- **Project Description:**
- The project's objective is to leverage operational analytics for end-to-end analysis of a company's operation. This analysis helps identify areas for improvement within the company. One of the key aspects of Operational Analytics is investigating metric spikes. This involves understanding and explaining sudden changes in key metrics, such as a dip in daily user engagement or a drop in sales.
- **Approach:**
- Through SQL queries and aggregations, the collected data will be explored and analyzed to uncover patterns, trends and anomalies. For this project, I have used My SQL to extract the required data from the given database using the Join function, subqueries, Aggregation, where condition, Group by, Distinct and other functions required.
- **Tech-Stack Used:**
- Used MySQL Workbench 8.0 community server version 8.0.33 which is owned by oracle.

1. **CaseStudy1:**

➢ **Jobs Reviewed Over Time:** Calculate the number of jobs reviewed per hour for each day in November 2020.

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

  - **Query:**

```sql
SELECT DISTINCT
    ds AS days,
    COUNT(job_id) / (SUM(time_spent) / 3600) AS no_of_jobs_reviewed
FROM
    job_data
GROUP BY days;
```

- **Result:**

days	no_of_jobs_reviewed
2020-11-30	180.0000
2020-11-29	180.0000
2020-11-28	218.1818
2020-11-27	34.6154
2020-11-26	64.2857
2020-11-25	80.0000

- **Insights:** The number of job reviewed per hour per day in November 2020 varies, with higher activity on some days and lower activity on others.

- **Throughput Analysis:** Calculate the 7-day rolling average of throughput (number of events per second).
- **Task:** 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:**

```sql
SELECT DISTINCT
    ds AS days,
    COUNT(job_id) / (SUM(time_spent) / 3600) AS no_of_jobs_reviewed
FROM
    job_data
GROUP BY days;
```

```sql
SELECT
    ds,
    ROUND((COUNT(event) / SUM(time_spent)), 2) AS daily_metric
FROM
    job_data
GROUP BY ds;
```

- **Result:**

7_day_rolling_throughput
0.03

ds	daily_metric
2020-11-30	0.05
2020-11-29	0.05
2020-11-28	0.06
2020-11-27	0.01
2020-11-26	0.02
2020-11-25	0.02

- **Insights:** The 7-day rolling average throughput of provides a smooth view of the data. I would prefer using the 7-day rolling average for throughput because it provides a more stable representation of performance trends. This can help In identifying long-term patterns.

➢ **Language Share Analysis:** Calculate the percentage share of each language in the last 30 days.

- **Task:** Write an SQL query to calculate the percentage share of each language over the last 30 days.

- **Query:**

```sql
SELECT
    language,
    ROUND(((COUNT(language) / 8) * 100), 2) AS percentage_share
FROM
    job_data
GROUP BY language;
```

- **Result:**

language	percentage_share
English	12.50
Arabic	12.50
Persian	37.50
Hindi	12.50
French	12.50
Italian	12.50

- **Insights:** The language distribution in last 30 days is relatively balanced, with Persian having the highest score.

➢ **Duplicate Rows Detection:** Identify duplicate rows in the data.

- **Task:** Write an SQL query to display duplicate rows from the job_data table.

- **Query:**

```
SELECT
    *
FROM
    job_data
GROUP BY ds , job_id , actor_id , event , language , time_spent , org
HAVING COUNT(*) > 1;
```

- **Result:**

	ds	job_id	actor_id	event	language	time_spent	org

- **Insights:** Here we see that there are not any duplicate rows in job_data table. But if there any then we to implement data validation mechanisms to prevent such duplicates.

2. CaseStudy2:

➢ **Weekly User Engagement:** Measure the activeness of users on a weekly basis.

- **Task:** Write an SQL query to calculate the weekly 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);
```

weeks	no_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
33	1225
34	1204
35	104

- **Insights:**
- User engagement around 30 weeks has shown some fluctuation over the time period.

➢ **Throughput Analysis:** Analyze the growth of users over time for a product.

- **Task:** Write an SQL query to calculate the user growth for the product.

- **Query:**

```
select week_num, year_num,
sum(active_users) over (order by week_num, year_num ) as cumulative_sum
from (
SELECT
    WEEK(activated_at) AS week_num,
    YEAR(activated_at) AS year_num,
    COUNT(DISTINCT user_id) AS active_users
FROM
    users
GROUP BY year_num , week_num
ORDER BY year_num , week_num) as alias;
```

- **Result:**

week_num	year_num	cumulative_sum
0	2013	23
0	2014	106
1	2013	136
1	2014	262
2	2013	310
2	2014	419
3	2013	455
3	2014	568
4	2013	598
4	2014	728
5	2013	776
5	2014	909
6	2013	947
6	2014	1082
7	2013	1124
7	2014	1249
8	2013	1283
8	2014	1412
9	2013	1455
9	2014	1588
10	2013	1620
10	2014	1774

week_num	year_num	cumulative_sum
11	2013	1805
11	2014	1935
12	2013	1968
12	2014	2116
13	2013	2155
13	2014	2322
14	2013	2357
14	2014	2519
15	2013	2562
15	2014	2726
16	2013	2772
16	2014	2951
17	2013	3000
17	2014	3170
18	2013	3214
18	2014	3377
19	2013	3434
19	2014	3619
20	2013	3658
20	2014	3834
21	2013	3883
21	2014	4066

week_num	year_num	cumulative_sum
22	2013	4120
22	2014	4316
23	2013	4366
23	2014	4562
24	2013	4607
24	2014	4836
25	2013	4893
25	2014	5100
26	2013	5156
26	2014	5357
27	2013	5409
27	2014	5631
28	2013	5703
28	2014	5918
29	2013	5985
29	2014	6206
30	2013	6273
30	2014	6511
31	2013	6578
31	2014	6771
32	2013	6842
32	2014	7087

week_num	year_num	cumulative_sum
33	2014	7421
34	2013	7499
34	2014	7758
35	2013	7821
35	2014	7839
36	2013	7911
37	2013	7996
38	2013	8086
39	2013	8170
40	2013	8257
41	2013	8330
42	2013	8429
43	2013	8518
44	2013	8614
45	2013	8705
46	2013	8793
47	2013	8895
48	2013	8992
49	2013	9108
50	2013	9232
51	2013	9334
52	2013	9381

- **Insights:** User growth has been positive over time with some fluctuation.

- **Weekly Retention Analysis:** Analyze the retention of users on a weekly basis after signing up for a product.
- **Task:** Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

Query

```sql
SELECT
    WEEK(occurred_at) AS weeks,
    COUNT(DISTINCT user_id) AS no_of_users
FROM
    events
WHERE
    event_type = 'signup_flow'
        AND event_name = 'complete_signup'
GROUP BY weeks
ORDER BY weeks;
```

Result

weeks	no_of_users
17	72
18	163
19	185
20	176
21	183
22	196
23	196
24	229
25	207
26	201
27	222
28	215
29	221
30	238
31	193
32	245
33	261
34	259
35	18

- **Insights:** Weekly user retention shows a gradual decline over time.

- **Weekly Engagement Per Device:** Measure the activeness of users on a weekly basis per device.
- **Task:** Write an SQL query to calculate the weekly engagement per device.

- **Query:**

```sql
SELECT
    WEEK(occurred_at) AS Weeks,
    device,
    COUNT(DISTINCT user_id) AS User_engagement
FROM
    events
WHERE
    event_type = 'engagement'
GROUP BY device , weeks
ORDER BY weeks;
```

- **Result:**

Weeks	device	User_engagement
17	acer aspire desktop	9
17	acer aspire notebook	20
17	amazon fire phone	4
17	asus chromebook	21
17	dell inspiron desktop	18
17	dell inspiron notebook	46
17	hp pavilion desktop	14
17	htc one	16
17	ipad air	27
17	ipad mini	19
17	iphone 4s	21
17	iphone 5	65
17	iphone 5s	42
17	kindle fire	6
17	lenovo thinkpad	86
17	mac mini	6
17	macbook air	54
17	macbook pro	143
17	nexus 10	16
17	nexus 5	40
17	nexus 7	18
17	nokia lumia 635	17

*This is just a sample output of only 19 rows. There are 491 rows return from the above query which could be handling in a single page.

- **Insights:** Engagement varies across different devices and weeks. Some devices show consistently higher engagement than others.

- **Email Engagement Analysis:** Analyze how users are engaging with the email service.
- **Task:** Write an SQL query to calculate the email engagement metrics.
- **Query:**

```sql
SELECT
    WEEK(occurred_at) AS Weeks,
    COUNT(DISTINCT CASE
            WHEN action = 'sent_weekly_digest' THEN user_id
        END) AS weekly_emails,
    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 email_opened,
    COUNT(DISTINCT CASE
            WHEN action = 'email_clickthrough' THEN user_id
        END) AS email_clickthrough
FROM
    email_events
GROUP BY weeks;
```

- **Result:**

Weeks	weekly_emails	reengagement_mail	email_opened	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
32	3897	200	1318	416
33	4012	264	1417	490
34	4111	261	1502	481
35	0	48	41	38

- **Insights:** The email engagement is divided in 4 parts **weekly_emails, reengagement_mail, email_opened, email_clickthrough** with respect to weeks.