JULY 2024

OPERATION ANALYTICS AND INVESTIGATING METRIC SPIKE

REPORT

Gunisha Chopra
DATA ANALYST

PROJECT 3

OPERATION ANALYTICS AND INVESTIGATING METRIC SPIKE

Advanced SQL

CASE STUDY 1

Project description:

Operational analytics is a category of business analytics that enables continuous monitoring of data and discovery of insights to help teams make better decisions on the fly. it processes real-time signals from various parts of a business to offer instant feedback. this analysis is further used to predict a company's fortune's overall growth or decline. It means better automation, better understanding between cross-functional teams, and more effective workflows. Operation Analytics is the analysis done for a company’s complete end-to-end operations.

the company then finds the areas in which it must improve.

SPIKE METRIC:

It is used by companies to analyse whether the user is engaged with our product or not. We also calculate its Lifetime value and see how many users are retained and how much is churn. Investigating metric spikes is also an important part of operation analytics as a Data Analyst you must be able to understand or make other teams understand questions like- Why is there a dip in daily engagement? Why have sales taken a dip? Etc. Questions like these must be answered daily and for that it is very important to investigate metric spike.

I am working as a Data Analyst Lead and am provided with different data
sets, and tables from which I must derive certain insights. And answer the questions asked by different departments.

APPROACH:

I have been provided with two data sets. The first one is a customer-care centre database which includes job ID, actor ID, event, language etc. And the second one is regarding an online platform like WhatsApp where I must create its Investigating metric spike.

Now I will create separate databases for both cases and then perform SQL queries to get insights on problems that my team or certain departments of the company may be facing.

TECH-STACK USED:

INSIGHTS:

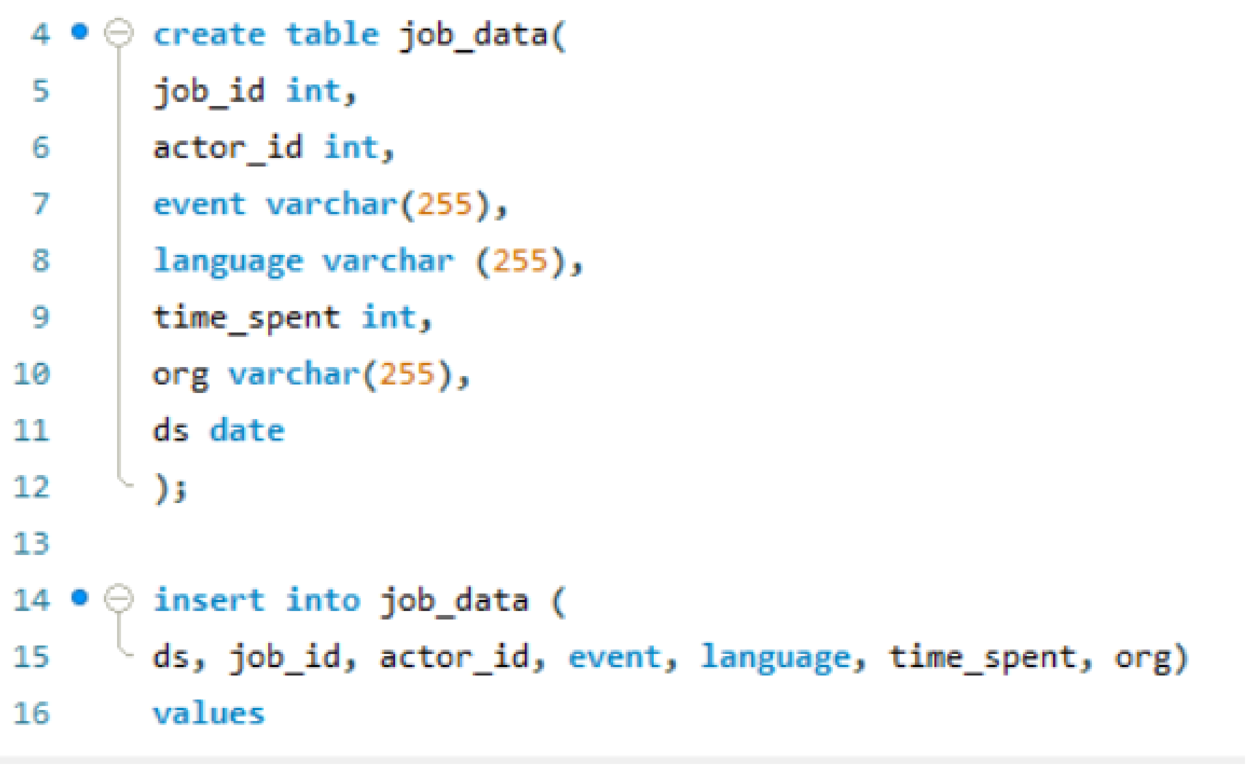
Case Study 1 (Job Data) job_data

1. job_id: unique identifier of jobs
2. actor_id: unique identifier of actor
3. event: decision/skip/transfer

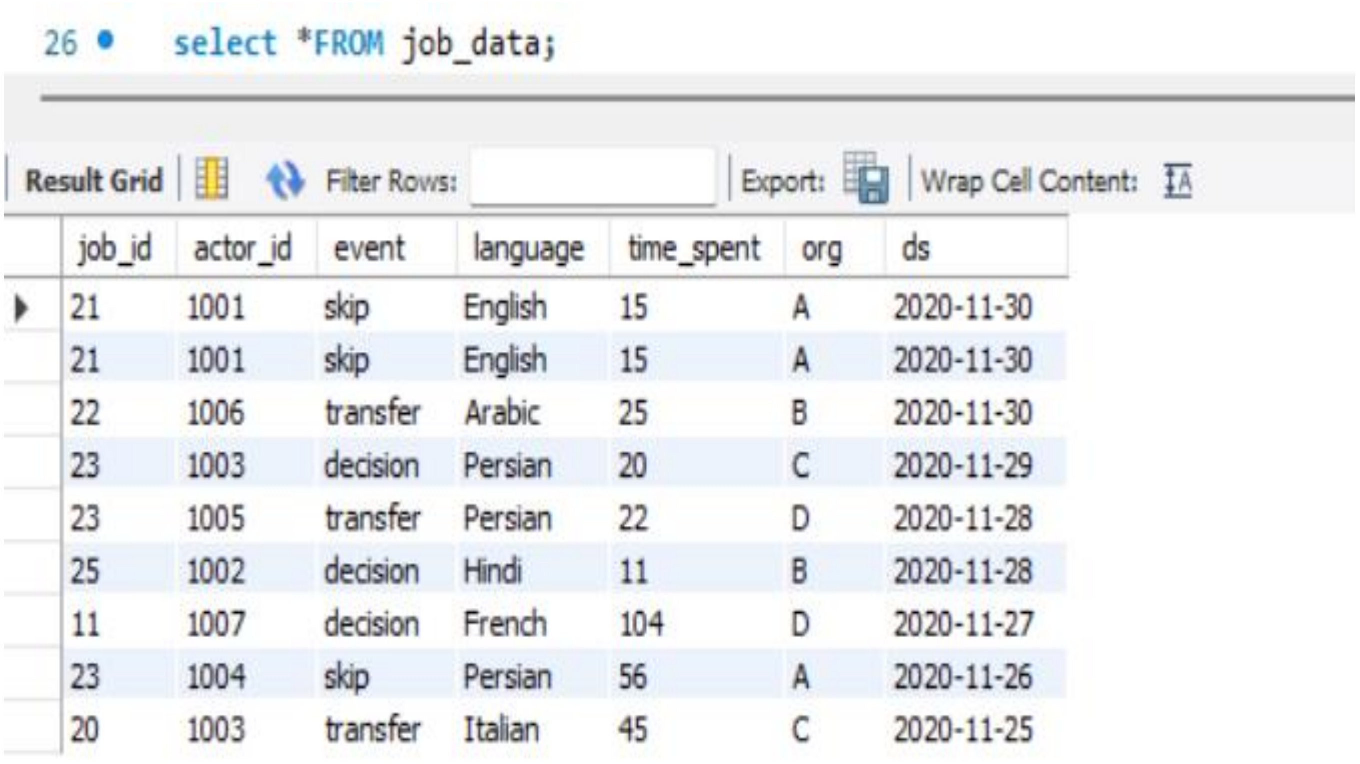
4. language: language of the content
5. time_spent: time spent to review the job in seconds
6. org: organisation of the actor
7. ds: date in the yyyy/mm/dd format.

It is stored in the form of text and we use Presto to run. No need for a date function

CREATING A DATA BASE:



```sql
create table job_data(
job_id int,
actor_id int,
event varchar(255),
language varchar (255),
time_spent int,
org varchar(255),
ds date
);

insert into job_data (
ds, job_id, actor_id, event, language, time_spent, org)
values
```



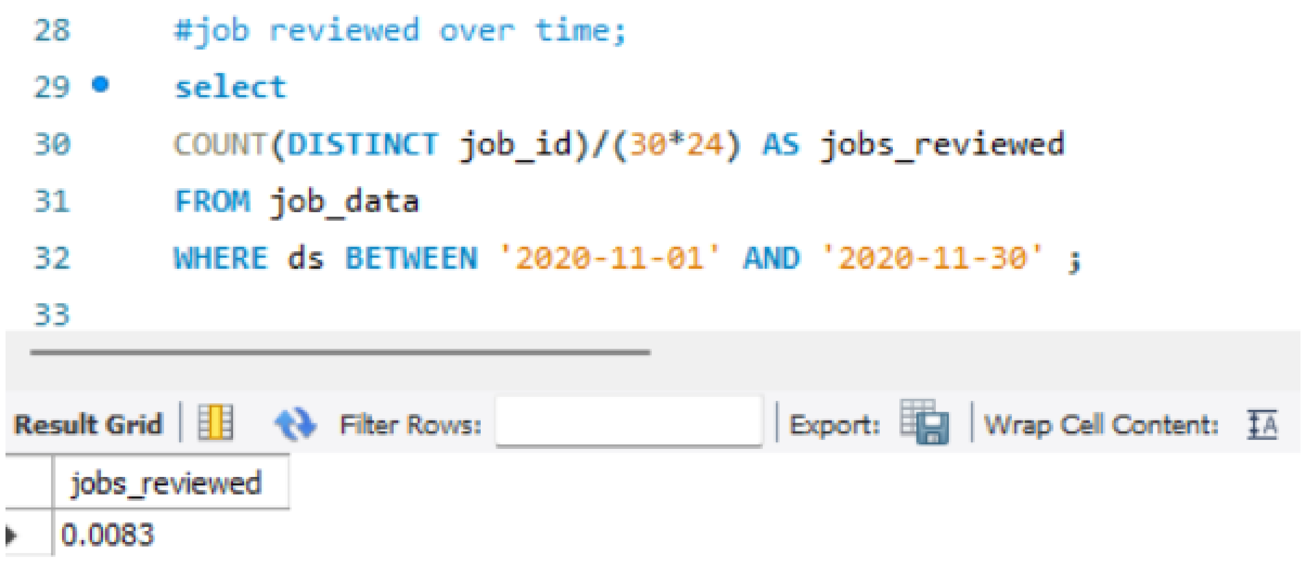
```
select *FROM job_data;
```

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

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

<u>**TASKS**</u>

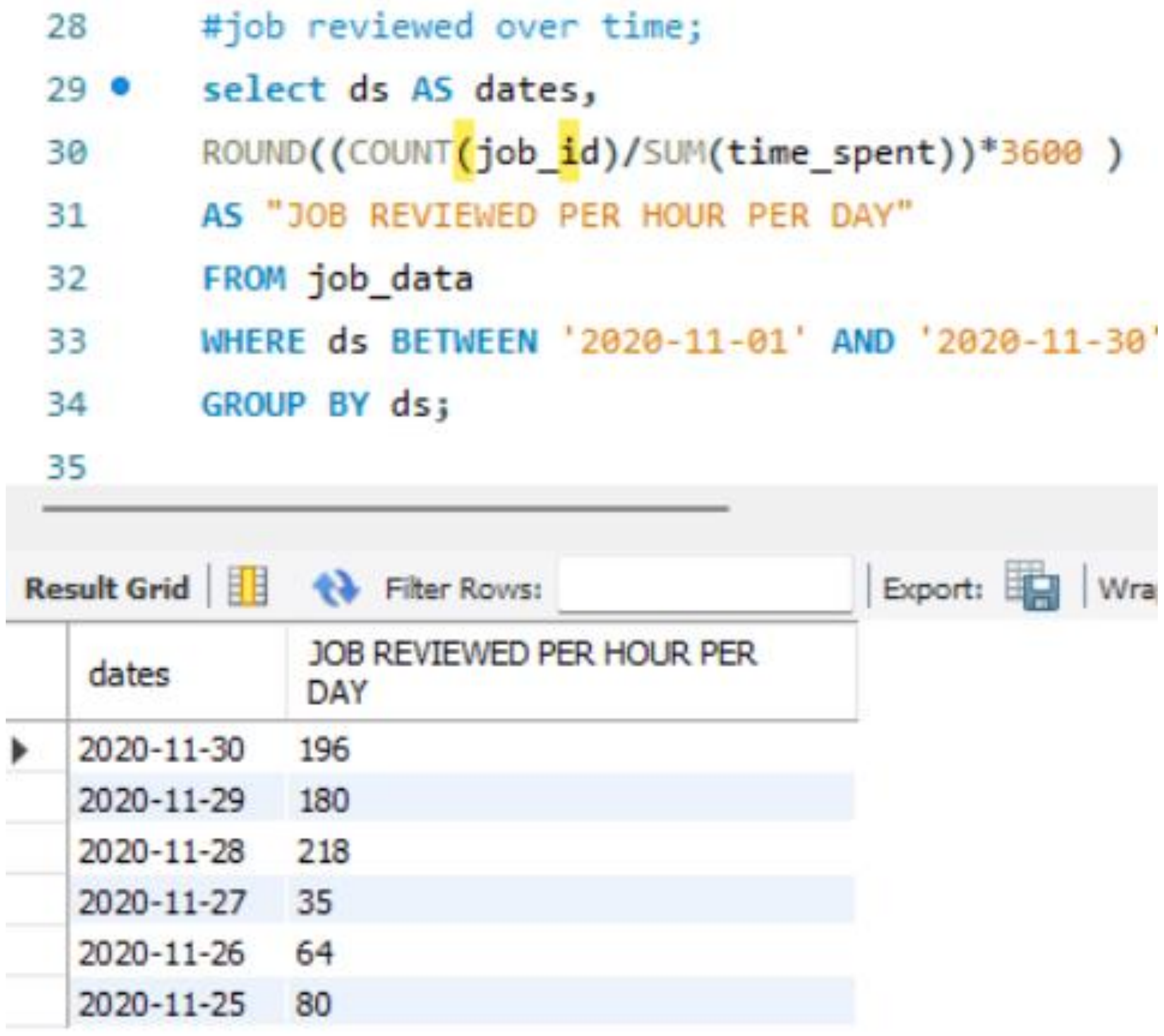
1.) <u>JOBS REVIEWED OVER TIME:</u> Number of jobs reviewed over time i.e. calculate the number of jobs reviewed per hour per day of November 2020.



```sql
#job reviewed over time;
select
COUNT(DISTINCT job_id)/(30*24) AS jobs_reviewed
FROM job_data
WHERE ds BETWEEN '2020-11-01' AND '2020-11-30' ;
```

jobs_reviewed
0.0083

#jobs reviewed each day of November:

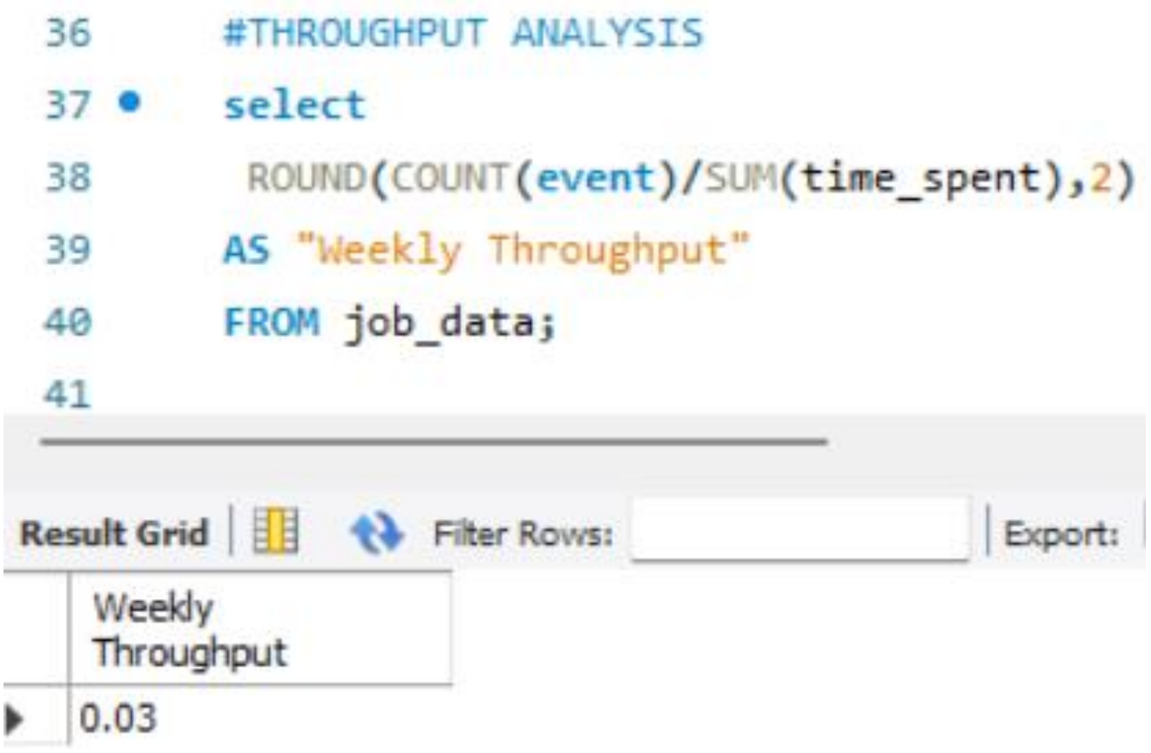


```sql
#job reviewed over time;
select ds AS dates,
ROUND((COUNT(job_id)/SUM(time_spent))*3600 )
AS "JOB REVIEWED PER HOUR PER DAY"
FROM job_data
WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY ds;
```

dates	JOB REVIEWED PER HOUR PER DAY
2020-11-30	196
2020-11-29	180
2020-11-28	218
2020-11-27	35
2020-11-26	64
2020-11-25	80

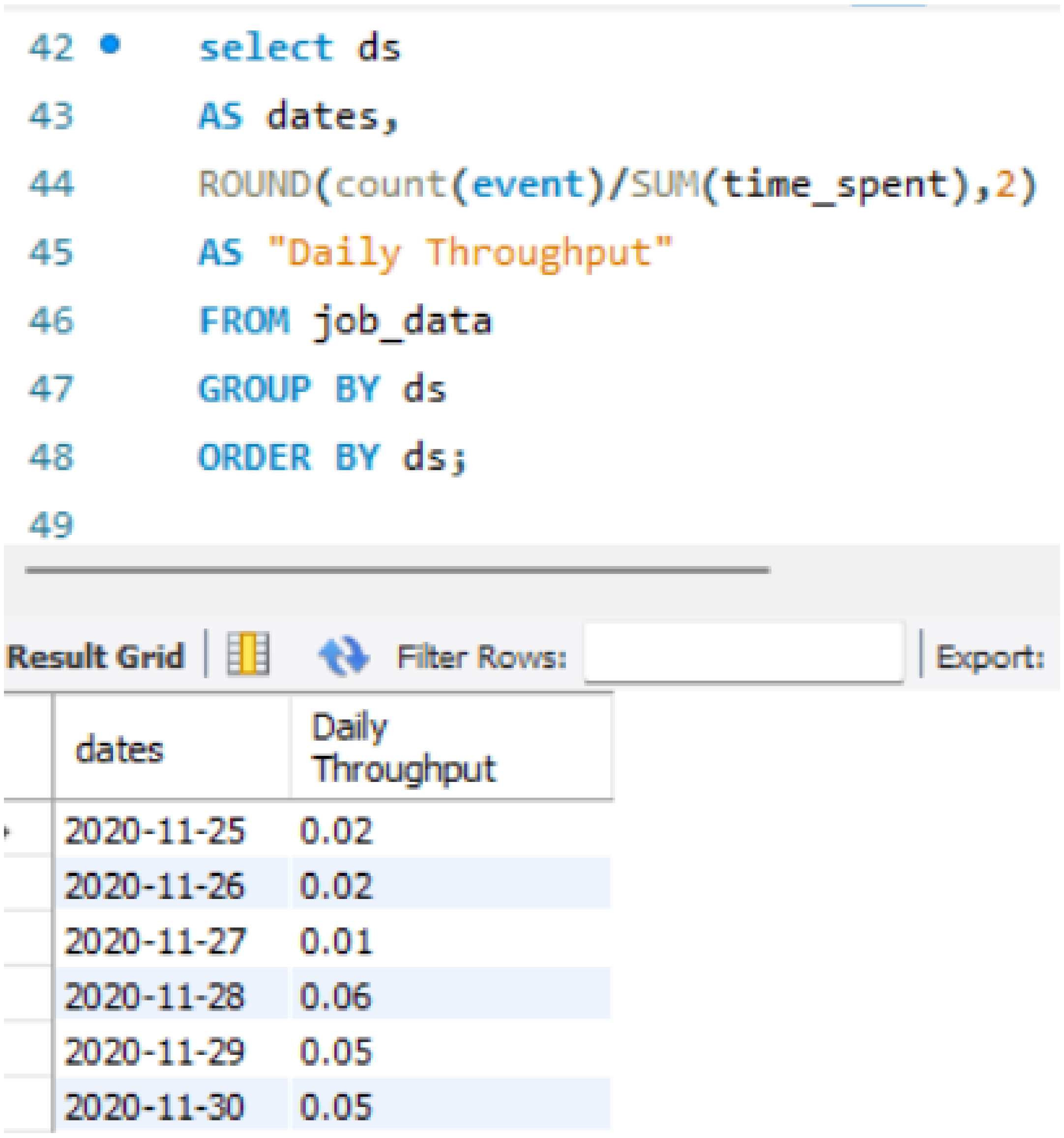
2.)**<u>THROUGHPUT ANALYSIS:</u>** Number of events happening per second.

#task is to calculate the 7-day rolling average of throughput.
For throughput, do you prefer daily metric or 7-day rolling and why?



```
#THROUGHPUT ANALYSIS
select
 ROUND(COUNT(event)/SUM(time_spent),2)
AS "Weekly Throughput"
FROM job_data;
```

```
Weekly
Throughput
0.03
```



```
select ds
AS dates,
ROUND(count(event)/SUM(time_spent),2)
AS "Daily Throughput"
FROM job_data
GROUP BY ds
ORDER BY ds;
```

dates	Daily Throughput
2020-11-25	0.02
2020-11-26	0.02
2020-11-27	0.01
2020-11-28	0.06
2020-11-29	0.05
2020-11-30	0.05

#On the date 2020-11-28 the Throughput is highest i.e. **0.06**.

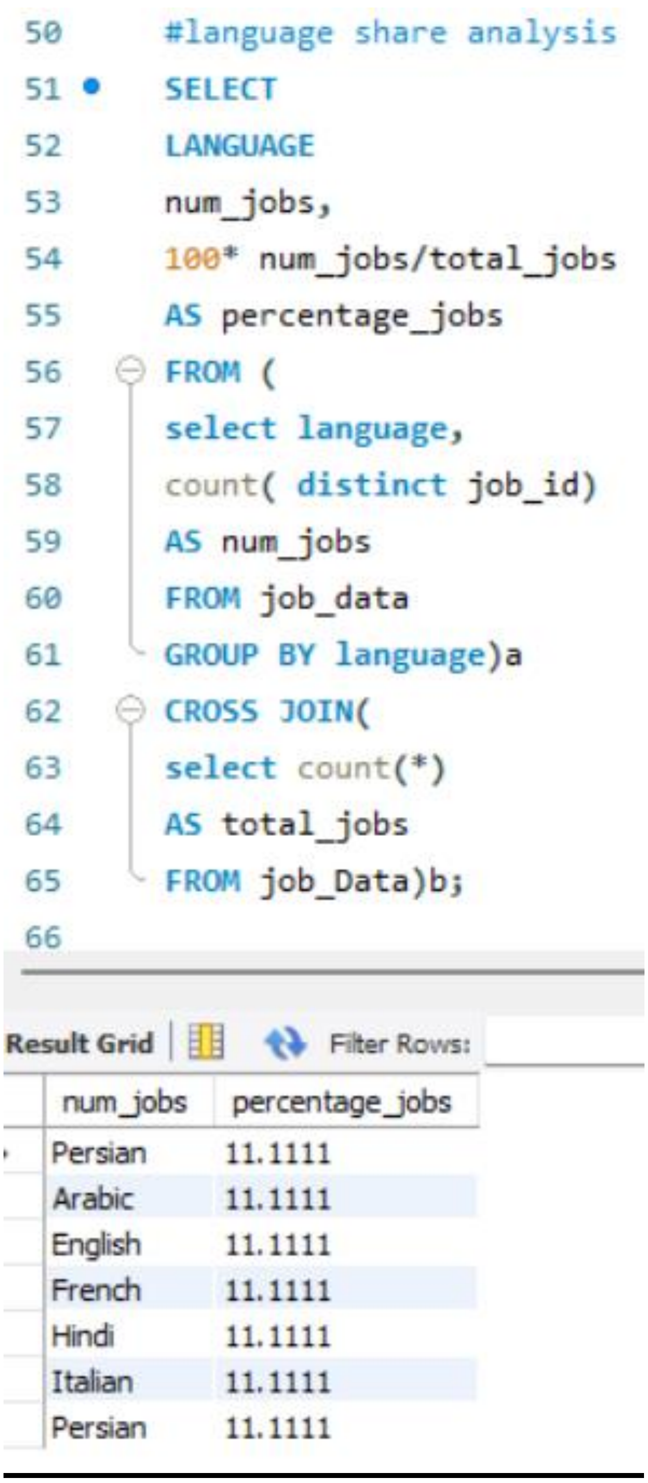
A 7-day moving average rolling average or throughput is a short-term
trend indicator. It is quite simply the average closing prices of the last

seven trading days. It is used because it appears to be particularly more
reactive to price changes. moving average to decide when to buy or sell.
moving averages at their simplest are trend indicators and very useful in
trending markets.

Metrics will always go up and down on a weekly and daily basis. You'll get numbers faster every day or minute if you want, As a result, rolling metrics are superb at showing that your metrics are trending up or down daily.

3.) **<u>Language Share Analysis</u>:** the percentage share of each language in the last 30 days.

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



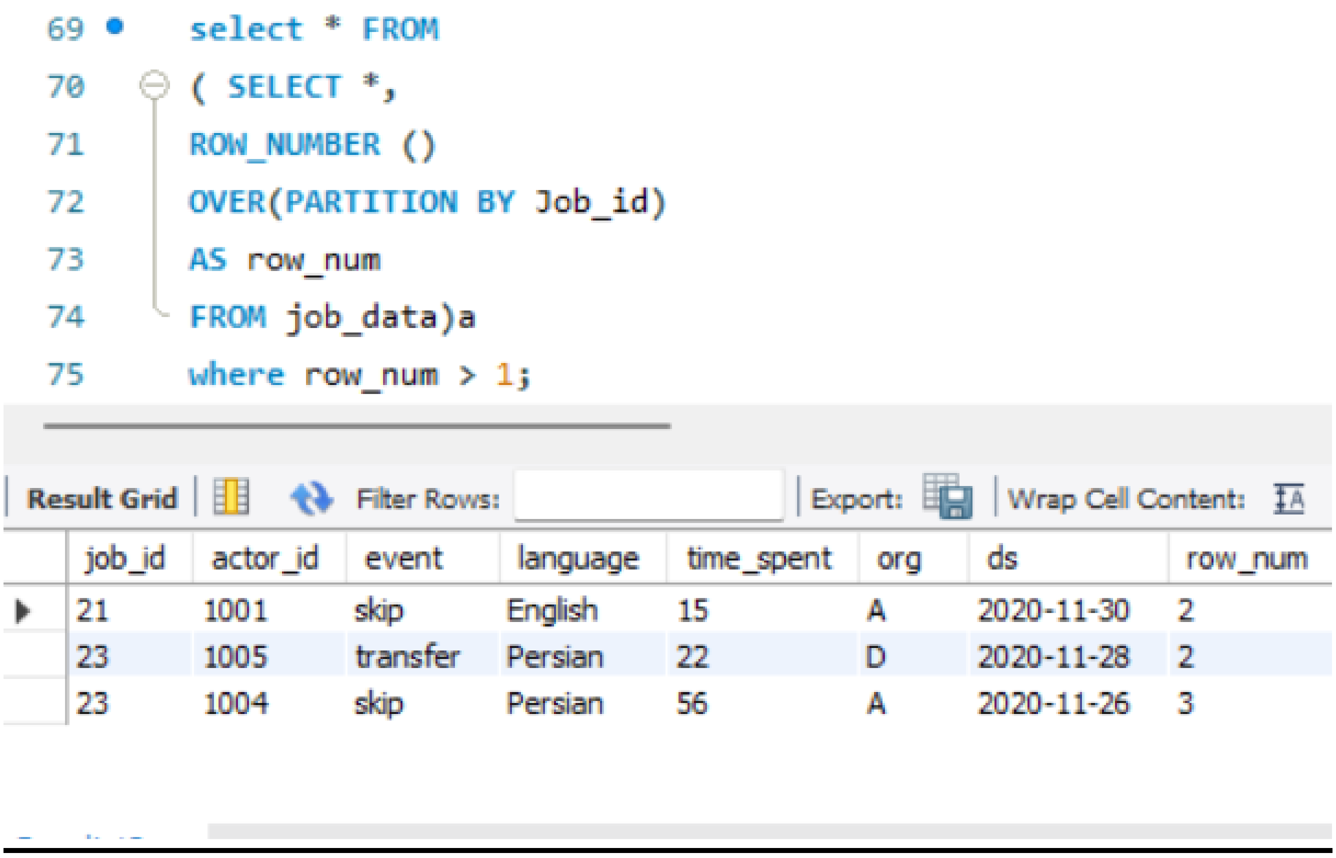
```
#language share analysis
SELECT
LANGUAGE
num_jobs,
100* num_jobs/total_jobs
AS percentage_jobs
FROM (
select language,
count( distinct job_id)
AS num_jobs
FROM job_data
GROUP BY language)a
CROSS JOIN(
select count(*)
AS total_jobs
FROM job_Data)b;
```

Result Grid | Filter Rows:

num_jobs	percentage_jobs
Persian	11.1111
Arabic	11.1111
English	11.1111
French	11.1111
Hindi	11.1111
Italian	11.1111
Persian	11.1111

4.) <u>**DUPLICATE ROWS DETECTION:**</u>

Identifying duplicate rows from the job_data table.



```sql
select * FROM
( SELECT *,
ROW_NUMBER ()
OVER(PARTITION BY Job_id)
AS row_num
FROM job_data)a
where row_num > 1;
```

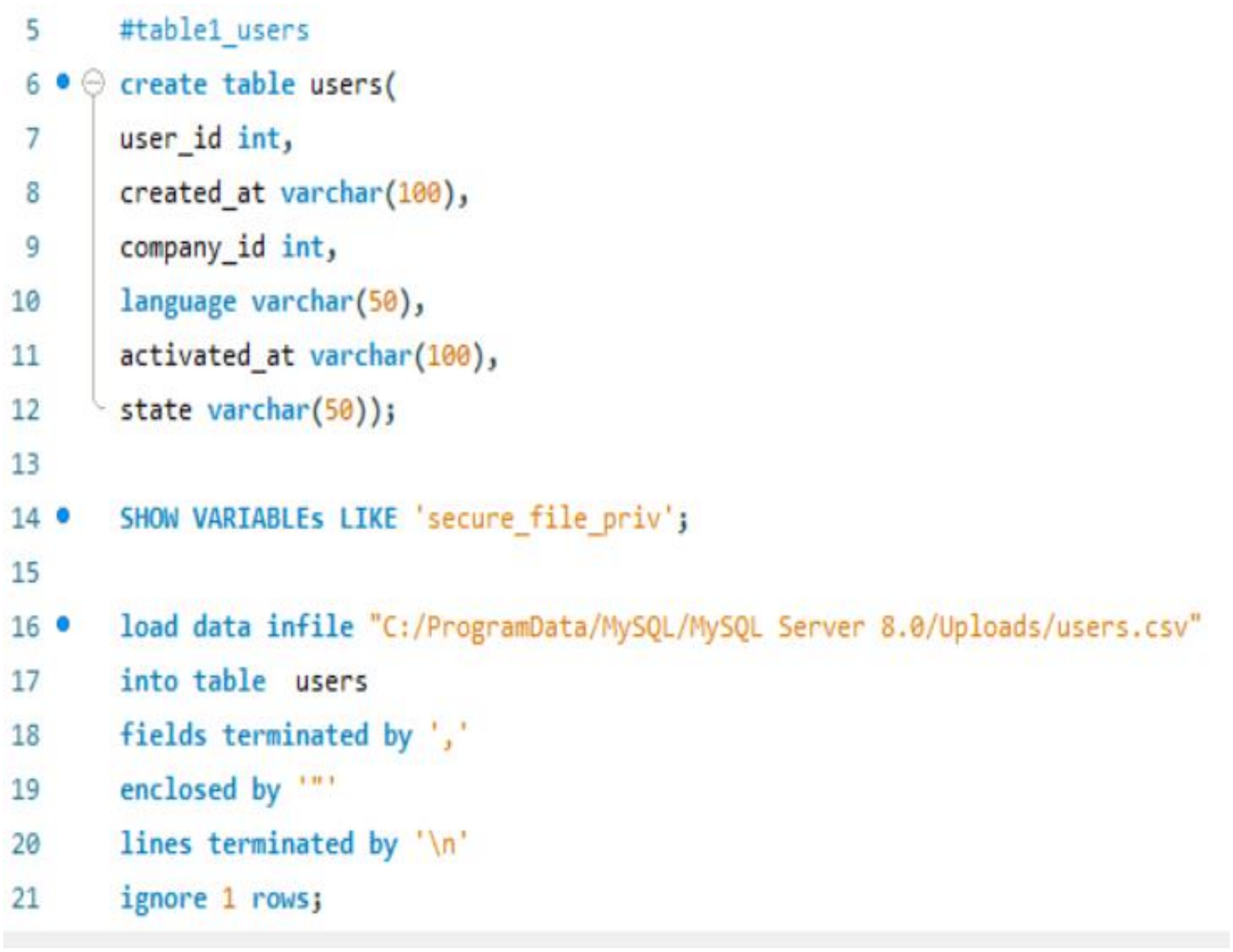
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

job_id	actor_id	event	language	time_spent	org	ds	row_num
21	1001	skip	English	15	A	2020-11-30	2
23	1005	transfer	Persian	22	D	2020-11-28	2
23	1004	skip	Persian	56	A	2020-11-26	3

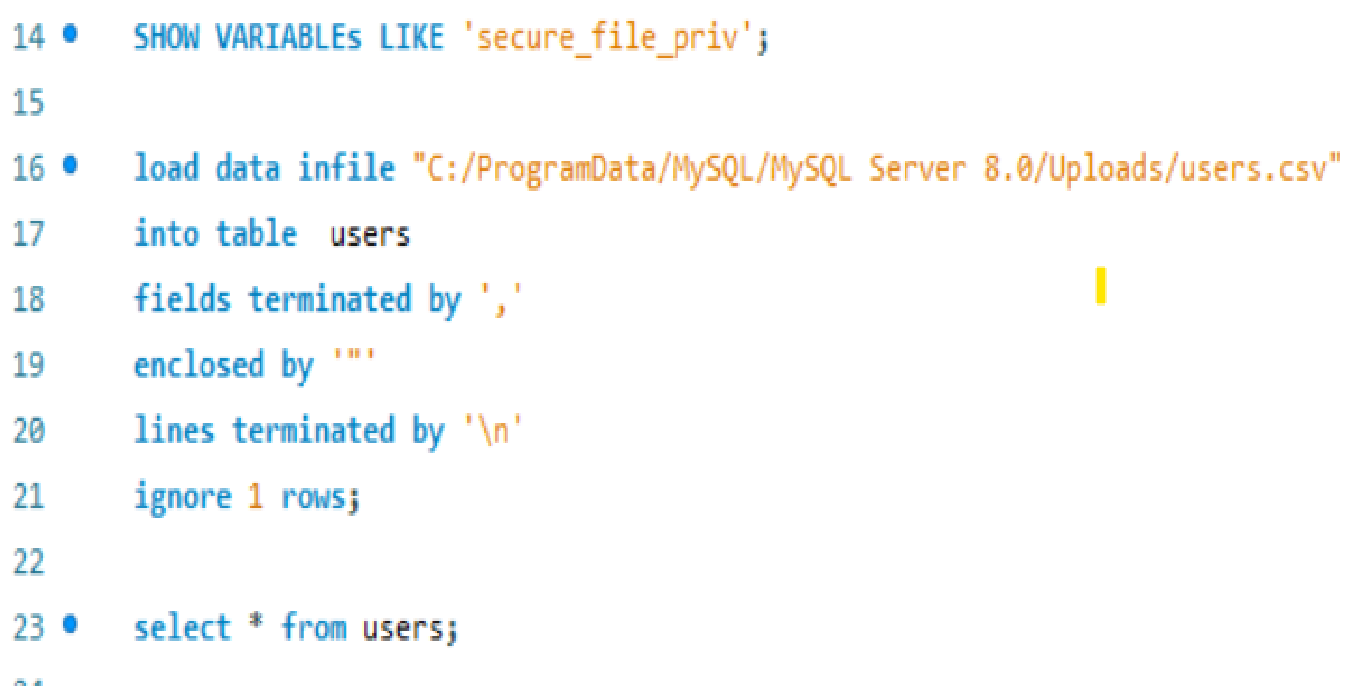
CASE STUDY 2

INVESTIGATING METRIC SPIKE

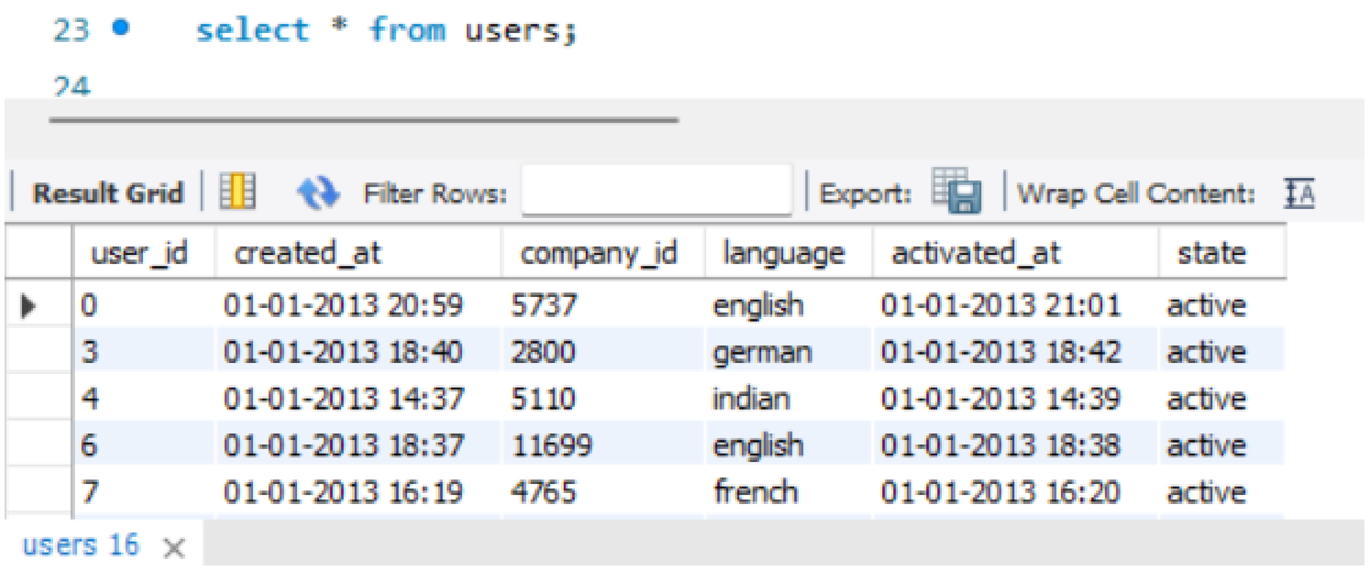
- **TABLE-1:** USERS
  This table includes one row per user, with descriptive information about
  that user's account.



```sql
#table1_users
create table users(
user_id int,
created_at varchar(100),
company_id int,
language varchar(50),
activated_at varchar(100),
state varchar(50));

SHOW VARIABLEs LIKE 'secure_file_priv';

load data infile "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users.csv"
into table  users
fields terminated by ','
enclosed by '"'
lines terminated by '\n'
ignore 1 rows;
```



```
SHOW VARIABLEs LIKE 'secure_file_priv';

load data infile "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users.csv"
into table  users
fields terminated by ','
enclosed by '"'
lines terminated by '\n'
ignore 1 rows;

select * from users;
```

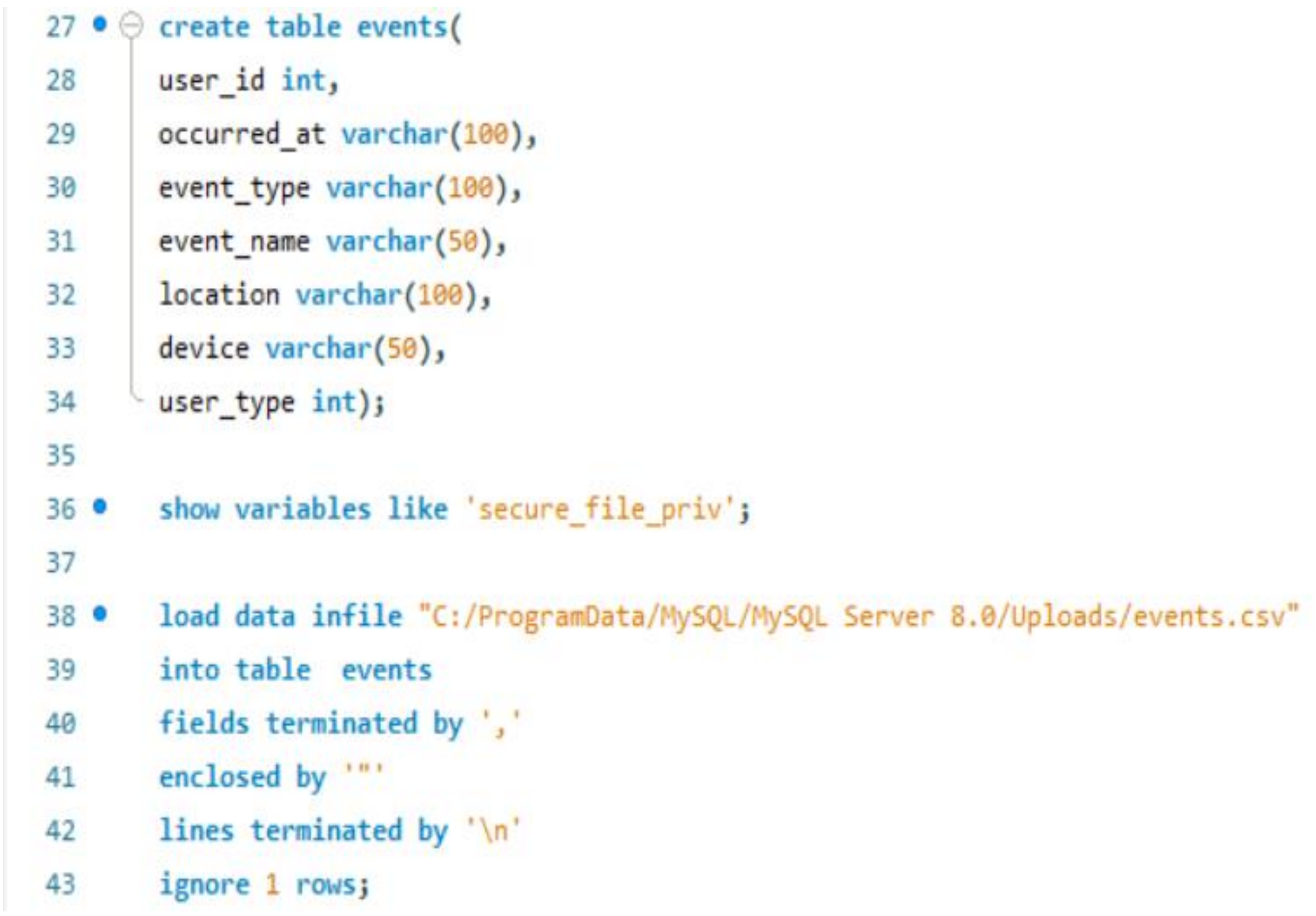


```
select * from users;
```

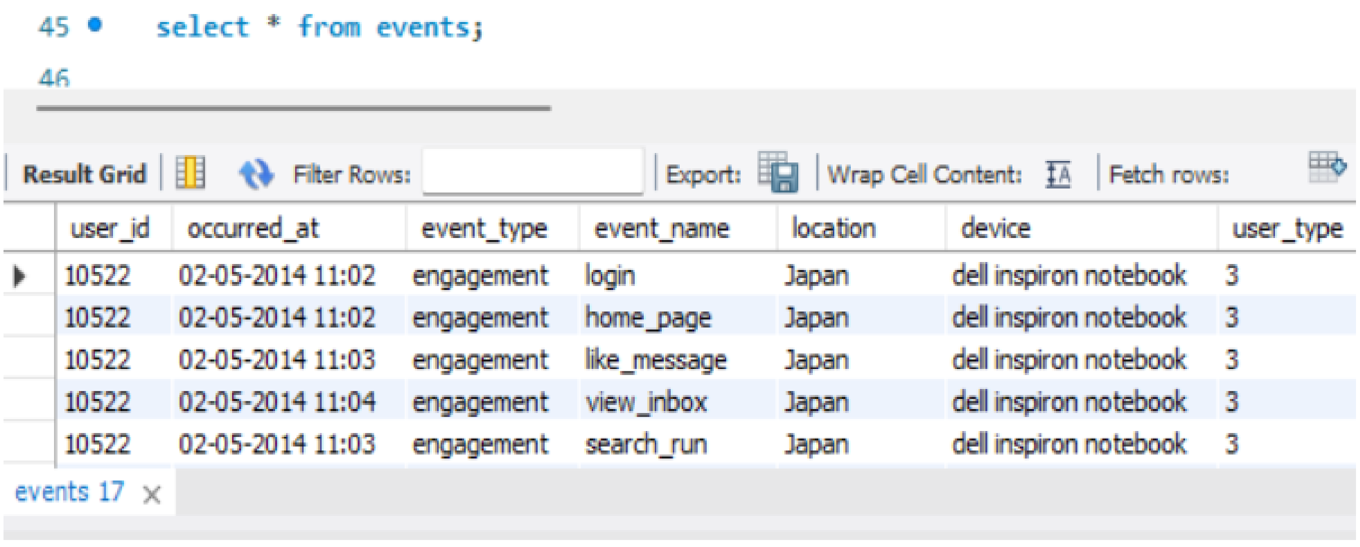
user_id	created_at	company_id	language	activated_at	state
0	01-01-2013 20:59	5737	english	01-01-2013 21:01	active
3	01-01-2013 18:40	2800	german	01-01-2013 18:42	active
4	01-01-2013 14:37	5110	indian	01-01-2013 14:39	active
6	01-01-2013 18:37	11699	english	01-01-2013 18:38	active
7	01-01-2013 16:19	4765	french	01-01-2013 16:20	active

- **<u>TABLE-2:</u>** <u>EVENTS</u>
  This table includes one row per event, where an event is an action that a user has taken. These events include login events, messaging events,

search events, events logged as users progress through a signup funnel, events around received emails.



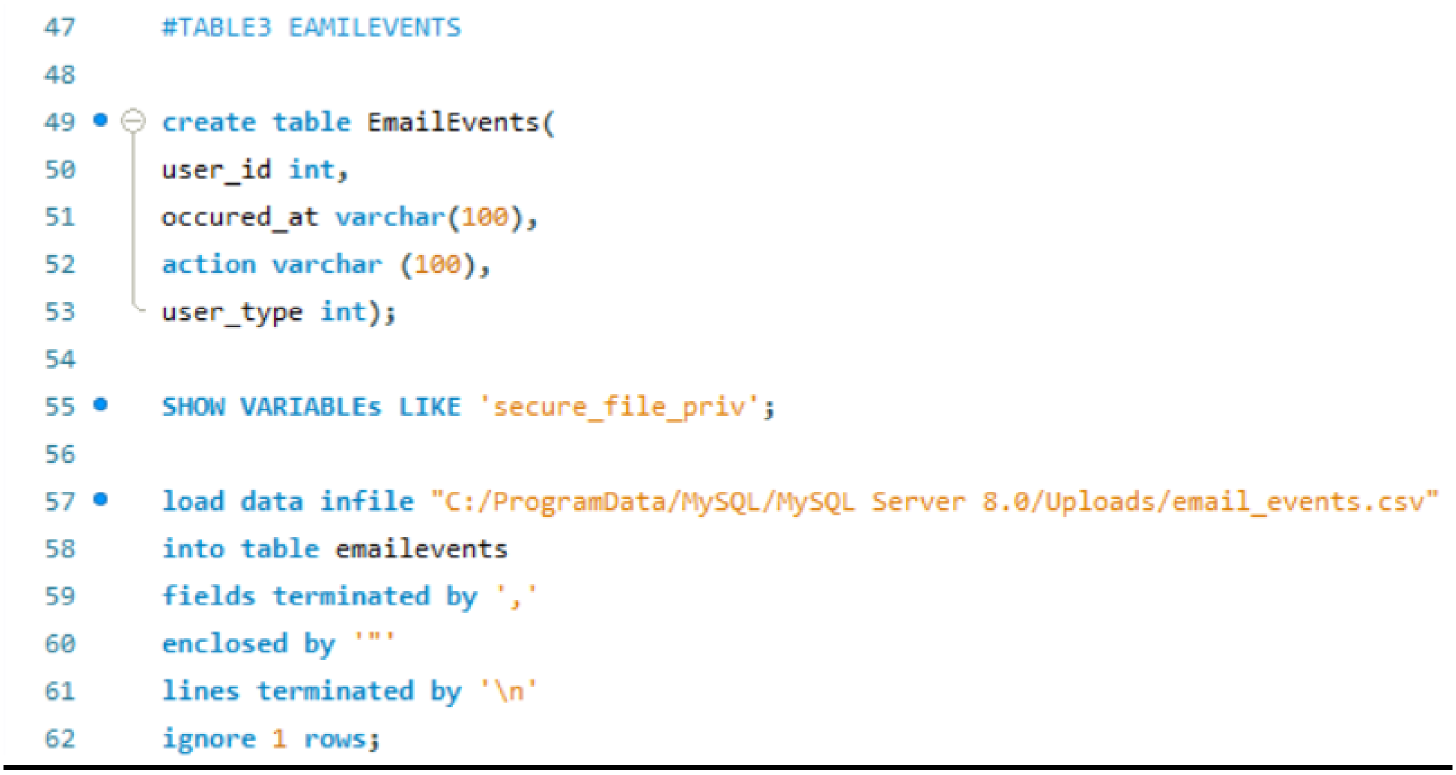
```
create table events(
user_id int,
occurred_at varchar(100),
event_type varchar(100),
event_name varchar(50),
location varchar(100),
device varchar(50),
user_type int);

show variables like 'secure_file_priv';

load data infile "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/events.csv"
into table  events
fields terminated by ','
enclosed by '"'
lines terminated by '\n'
ignore 1 rows;
```



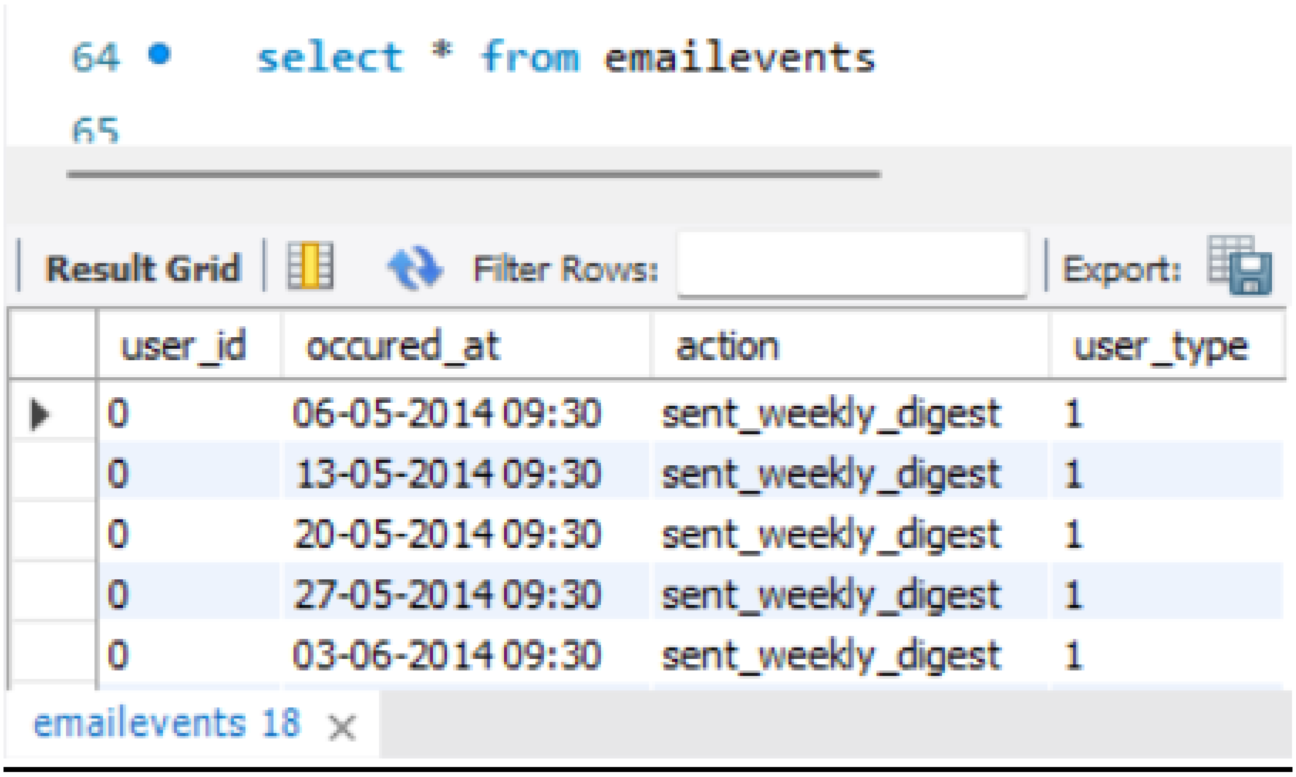
```
select * from events;
```

user_id	occurred_at	event_type	event_name	location	device	user_type
10522	02-05-2014 11:02	engagement	login	Japan	dell inspiron notebook	3
10522	02-05-2014 11:02	engagement	home_page	Japan	dell inspiron notebook	3
10522	02-05-2014 11:03	engagement	like_message	Japan	dell inspiron notebook	3
10522	02-05-2014 11:04	engagement	view_inbox	Japan	dell inspiron notebook	3
10522	02-05-2014 11:03	engagement	search_run	Japan	dell inspiron notebook	3

- **TABLE-3:** EMAIL EVENTS
  This table contains events specific to the sending of emails.
  It is similar in structure to the events table above.



```
#TABLE3 EAMILEVENTS

create table EmailEvents(
user_id int,
occured_at varchar(100),
action varchar (100),
user_type int);

SHOW VARIABLEs LIKE 'secure_file_priv';

load data infile "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/email_events.csv"
into table emailevents
fields terminated by ','
enclosed by '"'
lines terminated by '\n'
ignore 1 rows;
```



```
select * from emailevents
```

Result Grid | Filter Rows: | Export:

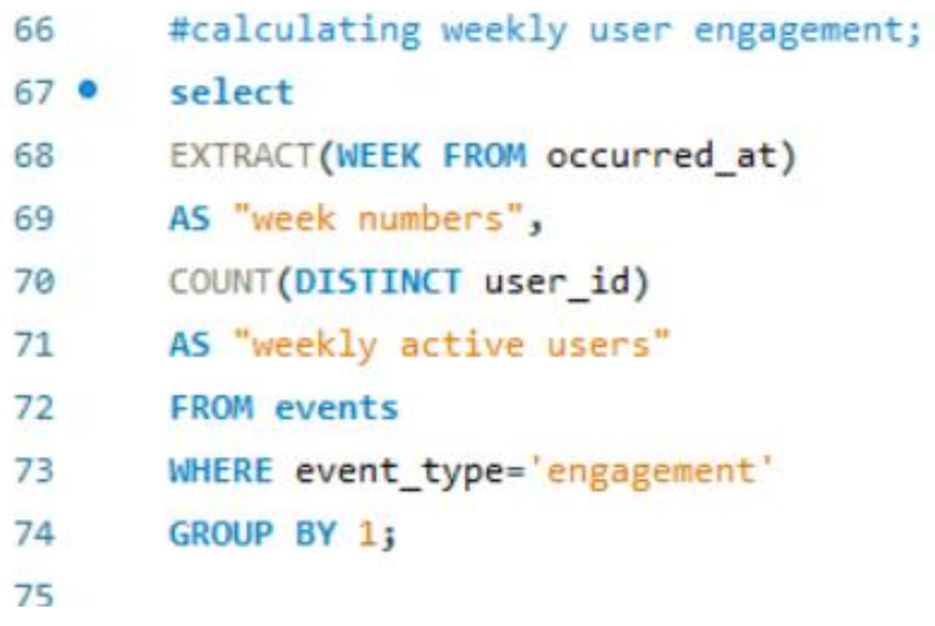
	user_id	occured_at	action	user_type
▶	0	06-05-2014 09:30	sent_weekly_digest	1
	0	13-05-2014 09:30	sent_weekly_digest	1
	0	20-05-2014 09:30	sent_weekly_digest	1
	0	27-05-2014 09:30	sent_weekly_digest	1
	0	03-06-2014 09:30	sent_weekly_digest	1

emailevents 18 ×

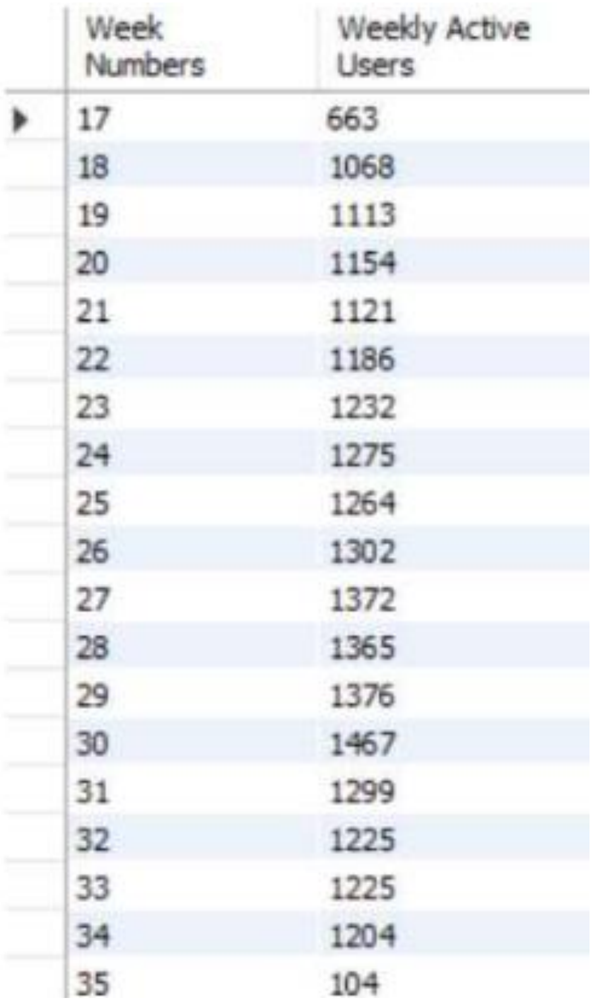
TASKS

1.) WEEKLY USER ENAGAGEMENT:

Measuring the activeness of the users every week.



```
#calculating weekly user engagement;
select
EXTRACT(WEEK FROM occurred_at)
AS "week numbers",
COUNT(DISTINCT user_id)
AS "weekly active users"
FROM events
WHERE event_type='engagement'
GROUP BY 1;
```

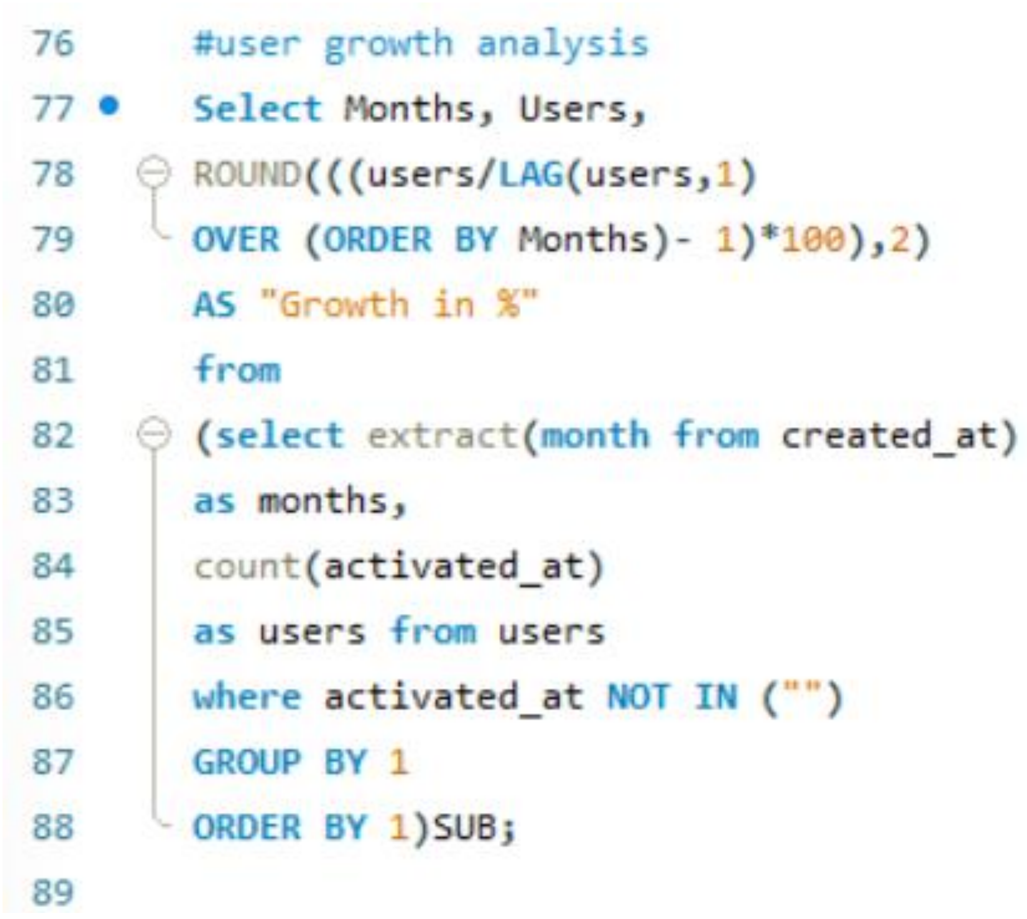


Week Numbers	Weekly Active Users
17	663
18	1068
19	1113
20	1154
21	1121
22	1186
23	1232
24	1275
25	1264
26	1302
27	1372
28	1365
29	1376
30	1467
31	1299
32	1225
33	1225
34	1204
35	104

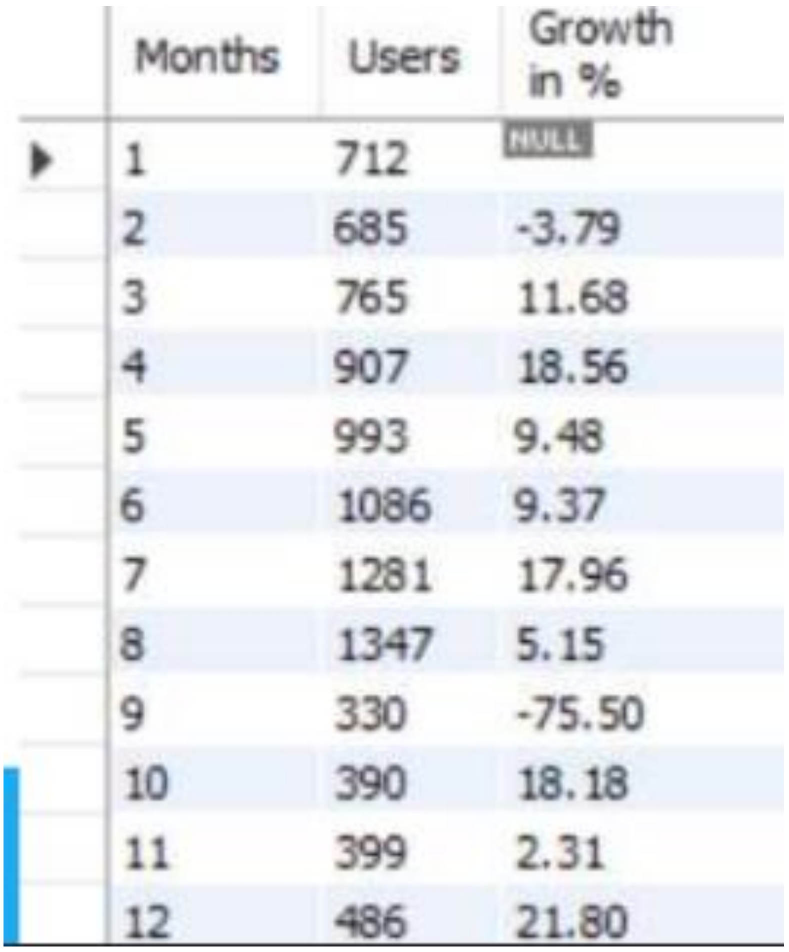
2.) <u>USER GROWTH ANALYSIS:</u>

Analyse the growth of users over time for a product.

User growth = number of active users per week.

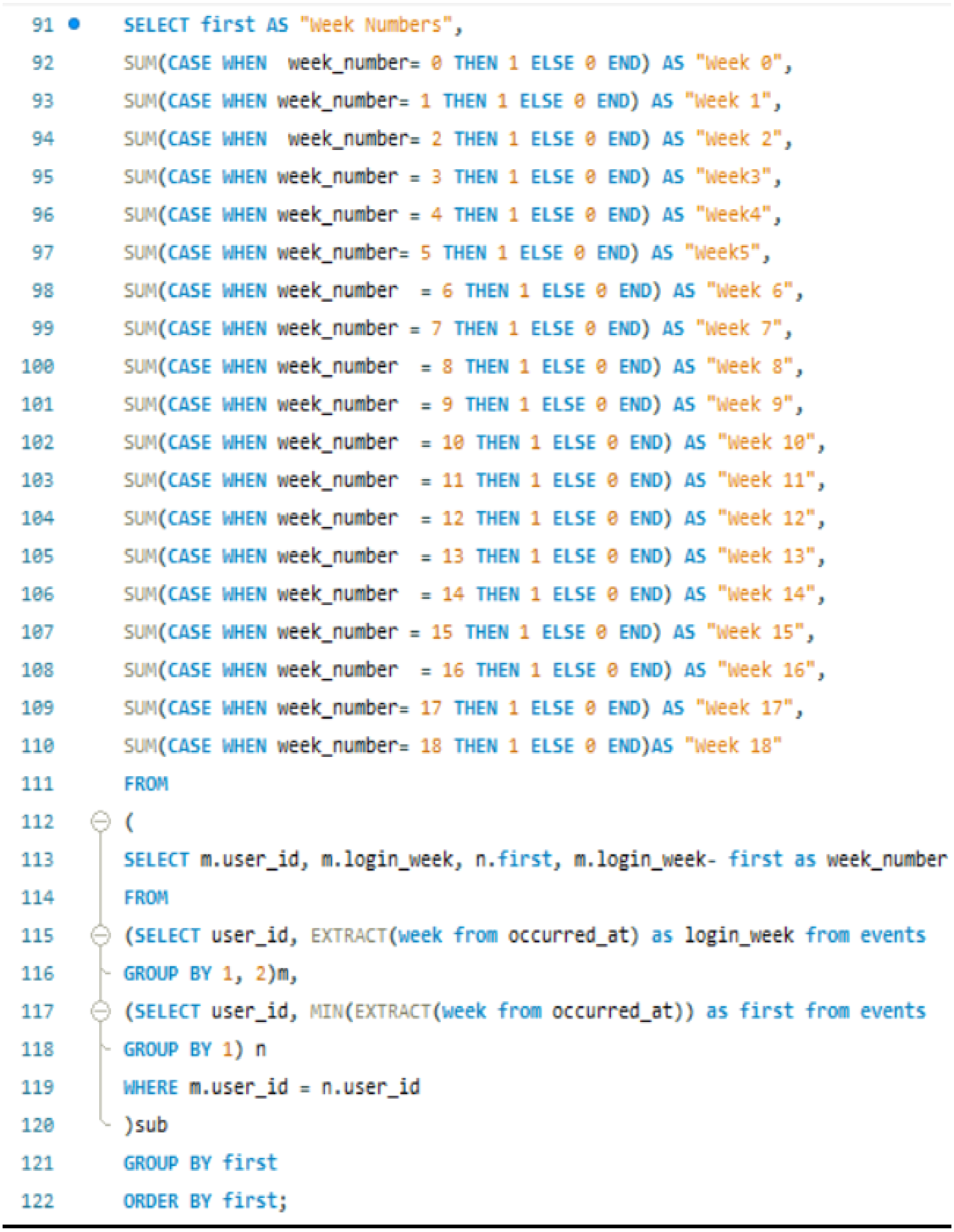


```sql
#user growth analysis
Select Months, Users,
ROUND(((users/LAG(users,1)
OVER (ORDER BY Months)- 1)*100),2)
AS "Growth in %"
from
(select extract(month from created_at)
as months,
count(activated_at)
as users from users
where activated_at NOT IN ("")
GROUP BY 1
ORDER BY 1)SUB;
```

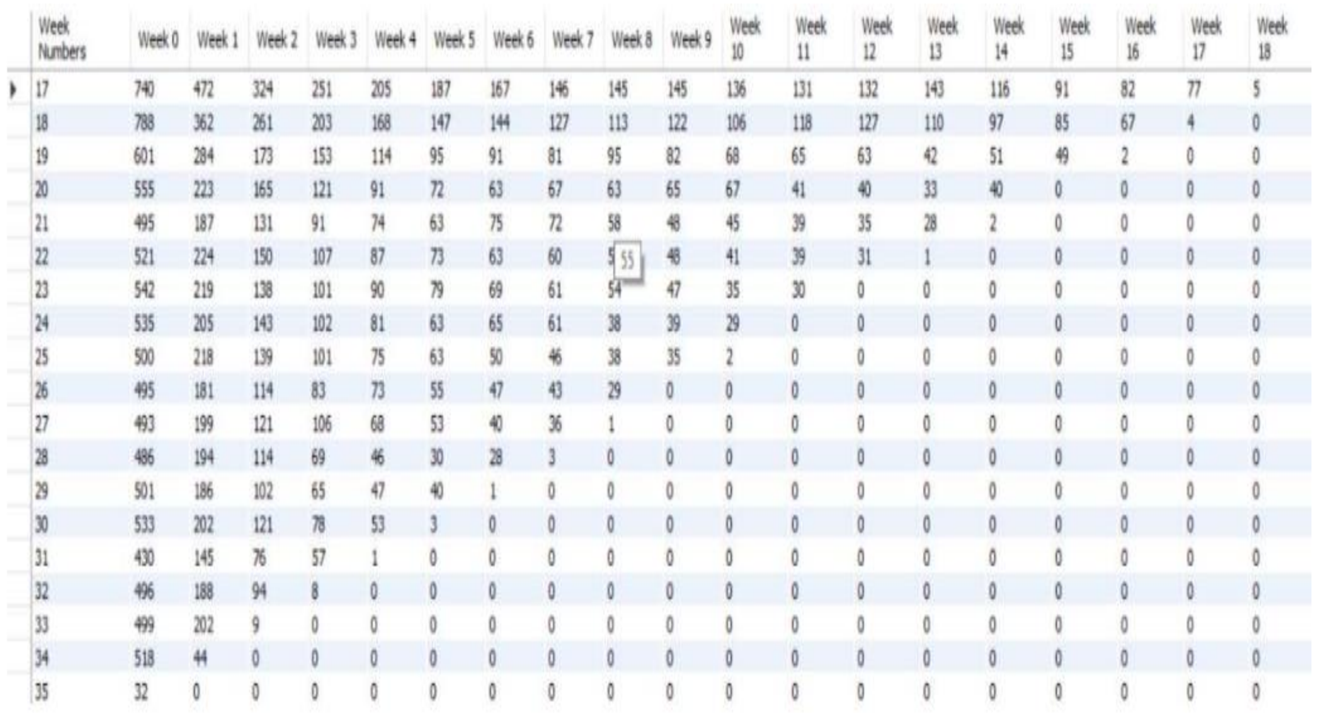


Months	Users	Growth in %
1	712	NULL
2	685	-3.79
3	765	11.68
4	907	18.56
5	993	9.48
6	1086	9.37
7	1281	17.96
8	1347	5.15
9	330	-75.50
10	390	18.18
11	399	2.31
12	486	21.80

3.) WEEKLY RETENTION ANALYSIS:



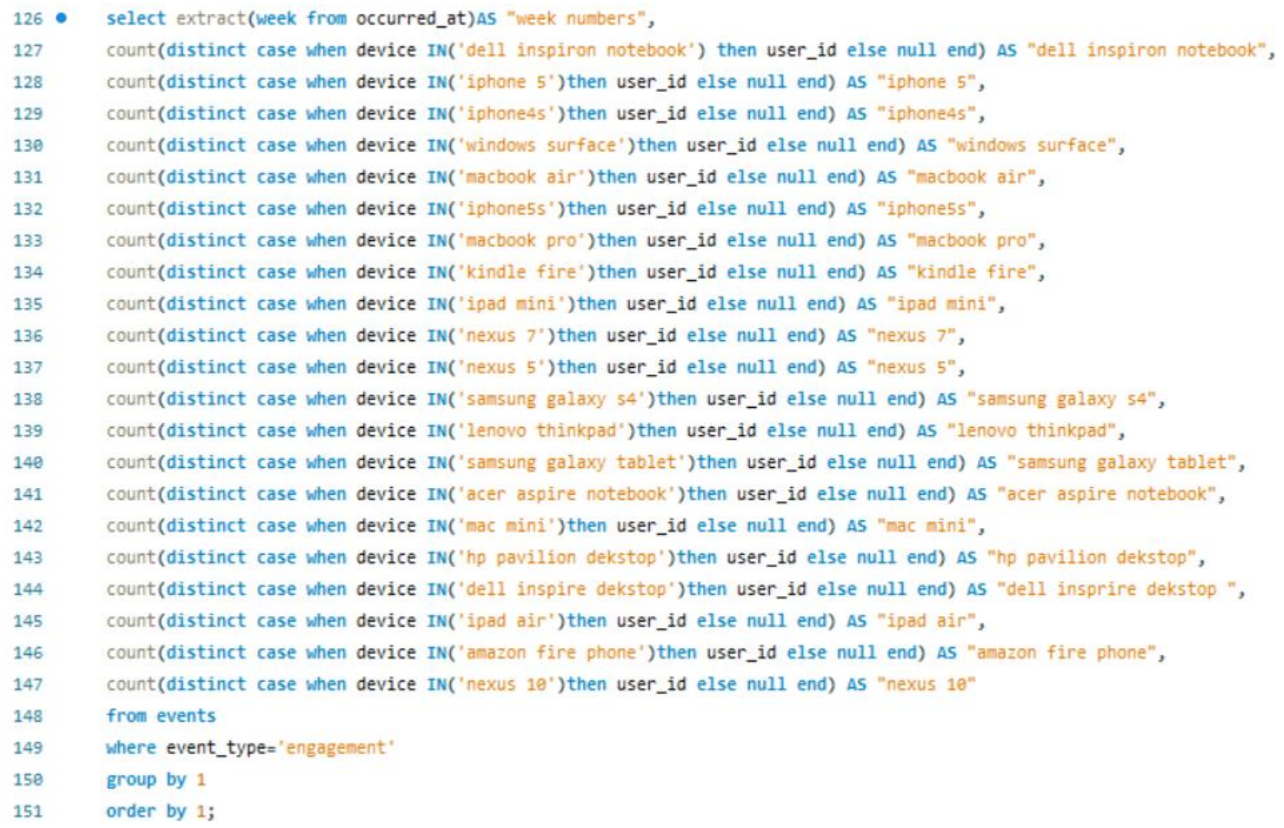
```sql
SELECT first AS "Week Numbers",
SUM(CASE WHEN  week_number= 0 THEN 1 ELSE 0 END) AS "Week 0",
SUM(CASE WHEN week_number= 1 THEN 1 ELSE 0 END) AS "Week 1",
SUM(CASE WHEN  week_number= 2 THEN 1 ELSE 0 END) AS "Week 2",
SUM(CASE WHEN week_number = 3 THEN 1 ELSE 0 END) AS "Week3",
SUM(CASE WHEN week_number = 4 THEN 1 ELSE 0 END) AS "Week4",
SUM(CASE WHEN week_number= 5 THEN 1 ELSE 0 END) AS "Week5",
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",
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- first as week_number
FROM
(SELECT user_id, EXTRACT(week from occurred_at) as login_week from events
GROUP BY 1, 2)m,
(SELECT user_id, MIN(EXTRACT(week from occurred_at)) as first from events
GROUP BY 1) n
WHERE m.user_id = n.user_id
)sub
GROUP BY first
ORDER BY first;
```



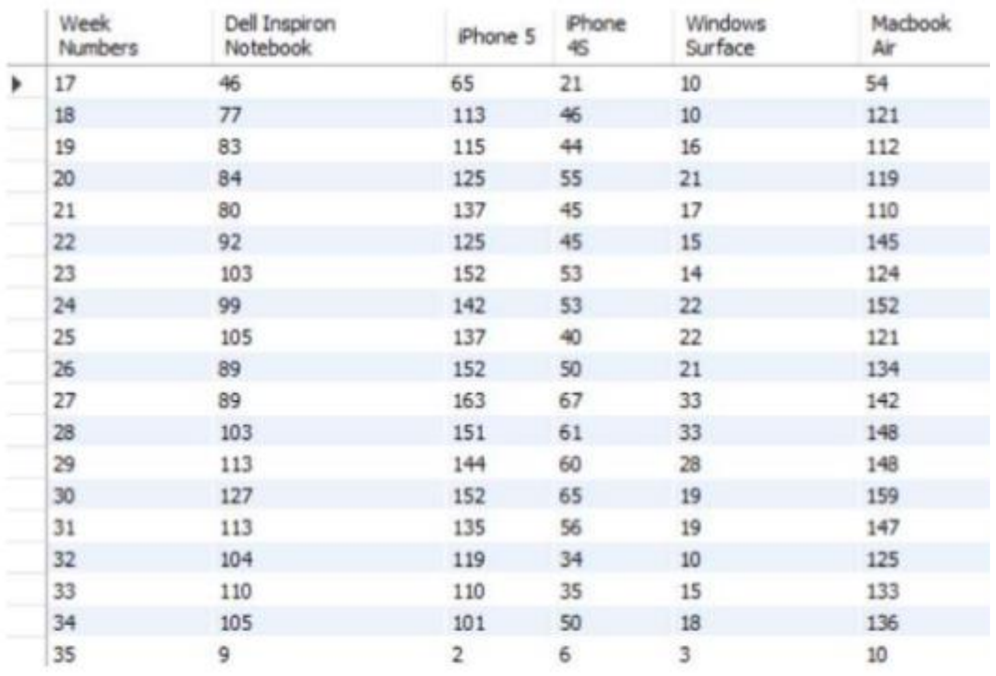
Week Numbers	Week 0	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16	Week 17	Week 18
17	740	472	324	251	205	187	167	146	145	145	136	131	132	143	116	91	82	77	5
18	788	362	261	203	168	147	144	127	113	122	106	118	127	110	97	85	67	4	0
19	601	284	173	153	114	95	91	81	95	82	68	65	63	42	51	49	2	0	0
20	555	223	165	121	91	72	63	67	63	65	67	41	40	33	40	0	0	0	0
21	495	187	131	91	74	63	75	72	58	48	45	39	35	28	2	0	0	0	0
22	521	224	150	107	87	73	63	60	55	48	41	39	31	1	0	0	0	0	0
23	542	219	138	101	90	79	69	61	54	47	35	30	0	0	0	0	0	0	0
24	535	205	143	102	81	63	65	61	38	39	29	0	0	0	0	0	0	0	0
25	500	218	139	101	75	63	50	46	38	35	2	0	0	0	0	0	0	0	0
26	495	181	114	83	73	55	47	43	29	0	0	0	0	0	0	0	0	0	0
27	493	199	121	106	68	53	40	36	1	0	0	0	0	0	0	0	0	0	0
28	486	194	114	69	46	30	28	3	0	0	0	0	0	0	0	0	0	0	0
29	501	186	102	65	47	40	1	0	0	0	0	0	0	0	0	0	0	0	0
30	533	202	121	78	53	3	0	0	0	0	0	0	0	0	0	0	0	0	0
31	430	145	76	57	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	496	188	94	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	499	202	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	518	44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

4.) **WEEKLY ENGAGEMENT PER DEVICE**:

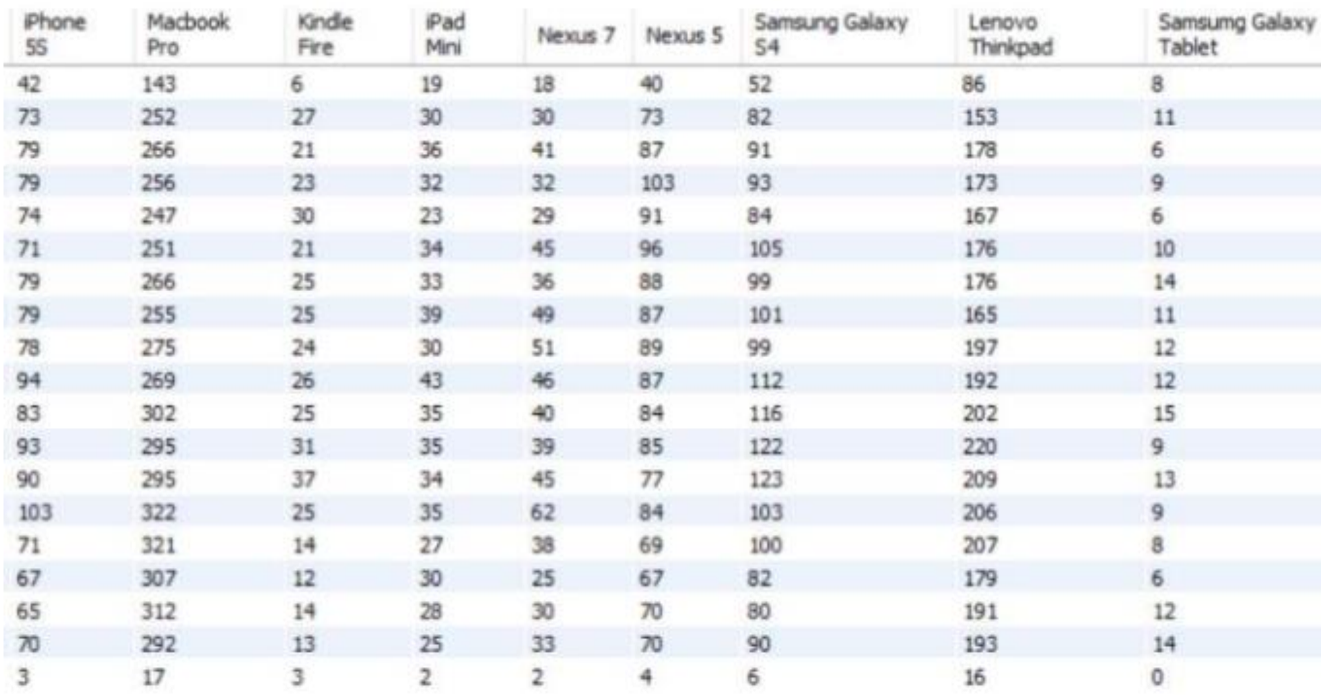
Measure the activeness of users weekly per device.



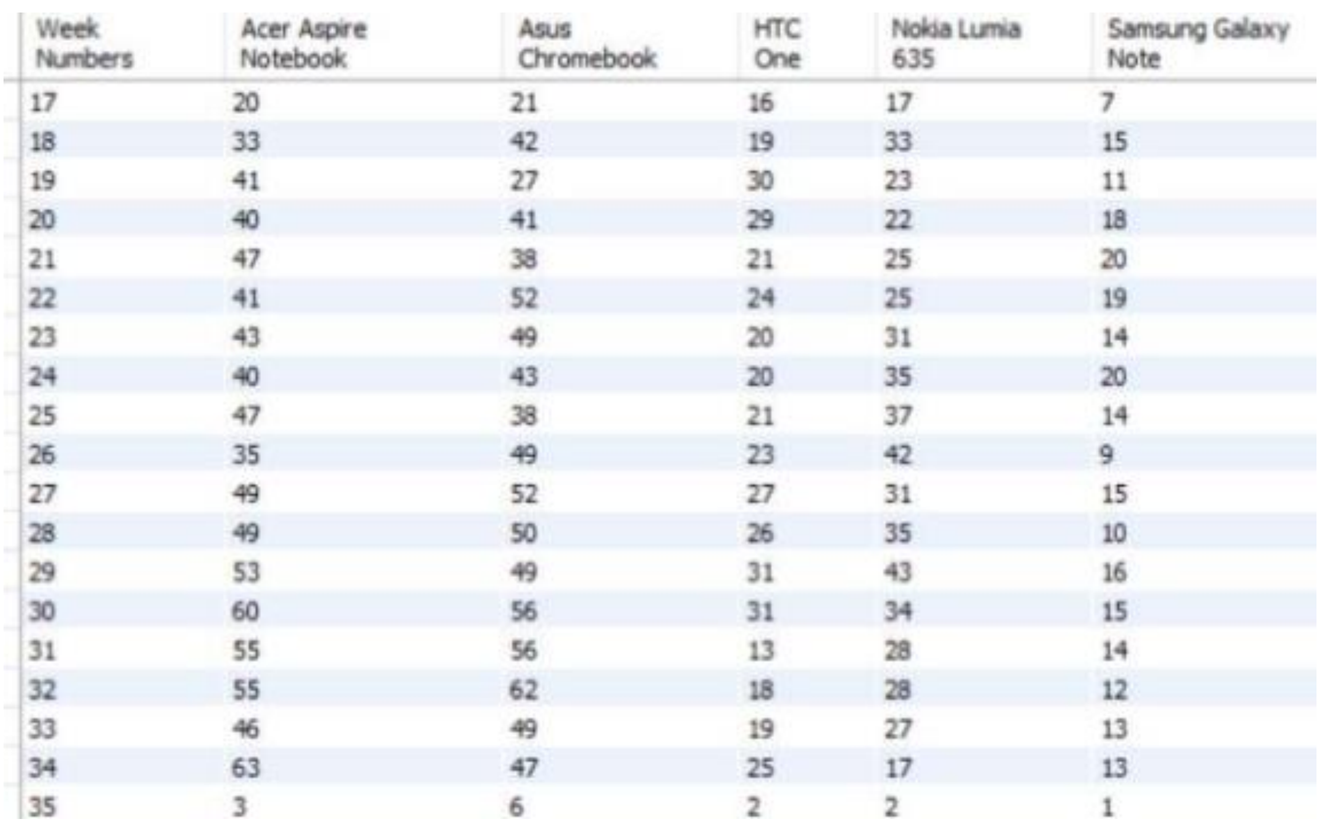
```sql
select extract(week from occurred_at)AS "week numbers",
count(distinct case when device IN('dell inspiron notebook') then user_id else null end) AS "dell inspiron notebook",
count(distinct case when device IN('iphone 5')then user_id else null end) AS "iphone 5",
count(distinct case when device IN('iphone4s')then user_id else null end) AS "iphone4s",
count(distinct case when device IN('windows surface')then user_id else null end) AS "windows surface",
count(distinct case when device IN('macbook air')then user_id else null end) AS "macbook air",
count(distinct case when device IN('iphone5s')then user_id else null end) AS "iphone5s",
count(distinct case when device IN('macbook pro')then user_id else null end) AS "macbook pro",
count(distinct case when device IN('kindle fire')then user_id else null end) AS "kindle fire",
count(distinct case when device IN('ipad mini')then user_id else null end) AS "ipad mini",
count(distinct case when device IN('nexus 7')then user_id else null end) AS "nexus 7",
count(distinct case when device IN('nexus 5')then user_id else null end) AS "nexus 5",
count(distinct case when device IN('samsung galaxy s4')then user_id else null end) AS "samsung galaxy s4",
count(distinct case when device IN('lenovo thinkpad')then user_id else null end) AS "lenovo thinkpad",
count(distinct case when device IN('samsung galaxy tablet')then user_id else null end) AS "samsung galaxy tablet",
count(distinct case when device IN('acer aspire notebook')then user_id else null end) AS "acer aspire notebook",
count(distinct case when device IN('mac mini')then user_id else null end) AS "mac mini",
count(distinct case when device IN('hp pavilion dekstop')then user_id else null end) AS "hp pavilion dekstop",
count(distinct case when device IN('dell inspire dekstop')then user_id else null end) AS "dell insprire dekstop ",
count(distinct case when device IN('ipad air')then user_id else null end) AS "ipad air",
count(distinct case when device IN('amazon fire phone')then user_id else null end) AS "amazon fire phone",
count(distinct case when device IN('nexus 10')then user_id else null end) AS "nexus 10"
from events
where event_type='engagement'
group by 1
order by 1;
```



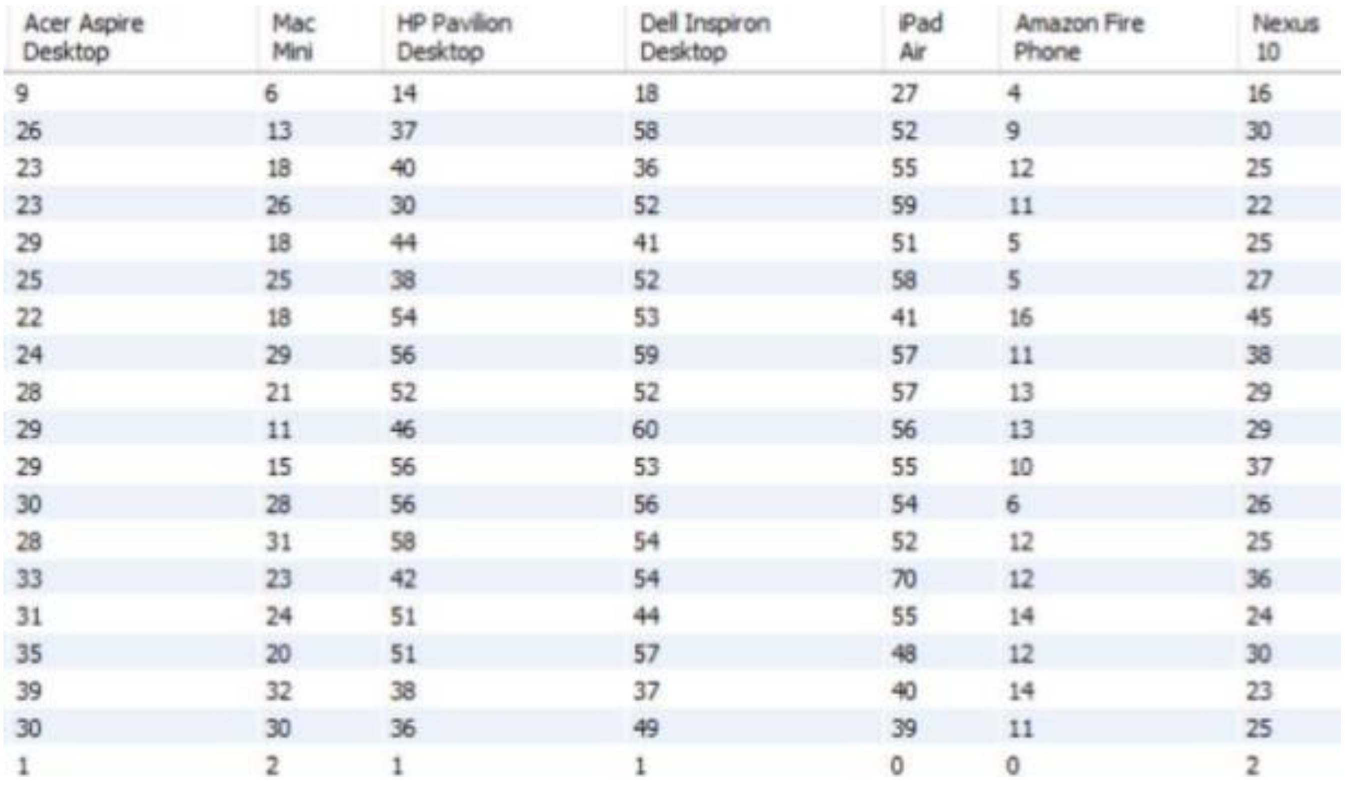
Week Numbers	Dell Inspiron Notebook	iPhone 5	iPhone 4S	Windows Surface	Macbook Air
17	46	65	21	10	54
18	77	113	46	10	121
19	83	115	44	16	112
20	84	125	55	21	119
21	80	137	45	17	110
22	92	125	45	15	145
23	103	152	53	14	124
24	99	142	53	22	152
25	105	137	40	22	121
26	89	152	50	21	134
27	89	163	67	33	142
28	103	151	61	33	148
29	113	144	60	28	148
30	127	152	65	19	159
31	113	135	56	19	147
32	104	119	34	10	125
33	110	110	35	15	133
34	105	101	50	18	136
35	9	2	6	3	10



iPhone 5S	Macbook Pro	Kindle Fire	iPad Mini	Nexus 7	Nexus 5	Samsung Galaxy S4	Lenovo Thinkpad	Samsumg Galaxy Tablet
42	143	6	19	18	40	52	86	8
73	252	27	30	30	73	82	153	11
79	266	21	36	41	87	91	178	6
79	256	23	32	32	103	93	173	9
74	247	30	23	29	91	84	167	6
71	251	21	34	45	96	105	176	10
79	266	25	33	36	88	99	176	14
79	255	25	39	49	87	101	165	11
78	275	24	30	51	89	99	197	12
94	269	26	43	46	87	112	192	12
83	302	25	35	40	84	116	202	15
93	295	31	35	39	85	122	220	9
90	295	37	34	45	77	123	209	13
103	322	25	35	62	84	103	206	9
71	321	14	27	38	69	100	207	8
67	307	12	30	25	67	82	179	6
65	312	14	28	30	70	80	191	12
70	292	13	25	33	70	90	193	14
3	17	3	2	2	4	6	16	0



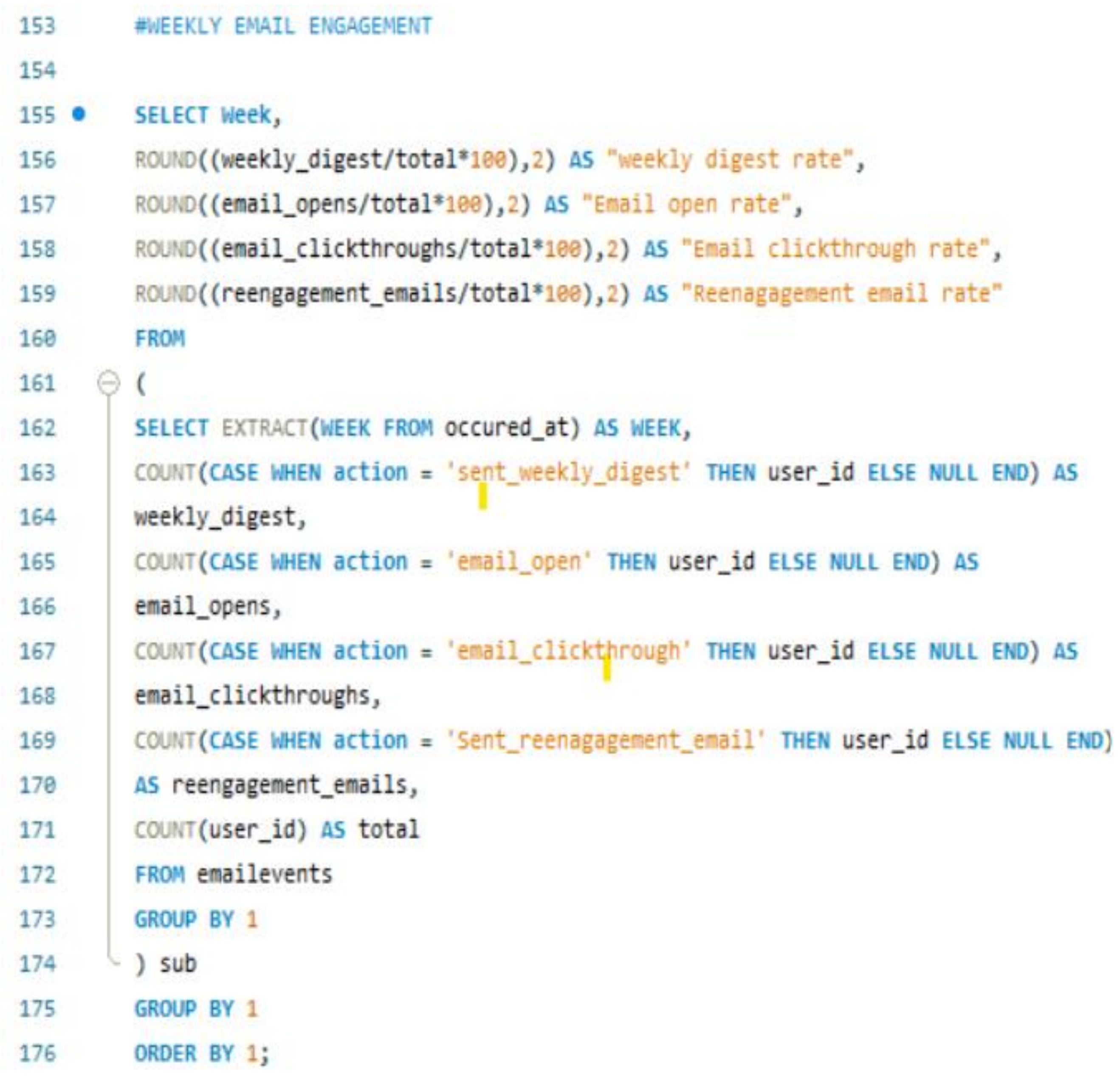
Week Numbers	Acer Aspire Notebook	Asus Chromebook	HTC One	Nokia Lumia 635	Samsung Galaxy Note
17	20	21	16	17	7
18	33	42	19	33	15
19	41	27	30	23	11
20	40	41	29	22	18
21	47	38	21	25	20
22	41	52	24	25	19
23	43	49	20	31	14
24	40	43	20	35	20
25	47	38	21	37	14
26	35	49	23	42	9
27	49	52	27	31	15
28	49	50	26	35	10
29	53	49	31	43	16
30	60	56	31	34	15
31	55	56	13	28	14
32	55	62	18	28	12
33	46	49	19	27	13
34	63	47	25	17	13
35	3	6	2	2	1



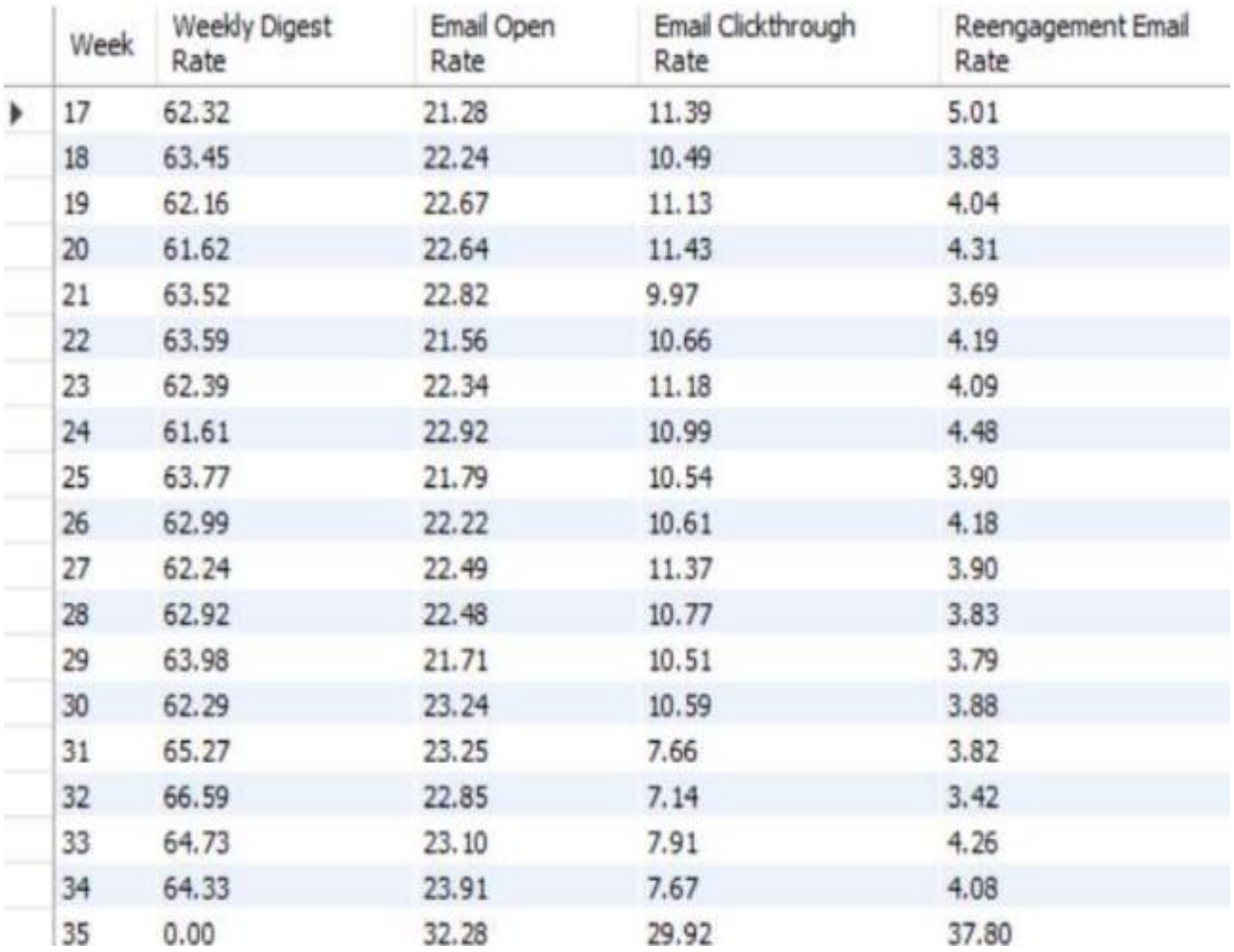
Acer Aspire Desktop	Mac Mini	HP Pavilion Desktop	Dell Inspiron Desktop	iPad Air	Amazon Fire Phone	Nexus 10
9	6	14	18	27	4	16
26	13	37	58	52	9	30
23	18	40	36	55	12	25
23	26	30	52	59	11	22
29	18	44	41	51	5	25
25	25	38	52	58	5	27
22	18	54	53	41	16	45
24	29	56	59	57	11	38
28	21	52	52	57	13	29
29	11	46	60	56	13	29
29	15	56	53	55	10	37
30	28	56	56	54	6	26
28	31	58	54	52	12	25
33	23	42	54	70	12	36
31	24	51	44	55	14	24
35	20	51	57	48	12	30
39	32	38	37	40	14	23
30	30	36	49	39	11	25
1	2	1	1	0	0	2

5.) EMAIL ENGAGEMENT ANALYSIS:

Analysing how users are engaging with the email service.



```
#WEEKLY EMAIL ENGAGEMENT

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 "Reenagagement 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_reenagagement_email' THEN user_id ELSE NULL END)
AS reengagement_emails,
COUNT(user_id) AS total
FROM emailevents
GROUP BY 1
) sub
GROUP BY 1
ORDER BY 1;
```



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

RESULTS:

How this project helped me: This project helps me to understand the importance of operation analytics. Through this project I am able to understand how the companies use metric spike as a secret weapon. With an informed and proactive approach, they can leverage insights to make data-backed decisions that optimize their strategy and boost ROI.

Challenges that I faced in this project: The challenge here is that the data in case study 2 is very huge. And because of the huge amount of data, SQL Workbench is very slow to import it. To tackle this challenge, I used **LOAD DATA** statements. Now, another problem arises in the column **user_type** in the **events** table **that has database int which is stopping the** import process. First, I need to change its datatype to text. Then restart the process of loading into the **events** table.

Conclusion: Operational Analytics tackles the problem by synchronising real-time data. Operational Analytics can aggregate data from multiple data sources into a cumulative, organised, actionable solution capable of delivering analytical models in real time to create individual customer profiles and a holistic view of operations for a company. This guarantees that your operational customers and systems are used efficiently. Whenever utilized correctly, operational analytics can achieve a significant positive effect on the general public and the world everywhere and increment the general efficiency of specific areas.

-end-