ANALYTICS PROJECT

Operation Analytics and Investigating Metric Spike

A Project Delivered By Using SQL techniques.

Presented By : Jacky Kumar

PROJECT DESCRIPTION:

Operation analytics is done for the complete end-to-end operations of the company. With its help, the company then finds areas it needs to improve. You work closely with the ops, support, and marketing team among others, and help them derive insight from the data they collect.

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

Presented By : Jacky Kumar

Investigation of metric spike

This is an important part of operation analytics because, as a Data Analyst, you must be able to understand-or make other teams understand-questions such as: Why is daily engagement dipping? Why have sales taken a dip? Etc. Such questions come up daily and, for that, investigating metric spikes becomes very important.

Presented By : Jacky Kumar

APPROACH

To approach an Operation & Metric Analytics project, start by defining clear objectives and identifying the key metrics (KPIs) that align with business goals. Collect relevant operational data from various sources to ensure a comprehensive dataset for analysis.

Goals 1

Define Objectives and Collect Data

Goals 2

Data Preparation and Exploration

Goals 3

Analyze data

Goals 4

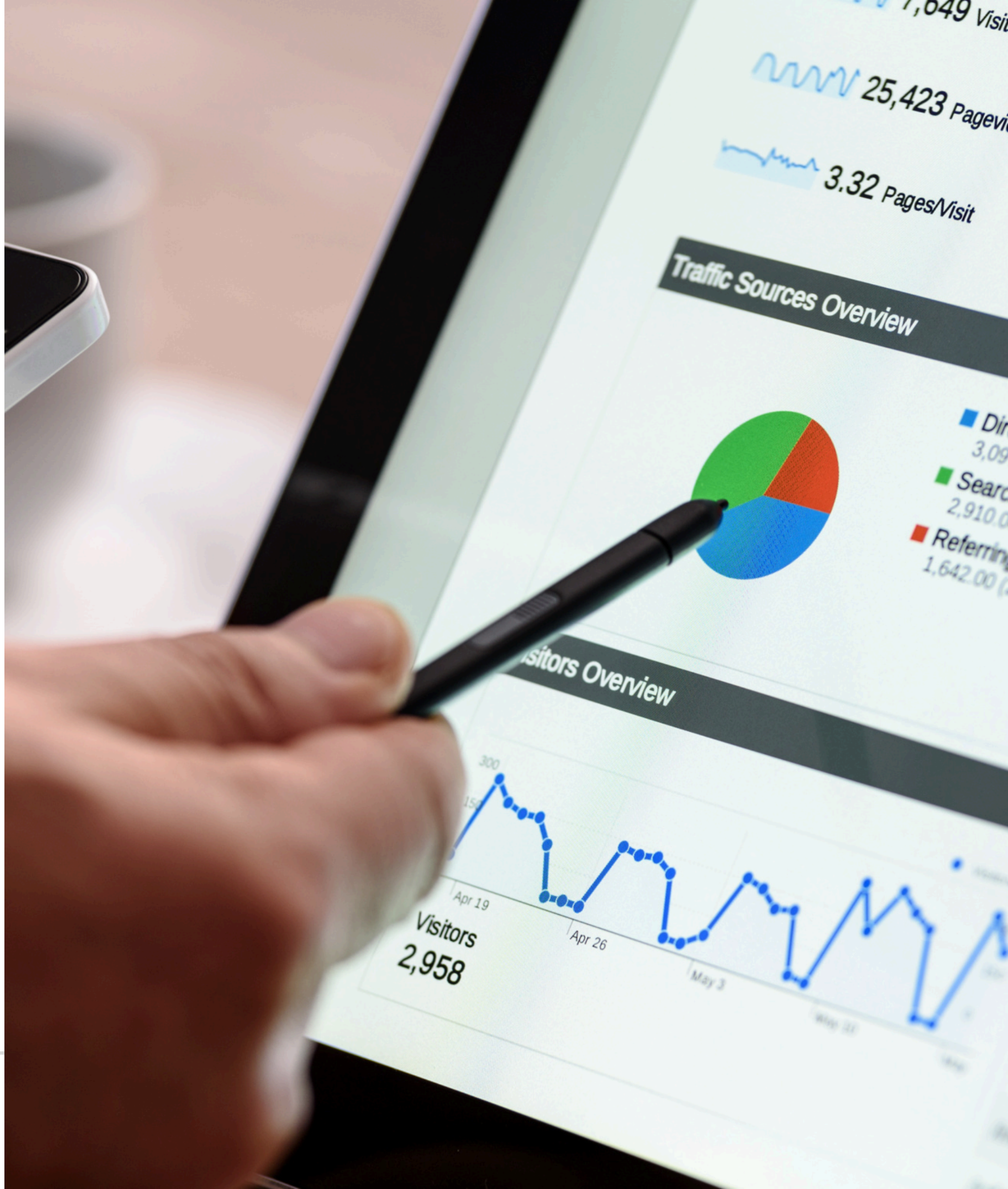
Generate Insights

TECH-STACK USED

This software provides better data security, more Built-in functions, and on- demand scalability; apart from this, it also provides an auto-completing feature that makes it easy for the developer to write the queries.

- **MySQL Workbench 8.0**

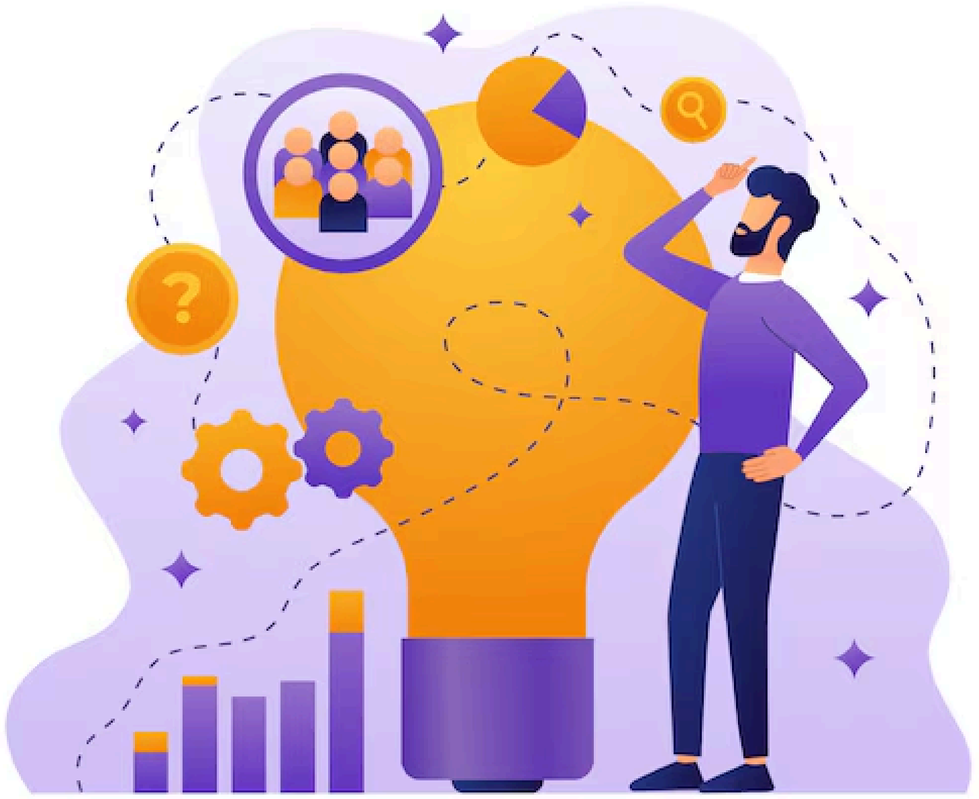
- **Microsoft Excel for importing and exporting the data**



MY INSIGHTS

- While making this project, I came to learn about SQL, how to implement the queries, and various in-built functions that can be used to get the data.

- ·I got a good exposure to My SQL, respectively, on how SQL and the functions could be used in analysing the data from the dataset, how the queries work and are executed, and how I am playing a major role in retrieving the data instantly without searching for it in the dataset.



DATABASE CREATION

Code

```
create database project3;

use project3;
```

Output

creating a table and importing data in the table

Code

```
create table job_data(
ds date,
job_id int not null,
actor_id int not null,
event varchar(50) not null,
language varchar(50) not null,
time_spent int not null,
org char(2)
);
```

Output

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	Aeabic	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

```
insert into job_data (ds, job_id, actor_id, event, language, time_spent, org)
Values('2020-11-30', 21, 1001, 'skip', 'English', 15, 'A'),
('2020-11-30', 22, 1006, 'transfer','Aeabic', 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');
```

Case Study 01.

Task 01. Number of jobs reviewed: Amount of job review over time.

Code

```
select ds as date,
count(job_id) as joint_job_id,
round((sum(time_spent) / 3600), 2) as total_time_per_hour,
round((count(job_id) / (sum(time_spent)/3600)),2) as job_review
from job_data
where
ds between '2020-11-01' and '2020-11-30'
group by ds
order by ds;
```

Output

date	joint_job_id	total_time_per_hour	job_review
2020-11-25	1	0.01	80.00
2020-11-26	1	0.02	64.29
2020-11-27	1	0.03	34.62
2020-11-28	2	0.01	218.18
2020-11-29	1	0.01	180.00
2020-11-30	2	0.01	180.00

Insights

We can say that 0.01 jobs reviewed per hour in nov 2020.

On 28th nov 2020 was highest job reviewed which was 218.18

Task 02. Throughput Analysis

Code

```
select round(count(event) / sum(time_spent), 2) As weekly_avg_throughput
from job_data;

select ds as dates,
round(count(event) / sum(time_spent),2) as daily_avg_throughput
from job_data
group by ds
order by ds;
```

Output

weekly_avg_throughput
0.03

dates	daily_avg_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

explain whether you prefer using the daily metric or the 7-day rolling average for throughput, and why.

- 7-Day Rolling Average: (Use Case: Ideal for long-term planning and performance evaluation.
- Daily Metric: (Use Case: Suitable for monitoring real-time or daily performance and identifying anomalies.

Conclusion: I would prefer the 7-day rolling average for throughput analysis because it reduces noise from daily variations and gives a clearer picture of overall trends. However, for addressing specific anomalies or urgent issues, daily metrics are more actionable.

Task 03. Language Share Analysis:

Code

```
select language, round(100* count(*) / total, 2) as percentage,
jd.total from job_data
cross join (select count(*)  as total
from job_data) as jd
group by language, jd.total;
```

Output

language	percentage	total
English	12.50	8
Aeabic	12.50	8
Persian	37.50	8
Hindi	12.50	8
French	12.50	8
Italian	12.50	8

Insights

We can say that Persian language is the highest language used with the number of percentage share of 37.50 incompared to other languages.

Task 04. Duplicate Rows Detection:

Code

```
select actor_id, count(*) as duplicate
from job_data
group by actor_id
having count(*) > 1;
```

Output

	actor_id	duplicate
▶	1003	2

Insights

Total of 8 rows, we have only 2 duplicate rowsThe

actor_id 1003 is the duplicate.

CASE STUDY 02.

Task 01. Weekly User Engagement:

Code

```
select extract(week from occurred_at) as week_num,
count(distinct user_id) as active_users
from events
where event_type = 'engagement'
group by week_num
order by week_num;
```

Output

week_num	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

Insights

01. The highest engagement in 30th weeks, with 1467 users.
02. The minumum engagement in 35th weeks, with 104 users.

Task 02. User Growth Analysis:

Code

```
select count(*) from users where state = 'active';
select distinct state from users;

with weekly_active_users as(
select extract(year from created_at) as year,
extract(week from created_at) as week_number,
count(distinct user_id) num_of_users
from users

group by year, week_number
)

select year, week_number, num_of_users,
sum(num_of_users) over (order by year, week_number) as cumulative_users
from weekly_active_users
order by year, week_number;
```

Output

year	week_num	num_of_u	cumulative
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

Insights

Highest number of users was in the 33rd weeks of 2014 with 261 new users reaching a total of 9104 users.

Task 03. Weekly Retention Analysis: Code

```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",
    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 occurred_at) AS login_week
        FROM
            events
        GROUP BY
            user_id,
            login_week
    ) m
    JOIN (
        SELECT
            user_id,
            MIN(EXTRACT(WEEK FROM occurred_at)) AS first
        FROM
            events
        GROUP BY
            user_id
    ) n
    ON m.user_id = n.user_id
) sub
GROUP BY
    first
ORDER BY
    first;
```

Output

week_	week	week	week	wee	weel	week_	week	wee	wee	wee	wee	wee	weel	week	week	weel	wee	wee	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

Insights

01. The users joined in the 17th week shows the longest period of users retention in 18th week.
02. The lowest users joined in 35th week is 18 users.

Task 04. Weekly Retention Analysis: Code

```sql
SELECT
    EXTRACT(WEEK FROM occurred_at) AS week_number,
    COUNT(DISTINCT CASE WHEN device = 'dell inspiron notebook' THEN user_id ELSE NULL END) AS dell_inspiron_noteboo
    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 = 'iphone 5s' THEN user_id ELSE NULL END) AS iphone_5s,
    COUNT(DISTINCT CASE WHEN device = 'ipad air' THEN user_id ELSE NULL END) AS ipad_air,
    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 = 'macbook pro' THEN user_id ELSE NULL END) AS macbook_pro,
    COUNT(DISTINCT CASE WHEN device = 'ipad mini' THEN user_id ELSE NULL END) AS ipad_mini,
    COUNT(DISTINCT CASE WHEN device = 'kindle fire' THEN user_id ELSE NULL END) AS kindle_fire,
    COUNT(DISTINCT CASE WHEN device = 'amazon fire phone' THEN user_id ELSE NULL END) AS amazon_fire_phone,
    COUNT(DISTINCT CASE WHEN device = 'nexus 5' THEN user_id ELSE NULL END) AS nexus_5,
    COUNT(DISTINCT CASE WHEN device = 'nexus 7' THEN user_id ELSE NULL END) AS nexus_7,
    COUNT(DISTINCT CASE WHEN device = 'nexus 10' THEN user_id ELSE NULL END) AS nexus_10,
    COUNT(DISTINCT CASE WHEN device = 'samsung_galaxy_s4' THEN user_id ELSE NULL END) AS samsung_galaxy_s4,
    COUNT(DISTINCT CASE WHEN device = 'samsung_galaxy_tablet' THEN user_id ELSE NULL END) AS samsung_galaxy_tablet,
    COUNT(DISTINCT CASE WHEN device = 'samsung_galaxy_note' THEN user_id ELSE NULL END) AS samsung_galaxy_note,
    COUNT(DISTINCT CASE WHEN device = 'lenovo thinkpad' THEN user_id ELSE NULL END) AS lenovo_thinkpad,
    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,
    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 = '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
FROM
    events
WHERE
    event_type = 'engagement'
GROUP BY
    week_number
ORDER BY
    week_number;
```

Output

week	dell	iphc	iph	iph	ipa	winc	mac	mac	ipa	kin	an	nex	nexu	nex	sar	sar	sa	leno	ace	asi	htc	nok	ma	hp_	dell_
17	46	65	21	42	27	10	54	143	19	6	4	40	18	16	0	0	0	86	20	21	16	17	6	14	18
18	77	113	46	73	52	10	121	252	30	27	9	73	30	30	0	0	0	153	33	42	19	33	13	37	58
19	83	115	44	79	55	16	112	266	36	21	12	87	41	25	0	0	0	178	41	27	30	23	18	40	36
20	84	125	55	79	59	21	119	256	32	23	11	103	32	22	0	0	0	173	40	41	29	22	26	30	52
21	80	137	45	74	51	17	110	247	23	30	5	91	29	25	0	0	0	167	47	38	21	25	18	44	41
22	92	125	45	71	58	15	145	251	34	21	5	96	45	27	0	0	0	176	41	52	24	25	25	38	52
23	103	152	53	79	41	14	124	266	33	25	16	88	36	45	0	0	0	176	43	49	20	31	18	54	53
24	99	142	53	79	57	22	152	255	39	25	11	87	49	38	0	0	0	165	40	43	20	35	29	56	59
25	105	137	40	78	57	22	121	275	30	24	13	89	51	29	0	0	0	197	47	38	21	37	21	52	52
26	89	152	50	94	56	21	134	269	43	26	13	87	46	29	0	0	0	192	35	49	23	42	11	46	60
27	89	163	67	83	55	33	142	302	35	25	10	84	40	37	0	0	0	202	49	52	27	31	15	56	53
28	103	151	61	93	54	33	148	295	35	31	6	85	39	26	0	0	0	220	49	50	26	35	28	56	56
29	113	144	60	90	52	28	148	295	34	37	12	77	45	25	0	0	0	209	53	49	31	43	31	58	54
30	127	152	65	##	70	19	159	322	35	25	12	84	62	36	0	0	0	206	60	56	31	34	23	42	54
31	113	135	56	71	55	19	147	321	27	14	14	69	38	24	0	0	0	207	55	56	13	28	24	51	44
32	104	119	34	67	48	10	125	307	30	12	12	67	25	30	0	0	0	179	55	62	18	28	20	51	57
33	110	110	35	65	40	15	133	312	28	14	14	70	30	23	0	0	0	191	46	49	19	27	32	38	37
34	105	101	50	70	39	18	136	292	25	13	11	70	33	25	0	0	0	193	63	47	25	17	30	36	49
35	9	2	6	3	0	3	10	17	2	3	0	4	2	2	0	0	0	16	3	6	2	2	2	1	1

Insights

01. The most users can use macbook pro(322 users on 30th weeks, and lenovo thinkpad(220 users on 28th weeks) also iphone_5 (163 users on the 27th weeks).

Task 05. Email Engagement Analysis:

Code

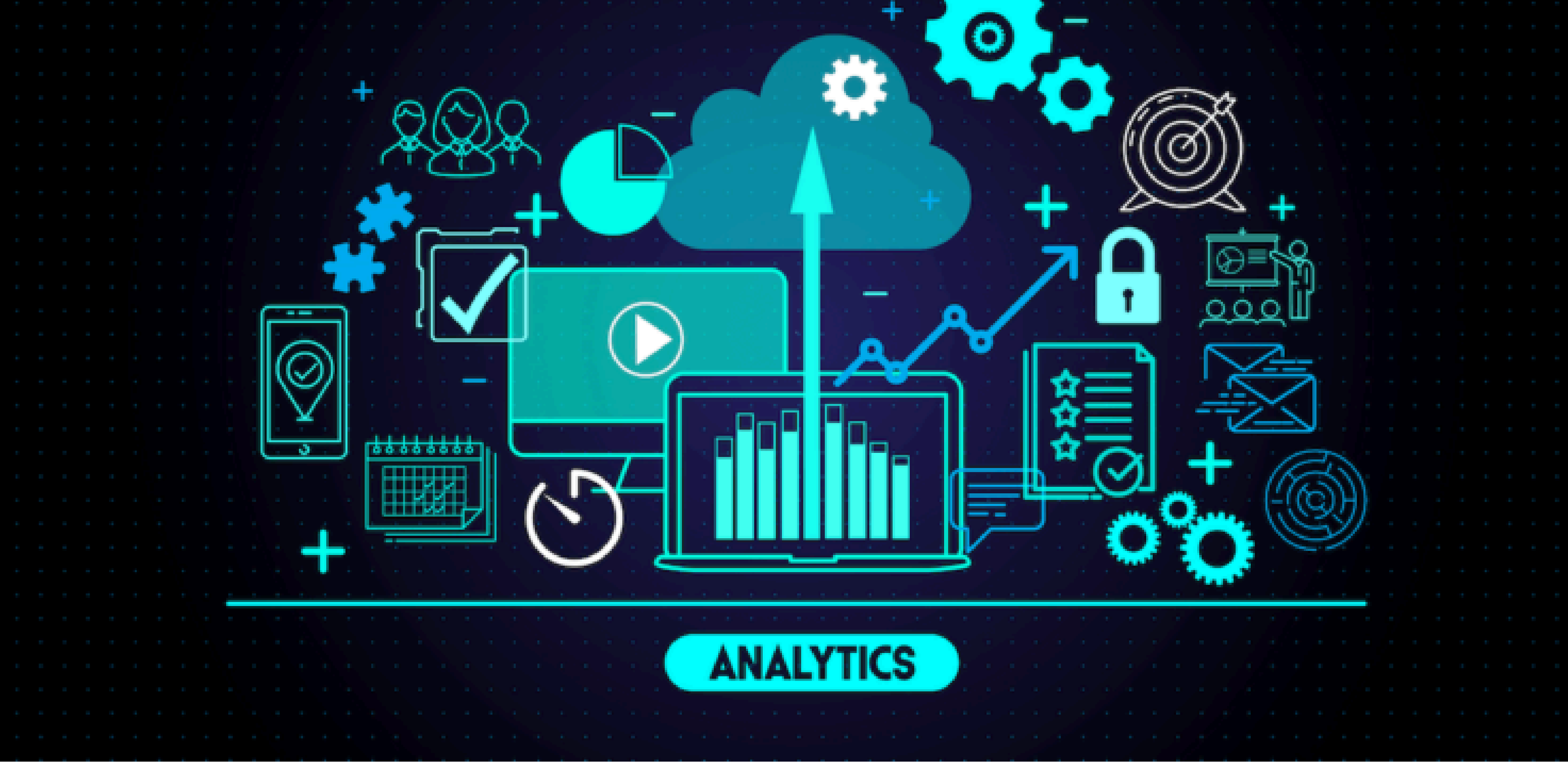
```
SELECT
    100.0 * SUM(CASE WHEN email_action = 'email-open' THEN 1 ELSE 0 END) /
    NULLIF(SUM(CASE WHEN email_action = 'email-sent' THEN 1 ELSE 0 END), 0) AS email_open_rate,

    100.0 * SUM(CASE WHEN email_action = 'email-clicked' THEN 1 ELSE 0 END) /
    NULLIF(SUM(CASE WHEN email_action = 'email-sent' THEN 1 ELSE 0 END), 0) AS email_clicked_rate
FROM (
    SELECT
        *,
        CASE
            WHEN action IN ('sent_weekly_digest', 'sent_reengagement_email') THEN 'email-sent'
            WHEN action = 'email_open' THEN 'email-open'
            WHEN action = 'email_clickthrough' THEN 'email-clicked'
            ELSE NULL
        END AS email_action
     FROM
         project3.email_events
) a;
```

Output

	email_open_rate	email_clicked_rate
▶	33.58339	14.78989

Insights

01. All emails sent around 33% were opened and 14% were only clicked..



THANKYOU