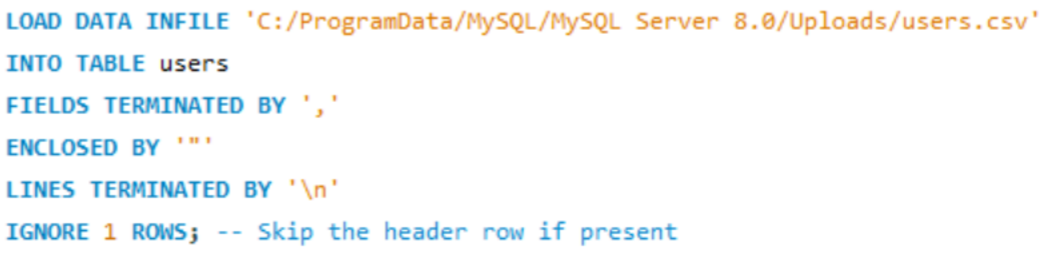
OPERATION ANALYTICS AND INVESTIGATING METRIC SPIKE

Advanced SQL

- **Project Description**: This project involves analysing a company's end-to-end operations. Operational Analytics is a crucial process that helps identify areas for improvement within the company. As a Data Analyst, I have worked closely with various teams, such as operations, support, and marketing, helping them derive valuable insights from the data they collect. One of the key aspects of Operational Analytics is investigating metric spikes. This involves understanding and explaining sudden changes in key metrics, such as a dip in daily user engagement or a drop in sales. As a Data Analyst, you'll need to answer these questions daily, making it crucial to understand how to investigate these metric spikes.

- **Approach**: As a lead Data Analyst these following steps were taken to analyse the data and provide valuable insights that would help improve the company's operations and understand sudden changes in key metrics.

  1. **Create Database and Tables:** First off, all a new database named Project3 was created and all the tables (Both for case study 1 and case study 2) provided in My-Excel csv format were imported in MYSQL workbench by following below query.



```
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; -- Skip the header row if present
```

  2. **Formulate queries**: Determined the specific questions I wanted to answer and tried to understand what result this query wants. Formulated SQL queries that retrieve the relevant data to answer those questions. Started with simple queries and gradually build more complex ones as needed.
  3. **Write Queries**: Used the SQL editor in MySQL Workbench to write and execute SQL queries. Ensured that the queries are accurate, efficient, and properly structured to extract the required information from the database.
  4. **Execute Queries**: Ran SQL queries against the database. Reviewed the query results to ensure they match with question expectations. Again, Refined queries as necessary to get the desired results.

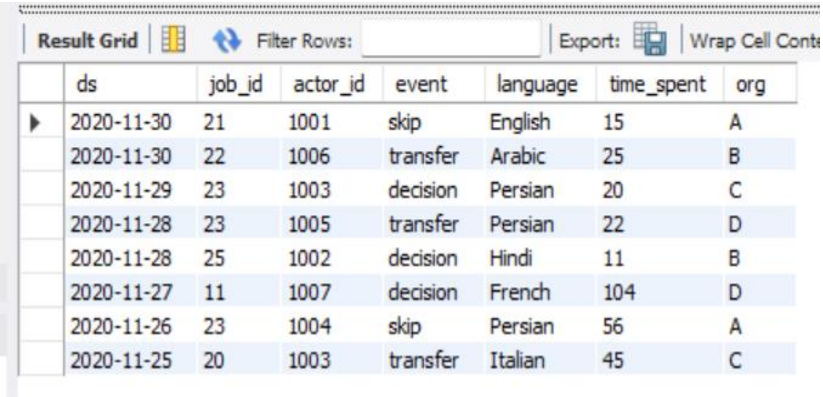
- **Tech-Stack Used**: In this project, MySQL Workbench 8.0 was used as a tool for creating databases, tables and writing SQL queries. To perform Operation Analytics and Investigating Metric Spike and answer questions posed by the management team.
  Microsoft Excel was used for importing data.
  Microsoft Word for Preparing the Project Report.

- **INSIGHT**: While working on this project I gained hands on experience with real life use cases. I gained knowledge on Advanced SQL queries and used it to answer some really interesting questions. For this project two case studies were given.

❖ Case Study 1: Job Data Analysis

Table named job_data with the following columns:

- job_id: Unique identifier of jobs
- actor_id: Unique identifier of actor
- event: The type of event (decision/skip/transfer).
- language: The Language of the content
- time_spent: Time spent to review the job in seconds.
- org: The Organization of the actor
- ds: The date in the format yyyy/mm/dd (stored as text).

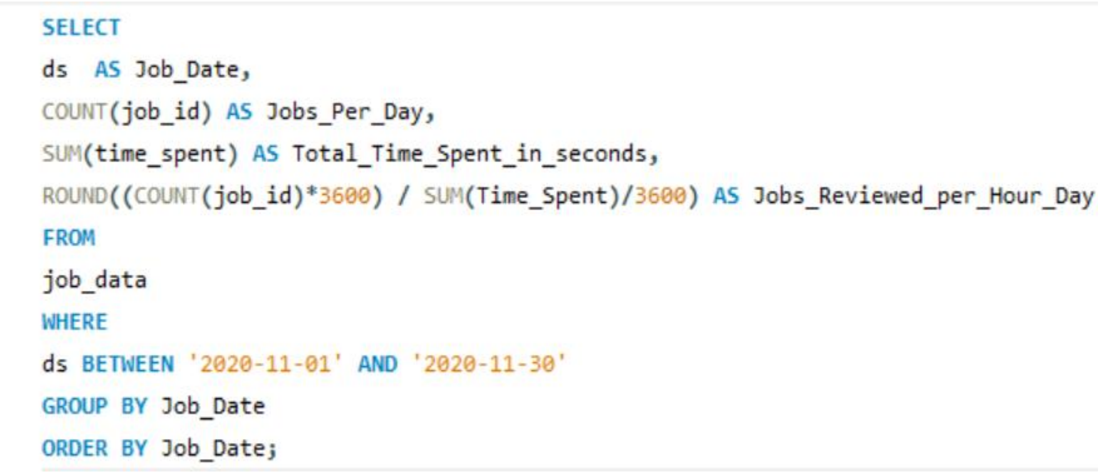


ds	job_id	actor_id	event	language	time_spent	org
2020-11-30	21	1001	skip	English	15	A
2020-11-30	22	1006	transfer	Arabic	25	B
2020-11-29	23	1003	decision	Persian	20	C
2020-11-28	23	1005	transfer	Persian	22	D
2020-11-28	25	1002	decision	Hindi	11	B
2020-11-27	11	1007	decision	French	104	D
2020-11-26	23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

Table 1: Job_data

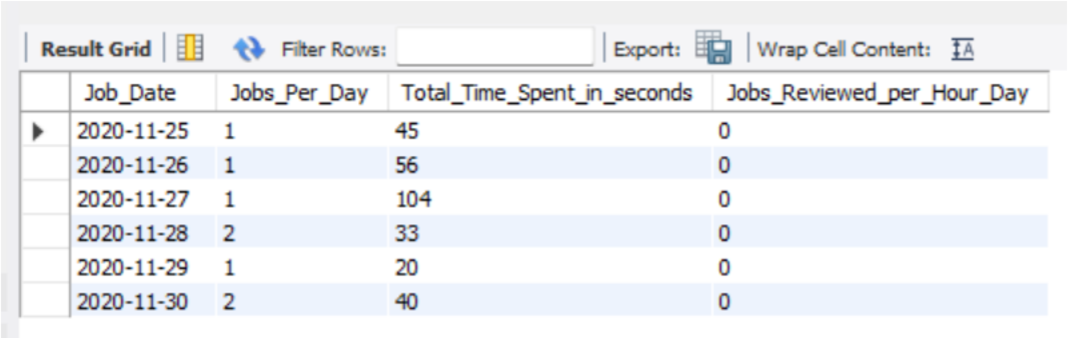
Tasks:

A. **Jobs Reviewed Over Time**:
   - ○ **Objective: Calculate the number of jobs reviewed per hour for each day in November 2020.**
   - ○ **SQL Task: Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.**



```sql
SELECT
ds  AS Job_Date,
COUNT(job_id) AS Jobs_Per_Day,
SUM(time_spent) AS Total_Time_Spent_in_seconds,
ROUND((COUNT(job_id)*3600) / SUM(Time_Spent)/3600) AS Jobs_Reviewed_per_Hour_Day
FROM
job_data
WHERE
ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY Job_Date
ORDER BY Job_Date;
```

SQL QUERY



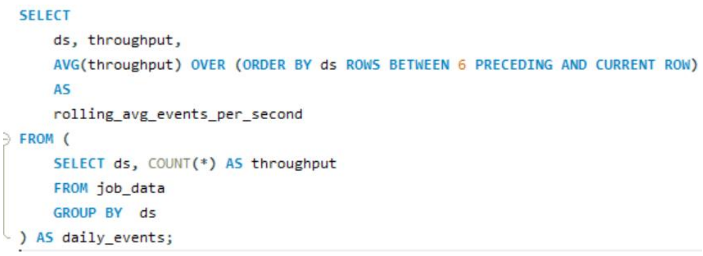
Job_Date	Jobs_Per_Day	Total_Time_Spent_in_seconds	Jobs_Reviewed_per_Hour_Day
2020-11-25	1	45	0
2020-11-26	1	56	0
2020-11-27	1	104	0
2020-11-28	2	33	0
2020-11-29	1	20	0
2020-11-30	2	40	0

OUTPUT

INSIGHT: For each day in November 2020 no. of. Jobs reviewed per hour is 0.

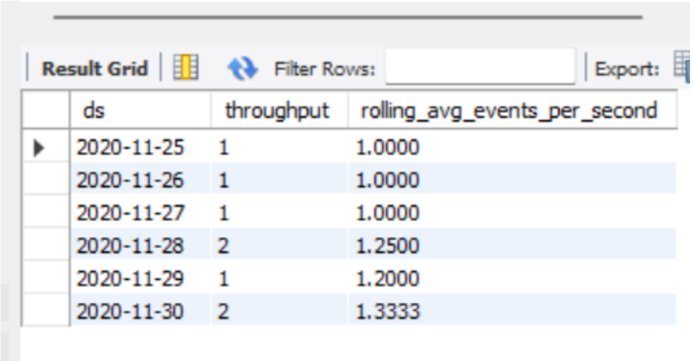
B. Throughput Analysis:

- **Objective: Calculate the 7-day rolling average of throughput (number of events per second).**
- **SQL 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.**



```sql
SELECT
    ds, throughput,
    AVG(throughput) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW)
    AS
    rolling_avg_events_per_second
FROM (
    SELECT ds, COUNT(*) AS throughput
    FROM job_data
    GROUP BY  ds
) AS daily_events;
```

SQL QUERY



ds	throughput	rolling_avg_events_per_second
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

OUTPUT

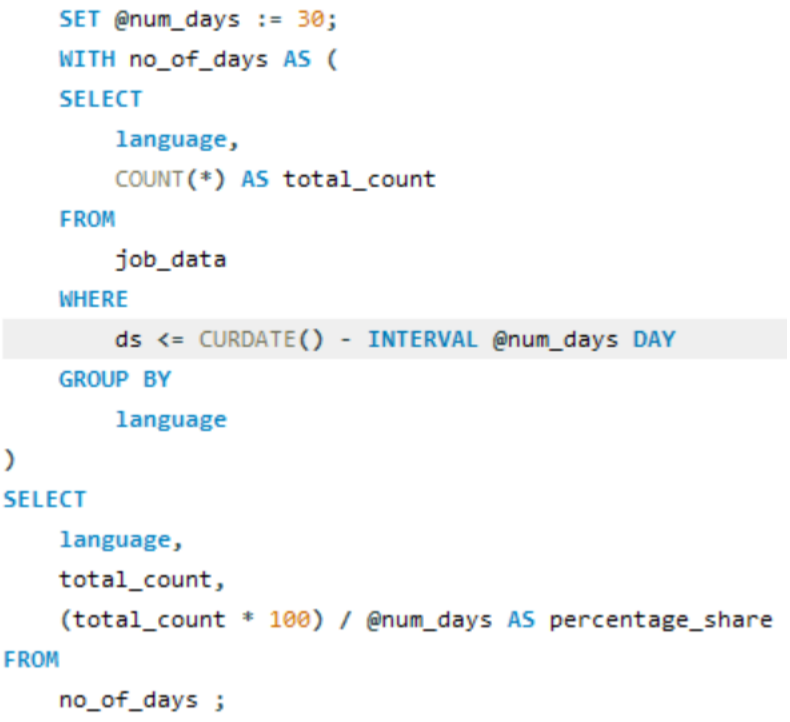
INSIGHT: The rolling average throughput is calculated by taking the average of throughput values over a sliding window of a specified number of days. In this case, we're using a 7-day rolling average, which means for each row in the result, the average is calculated over the current day and the preceding 6 days.

So, for each row in the result set, the rolling average throughput is calculated by summing the throughput values for the current day and the preceding 6 days, and then dividing that sum by 7.

This rolling average provides a smoothed average over a specified period, helping to identify trends or patterns in the data while reducing the effects of short-term fluctuations.

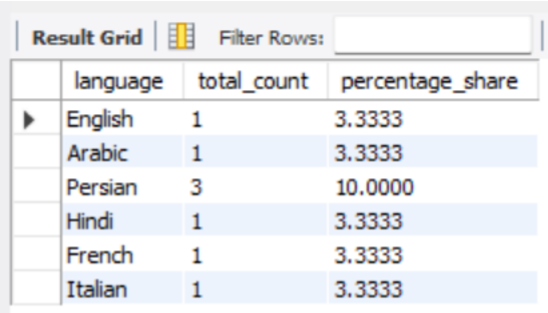
C. Language Share Analysis:

- **Objective: Calculate the percentage share of each language in the last 30 days.**
- **SQL Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.**



```
    SET @num_days := 30;
    WITH no_of_days AS (
    SELECT
        language,
        COUNT(*) AS total_count
    FROM
        job_data
    WHERE
        ds <= CURDATE() - INTERVAL @num_days DAY
    GROUP BY
        language
)
SELECT
    language,
    total_count,
    (total_count * 100) / @num_days AS percentage_share
FROM
    no_of_days ;
```

SQL QUERY



language	total_count	percentage_share
English	1	3.3333
Arabic	1	3.3333
Persian	3	10.0000
Hindi	1	3.3333
French	1	3.3333
Italian	1	3.3333

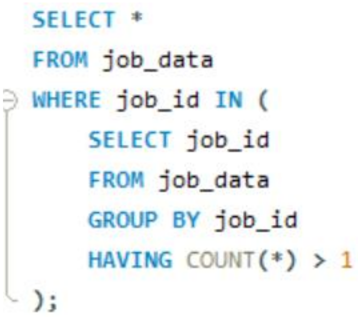
OUTPUT

INSIGHT: To calculate the percentage share of each language over the last 30 days in SQL, aggregate the data by language and then calculate the percentage share based on the count of each language occurrence within the last 30 days compared to the total count of all languages within the same period.

Persian language has the highest percentage share among all other languages.

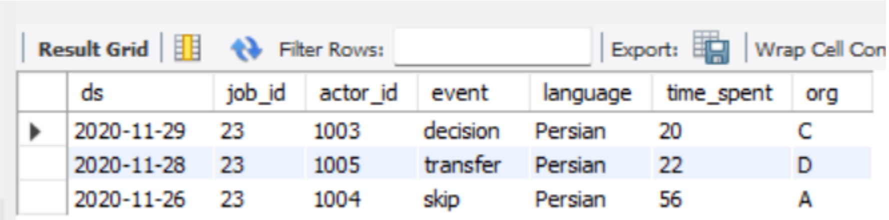
D. Duplicate Rows Detection:

- **Objective: Identify duplicate rows in the data.**
- **SQL Task: Write an SQL query to display duplicate rows from the job_data table.**



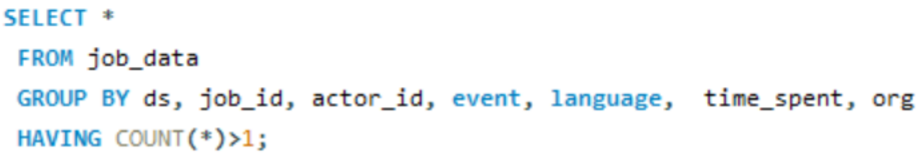
```
SELECT *
FROM job_data
WHERE job_id IN (
    SELECT job_id
    FROM job_data
    GROUP BY job_id
    HAVING COUNT(*) > 1
);
```

SQL Query



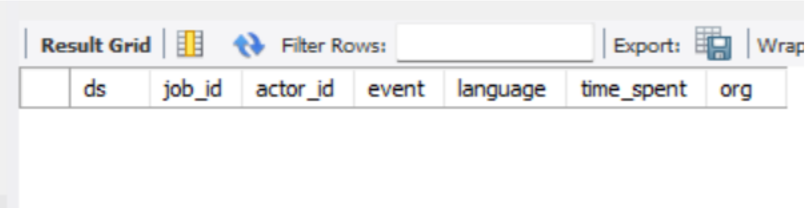
ds	job_id	actor_id	event	language	time_spent	org
2020-11-29	23	1003	decision	Persian	20	C
2020-11-28	23	1005	transfer	Persian	22	D
2020-11-26	23	1004	skip	Persian	56	A

OUTPUT: Duplicate rows using job_id column



```
SELECT *
 FROM job_data
 GROUP BY ds, job_id, actor_id, event, language,  time_spent, org
 HAVING COUNT(*)>1;
```

SQL Query: Duplicate using entire column



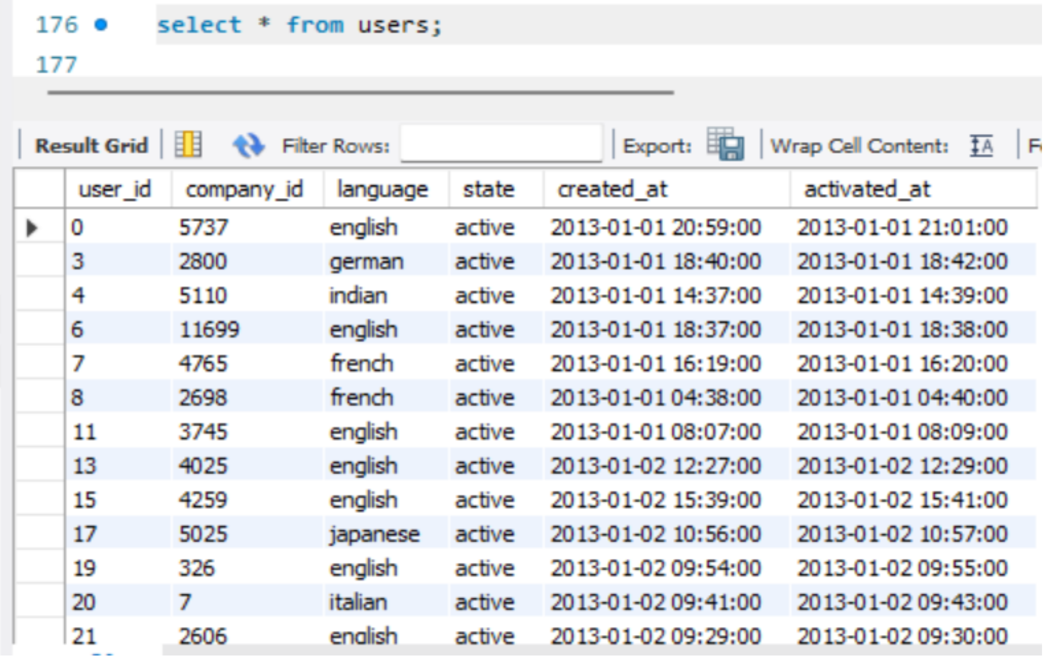
ds	job_id	actor_id	event	language	time_spent	org

OUTPUT: No rows are duplicate in entire row

INSIGHT: There were no rows duplicate in the entire columns. But based on job_id columns 3 rows were found to be duplicate.

❖ Case Study 2: Investigating Metric Spike

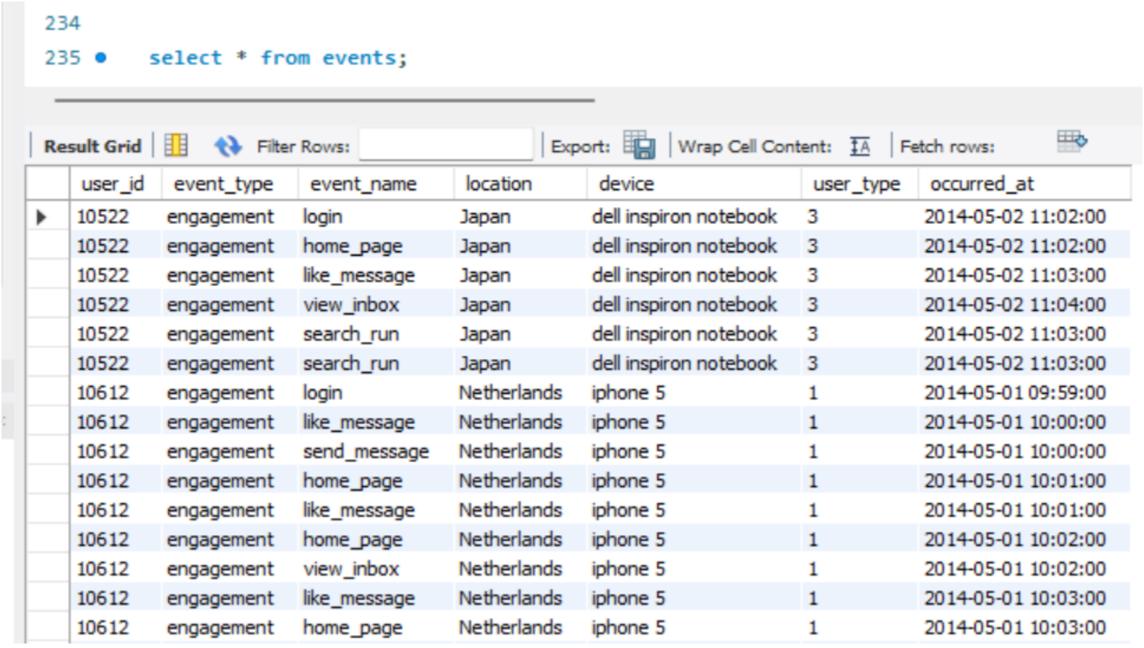
Following are the three tables:
users: Contains one row per user, with descriptive information about that user's account.
events: Contains one row per event, where an event is an action that a user has taken (e.g., login, messaging, search).
email events: Contains events specific to the sending of emails.



```
select * from users;
```

user_id	company_id	language	state	created_at	activated_at
0	5737	english	active	2013-01-01 20:59:00	2013-01-01 21:01:00
3	2800	german	active	2013-01-01 18:40:00	2013-01-01 18:42:00
4	5110	indian	active	2013-01-01 14:37:00	2013-01-01 14:39:00
6	11699	english	active	2013-01-01 18:37:00	2013-01-01 18:38:00
7	4765	french	active	2013-01-01 16:19:00	2013-01-01 16:20:00
8	2698	french	active	2013-01-01 04:38:00	2013-01-01 04:40:00
11	3745	english	active	2013-01-01 08:07:00	2013-01-01 08:09:00
13	4025	english	active	2013-01-02 12:27:00	2013-01-02 12:29:00
15	4259	english	active	2013-01-02 15:39:00	2013-01-02 15:41:00
17	5025	japanese	active	2013-01-02 10:56:00	2013-01-02 10:57:00
19	326	english	active	2013-01-02 09:54:00	2013-01-02 09:55:00
20	7	italian	active	2013-01-02 09:41:00	2013-01-02 09:43:00
21	2606	english	active	2013-01-02 09:29:00	2013-01-02 09:30:00

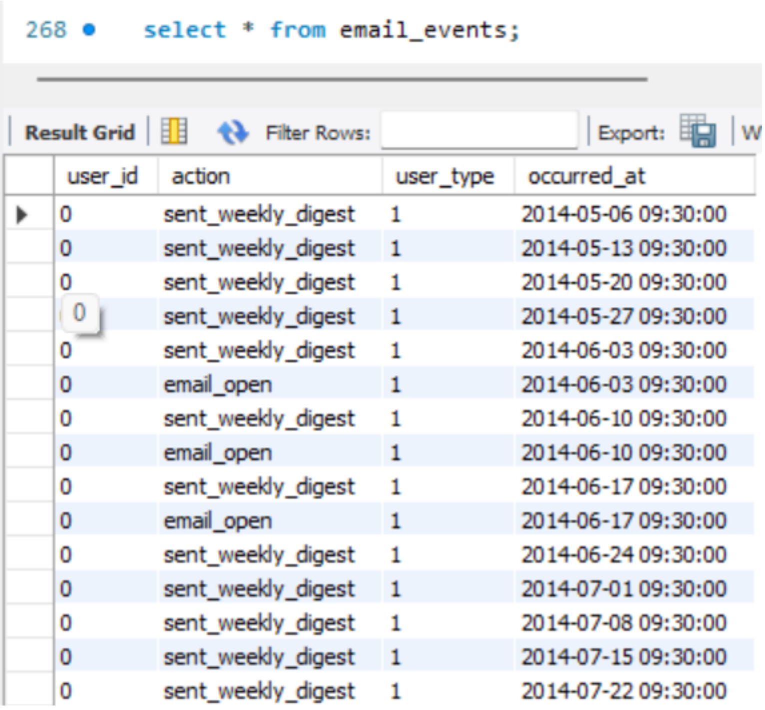
Table 1: Portion of Users table



```sql
select * from events;
```

user_id	event_type	event_name	location	device	user_type	occurred_at
10522	engagement	login	Japan	dell inspiron notebook	3	2014-05-02 11:02:00
10522	engagement	home_page	Japan	dell inspiron notebook	3	2014-05-02 11:02:00
10522	engagement	like_message	Japan	dell inspiron notebook	3	2014-05-02 11:03:00
10522	engagement	view_inbox	Japan	dell inspiron notebook	3	2014-05-02 11:04:00
10522	engagement	search_run	Japan	dell inspiron notebook	3	2014-05-02 11:03:00
10522	engagement	search_run	Japan	dell inspiron notebook	3	2014-05-02 11:03:00
10612	engagement	login	Netherlands	iphone 5	1	2014-05-01 09:59:00
10612	engagement	like_message	Netherlands	iphone 5	1	2014-05-01 10:00:00
10612	engagement	send_message	Netherlands	iphone 5	1	2014-05-01 10:00:00
10612	engagement	home_page	Netherlands	iphone 5	1	2014-05-01 10:01:00
10612	engagement	like_message	Netherlands	iphone 5	1	2014-05-01 10:01:00
10612	engagement	home_page	Netherlands	iphone 5	1	2014-05-01 10:02:00
10612	engagement	view_inbox	Netherlands	iphone 5	1	2014-05-01 10:02:00
10612	engagement	like_message	Netherlands	iphone 5	1	2014-05-01 10:03:00
10612	engagement	home_page	Netherlands	iphone 5	1	2014-05-01 10:03:00

Table 2: Portion of Events table



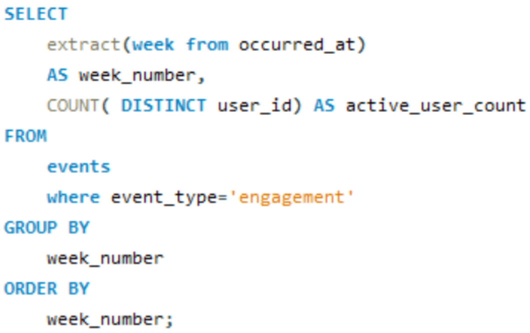
```sql
select * from email_events;
```

user_id	action	user_type	occurred_at
0	sent_weekly_digest	1	2014-05-06 09:30:00
0	sent_weekly_digest	1	2014-05-13 09:30:00
0	sent_weekly_digest	1	2014-05-20 09:30:00
0	sent_weekly_digest	1	2014-05-27 09:30:00
0	sent_weekly_digest	1	2014-06-03 09:30:00
0	email_open	1	2014-06-03 09:30:00
0	sent_weekly_digest	1	2014-06-10 09:30:00
0	email_open	1	2014-06-10 09:30:00
0	sent_weekly_digest	1	2014-06-17 09:30:00
0	email_open	1	2014-06-17 09:30:00
0	sent_weekly_digest	1	2014-06-24 09:30:00
0	sent_weekly_digest	1	2014-07-01 09:30:00
0	sent_weekly_digest	1	2014-07-08 09:30:00
0	sent_weekly_digest	1	2014-07-15 09:30:00
0	sent_weekly_digest	1	2014-07-22 09:30:00

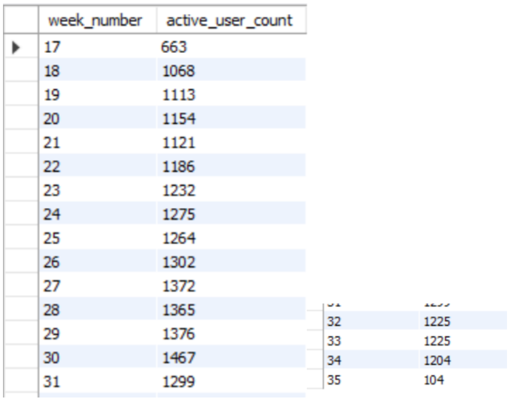
Table 3: Portion of Email_events table.

Tasks:

A. **Weekly User Engagement:**
   - **Objective: Measure the activeness of users on a weekly basis.**
   - **Your Task: Write an SQL query to calculate the weekly user engagement.**



```
SELECT
    extract(week from occurred_at)
    AS week_number,
    COUNT( DISTINCT user_id) AS active_user_count
FROM
    events
    where event_type='engagement'
GROUP BY
    week_number
ORDER BY
    week_number;
```



week_number	active_user_count
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

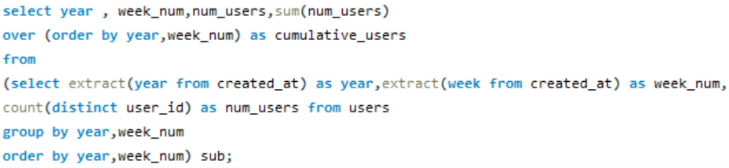
SQL Query OUTPUT

INSIGHT: Measuring the activeness of users on a weekly basis typically involves determining how many unique users were active during each week. "Active" users are those who performed some form of activity within the given week.

Based on the given data users were highly active in week 30 and least active in week 35.

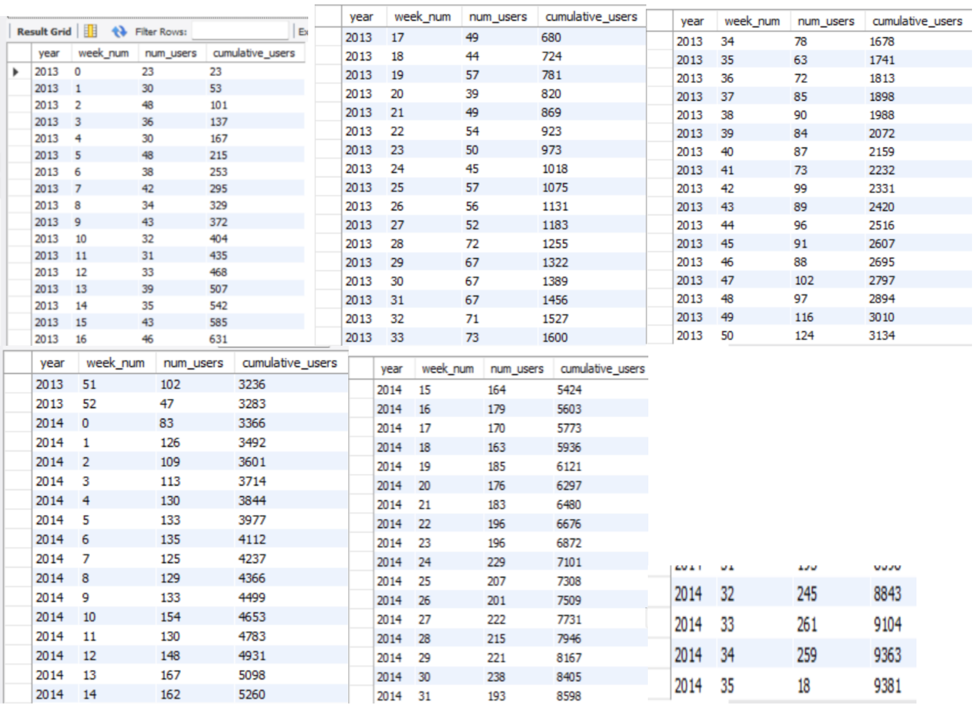
B. **User Growth Analysis:**

- **Objective: Analyse the growth of users over time for a product.**
- **Your Task: Write an SQL query to calculate the user growth for the product.**



```
select year , week_num,num_users,sum(num_users)
over (order by year,week_num) as cumulative_users
from
(select extract(year from created_at) as year,extract(week from created_at) as week_num,
count(distinct user_id) as num_users from users
group by year,week_num
order by year,week_num) sub;
```

SQL Query and OUTPUT



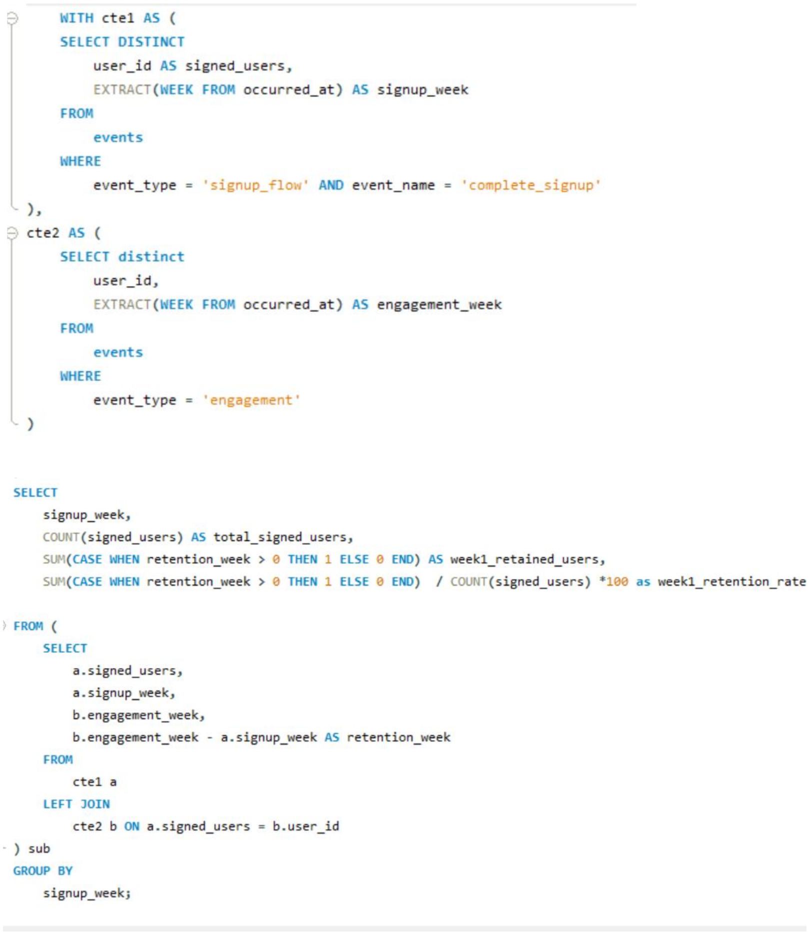
year	week_num	num_users	cumulative_users
2013	0	23	23
2013	1	30	53
2013	2	48	101
2013	3	36	137
2013	4	30	167
2013	5	48	215
2013	6	38	253
2013	7	42	295
2013	8	34	329
2013	9	43	372
2013	10	32	404
2013	11	31	435
2013	12	33	468
2013	13	39	507
2013	14	35	542
2013	15	43	585
2013	16	46	631
2013	17	49	680
2013	18	44	724
2013	19	57	781
2013	20	39	820
2013	21	49	869
2013	22	54	923
2013	23	50	973
2013	24	45	1018
2013	25	57	1075
2013	26	56	1131
2013	27	52	1183
2013	28	72	1255
2013	29	67	1322
2013	30	67	1389
2013	31	67	1456
2013	32	71	1527
2013	33	73	1600
2013	34	78	1678
2013	35	63	1741
2013	36	72	1813
2013	37	85	1898
2013	38	90	1988
2013	39	84	2072
2013	40	87	2159
2013	41	73	2232
2013	42	99	2331
2013	43	89	2420
2013	44	96	2516
2013	45	91	2607
2013	46	88	2695
2013	47	102	2797
2013	48	97	2894
2013	49	116	3010
2013	50	124	3134
2013	51	102	3236
2013	52	47	3283
2014	0	83	3366
2014	1	126	3492
2014	2	109	3601
2014	3	113	3714
2014	4	130	3844
2014	5	133	3977
2014	6	135	4112
2014	7	125	4237
2014	8	129	4366
2014	9	133	4499
2014	10	154	4653
2014	11	130	4783
2014	12	148	4931
2014	13	167	5098
2014	14	162	5260
2014	15	164	5424
2014	16	179	5603
2014	17	170	5773
2014	18	163	5936
2014	19	185	6121
2014	20	176	6297
2014	21	183	6480
2014	22	196	6676
2014	23	196	6872
2014	24	229	7101
2014	25	207	7308
2014	26	201	7509
2014	27	222	7731
2014	28	215	7946
2014	29	221	8167
2014	30	238	8405
2014	31	193	8598
2014	32	245	8843
2014	33	261	9104
2014	34	259	9363
2014	35	18	9381

INSIGHT: To calculate the user growth for a product, the total number of users between two different periods of time were compared.

The above query gives the total number of users registered in week of the year, as well as the cumulative number of users up to that week. We can then analyse the growth trend over time based on this data. From the output we observe that for some week users increases and in some week user decreases.

C. **Weekly Retention Analysis:**

   - **Objective: Analyse the retention of users on a weekly basis after signing up for a product.**
   - **Your Task: Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.**

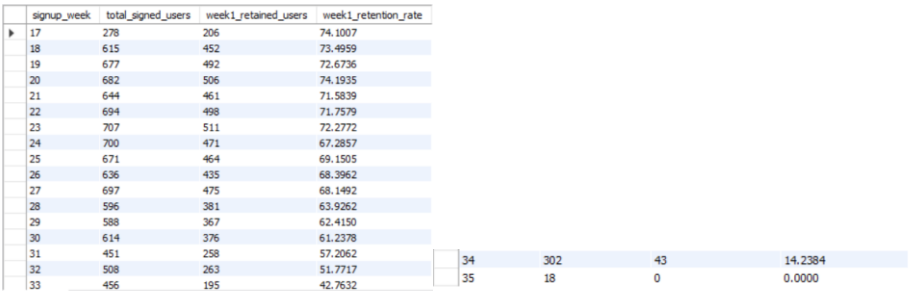


```
    WITH cte1 AS (
    SELECT DISTINCT
        user_id AS signed_users,
        EXTRACT(WEEK FROM occurred_at) AS signup_week
    FROM
        events
    WHERE
        event_type = 'signup_flow' AND event_name = 'complete_signup'
),
cte2 AS (
    SELECT distinct
        user_id,
        EXTRACT(WEEK FROM occurred_at) AS engagement_week
    FROM
        events
    WHERE
        event_type = 'engagement'
)


SELECT
    signup_week,
    COUNT(signed_users) AS total_signed_users,
    SUM(CASE WHEN retention_week > 0 THEN 1 ELSE 0 END) AS week1_retained_users,
    SUM(CASE WHEN retention_week > 0 THEN 1 ELSE 0 END)  / COUNT(signed_users) *100 as week1_retention_rate

FROM (
    SELECT
        a.signed_users,
        a.signup_week,
        b.engagement_week,
        b.engagement_week - a.signup_week AS retention_week
    FROM
        cte1 a
    LEFT JOIN
        cte2 b ON a.signed_users = b.user_id
) sub
GROUP BY
    signup_week;
```

SQL QUERY



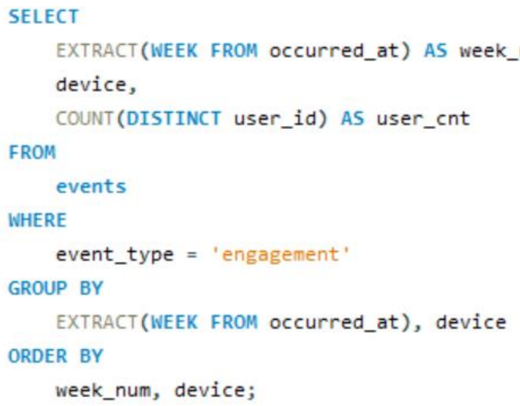
signup_week	total_signed_users	week1_retained_users	week1_retention_rate
17	278	206	74.1007
18	615	452	73.4959
19	677	492	72.6736
20	682	506	74.1935
21	644	461	71.5839
22	694	498	71.7579
23	707	511	72.2772
24	700	471	67.2857
25	671	464	69.1505
26	636	435	68.3962
27	697	475	68.1492
28	596	381	63.9262
29	588	367	62.4150
30	614	376	61.2378
31	451	258	57.2062
32	508	263	51.7717
33	456	195	42.7632
34	302	43	14.2384
35	18	0	0.0000

INSIGHT: Weekly retention refers to the percentage of users who continue to engage with a product or service over time, specifically on a weekly basis, after initially signing up or starting to use the product. It measures the ability of a product to retain its users over consecutive weeks. This query weekly retention rate after first week for each sign-up cohort.

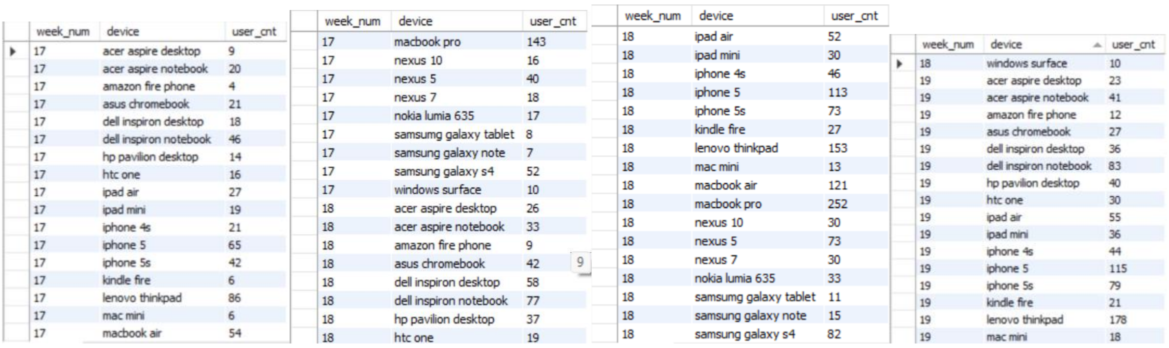
This analysis allows businesses to track the effectiveness of their retention strategies and assess user satisfaction and loyalty.

We observe that in week 34 approximately 14% are retained where as in week 20 approximately 74% of total users are retained.

D. **Weekly Engagement Per Device:**
   - **Objective: Measure the activeness of users on a weekly basis per device.**
   - **Your Task: Write an SQL query to calculate the weekly engagement per device.**



```
SELECT
    EXTRACT(WEEK FROM occurred_at) AS week_num,
    device,
    COUNT(DISTINCT user_id) AS user_cnt
FROM
    events
WHERE
    event_type = 'engagement'
GROUP BY
    EXTRACT(WEEK FROM occurred_at), device
ORDER BY
    week_num, device;
```



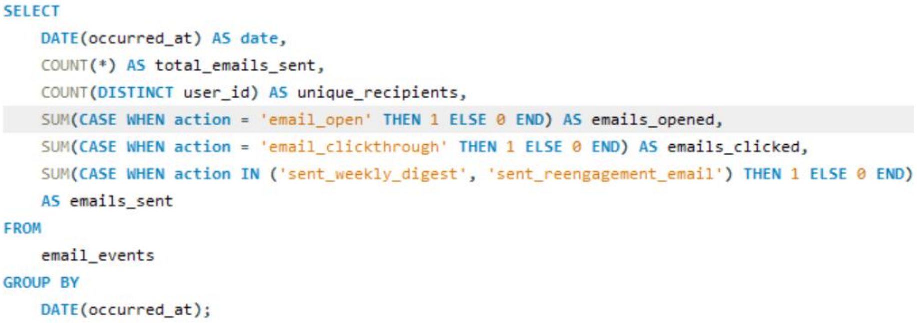
week_num	device	user_cnt
17	acer aspire desktop	9
17	acer aspire notebook	20
17	amazon fire phone	4
17	asus chromebook	21
17	dell inspiron desktop	18
17	dell inspiron notebook	46
17	hp pavilion desktop	14
17	htc one	16
17	ipad air	27
17	ipad mini	19
17	iphone 4s	21
17	iphone 5	65
17	iphone 5s	42
17	kindle fire	6
17	lenovo thinkpad	86
17	mac mini	6
17	macbook air	54
17	macbook pro	143
17	nexus 10	16
17	nexus 5	40
17	nexus 7	18
17	nokia lumia 635	17
17	samsumg galaxy tablet	8
17	samsung galaxy note	7
17	samsung galaxy s4	52
17	windows surface	10
18	acer aspire desktop	26
18	acer aspire notebook	33
18	amazon fire phone	9
18	asus chromebook	42
18	dell inspiron desktop	58
18	dell inspiron notebook	77
18	hp pavilion desktop	37
18	htc one	19
18	ipad air	52
18	ipad mini	30
18	iphone 4s	46
18	iphone 5	113
18	iphone 5s	73
18	kindle fire	27
18	lenovo thinkpad	153
18	mac mini	13
18	macbook air	121
18	macbook pro	252
18	nexus 10	30
18	nexus 5	73
18	nexus 7	30
18	nokia lumia 635	33
18	samsumg galaxy tablet	11
18	samsung galaxy note	15
18	samsung galaxy s4	82
18	windows surface	10
19	acer aspire desktop	23
19	acer aspire notebook	41
19	amazon fire phone	12
19	asus chromebook	27
19	dell inspiron desktop	36
19	dell inspiron notebook	83
19	hp pavilion desktop	40
19	htc one	30
19	ipad air	55
19	ipad mini	36
19	iphone 4s	44
19	iphone 5	115
19	iphone 5s	79
19	kindle fire	21
19	lenovo thinkpad	178
19	mac mini	18

*more rows are there which are not shown here

OUTPUT

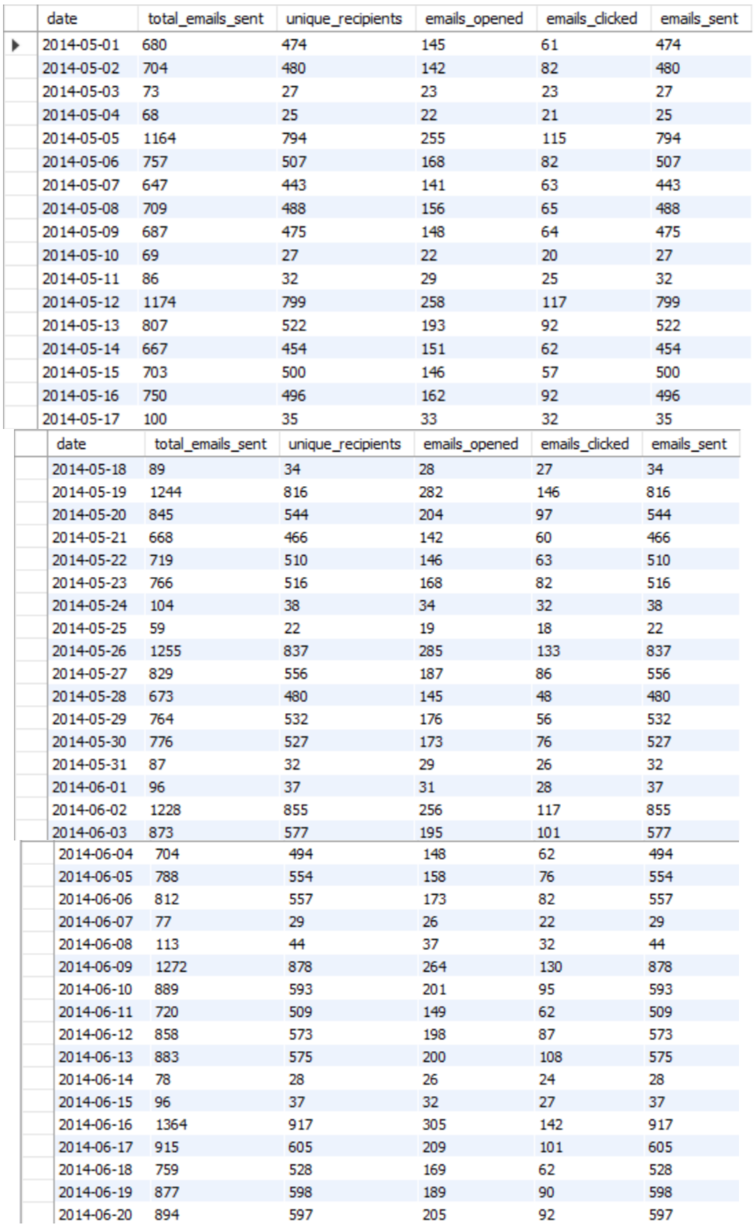
INSIGHTS: This query gives the weekly engagement count for each device. In week 30 device MacBook pro was used by highest no. of users that is 322.

E. **Email Engagement Analysis:**
   - **Objective: Analyse how users are engaging with the email service.**
   - **Your Task: Write an SQL query to calculate the email engagement metrics.**



```
SELECT
    DATE(occurred_at) AS date,
    COUNT(*) AS total_emails_sent,
    COUNT(DISTINCT user_id) AS unique_recipients,
    SUM(CASE WHEN action = 'email_open' THEN 1 ELSE 0 END) AS emails_opened,
    SUM(CASE WHEN action = 'email_clickthrough' THEN 1 ELSE 0 END) AS emails_clicked,
    SUM(CASE WHEN action IN ('sent_weekly_digest', 'sent_reengagement_email') THEN 1 ELSE 0 END)
    AS emails_sent
FROM
    email_events
GROUP BY
    DATE(occurred_at);
```

SQL QUERY



date	total_emails_sent	unique_recipients	emails_opened	emails_clicked	emails_sent
2014-05-01	680	474	145	61	474
2014-05-02	704	480	142	82	480
2014-05-03	73	27	23	23	27
2014-05-04	68	25	22	21	25
2014-05-05	1164	794	255	115	794
2014-05-06	757	507	168	82	507
2014-05-07	647	443	141	63	443
2014-05-08	709	488	156	65	488
2014-05-09	687	475	148	64	475
2014-05-10	69	27	22	20	27
2014-05-11	86	32	29	25	32
2014-05-12	1174	799	258	117	799
2014-05-13	807	522	193	92	522
2014-05-14	667	454	151	62	454
2014-05-15	703	500	146	57	500
2014-05-16	750	496	162	92	496
2014-05-17	100	35	33	32	35
2014-05-18	89	34	28	27	34
2014-05-19	1244	816	282	146	816
2014-05-20	845	544	204	97	544
2014-05-21	668	466	142	60	466
2014-05-22	719	510	146	63	510
2014-05-23	766	516	168	82	516
2014-05-24	104	38	34	32	38
2014-05-25	59	22	19	18	22
2014-05-26	1255	837	285	133	837
2014-05-27	829	556	187	86	556
2014-05-28	673	480	145	48	480
2014-05-29	764	532	176	56	532
2014-05-30	776	527	173	76	527
2014-05-31	87	32	29	26	32
2014-06-01	96	37	31	28	37
2014-06-02	1228	855	256	117	855
2014-06-03	873	577	195	101	577
2014-06-04	704	494	148	62	494
2014-06-05	788	554	158	76	554
2014-06-06	812	557	173	82	557
2014-06-07	77	29	26	22	29
2014-06-08	113	44	37	32	44
2014-06-09	1272	878	264	130	878
2014-06-10	889	593	201	95	593
2014-06-11	720	509	149	62	509
2014-06-12	858	573	198	87	573
2014-06-13	883	575	200	108	575
2014-06-14	78	28	26	24	28
2014-06-15	96	37	32	27	37
2014-06-16	1364	917	305	142	917
2014-06-17	915	605	209	101	605
2014-06-18	759	528	169	62	528
2014-06-19	877	598	189	90	598
2014-06-20	894	597	205	92	597

OUTPUT

INSIGHT: the above query calculates various email engagement metrics such as total emails sent, unique recipients, emails opened, emails clicked, and emails sent for each date.