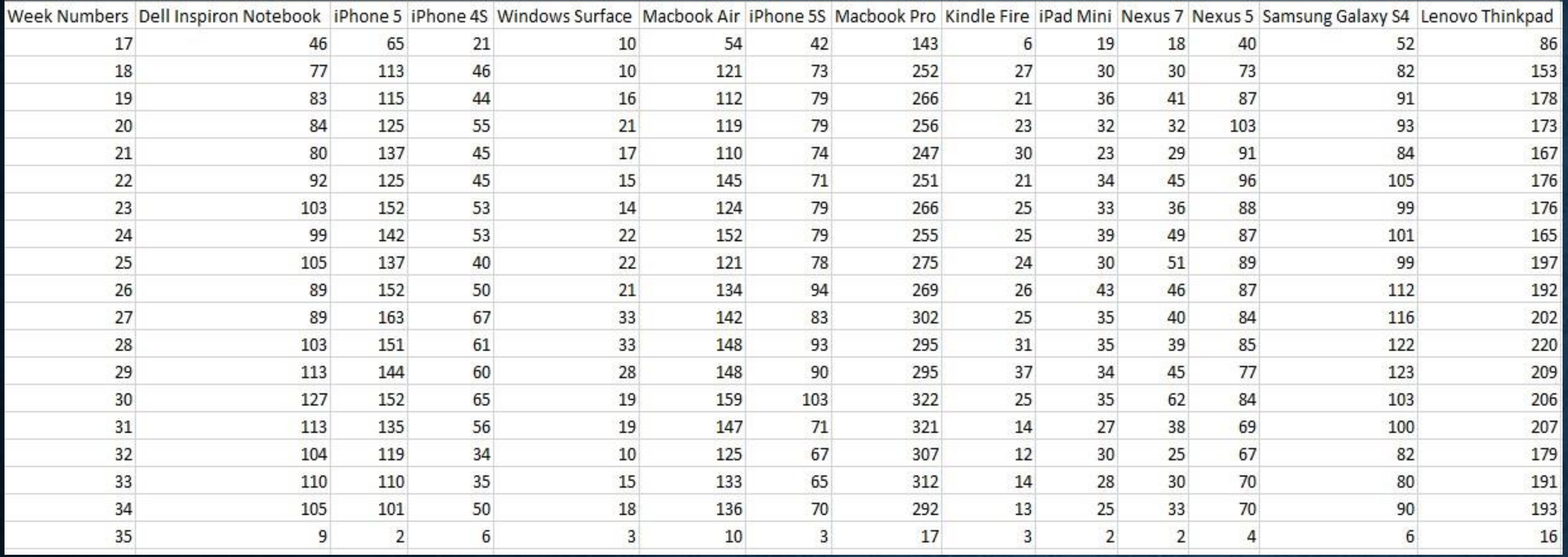


```sql
COUNT(DISTINCT CASE WHEN device = 'htc one' THEN user_id ELSE NULL END) AS `HTC One`,
COUNT(DISTINCT CASE WHEN device = 'nokia lumia 635' THEN user_id ELSE NULL END) AS `Nokia Lumia 635`,
COUNT(DISTINCT CASE WHEN device = 'samsung galaxy note' THEN user_id ELSE NULL END) AS `Samsung Galaxy Note`,
COUNT(DISTINCT CASE WHEN device = 'acer aspire desktop' THEN user_id ELSE NULL END) AS `Acer Aspire Desktop`,
COUNT(DISTINCT CASE WHEN device = 'mac mini' THEN user_id ELSE NULL END) AS `Mac Mini`,
COUNT(DISTINCT CASE WHEN device = 'hp pavilion desktop' THEN user_id ELSE NULL END) AS `HP Pavilion Desktop`,
COUNT(DISTINCT CASE WHEN device = 'dell inspiron desktop' THEN user_id ELSE NULL END) AS `Dell Inspiron Desktop`,
COUNT(DISTINCT CASE WHEN device = 'ipad air' THEN user_id ELSE NULL END) AS `iPad Air`,
COUNT(DISTINCT CASE WHEN device = 'amazon fire phone' THEN user_id ELSE NULL END) AS `Amazon Fire Phone`,
COUNT(DISTINCT CASE WHEN device = 'nexus 10' THEN user_id ELSE NULL END) AS `Nexus 10`


FROM events


WHERE event_type = 'engagement'
GROUP BY `Week Numbers`
ORDER BY `Week Numbers`;
```

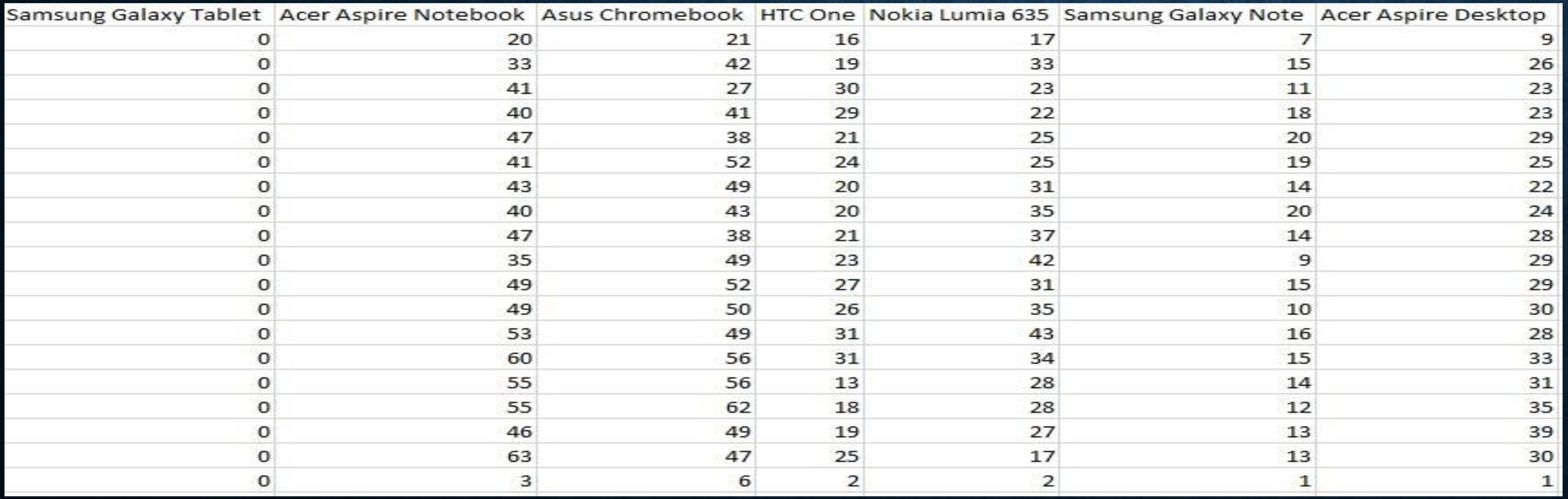
| Case Study 2: Investigating Metric Spike |

4. Weekly Engagement Per Device: Calculate the weekly engagement per device.

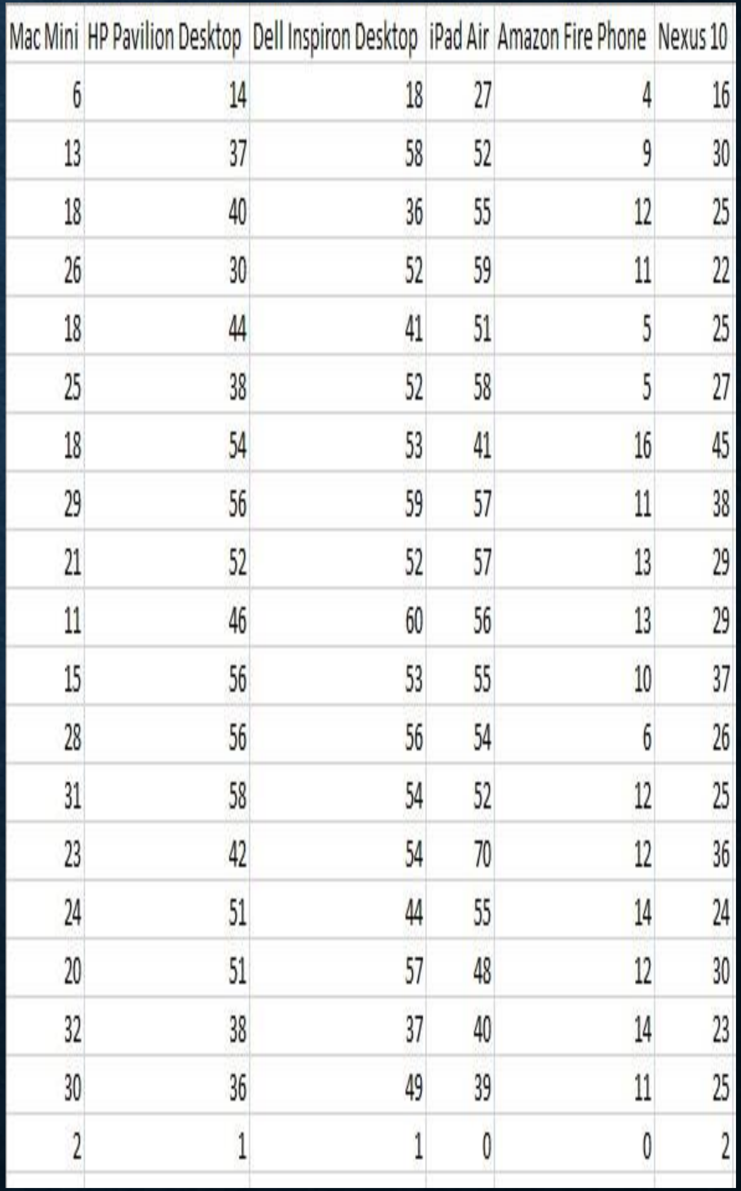
Output



Week Numbers	Dell Inspiron Notebook	iPhone 5	iPhone 4S	Windows Surface	Macbook Air	iPhone 5S	Macbook Pro	Kindle Fire	iPad Mini	Nexus 7	Nexus 5	Samsung Galaxy S4	Lenovo Thinkpad
17	46	65	21	10	54	42	143	6	19	18	40	52	86
18	77	113	46	10	121	73	252	27	30	30	73	82	153
19	83	115	44	16	112	79	266	21	36	41	87	91	178
20	84	125	55	21	119	79	256	23	32	32	103	93	173
21	80	137	45	17	110	74	247	30	23	29	91	84	167
22	92	125	45	15	145	71	251	21	34	45	96	105	176
23	103	152	53	14	124	79	266	25	33	36	88	99	176
24	99	142	53	22	152	79	255	25	39	49	87	101	165
25	105	137	40	22	121	78	275	24	30	51	89	99	197
26	89	152	50	21	134	94	269	26	43	46	87	112	192
27	89	163	67	33	142	83	302	25	35	40	84	116	202
28	103	151	61	33	148	93	295	31	35	39	85	122	220
29	113	144	60	28	148	90	295	37	34	45	77	123	209
30	127	152	65	19	159	103	322	25	35	62	84	103	206
31	113	135	56	19	147	71	321	14	27	38	69	100	207
32	104	119	34	10	125	67	307	12	30	25	67	82	179
33	110	110	35	15	133	65	312	14	28	30	70	80	191
34	105	101	50	18	136	70	292	13	25	33	70	90	193
35	9	2	6	3	10	3	17	3	2	2	4	6	16



Samsung Galaxy Tablet	Acer Aspire Notebook	Asus Chromebook	HTC One	Nokia Lumia 635	Samsung Galaxy Note	Acer Aspire Desktop
0	20	21	16	17	7	9
0	33	42	19	33	15	26
0	41	27	30	23	11	23
0	40	41	29	22	18	23
0	47	38	21	25	20	29
0	41	52	24	25	19	25
0	43	49	20	31	14	22
0	40	43	20	35	20	24
0	47	38	21	37	14	28
0	35	49	23	42	9	29
0	49	52	27	31	15	29
0	49	50	26	35	10	30
0	53	49	31	43	16	28
0	60	56	31	34	15	33
0	55	56	13	28	14	31
0	55	62	18	28	12	35
0	46	49	19	27	13	39
0	63	47	25	17	13	30
0	3	6	2	2	1	1

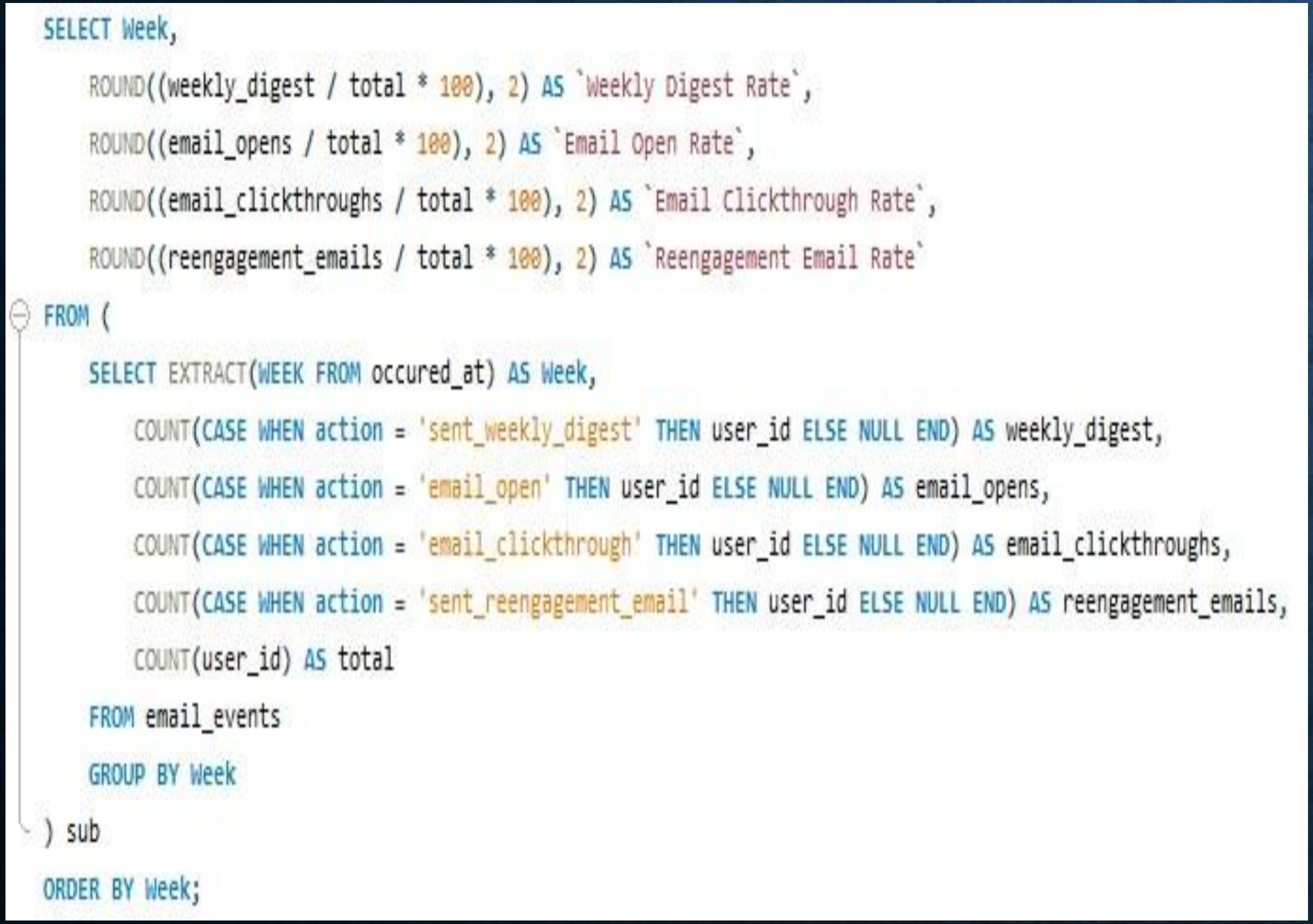


Mac Mini	HP Pavilion Desktop	Dell Inspiron Desktop	iPad Air	Amazon Fire Phone	Nexus 10
6	14	18	27	4	16
13	37	58	52	9	30
18	40	36	55	12	25
26	30	52	59	11	22
18	44	41	51	5	25
25	38	52	58	5	27
18	54	53	41	16	45
29	56	59	57	11	38
21	52	52	57	13	29
11	46	60	56	13	29
15	56	53	55	10	37
28	56	56	54	6	26
31	58	54	52	12	25
23	42	54	70	12	36
24	51	44	55	14	24
20	51	57	48	12	30
32	38	37	40	14	23
30	36	49	39	11	25
2	1	1	0	0	2

| Case Study 2: Investigating Metric Spike |

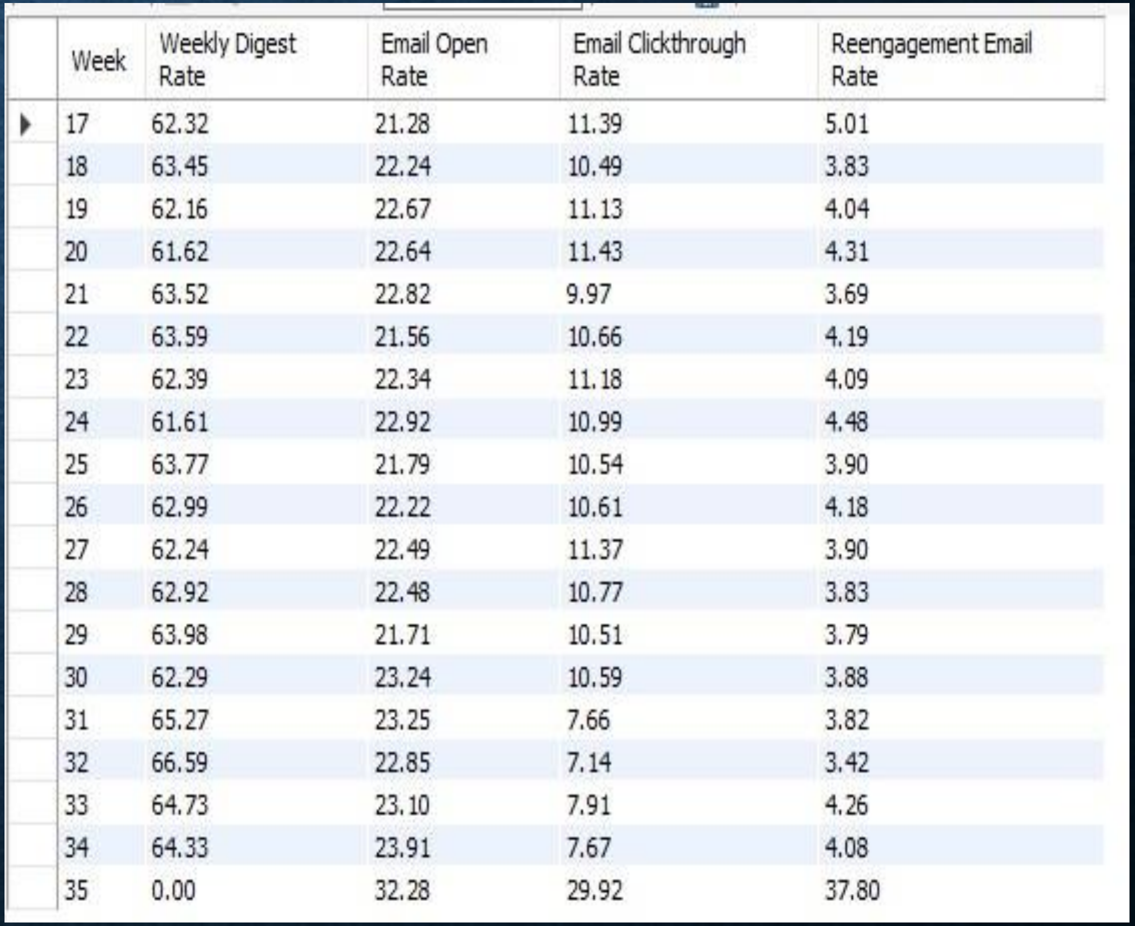
5. Email Engagement Analysis: Calculate the email engagement metrics.

Input



```
SELECT Week,
    ROUND((weekly_digest / total * 100), 2) AS `Weekly Digest Rate`,
    ROUND((email_opens / total * 100), 2) AS `Email Open Rate`,
    ROUND((email_clickthroughs / total * 100), 2) AS `Email Clickthrough Rate`,
    ROUND((reengagement_emails / total * 100), 2) AS `Reengagement Email Rate`
FROM (
    SELECT EXTRACT(WEEK FROM occured_at) AS Week,
        COUNT(CASE WHEN action = 'sent_weekly_digest' THEN user_id ELSE NULL END) AS weekly_digest,
        COUNT(CASE WHEN action = 'email_open' THEN user_id ELSE NULL END) AS email_opens,
        COUNT(CASE WHEN action = 'email_clickthrough' THEN user_id ELSE NULL END) AS email_clickthroughs,
        COUNT(CASE WHEN action = 'sent_reengagement_email' THEN user_id ELSE NULL END) AS reengagement_emails,
        COUNT(user_id) AS total
    FROM email_events
    GROUP BY Week
) sub
ORDER BY Week;
```

Output



Week	Weekly Digest Rate	Email Open Rate	Email Clickthrough Rate	Reengagement Email Rate
17	62.32	21.28	11.39	5.01
18	63.45	22.24	10.49	3.83
19	62.16	22.67	11.13	4.04
20	61.62	22.64	11.43	4.31
21	63.52	22.82	9.97	3.69
22	63.59	21.56	10.66	4.19
23	62.39	22.34	11.18	4.09
24	61.61	22.92	10.99	4.48
25	63.77	21.79	10.54	3.90
26	62.99	22.22	10.61	4.18
27	62.24	22.49	11.37	3.90
28	62.92	22.48	10.77	3.83
29	63.98	21.71	10.51	3.79
30	62.29	23.24	10.59	3.88
31	65.27	23.25	7.66	3.82
32	66.59	22.85	7.14	3.42
33	64.73	23.10	7.91	4.26
34	64.33	23.91	7.67	4.08
35	0.00	32.28	29.92	37.80