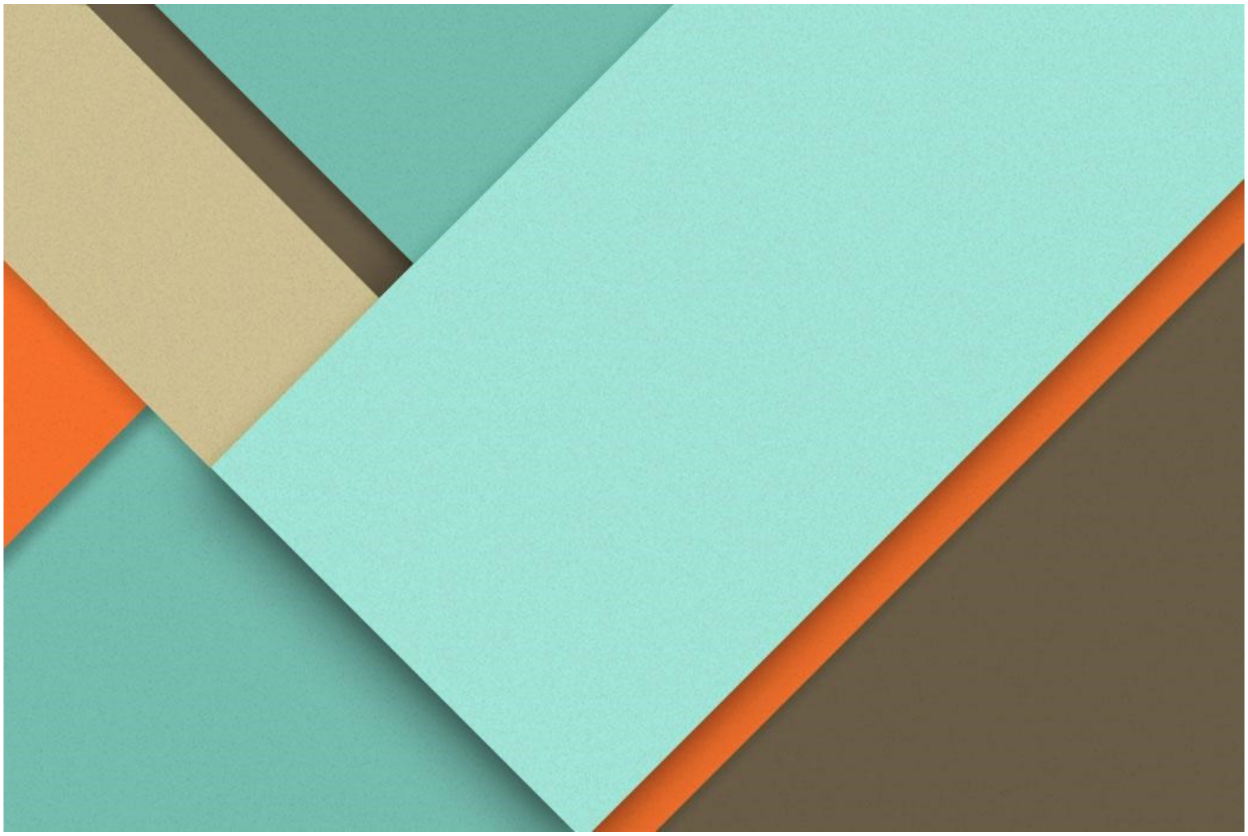


Operation Analytics and Investigating Metric Spike

—

03. 07. 2023

Project Description

Operation Analytics is the analysis done for the complete end to end operations of a company. With the help of this, the company then finds the areas on which it must improve upon. You work closely with the ops team, support team, marketing team, etc and help them derive insights out of the data they collect.

Being one of the most important parts of a company, this kind of analysis is further used to predict the overall growth or decline of a company's fortune. It means better automation, better understanding between cross-functional teams, and more effective workflows.

Investigating metric spike is also an important part of operation analytics as being 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 its very important to investigate metric spike.

Below are the two operations mentioning the answers for the related questions:

Case Study 1 (Job Data)

Below is the structure of the table with the definition of each column that you must work on:

- **Table-1:** job_data
    - **job_id:** unique identifier of jobs
    - **actor_id:** unique identifier of actor
    - **event:** decision/skip/transfer
    - **language:** language of the content
    - **time_spent:** time spent to review the job in seconds
    - **org:** organization of the actor
    - **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 date function

Use the dataset attached in the Dataset section below the project images then answer the questions that follows

1. **Number of jobs reviewed:** Amount of jobs reviewed over time.
   Task: Calculate the number of jobs reviewed per hour per day for November 2020?
2. **Throughput:** It is the no. of events happening per second.
   Task: 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?
3. **Percentage share of each language:** Share of each language for different contents.
   Task: Calculate the percentage share of each language in the last 30 days?
4. **Duplicate rows:** Rows that have the same value present in them.
   Task: Let's say you see some duplicate rows in the data. How will you display duplicates from the table?

Case Study 2 (Investigating metric spike)

The structure of the table with the definition of each column that you must work on is present in the project image

- **Table-1:** users
  This table includes one row per user, with descriptive information about that user's account.
- **Table-2:** events
  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.
- **Table-3:** email_events
  This table contains events specific to the sending of emails. It is similar in structure to the events table above.

Use the dataset attached in the Dataset section below the project images then answer the questions that follows

1. **User Engagement:** To measure the activeness of a user. Measuring if the user finds quality in a product/service.
   Task: Calculate the weekly user engagement?
2. **User Growth:** Amount of users growing over time for a product.
   Task: Calculate the user growth for product?
3. **Weekly Retention:** Users getting retained weekly after signing-up for a product.
   Task: Calculate the weekly retention of users-sign up cohort?

4. **Weekly Engagement:** To measure the activeness of a user. Measuring if the user finds quality in a product/service weekly.
   Task: Calculate the weekly engagement per device?
5. **Email Engagement:** Users engaging with the email service.
   Task: Calculate the email engagement metrics?

Approach

- First, I created the database and tables in MySQL Workbench, then importing the csv data provided using load data infile methods for tables having large number of rows, further describing tables checking datatypes of columns to modify column if there is different data types and used table import wizard for tables having small number of rows.
- After creating the required tables and data insertion, I spent some time understanding each column in the tables in Microsoft Excel and then I carried my analysis by writing SQL queries in MySQL Workbench and showcasing analysis using Tableau.

Tech-Stack Used

- **MySQL Workbench v8.0.33.0 community edition** is used for project execution in order to query the database and gather the required results.
- **Microsoft Excel Office 2019 version** was used to understand columns and datasets of different csv files. The Datasets can be referred here.
- **Tableau Desktop v2019.4.1 professional edition** is used to create visual representation of results for creation of graphs and other charts to understand the result better and make better decisions.

Results and Insights

Case Study-1 (Job Data)

1.1. Number of jobs reviewed per hour per day for November 2020

```sql
SELECT
    ds as date,
    COUNT(*) AS job_reviewed,
    sum(time_spent) / 3600 AS time_spent_per_hour
FROM
    jobs
WHERE
    ds >= '2020-11-01' AND ds < '2020-12-01'
GROUP BY date;
```

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

date	job_reviewed	time_spent_per_hour
2020-11-24	1	0.0158
2020-11-25	1	0.0125
2020-11-26	1	0.0156
2020-11-27	1	0.0289
2020-11-28	2	0.0092
2020-11-29	1	0.0056
2020-11-30	2	0.0111

1.2. 7-day rolling average of throughput

```sql
CREATE TEMPORARY TABLE JOBS_REVIEWED
(
SELECT ds,count(distinct job_id) as jobs_reviewed,CAST(COUNT(DISTINCT JOB_ID)/86400 AS DECIMAL(10,10)) AS THROUGHPUT
    from jobs
    group by ds
    order by ds
);
SELECT ds,jobs_reviewed,throughput
,avg(throughput) OVER(ORDER BY DS ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS throughput_7day
FROM JOBS_REVIEWED;
```

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

ds	jobs_reviewed	throughput	throughput_7day
2020-11-24	1	0.0000115740	0.00001157400000
2020-11-25	1	0.0000115740	0.00001157400000
2020-11-26	1	0.0000115740	0.00001157400000
2020-11-27	1	0.0000115740	0.00001157400000
2020-11-28	2	0.0000231480	0.00001322742857
2020-11-29	1	0.0000115740	0.00001322742857
2020-11-30	2	0.0000231480	0.00001488085714

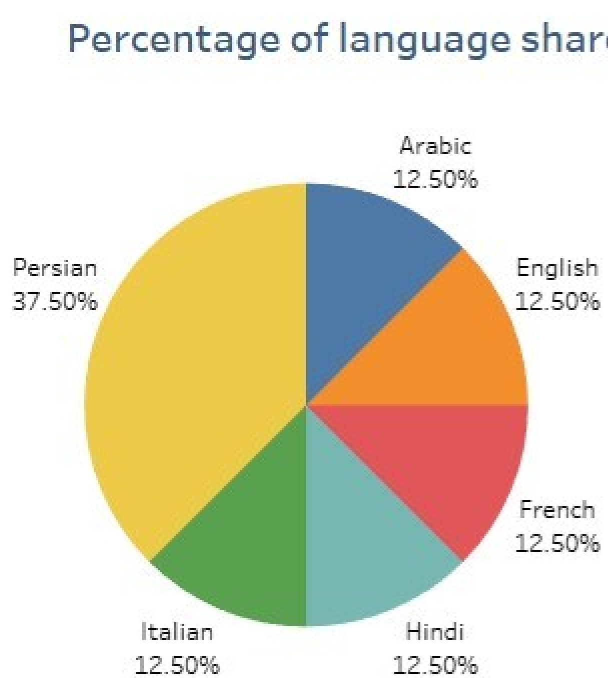
1.3. Percentage share of each language in the last 30 days i.e November month.

```sql
-- Percentage share of each language: Share of each language for different contents in last 30 days.

SELECT
    language, count(job_id) as jobs_applied,
    COUNT(*) / (SELECT COUNT(*) FROM jobs WHERE ds >= '2020-11-01' AND ds < '2020-12-01') * 100
    AS percentage_share_of_language
FROM
    jobs
WHERE
    ds >= '2020-11-01' AND ds < '2020-12-01'
GROUP BY language;
```

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

language	jobs_applied	percentage_share_of_language
French	4	12.5000
Persian	12	37.5000
Italian	4	12.5000
English	4	12.5000
Arabic	4	12.5000
Hindi	4	12.5000



1.4. Display of duplicate rows from the table.

```
select * from
(
select *,
row_number() over (partition by job_id order by ds) as rownum
from jobs
)as t
where rownum>1;
```

ds	job_id	actor_id	event	language	time_spent	org	rownum
2020-10-17	3	1011	skip	Persian	63	A	2
2020-11-10	3	1014	skip	Persian	27	A	3
2020-11-07	10	1020	decision	Hindi	14	B	2
2020-09-27	11	1003	decision	Persian	61	C	2
2020-11-27	11	1007	decision	French	104	D	3
2020-09-16	12	1020	transfer	Italian	80	C	2
2020-11-26	23	1004	skip	Persian	56	A	2
2020-11-28	23	1005	transfer	Persian	22	D	3

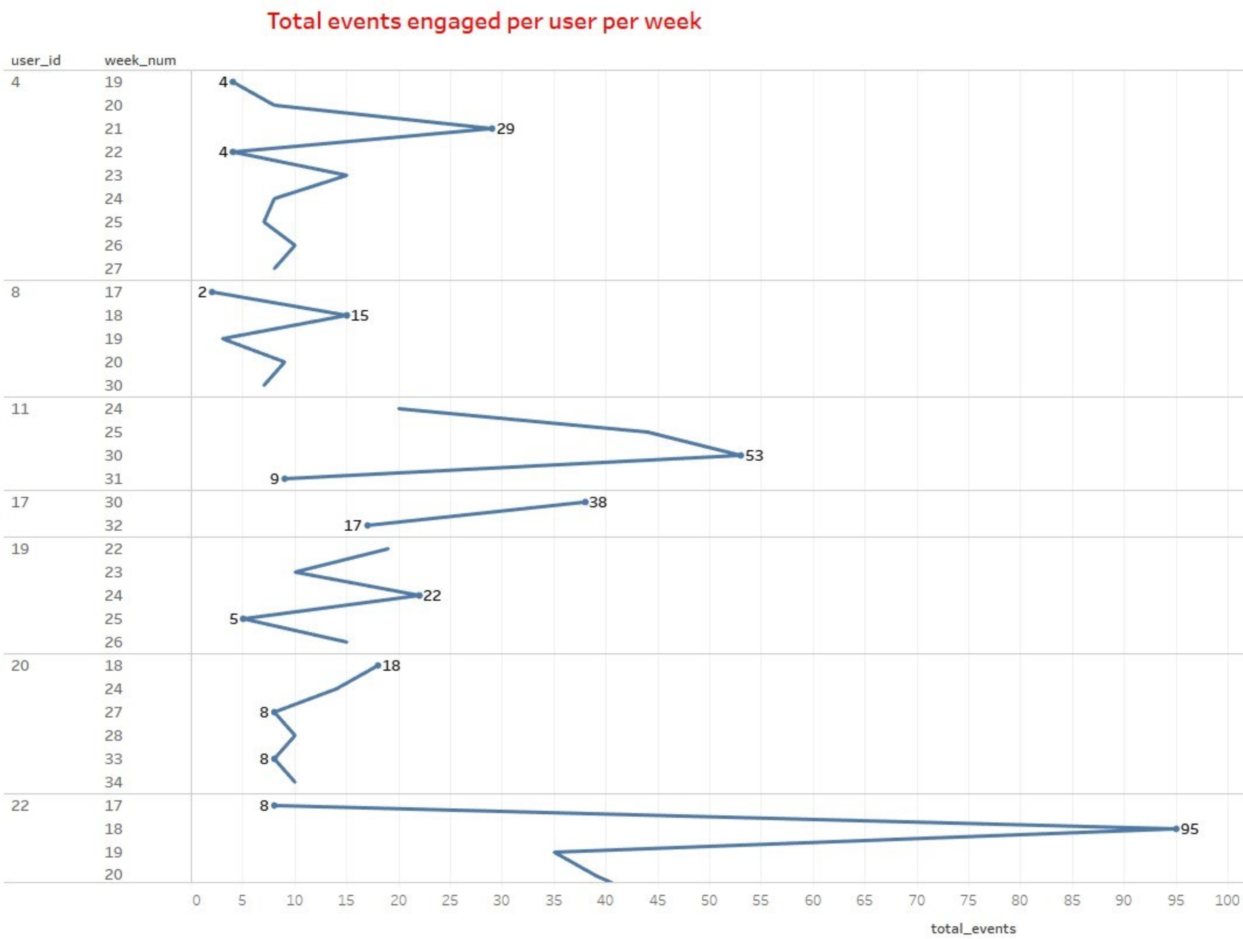
Case Study 2 (Investigating metric spike)

2.1. weekly user engagement

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

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

user_id	week_num	total_events
4	19	4
4	20	8
4	21	29
4	22	4
4	23	15
4	24	8
4	25	7
4	26	10

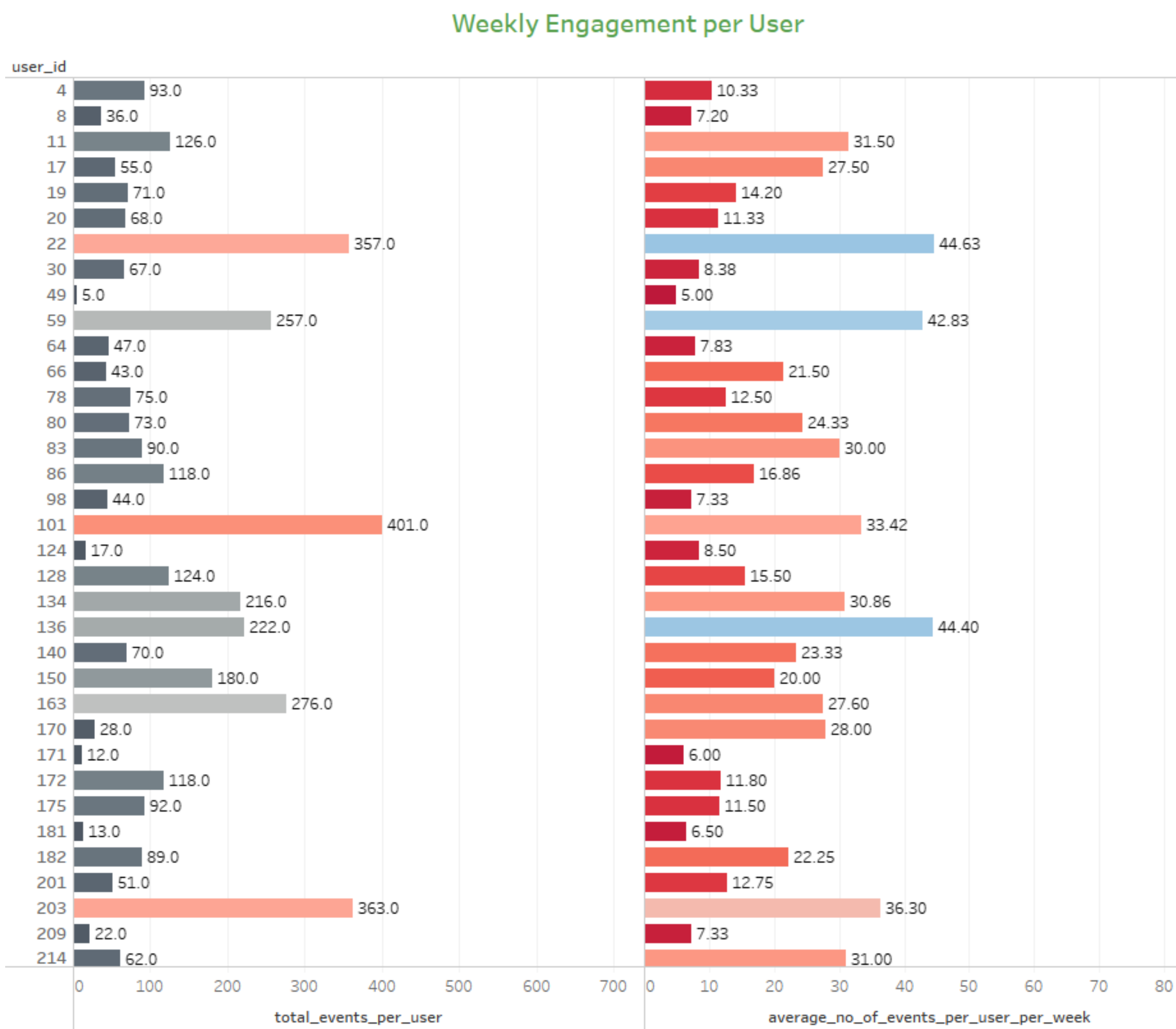


Above visualization, shows the highest and lowest number of events per week for each user.

```sql
SELECT
    user_id,
    sum(event_name) as total_events_per_user,
    AVG(event_name) AS average_no_of_events_per_user_per_week
FROM (
    SELECT
        user_id,
        COUNT(event_name) AS event_name
    FROM events
    WHERE
        event_type='engagement'
    GROUP BY user_id, week(occurred_at)
) subquery
GROUP BY user_id
ORDER BY user_id;
```

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

user_id	total_events_per_user	average_no_of_events_per_user_per_week
4	93	10.3333
8	36	7.2000
11	126	31.5000
17	55	27.5000
19	71	14.2000
20	68	11.3333
22	357	44.6250
30	67	8.3750


Weekly Engagement per User
user_id
4
8
11
17
19
20
22
30
49
59
64
66
78
80
83
86
98
101
124
128
134
136
140
150
163
170
171
172
175
181
182
201
203
209
214
93.0
36.0
126.0
55.0
71.0
68.0
357.0
67.0
5.0
257.0
47.0
43.0
75.0
73.0
90.0
118.0
44.0
401.0
17.0
124.0
216.0
222.0
70.0
180.0
276.0
28.0
12.0
118.0
92.0
13.0
89.0
51.0
363.0
22.0
62.0
10.33
7.20
31.50
27.50
14.20
11.33
44.63
8.38
5.00
42.83
7.83
21.50
12.50
24.33
30.00
16.86
7.33
33.42
8.50
15.50
30.86
44.40
23.33
20.00
27.60
28.00
6.00
11.80
11.50
6.50
22.25
12.75
36.30
7.33
31.00
0 100 200 300 400 500 600 700
total_events_per_user
0 10 20 30 40 50 60 70 80
average_no_of_events_per_user_per_week

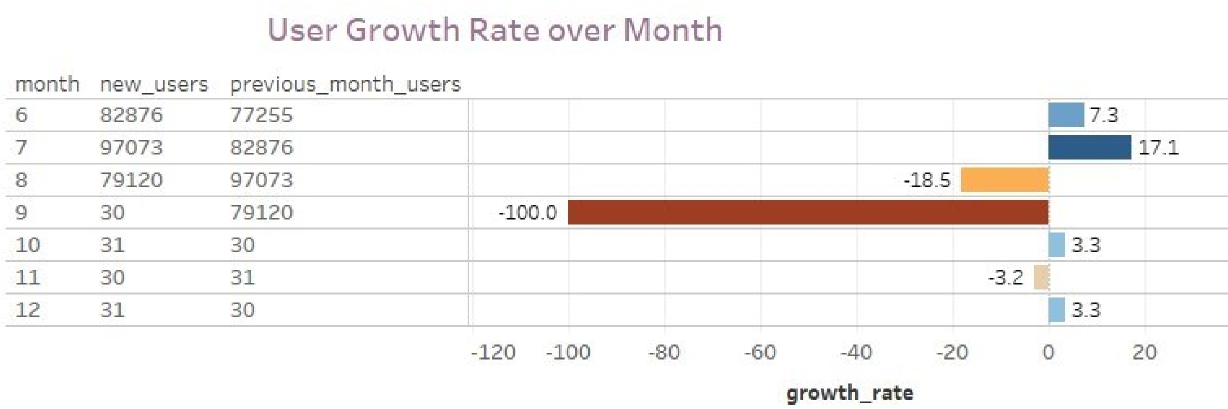
2.2. User Growth rate

```sql
SELECT
    extract(month from occurred_at) AS month,
    COUNT(*) AS new_users,
    LAG(COUNT(*)) OVER (ORDER BY extract(month from occurred_at)) AS previous_month_users,

    (COUNT(*) - LAG(COUNT(*)) OVER (ORDER BY extract(month from occurred_at))) /
    LAG(COUNT(*)) OVER (ORDER BY extract(month from occurred_at)) * 100 AS growth_rate
FROM
    events
WHERE
    occurred_at >= '2014-01-01'
    AND occurred_at< '2015-01-01'
GROUP BY
    extract(month from occurred_at)
ORDER BY
    extract(month from occurred_at);
```

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

month	new_users	previous_month_users	growth_rate
5	77255	NULL	NULL
6	82876	77255	7.2759
7	97073	82876	17.1304
8	79120	97073	-18.4943
9	30	79120	-99.9621
10	31	30	3.3333
11	30	31	-3.2258
12	31	30	3.3333



July Month has the highest user growth rate.

```sql
SELECT
    extract(month from occurred_at) AS month, extract(week from occurred_at) as week,
    COUNT(*) AS new_users,
    LAG(COUNT(*)) OVER (ORDER BY extract(month from occurred_at)) AS previous_new_users,
    (COUNT(*) - LAG(COUNT(*)) OVER (ORDER BY extract(month from occurred_at))) /
    LAG(COUNT(*)) OVER (ORDER BY extract(month from occurred_at)) * 100 AS growth_rate
FROM
    events
WHERE
    occurred_at >= '2014-01-01'
    AND occurred_at< '2015-01-01'
GROUP BY
    extract(month from occurred_at), extract(week from occurred_at)
ORDER BY
    extract(month from occurred_at);
```

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

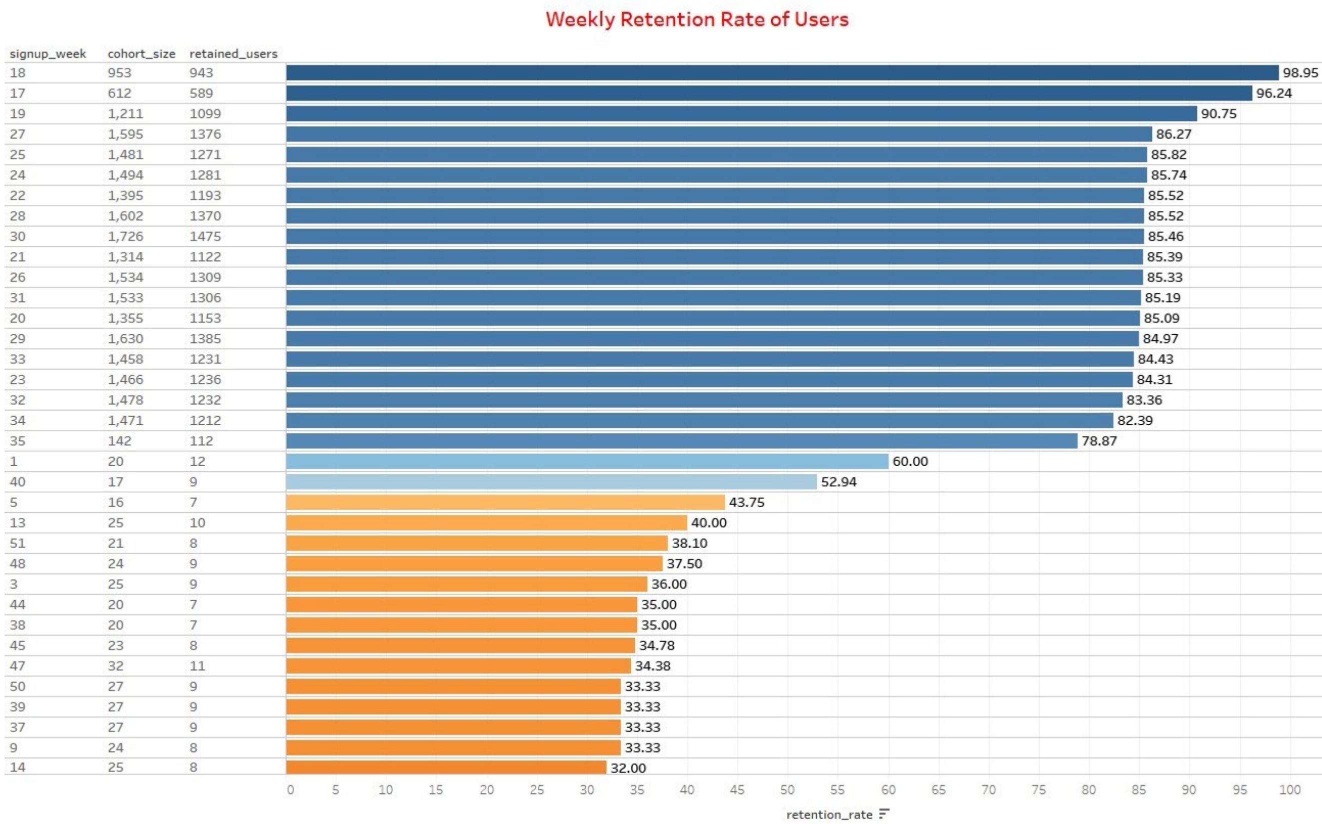
month	week	new_users	previous_new_users	growth_rate
5	17	7404	NULL	NULL
5	18	15850	7404	114.0735
5	19	17155	15850	8.2334
5	20	18775	17155	9.4433
5	21	18071	18775	-3.7497
6	22	19444	18071	7.5978
6	23	19317	19444	-0.6532
6	24	20217	19317	4.6591

2.3. Weekly Retention rate: Users engaging product weekly after signing-up for a product

```sql
SELECT
    signup_week,
    cohort_size,
    retained_users,
    (retained_users / cohort_size) * 100 AS retention_rate
FROM (
    SELECT
        signup_week,
        COUNT(DISTINCT user_id) AS cohort_size,
        count(DISTINCT CASE WHEN signup_week < activity_week AND event_type = 'signup_flow'
        and event_name='complete_signup' then user_id END) AS signup__users,
        count(DISTINCT CASE WHEN activity_week >= signup_week AND event_type = 'engagement'
        and event_name='login' then user_id END) AS retained_users
    FROM (
        SELECT
            user_id,
            event_type, event_name,
            extract(week from occurred_at) AS signup_week,
            extract(week from occurred_at) AS activity_week
        FROM
            events
    ) sub1
    GROUP BY
        signup_week
) sub2
ORDER BY
    signup_week;
```

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

signup_week	cohort_size	retained_users	retention_rate
0	17	4	23.5294
1	20	12	60.0000
2	27	5	18.5185
3	25	9	36.0000
4	27	6	22.2222
5	16	7	43.7500
6	19	6	31.5789
7	21	6	28.5714

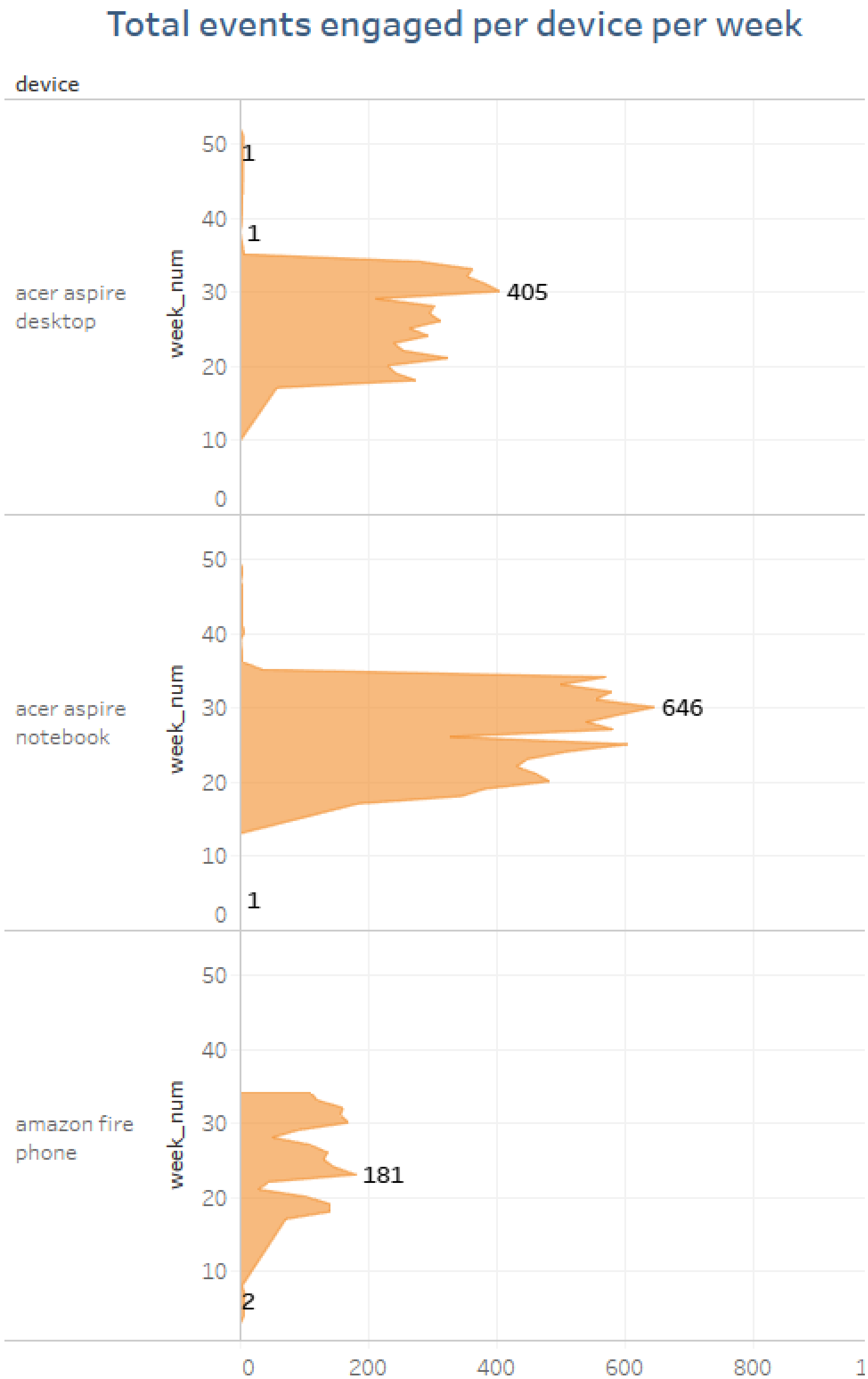


Week-18 has the highest retention rate of signup users.

2.4. Weekly Engagement per Device

```sql
SELECT
    device,
    EXTRACT(WEEK FROM occurred_at) AS week_num,
    COUNT(event_name) AS total_events
FROM
    events
WHERE
    event_type = 'engagement'
GROUP BY device , week_num;
```

device	week_num	total_events
dell inspiron notebook	17	485
dell inspiron notebook	18	877
iphone 5	18	1273
iphone 5	19	1166
iphone 5	20	1281
iphone 4s	20	607
windows surface	20	189
macbook air	21	1279

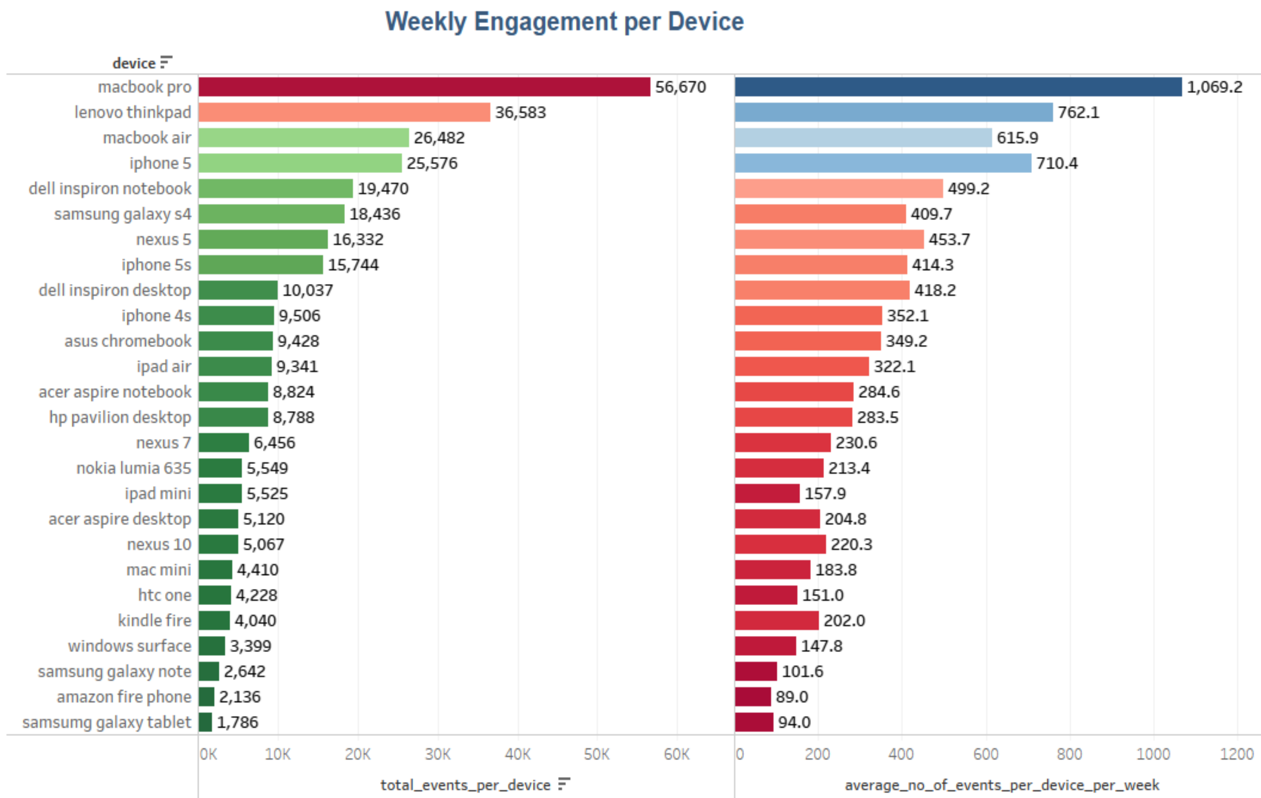


Above visualization, shows highest and lowest point of device engagement per week.

```sql
SELECT
    device,
    sum(event_name) as total_events_per_device,
    AVG(event_name) AS average_no_of_events_per_device_per_week
FROM (
    SELECT
        device,
        COUNT(event_name) AS event_name
    FROM events
    where
        event_type='engagement'
    GROUP BY device, week(occurred_at)
) subquery
GROUP BY device;
```

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

device	total_events_per_device	average_no_of_events_per_device_per_week
dell inspiron notebook	19470	499.2308
iphone 5	25576	710.4444
iphone 4s	9506	352.0741
windows surface	3399	147.7826
macbook air	26482	615.8605
iphone 5s	15744	414.3158
macbook pro	56670	1069.2453
kindle fire	4040	202.0000



2.5. Email-engagement metrics

```sql
SELECT
    ee.action,
    COUNT(ee.action) AS event_count,
    COUNT(*) / (select count(*) from email_events) * 100 AS percentage_share
FROM
    email_events ee
GROUP BY
    ee.action;
```

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

action	event_count	percentage_share
sent_weekly_digest	57267	63.3562
email_open	20459	22.6344
email_clickthrough	9010	9.9680
sent_reengagement_email	3653	4.0414

Email Engagement Metrics