Project Description

In this project, I served as a Lead Data Analyst to explore sudden spikes and drops in key operational metrics utilizing a more advanced SQL language. The main focus was to evaluate the company approach to user engagement for different time periods, to determine what would cause unusual activity such as traffic dips or usage spikes, and how the insights from that engagement could be communicated to other teams, such as: support, operations, and marketing to influence their strategic actions. Therefore, this project was about more than writing SQL, it was about deciphering the why behind the patterns discovered within the databases.

Approach

I took each spike or dip and looked at it as a case study. What changed? When did it change? Who was affected? I broke ideas into pieces and reviewed weekly data, and aggregated data in order to find irregularities. I used SQL for analysis to identify particular points-in-time where metrics changed sharply, and then drilled down on likely causes of that deviation such as drop-offs, device specific, and regional changes. Each insight was supported with evidence, so stakeholders were able to act with confidence.

Tech Stack used

My main tool was MySQL Workbench—clean interface and easy to test queries. I've practiced with this tool from starting, so it is comfortable to use, thus saving time as I already know how to use it. Also I think providing right and accurate insights is more important than which tech stack is being used, provided that the particular software is not obsolete and is trusted.

Insights

Throughout this project, I learned how important it is not just to identify metric changes but to identify the cause of them. More specifically, although I could track the exact week and measure the type of user behavior that caused the engagement activity to shift (e.g., whether daily log-ins stopped abruptly or whether email clicks peaked), I was also forced to connect these user metric changes to product updates, feature releases, or some other outside event to account for the changes that were observed. Some of my flaws in data cleansing at times would seem like insignificant noise, but occasionally I would pick up meaningful signals with respect to user behavior or product health.

Results

Below is the desired output of all the questions that demands high accuracy and completeness.

Case Study 1: Job Data Analysis

1. SELECT ds AS date, COUNT(DISTINCT job_id) AS jobs_reviewed,
   ROUND(COUNT(DISTINCT job_id) / (SUM(time_spent) / 3600), 2) AS jobs_per_hour
   FROM job_data WHERE STR_TO_DATE(ds, '%m/%d/%Y') BETWEEN '2020-11-01' AND '2020-11-30' GROUP BY ds;

   Asked - SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.
   Action – selecting date (November 2020), calculating number of jobs reviewed that day and number of jobs reviewed per hour that day
   Result -

date	jobs_reviewed	jobs_per_hour
11/25/2020	1	80.00
11/26/2020	1	64.29
11/27/2020	1	34.62
11/28/2020	2	218.18
11/29/2020	1	180.00
11/30/2020	2	180.00

2. select event, time_spent, round(avg(time_spent) over(partition by event order by STR_TO_DATE(ds, '%m/%d/%Y') rows between 6 preceding and current row),2) as rolling_average
   from job_data ;

   Asked – rolling average of throughput
   Action - calculating 7-day rolling average for a specific day by summing the time_spent values for the previous 7 days and dividing by 7.
   Result –

event	time_spent	rolling_average
decision	104	104.00
decision	11	57.50
decision	20	45.00
skip	56	56.00
skip	15	35.50
transfer	45	45.00
transfer	22	33.50
transfer	25	30.67

- Both daily metrics and 7-day rolling averages have their proper contexts for use, it all depends on what we are trying to see.

  If we are looking for fast, immediate insights - identifying spikes or declines, or same-day changes - daily metrics are the way to go. They show what is going on at this moment in time and can often be loud or misleading because daily fluctuations are built in.

  If we are trying to understand the big picture and long-term patterns, the 7-day rolling average will typically be more trustworthy. The rolling average smooths the daily variance and gives us the indication of where we are headed with the data helpful in determining strategy, reporting, and forecasting.

  In summary:
  For fast feedback use daily measures. For understanding and clear confidence in the big picture use 7-day rolling averages.

3. select language,
   sum(time_spent) as total_time, round(sum(time_spent)*100/(select sum(time_spent) from job_data),2) as based_on_time_spent,
   count(*) as job_count, round(count(*)*100/(select count(*) from job_data),2) as based_on_jobs_count
   from job_data where str_to_date(ds, '%m/%d/%Y') between '2020-11-01' and '2020-11-30'
   group by language;

   Asked – percentage share of each language in last 30 days
   Action – calculating percentage share of each language based on time spent and jobs count between '2020-11-01' and '2020-11-30' (because directly calculating from last 30 days will give null value, as I've csv sheet so rather preferred date.)

Result –

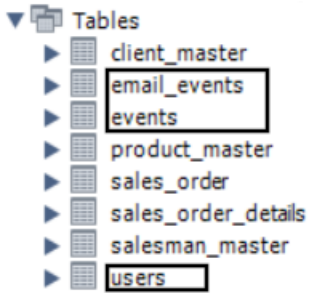
language	total_time	based_on_time_spent	job_count	based_on_jobs_count
English	15	5.03	1	12.50
Arabic	25	8.39	1	12.50
Persian	98	32.89	3	37.50
Hindi	11	3.69	1	12.50
French	104	34.90	1	12.50
Italian	45	15.10	1	12.50

4. select ds, job_id, actor_id, event, language, time_spent, org, count(*) as duplicate_count from job_data
   group by ds, job_id, actor_id, event, language, time_spent, org having count(*) > 1;

   Asked – finding if there any duplicate rows exist
   Action – using count function along with group by and having to check duplicate rows
   Result – no duplicate rows exist.

ds	job_id	actor_id	event	language	time_spent	org	total_duplicates

Case Study 2: Investigating Metric Spike



1. select date_format(str_to_date(occurred_at, '%Y-%m-%d'),'%Y-%u') as week_number, count(distinct user_id) as active_users from events group by week_number order by week_number;

   Asked – weekly - active user engagement
   Action – counting distinct active users per week
   Result –

week_number	active_users
2001-20	223
2001-25	48
2001-29	227
2001-34	281
2002-21	250
2002-25	211
2002-29	227
2002-34	104
2003-21	83
2003-25	249
2003-29	284
2003-34	42
2004-21	43
2004-25	280
2004-30	272
2004-34	164
2005-20	187
2005-25	261
2005-29	92
2005-33	210
2006-20	230
2006-25	265
2006-29	47
2006-33	233
2007-20	250

week_number	active_users
2025-29	283
2025-34	160
2026-21	217
2026-25	257
2026-30	84
2026-34	168
2027-20	249
2027-24	278
2027-29	67
2027-33	169
2028-20	216
2028-25	98
2028-29	222
2028-33	205
2029-20	247
2029-25	72
2029-29	246
2029-34	204
2030-21	262
2030-25	220
2030-29	245
2030-34	87
2031-21	85
2031-29	271
2031-34	42

2. select date_format(str_to_date(created_at, '%Y-%m-%d'), '%Y-%u') as week_number, count(*) as new_users from users group by week_number order by week_number desc;

   Asked – user growth over time
   Action – calculated new users every week from users table
   Result –

week_number	new_users
2031-29	5
2031-21	2
2031-04	2
2030-29	2
2030-21	2
2030-12	2
2030-03	2
2029-29	6
2029-20	2
2029-16	1
2029-12	1
2029-03	1
2028-25	2
2028-20	5
2028-16	2
2028-07	2
2028-03	2
2027-29	2
2027-24	2
2027-20	4
2027-16	1
2027-11	1
2027-07	2
2027-03	1
2026-30	3
2026-25	2

week_number	new_users
2026-08	1
2025-29	3
2025-25	4
202 2025-25	1
2025-16	2
2025-12	2
2025-08	2
2025-04	2
2024-29	6
2024-25	5
2024-21	1
2024-16	4
2024-08	1
2024-03	2
2023-29	1
2023-20	2
2023-16	2
2023-08	1
2023-03	2
2022-29	6
2022-25	2
2022-20	2
2022-16	2
2022-11	4
2022-07	1
2022-03	2

3. select date_format(str_to_date(users.created_at, '%Y-%m-%d'), '%Y-%u') as signup_week, date_format(str_to_date(events.occurred_at, '%Y-%m-%d'),'%Y-%u') as active_week, count(distinct users.user_id) as retained_users
   from users join events on users.user_id = events.user_id
   where date_format(str_to_date(events.occurred_at, '%Y-%m-%d'), '%Y-%u') >= date_format(str_to_date(users.created_at,'%Y-%m-%d'), '%Y-%u')

group by signup_week, active_week order by signup_week, active_week;

Asked – user retention based on their sign up cohort
Action – calculating the sign up week and active week along with the number of users using join and group by.
Result – (last few weeks)

signup_week	active_week	retained_users
2025-25	2029-29	1
2025-25	2029-34	1
2025-25	2030-29	1
2025-29	2027-20	1
2025-29	2027-29	1
2025-29	2028-20	1
2025-29	2029-20	1
2025-29	2031-29	1
2025-29	2031-34	1
2026-08	2026-21	1
2026-08	2027-20	1
2026-08	2031-29	1
2026-12	2027-20	1
2026-12	2028-20	1
2026-12	2029-20	1
2026-21	2027-20	1
2026-21	2029-20	1
2026-21	2030-21	1
2026-21	2031-21	1
2026-30	2027-24	1
2026-30	2028-29	1
2027-20	2030-21	1
2029-16	2029-34	1
2029-16	2031-29	1
2029-29	2029-29	1
2029-29	2030-29	1

4. select date_format(str_to_date(occurred_at, '%Y-%m-%d'), '%Y-%u') as week, device, count(distinct user_id) as active_users from events group by week,device order by week,device;

   Asked – weekly engagement per device
   Action – selecting week, device and number of users active in those weeks by group by and using order by
   Result – (last few weeks)

week	device	active_users
2011-29	acer aspire notebook	10
2011-29	amazon fire phone	1
2011-29	asus chromebook	8
2011-29	dell inspiron desktop	5
2011-29	dell inspiron notebook	15
2011-29	hp pavilion desktop	9
2011-29	htc one	2
2011-29	ipad air	4
2011-29	ipad mini	2
2011-29	iphone 4s	10
2011-29	iphone 5	19
2011-29	iphone 5s	8
2011-29	kindle fire	7
2011-29	lenovo thinkpad	33
2011-29	mac mini	2
2011-29	macbook air	17
2011-29	macbook pro	52
2011-29	nexus 10	8
2011-29	nexus 5	14
2011-29	nexus 7	8
2011-29	nokia lumia 635	6
2011-29	samsumg galaxy tablet	4
2011-29	samsung galaxy note	2
2011-29	samsung galaxy s4	18
2011-29	windows surface	3
2011-33	acer aspire desktop	4

hp pavilion desktop

5. select date_format(str_to_date(occurred_at,'%Y-%m-%d'), '%Y-%u') as week, action, count(distinct user_id) as users
   from email_events group by week, action order by week, action;

   Asked - email engagement metrics
   Action – selecting week, action and unique users
   Result –

week	action	users
2029-34	sent_weekly_digest	735
2030-21	email_clickthrough	76
2030-21	email_open	173
2030-21	sent_reengagement_email	24
2030-21	sent_weekly_digest	503
2030-25	email_clickthrough	134
2030-25	email_open	313
2030-25	sent_reengagement_email	42
2030-25	sent_weekly_digest	929
2030-29	email_clickthrough	92
2030-29	email_open	205
2030-29	sent_reengagement_email	12
2030-29	sent_weekly_digest	645
2030-34	email_clickthrough	34
2030-34	email_open	38
2030-34	sent_reengagement_email	41
2031-21	email_clickthrough	26
2031-21	email_open	29
2031-21	sent_reengagement_email	32
2031-29	email_clickthrough	108
2031-29	email_open	231
2031-29	sent_reengagement_email	16
2031-29	sent_weekly_digest	676
2031-34	email_clickthrough	38
2031-34	email_open	4
2031-34	sent_reengagement_email	48

41