Operation and Metric Analytics

Project Description

- This project is about analysing and evaluating two data sets, one for job data and the other for user engagement and email engagement. This project finds to important insights and metrics through data analysis to evaluate the performance and growth of different aspects of data sets.

- In other hand, I have uncovered valuable insights using SQL queries , such as user engagement, retention rates and workflow optimization. These data driven findings will guide better decision making and enhance overall performance.

Approach and Tech Stack used

- Get the information from given description about data and understand the problem.
- Go through the row data and understand the variable and attributes as given.
- Use MYSQL Workbench to import and create new database. Do some modification in row data and start writing queries as questions asked in order to achieve the results.
- Execute the queries. If there are any errors in the code, modify the code and fix the code without any errors.
- Once done with all queries and cross checked the queries and save the file.

- In this project I have used MySQL Workbench 8.0, a powerful SQL development and database design tool. This tool facilitated data exploration, query development, and result visualization. The SQL language was employed to manipulate and analyze datasets efficiently.

Insights and Results

Case Study 1: Job Data Analysis

- Table : job_data
- Columns : job_id, actor_id, event, language, time_spent, org, ds
- Tasks : Jobs reviewed over time, Throughput analysis, language Share analysis, Duplicate rows detection

Task 1 : Jobs reviewed over time

- Calculate the number of jobs reviewed per hour for each day in November 2020.
- To find the number of jobs reviewed per hour per day of November 2020:
  - We will use the data from job_id columns of the job_data table.
  - Then we will divide the total count of job_id (distinct and non-distinct) by (30 days * 24 hours) for finding the number of jobs reviewed per day

- SQL Query : select count(distinct job_id)/(30*24) as number_of_jobs_reviewed_per_day_distinct from job_data;

Result Grid | Filter Rows:

	number_of_jobs_reviewed_per_day_distinct
▶	0.0083

Task 2 : Throughput Analysis

- Calculate the 7-day rolling average of throughput (number of events per second)
- For calculating the throughput we will be using the 7-day rolling because 7-day rolling gives us the average for all the days right from day 1 to day 7 whereas daily metric gives us average for only that particular day itself. For calculating the 7-day rolling daily metric average of throughput:-
  - We will be first taking the count of job_id(distinct and non-distinct) and ordering them w.r.t ds (date of interview)
  - Then by using the ROW function we will be considering the rows between 6 preceding rows and the current row
  - Then we will be taking the average of the jobs_reviewed

- SQL Query: SELECT ds AS date, job_revewied, AVG(job_revewied) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS seven_day_rolling_avg from(select ds, count(job_id) as job_revewied
  from job_data
  group by ds
  order by ds)a;

date	job_revewied	seven_day_rolling_avg
2020-11-25	1	1.0000
2020-11-26	1	1.0000
2020-11-27	1	1.0000
2020-11-28	2	1.2500
2020-11-29	1	1.2000
2020-11-30	2	1.3333

Task 3 : Language Share Analysis

- Calculate the percentage share of each language in the last 30 days.
- SQL Query: SELECT job_id, language, COUNT(language) AS count_language, (COUNT(language) * 100.0 / (SELECT COUNT(*) FROM job_data)) AS percentage_share_of_language FROM job_data GROUP BY language, job_id;

job_id	language	count_language	percentage_share_of_language
21	English	1	12.50000
22	Arabic	1	12.50000
23	Persian	3	37.50000
25	Hindi	1	12.50000
11	French	1	12.50000
20	Italian	1	12.50000

Task 4 : Duplicate Rows Detection

- Identify duplicate rows in the data.
- Let's say you see some duplicate rows in the data. How will you display duplicates from the table? To view the duplicate rows having the same value we will:
  - First decide in which do we need to find the duplicate row values
  - After deciding the column(parameter) we will use the ROW_NUMBER function to find the row numbers having the same value
  - Then we will portioning the ROW_NUMBER function over the column (parameter) that we decided i.e. job_id
  - Then using the WHERE function we will find the row_num having value greater than 1 i.e. row_num > 1 based on the occurrence of the job_id in the table
- SQL Query: SELECT *FROM (SELECT *, ROW_NUMBER() OVER (PARTITION BY job_id ORDER BY job_id) AS rownumber FROM job_data) AS subquery where rownumber > 1 ;

ds	job_id	actor_id	event	language	time_spent	org	rownumber
2020-11-28	23	1005	transfer	Persian	22	D	2
2020-11-26	23	1004	skip	Persian	56	A	3

Case Stud

- Table : users, events, email_events
- Tasks :Weekly User Engagement, User Growth Analysis, Weekly Retention Analysis, Weekly Engagement Per Device, Email Engagement Analysis

Task 1 : Wee

- Measure the activeness of users on a weekly basis.
- Measuring if the user finds quality in a product/service.
- SQL Query : SELECT extract(week from occured_at) as week_number, count(distinct user_id) as number_of_users FROM events group by week_number;

Result Grid Filter Rows:

week_number	number_of_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

Result 5

Task 2 : User Growth Analysis:

- Analyze the growth of users over time for a product.
- SQL Query: select year_num, week_num, num_active_users, SUM(num_active_users) OVER (ORDER BY year_num, week_num ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS cum_active_usersfrom (select extract(year from a.created_at) as year_num, extract(week from a.created_at) as week_num, count(distinct user_id) as num_active_usersfrom users_t2 WHERE state LIKE '%active%' group by year_num,week_num order by year_num,week_num) a;

year_num	week_num	num_active_users	cum_active_users
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

year_num	week_num	num_active_users	cum_active_users
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

- SQL Query: select count(*) from users_t2 where state = 'active';

total_active_users
9381

Task 3 : Weekly Retention Analysis:

- Analyze the retention of users on a weekly basis after signing up for a product.
- SQL Query: SELECT DISTINCT user_id, COUNT(user_id), SUM(CASE WHEN retention_week = 1 THEN 1 ELSE 0 END) AS per_week_retention FROM (SELECT a.user_id, a.signup_week, b.engagement_week, b.engagement_week a.signup_week AS retention_week FROM ((SELECT DISTINCT user_id, EXTRACT(WEEK FROM occured_at) AS signup_week FROM events WHERE event_type = 'signup_flow' AND event_name = 'complete_signup') a LEFT JOIN (SELECT DISTINCT user_id, EXTRACT(WEEK FROM occured_at) AS engagement_week FROM events WHERE event_type = 'engagement') b ON a.user_id = b.user_id)) d GROUP BY user_id ORDER BY user_id;

user_id	COUNT(user_id)	per_week_retention
11768	1	0
11770	1	0
11775	2	1
11778	3	0
11779	5	1
11780	2	1
11785	1	0
11787	3	1
11791	2	1
11793	6	1
11795	2	1
11798	6	1

Task 4 : Email Engagement Analysis:

- Analyze how users are engaging with the email service.
- SQL Query: SELECT * from email_events;
  select 100 * sum
  (CASE WHEN email_cat= 'email_opened' then 1 else 0 end)/sum(case when email_cat = 'email_sent' then 1 else 0 end) as email_opening_rate, 100 * sum(case when email_cat = 'email_clicked' then 1 else 0 end)/sum(case when email_cat = 'email_sent' then 1 else 0 end)
  as email_clicking_rate
  FROM (select* , CASE when action in ('sent_weekly_digest','sent_reengagement_email‘) then 'email_sent'when action in ('email_open') then 'email_opened'WHEN action in ('email_clickthrough') then 'email_clicked' end as email_cat from email_events) a;

Result Grid | Filter Rows:

	number_of_jobs_reviewed_per_day
▶	0.0111

Task 5 : Weekly Engagement Per Device:

- Measure the activeness of users on a weekly basis per device.
- SQL Query: SELECT EXTRACT(week from occured_at) as week_num,
  extract(year from occured_at) as year_num,device,
  count(distinct user_id) as no_of_users from events
  where event_type = 'engagement‘
  group by 1,2,3order by 1,2,3;

week_num	year_num	device	no_of_u
17	2014	acer aspire desktop	9
17	2014	acer aspire notebook	20
17	2014	amazon fire phone	4
17	2014	asus chromebook	21
17	2014	dell inspiron desktop	18
17	2014	dell inspiron notebook	46
17	2014	hp pavilion desktop	14
17	2014	htc one	16
17	2014	ipad air	27
17	2014	ipad mini	19
17	2014	iphone 4s	21
17	2014	iphone 5	65
17	2014	iphone 5s	42
17	2014	kindle fire	6
17	2014	lenovo thinkpad	86
17	2014	mac mini	6
17	2014	macbook air	54

THANK YOU.