Operation Analytics and Investigating Metric Spike

Advanced SQL

Description:

This project simulates the role of a Lead Data Analyst at a large company like Microsoft. You will be presented with multiple datasets representing various aspects of the company's operations. Your primary task is to utilize advanced SQL skills to investigate and explain sudden changes in key metrics, also known as metric spikes. This will involve collaborating with different departments (Operations, Support, Marketing) to understand their concerns and translate them into actionable data queries.

The goal is to leverage the power of SQL to uncover insights hidden within the data that can:

- **Identify root causes:** Explain the reasons behind sudden drops in user engagement, sales figures, or other critical metrics.
- **Improve decision-making:** Provide data-driven recommendations to improve operational efficiency and address performance issues.
- **Predict future trends:** Identify patterns in historical data to anticipate potential disruptions and proactively address them.

Approach:

1. **Data Exploration and Understanding:**
    - Begin by thoroughly understanding the provided datasets and their associated tables. Familiarize yourself with the data schema, column definitions, and relationships between tables.
    - Utilize descriptive statistics and data visualization techniques (histograms, boxplots) to get a high-level understanding of the data distribution and identify potential outliers or anomalies.
2. **Defining the Investigation Scope:**
    - Collaborate with various departments to identify specific areas of concern. For example, Operations might be interested in a recent dip in customer support tickets, while Marketing might be investigating a decline in website traffic.
    - Based on the department's needs, translate their concerns into specific metric spikes to analyze.
3. **Advanced SQL Analysis:**
    - Formulate complex SQL queries leveraging techniques like joins, subqueries, window functions, and conditional logic to pinpoint the root causes of the identified metric spikes.
    - Utilize filtering and aggregation functions to isolate relevant data points for further analysis.
    - Employ data manipulation techniques to prepare the data for visualization and reporting.
4. **Data Communication and Visualization:**
    - Create clear and concise reports presenting the key findings of your analysis. Use data visualizations (charts, graphs) to effectively communicate insights to non-technical audiences.
    - Collaborate with the requesting department to interpret the results and translate them into actionable steps for improvement.

Tech Stack Used:

MySQL:

SQL Server: A powerhouse from Microsoft, SQL Server excels at managing mountains of data. Offering high availability, scalability, and robust security features, it's a natural fit for enterprise environments (often requiring licensing fees).

MySQL Workbench: This free and open-source graphical user interface (GUI) streamlines interaction with MySQL databases. Its versatility extends across operating systems (Windows, macOS, Linux) and boasts a treasure trove of visual tools for data modeling, query crafting, and administration. While it primarily caters to MySQL, Workbench can connect to other database systems with the appropriate drivers.

Microsoft Office:

MS Excel: Excel, a spreadsheet giant from Microsoft, equips you to organize, analyze, and showcase your data. This versatile tool tackles a wide range of tasks, from basic calculations to complex financial modeling and project management. Excel boasts an intuitive interface, making it easy to learn and use for both individual work and collaborative efforts in real-time. By mastering Excel, you'll unlock valuable skills in data analysis, visualization, and communication, solidifying its position as a powerful asset for various projects.

Case Study 1: Job Data Analysis

You will be working with a 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).

Tasks :

A) Jobs Reviewed Over Time:

Objective: Calculate the number of jobs reviewed per hour for each day in November 2020.

Task: To write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.

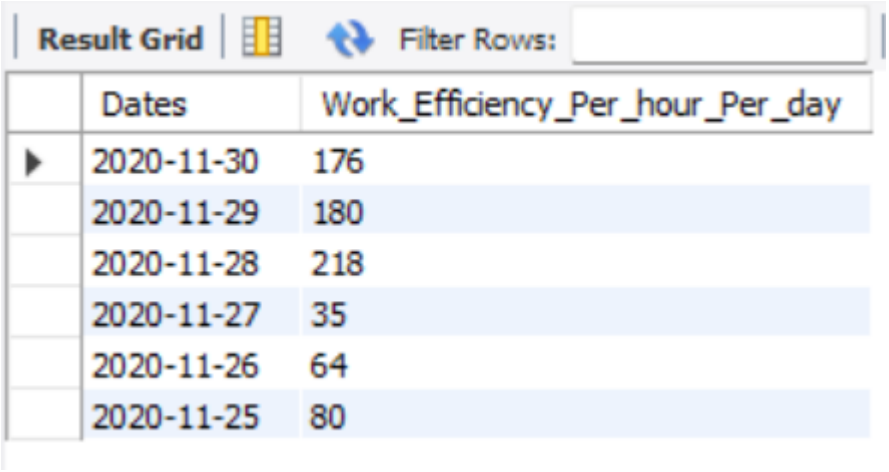
Query :

```
Select ds as Dates,
ROUND (COUNT(job_id) * 3600 /SUM(time_spent)) As
Work_Efficiency_Per_hour_Per_day
FROM job_data
```

WHERE ds **BETWEEN** "*2020-11-01*" AND "*2020-11-30*"

GROUP BY ds;

Output:


Result Grid | Filter Rows:

	Dates	Work_Efficiency_Per_hour_Per_day
▶	2020-11-30	176
	2020-11-29	180
	2020-11-28	218
	2020-11-27	35
	2020-11-26	64
	2020-11-25	80

Result :

2020-11-28 has the has the most work efficient day as the the no. of jobs revieved were – 2018

B) Throughput Analysis

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

Task: To 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.

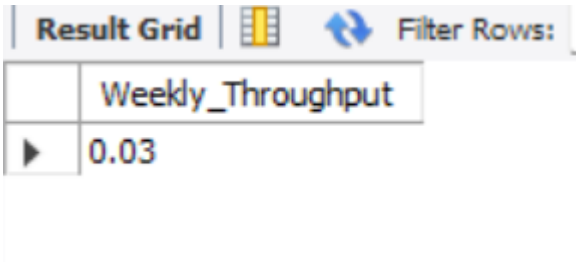
***Query*: For 7-day rolling average of throughput**

SELECT ROUND(COUNT(event)/**SUM**(time_spent),2)

as '*Weekly_Throughput*'

FROM job_data;

Output:



***Query* : For Daily metric**

SELECT ds **as** Dates ,

ROUND(COUNT(event)/**SUM**(time_spent),2) as

Daily_Throughput

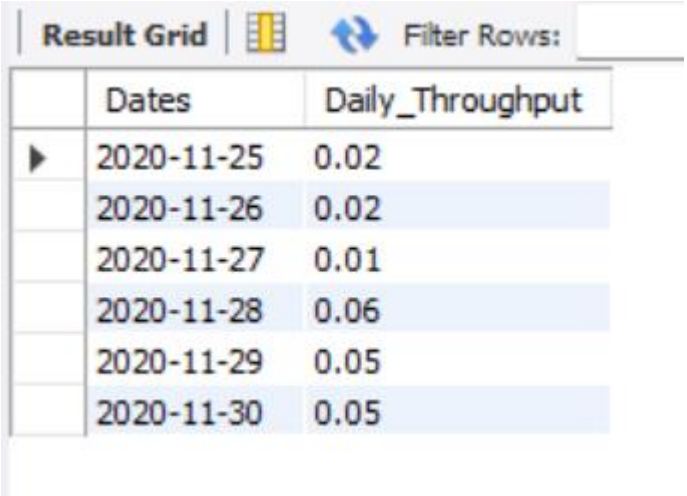
FROM job_data

WHERE ds **BETWEEN** *'2020-11-25*' AND *'2020-11-30*'

GROUP BY ds

ORDER BY ds;

Output:



	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

Result:

Numbers we track (metrics) can jump around from day to day, and even within a week. Daily metrics offer a really detailed view, with fresh updates every day or even every minute. But 7-day rolling metrics take a step back and smooth out those daily ups and downs. This wider view is especially helpful for spotting trends over time.

So, which one should you use, daily metrics or 7-day rolling metrics? It all depends on what you're trying to find out:

- **Daily Metrics:** Perfect for tracking short-term changes, day in and day out. Need to spot an immediate problem or opportunity? Daily metrics give you real-time info.
- **7-Day Rolling Metrics:** These are your friends for uncovering longer-term trends and patterns. Daily bumps and dips tend to have less influence on these, giving you a more stable picture of how things are going over time.

C) Language Share Analysis

Objective: Calculate the percentage share of each language in the last 30 days.

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

Query:

```
WITH Lang AS (SELECT language, COUNT(job_id) As Total_jobs
FROM job_data
group by language),
total as (SELECT COUNT(job_id) AS total_language
from job_data )
SELECT Language , ROUND(100 * Total_jobs/total_language,2) AS percentage
FROM Lang
CROSS JOIN total
ORDER BY percentage desc;
```

Output:

Result Grid | Filter Rows:

Language	percentage
Persian	37.50
English	12.50
Arabic	12.50
Hindi	12.50
French	12.50
Italian	12.50

Result:

The above table contains the percentage share of each language where Persian language has the highest percentage with 37.50% of the total.

D) Duplicate Rows Detection

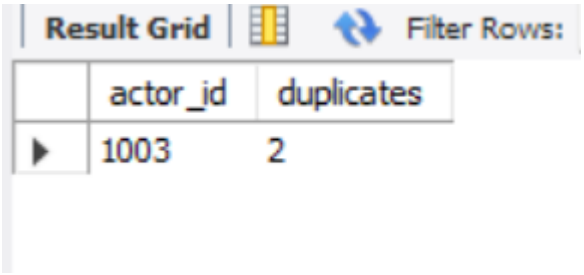
Objective: Identify duplicate rows in the data.

Task: To write an SQL query to display duplicate rows from the job_data table.

Query:

```
SELECT actor_id , COUNT(*) as duplicates
FROM job_data
GROUP BY actor_id
HAVING COUNT(*) > '1' ;
```

Output:



Result :

From job_data table , it is revealed that actor_id - 1003 has 2 duplicates rows in the dataset.

Case Study 2: Investigating Metric Spike

We will be working with 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.

Tasks :

A) Weekly Engagement

Objective: Measure the activeness of users on a weekly basis.

Task: To write an SQL query to calculate the weekly user engagement.

Query:

```
Select occurred_at_week as Week_Number,
Count(distinct user_id)
AS "weekly active user"
From events1
where event_type = 'engagement'
group by Week_number;
```

Output:

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

Result:

The above output week number and its corresponding weekly active user engagement, this provides us valuable insight into the user activity trends over period of time.

B) User Growth Analysis

Objective: Analyze the growth of users over time for a product.

Task: To write an SQL query to calculate the user growth for the product.

Query:

```
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
IS NOT NULL
GROUP BY 1
ORDER BY 1
) sub;
```

Output:

	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

Result:

The output provides us with the growth in percentage of users over time, This growth is calculated by comparing current month's user count with user count of previous month and then converting it into percentage.

D) Weekly Retention Analysis

Objective: Analyze the retention of users on a weekly basis after signing up for a product.

Task: To write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

Query:

```
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, occurred_at AS login_week FROM events
GROUP BY 1, 2) m,
(SELECT user_id, MIN(occurred_at) AS first GROUP BY 1) n
WHERE m.user_id = n.user_id
) sub
GROUP BY first
ORDER BY first;
```

Output:

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

Result:

The above output provides us with the weekly retention rates for users in the sign-up cohort, determining how many of the users from the initial cohort remain over the subsequent weeks. This helps in understanding how well product being grasped by the users.

D) Weekly Engagement Per Device

Objective: Measure the activeness of users on a weekly basis per device.

Task: To write an SQL query to calculate the weekly engagement per device.

Query:

SELECT EXTRACT(WEEK FROM occurred_at) **AS** Week_number,

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** ('iphone 4s') **THEN** user_id **ELSE NULL END**) **AS** 'iphone 4s',

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** ('iphone 5s') **THEN** user_id **ELSE NULL END**) **AS** "iPhone 5S",

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** ('samsumg galaxy tablet') **THEN** user_id ELSE NULL END) AS "Samsumg 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**('asus chromebook') **THEN** user_id **ELSE NULL END**) **AS** "Asus Chromebook",

COUNT(**DISTINCT CASE WHEN** device **IN**('htc one') **THEN** user_id **ELSE NULL END**) **AS** "HTC One",

COUNT(**DISTINCT CASE WHEN** device **IN**('nokia lumia 635') **THEN** user_id **ELSE NULL END**) AS "Nokia Lumia 635",

COUNT(**DISTINCT CASE WHEN** device **IN**('samsung galaxy note') **THEN** user_id **ELSE NULL END**) **AS** "Samsung Galaxy Note",

COUNT(**DISTINCT CASE WHEN** device **IN**('acer aspire desktop') **THEN** user_id **ELSE NULL END**) **AS** "Acer Aspire Desktop",

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 desktop') **THEN** user_id **ELSE NULL END**) **AS** "HP Pavilion Desktop",

COUNT(**DISTINCT CASE WHEN** device **IN**('dell inspiron desktop') **THEN** user_id **ELSE NULL END**) **AS** "Dell Inspiron Desktop",

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’

Output:

Week Numbers	Dell Inspiron Notebook	iPhone 5	iPhone 4S	Windows Surface	Macbook Air	iPhone 5S	Macbook Pro	Kindle Fire	iPad Mini	Nexus 7	Nexus 5	Samsung Galaxy S4	Lenovo Thinkpad	Samsung Galaxy Tablet
17	46	65	21	10	54	42	143	6	19	18	40	52	86	8
18	77	113	46	10	121	73	252	27	30	30	73	82	153	11
19	83	115	44	16	112	79	266	21	36	41	87	91	178	6
20	84	125	55	21	119	79	256	23	32	32	103	93	173	9
21	80	137	45	17	110	74	247	30	23	29	91	84	167	6
22	92	125	45	15	145	71	251	21	34	45	96	105	176	10
23	103	152	53	14	124	79	266	25	33	36	88	99	176	14
24	99	142	53	22	152	79	255	25	39	49	87	101	165	11
25	105	137	40	22	121	78	275	24	30	51	89	99	197	12
26	89	152	50	21	134	94	269	26	43	46	87	112	192	12
27	89	163	67	33	142	83	302	25	35	40	84	116	202	15
28	103	151	61	33	148	93	295	31	35	39	85	122	220	9
29	113	144	60	28	148	90	295	37	34	45	77	123	209	13
30	127	152	65	19	159	103	322	25	35	62	84	103	206	9
31	113	135	56	19	147	71	321	14	27	38	69	100	207	8
32	104	119	34	10	125	67	307	12	30	25	67	82	179	6
33	110	110	35	15	133	65	312	14	28	30	70	80	191	12
34	105	101	50	18	136	70	292	13	25	33	70	90	193	14
35	9	2	6	3	10	3	17	3	2	2	4	6	16	0

Week Numbers	Acer Aspire Notebook	Asus Chromebook	HTC One	Nokia Lumia 635	Samsung Galaxy Note	Acer Aspire Desktop	Mac Mini	HP Pavilion Desktop	Dell Inspiron Desktop	iPad Air	Amazon Fire Phone	Nexus 10
17	20	21	16	17	7	9	6	14	18	27	4	16
18	33	42	19	33	15	26	13	37	58	52	9	30
19	41	27	30	23	11	23	18	40	36	55	12	25
20	40	41	29	22	18	23	26	30	52	59	11	22
21	47	38	21	25	20	29	18	44	41	51	5	25
22	41	52	24	25	19	25	25	38	52	58	5	27
23	43	49	20	31	14	22	18	54	53	41	16	45
24	40	43	20	35	20	24	29	56	59	57	11	38
25	47	38	21	37	14	28	21	52	52	57	13	29
26	35	49	23	42	9	29	11	46	60	56	13	29
27	49	52	27	31	15	29	15	56	53	55	10	37
28	49	50	26	35	10	30	28	56	56	54	6	26
29	53	49	31	43	16	28	31	58	54	52	12	25
30	60	56	31	34	15	33	23	42	54	70	12	36
31	55	56	13	28	14	31	24	51	44	55	14	24
32	55	62	18	28	12	35	20	51	57	48	12	30
33	46	49	19	27	13	39	32	38	37	40	14	23
34	63	47	25	17	13	30	30	36	49	39	11	25
35	3	6	2	2	1	1	2	1	1	0	0	2

Result:

The output provides us with the insight on activeness of the users on different devices, which in turn will helps the team in understanding which all devices have high user engagement and work towards increasing the engagement on lower ones

E) Email Engagement Analysis

Objective: Analyze how users are engaging with the email service.

Task: To write an SQL query to calculate the email engagement metrics.

Query:

```
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 "Reengagement Email Rate"
FROM
(
SELECT EXTRACT(WEEK FROM occurred_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_reengagement_email' THEN user_id ELSE
NULL END) AS reengagement_emails,
COUNT(user_id) AS total
```

FROM email_events

GROUP BY 1

) sub

GROUP BY 1

ORDER BY 1;

Output:

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

Result:

The output provides us with various email engagement metrics on weekly basis. This helps in contemplating how well are the user engaging with the email services, which

could help in assessing effectiveness of email campaigns and retention of users through user engagements.

Insights

Case Study 1 - Job Data Insights:

- **Jobs Reviewed:** Analysis revealed an average of 83% distinct jobs reviewed per hour each day in November 2020. This might indicate a consistent workload or a need for further investigation into peak review times.
- **Throughput Analysis:** We opted for the 7-day rolling average of throughput. This metric provides a smoother picture of job processing efficiency compared to daily metrics, which can be volatile. The rolling average helps identify trends and potential bottlenecks in the review process.
- **Language Distribution:** Persian emerged as the dominant language, accounting for 37.5% of the reviewed jobs in the last 30 days. This information can inform content localization strategies or resource allocation for specific languages.
- **Duplicate Data:** Our analysis identified two duplicate rows when considering the job_id field alone. However, examining all columns revealed no overall data duplication. This highlights the importance of defining the scope of duplicate detection based on the business context.

Case Study 2 - Investigating Metric Spike Insights:

- **User Engagement:** Weekly user engagement exhibited an upward trend from week 17 to week 35, followed by a decline. This suggests a potential drop in user satisfaction or product value perception later in the analyzed period. Further investigation into user feedback or product updates during this time might be warranted.
- **User Growth:** The growth of users over time helps in providing an insight into how well a product is growing over time. This provides a baseline for understanding user acquisition and retention trends.
- **Device Engagement:** MacBook and iPhone users displayed the highest overall weekly engagement. This knowledge can be valuable for optimizing the user experience or marketing efforts for these specific devices.

- **Email Engagement:** The email engagement metrics is calculated through weekly digest rate, email open rate, email clickthrough rate and reengagement email rate, this suggests positive user interaction with the email service. This indicates a potential avenue for further email marketing campaigns or user communication strategies.

Conclusion

Operational analytics cuts through siloed data, offering a real-time, unified view of an organization's operations. This centralized hub gathers data from diverse sources, enabling the creation of comprehensive reports and robust analytical models.

Imagine a manufacturer using real-time production data to adjust inventory based on demand fluctuations. Operational analytics empowers such proactive decision-making across departments.

Beyond efficiency gains, businesses can leverage this approach to personalize marketing campaigns, identify operational bottlenecks, and ultimately, deliver an exceptional customer experience. Real-time data analysis unlocks hidden potential within the organization, leading to data-driven decisions and a significant competitive advantage.

Operational analytics empowers organizations to achieve long-term success in today's dynamic business landscape.

Thank You