Operation Analytics and Investigating Metric Spike

Description & Tech Stack

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.

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.

MySQL CE 8.0 was used to perform the analysis on the given database.

INSIGHTS

CASE STUDY 1

For the Case Study 1, Job Data table with data about jobs and the actors for the respective job, the organization of the actor, the event description is it decision, skip or transfer, languages and the date of the job.

This table data is used to identify the number of jobs reviewed, throughput, percentage share of each language and if there are any duplicate rows

The Operation Analysis helps company identify what they should be working upon which increases the efficiency of the company

01 Number of Jobs Reviewed

Number of jobs reviewed: Amount of jobs reviewed over time.

Task: Calculate the number of jobs reviewed per hour per day for November 2020?

Highest Number of jobs reviewed is 218 on 28th November.

```sql
-- A) Number of jobs reviewed: Amount of jobs reviewed over time.
select ds as Date,
round((count(distinct job_id)/sum(time_spent))*3600) as `Time spent`
from job_data
where ds between '2020-11-01' and '2020-11-30'
group by ds
order by `Time spent` desc;
```

	Date	Time spent
▶	2020-11-28	218
	2020-11-29	180
	2020-11-30	180
	2020-11-25	80
	2020-11-26	64
	2020-11-27	35

02 Throughput

Throughput: It is the no. of events happening per second.
Your task: Let's say the above metric is called throughput. Calculate 7 day rolling average of throughput?
The rolling average is consistent and helps understand the trend over time without much fluctuations compared to the daily metric.

```sql
-- B)Throughput: It is the no. of events happening per second.
select ds as date,
round(count(event)/sum(time_spent),2) as Daily_Throughput
from job_data
group by ds
order by ds;

#7-day-throughput
select round(count(event)/sum(time_spent),2) as Weekly_throughput
from job_data;
```

	date	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

	Weekly_throughput
▶	0.03

03 Percentage share of each language

Percentage share of each language: Share of each language for different contents.

Your task: Calculate the percentage share of each language in the last 30 days?

Persian has the largest share among all the languages being used.

```sql
-- C) Percentage share of each language: Share of each language for different events.
select language,
round(100*((count(*)/(select count(*) from job_data))) ,2) as `Percentage Share`
from job_data
group by language
order by `Percentage Share` desc;
```

language	Percentage Share
Persian	37.50
English	12.50
Arabic	12.50
Hindi	12.50
French	12.50
Italian	12.50

04 Duplicate rows

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?

No Duplicates Found

```sql
-- D) Duplicate rows: Rows that have the same value present in them.
select job_id, actor_id, count(*) as Count
from job_data
group by job_id, actor_id
having Count>1;
```

	job_id	actor_id	Count

CASE STUDY 2

Investigating Metric Spike lets the team understand reasons behind the reason if there is a dip daily engagement . The data to be analyzed has tables:

- **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.

01 User Engagement

User Engagement: To measure the activeness of a user. Measuring if the user finds quality in a product/service.
Your task: Calculate the weekly user engagement?

The events can be grouped by week they occurred at to collect the users engagement. This shows that the user engagement has increased then decreased.

```
-- A)User Engagement
SELECT week(occurred_at) as Week,
count(DISTINCT user_id)as Weekly_User_engagement
FROM events
GROUP BY week(occurred_at)
ORDER BY week(occurred_at);
```

Week	Weekly_User_engagement
17	740
18	1260
19	1287
20	1351
21	1299
22	1381
23	1446
24	1471
25	1459
26	1509
27	1573
28	1577
29	1607
30	1706
31	1514
32	1454
33	1438
34	1443
35	118

02 User Growth

User Growth: Amount of users growing over time for a product.
Your task: Calculate the user growth for product?

The last row of the table, shows that there are **9381** active users.

```
SET @g := 0;
SELECT a.no_of_users, a.date,
( @g := @g + a.no_of_users ) as user_growth
FROM
( SELECT count(user_id) as no_of_users,
date(created_at) as date
FROM users WHERE state = "active"
GROUP BY date(created_at) ) a;
```

no_of_users	date	user_growth
7	2013-01-01	7
7	2013-01-02	14
6	2013-01-03	20
1	2013-01-04	21
2	2013-01-05	23
3	2013-01-06	26
4	2013-01-07	30
2	2013-01-08	32
6	2013-01-09	38
6	2013-01-10	44
6	2013-01-11	50
3	2013-01-12	53

03 Weekly Retention

Weekly Retention: Users getting retained weekly after signing-up for a product.
Task: Calculate the weekly retention of users-sign up cohort?

The retention chart is shown in the next page

```
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 - 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,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	5	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

04 Weekly Engagement

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?

Mac Book Pro has the highest user engagement and it can be analyzed per week.

```
SELECT week(occurred_at) as Weeks,
device, count(distinct user_id)as User_engagement
FROM events
GROUP BY device,week(occurred_at)
ORDER BY week(occurred_at);
```

Weeks	device	User_engagement
17	acer aspire desktop	12
17	acer aspire notebook	23
17	amazon fire phone	4
17	asus chromebook	23
17	dell inspiron desktop	20
17	dell inspiron notebook	48
17	hp pavilion desktop	17
17	htc one	19
17	ipad air	29
17	ipad mini	19
17	iphone 4s	27
17	iphone 5	69
17	iphone 5s	47
17	kindle fire	6
17	lenovo thinkpad	94
17	mac mini	7
17	macbook air	61
17	macbook pro	154

05 Email Engagement

Email Engagement: Users engaging with the email service.

Task: Calculate the email engagement metrics?

```sql
SELECT week(occurred_at) as Week,
count( DISTINCT ( CASE WHEN action = "sent_weekly_digest" THEN user_id end )) as weekly_digest,
count( distinct ( CASE WHEN action = "sent_reengagement_email" THEN user_id end )) as reengagement_mail,
count( distinct ( CASE WHEN action = "email_open" THEN user_id end )) as opened_email,
count( distinct ( CASE WHEN action = "email_clickthrough" THEN user_id end )) as email_clickthrough
FROM emails
GROUP BY week(occurred_at)
ORDER BY week(occurred_at);
```

Re-Engagement ,
opened_email,
email_clickthrough
are described as per the email events table from the database.

These insights can be used to identify email strategies and sent to improve user interaction with the application.

Week	weekly_digest	reengagement_mail	opened_email	email_clickthrough
17	908	73	310	166
18	2602	157	900	425
19	2665	173	961	476
20	2733	191	989	501
21	2822	164	996	436
22	2911	192	965	478
23	3003	197	1057	529
24	3105	226	1136	549
25	3207	196	1084	524
26	3302	219	1149	550
27	3399	213	1207	613
28	3499	213	1228	594
29	3592	213	1201	583
30	3706	231	1363	625
31	3793	222	1338	444

Thank You!