Operation Analytics and Investigating Metric Spike

Project Description:

In this project, the primary objective was to analyse the data provided by the company. The main task is to obtain valuable insights and answer the questions asked by different teams. This information will be used by operation departments like support and marketing team to find the efficiency of their campaigns. In case study 1, they gave a table named job_data while, In the case study 2 there are users, events1 and email_events1 tables.

Approach:

I was planned to do whole project using SQL workbench. Unfortunately, case study 2 consisted of large data which was needed to import was unable to get inserted so I used Ms SQL server and worked fine.

Firstly, I am need to create a database to import dataset provided by the company. Then I need to import the data to the application to perform various analysis on the data to produce valuable information to different departments. By performing various analysis, I think I will be able to solve all the question raised by the company. Most of them should be answered on a daily basis or weekly basis.

Tech-Stack Used:

- MySQL workbench for small data especially in the first case study.
- MS SQL server used for big data and the import speed of data 100x faster.
- Google Drive to upload and store the Project

CASE STUDY - I (JOB DATA)

A. Number of jobs reviewed:

OBJECTIVE:

Calculate the number of jobs reviewed per hour per day for November 2020.

Code used to get the result:

```
SELECT
    ds AS date,
    ROUND(COUNT(job_id) / SUM(time_spent) * 3600) AS `No. of Jobs reviewed per day`
FROM
    job_data
GROUP BY ds
HAVING ds BETWEEN '2020-11-01' AND '2020-11-30';
```

Result:

	date	No. of Jobs reviewed per day
▶	2020-11-30	180
	2020-11-29	180
	2020-11-28	218
	2020-11-27	35
	2020-11-26	64
	2020-11-25	80

Conclusion: number of jobs reviewed on a day is 28/11/2020 with 218.

B. Throughput:

OBJECTIVE:

Calculate 7 day rolling average of throughput.

Code used to get the result:

Weekly Throughput:

```
SELECT
    ROUND(COUNT(event) / SUM(time_spent), 3) AS `Events per second`
FROM
```

```
    job_data;
```

Result :

	Events per second
▶	0.027

Daily Throughput:

```
select ds,round(count(event)/sum(time_spent),3) as`Events per second`
from job_data
group by ds
order by ds;
```

Result :

	ds	Events per second
▶	2020-11-25	0.022
	2020-11-26	0.018
	2020-11-27	0.010
	2020-11-28	0.061
	2020-11-29	0.050
	2020-11-30	0.050

Conclusion: The throughout on 7 day rolling basics is showed above and daily metric is 0.027

C. Percentage share of each language:

OBJECTIVE:

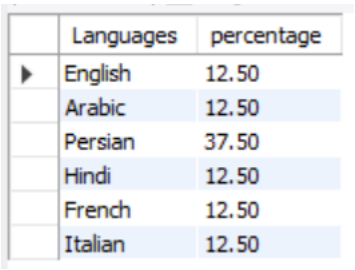
Calculate the percentage share of languages.

Code used to get the result:

```
SELECT
    LANGUAGE AS Languages,
    ROUND(100 * COUNT(*) / sub.total, 2) AS percentage
FROM
    JOB_DATA
        CROSS JOIN
    (SELECT
        COUNT(*) AS total
    FROM
        job_data) AS sub
```

```
GROUP BY LANGUAGE , sub.total;
```

Result :



	Languages	percentage
▶	English	12.50
	Arabic	12.50
	Persian	37.50
	Hindi	12.50
	French	12.50
	Italian	12.50

Conclusion: Persian has the highest share with 37.5%

D. Duplicate rows:

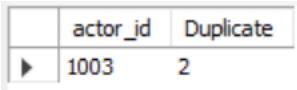
OBJECTIVE:

To find the duplicate row in the row.

Code used to get the result:

```
SELECT
    actor_id, COUNT(*) AS Duplicate
FROM
    job_data
GROUP BY actor_id
HAVING COUNT(*) > 1;
```

Result:



	actor_id	Duplicate
▶	1003	2

Conclusion: There was a duplicate row with actor id :1003

CASE STUDY - II

While importing the data I faced numerous problems in MySQL workbench, especially the importing speed of the data was too slow and it collapse in the middle so some of the questions were answered using the MS SQL Server which has 100 times faster importing speed as compared to workbench.

A. Duplicate rows:

OBJECTIVE:

To calculate the weekly user engagement.

Code used to get the result:

```
SELECT DATEPART(week, occurred_at) as weekno, COUNT(DISTINCT user_id) as
"Weekly engagement"
from Process_analytics.dbo.events
WHERE event_type = 'engagement'
GROUP BY DATEPART(week, occurred_at)
order by weekno
```

Result:

Results | Messages

	weekno	Weekly engagement
1	18	663
2	19	1068
3	20	1113
4	21	1154
5	22	1121
6	23	1186
7	24	1232
8	25	1275
9	26	1264
10	27	1302
11	28	1372
12	29	1365
13	30	1376
14	31	1467
15	32	1299
16	33	1225
17	34	1225
18	35	1204
19	36	104

B. Duplicate rows:

OBJECTIVE:

To calculate the weekly user engagement.

Code used to get the result:

```
select month, user,
round(((USER/lag(user , 1) over( order by month)-1)*100),2) as `Percentage of Growth`
from (select extract(month from created_at) as month,
count( activated_at) as user
from users
where state = 'active'
group by MONTH)  sub
```

Result:

	month	user	Percentage of Growth
▶	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

C. Duplicate rows:

OBJECTIVE:

To calculate the weekly user engagement.

Code used to get the result:

```
SELECT first as `Week Number`,
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 - n.first) AS week_number
   FROM
   (SELECT user_id, extract(week from occurred_at) AS login_week
      FROM `operation analytics`.`table-2 events`
      GROUP BY user_id, extract(week from occurred_at)) m
   JOIN
   (SELECT user_id, MIN(extract(week from occurred_at)) AS first
      FROM `operation analytics`.`table-2 events`
      GROUP BY user_id) n
 ON m.user_id = n.user_id) sub
GROUP BY first
ORDER BY first;
```

Result:

	Week Number	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
1	18	740	472	324	251	205	187	167	146	145	145	136	131	132	143	116	91	82	77	5
2	19	788	362	261	203	168	147	144	127	113	122	106	118	127	110	97	85	67	4	0
3	20	601	284	173	153	114	95	91	81	95	82	68	65	63	42	51	49	2	0	0
4	21	555	223	165	121	91	72	63	67	63	65	67	41	40	33	40	0	0	0	0
5	22	495	187	131	91	74	63	75	72	58	48	45	39	35	28	2	0	0	0	0
6	23	521	224	150	107	87	73	63	60	55	48	41	39	31	1	0	0	0	0	0
7	24	542	219	138	101	90	79	69	61	54	47	35	30	0	0	0	0	0	0	0
8	25	535	205	143	102	81	63	65	61	38	39	29	0	0	0	0	0	0	0	0
9	26	500	218	139	101	75	63	50	46	38	35	2	0	0	0	0	0	0	0	0
10	27	495	181	114	83	73	55	47	43	29	0	0	0	0	0	0	0	0	0	0
11	28	493	199	121	106	68	53	40	36	1	0	0	0	0	0	0	0	0	0	0
12	29	486	194	114	69	46	30	28	3	0	0	0	0	0	0	0	0	0	0	0
13	30	501	186	102	65	47	40	1	0	0	0	0	0	0	0	0	0	0	0	0
14	31	533	202	121	78	53	3	0	0	0	0	0	0	0	0	0	0	0	0	0
15	32	430	145	76	57	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	33	496	188	94	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	34	499	202	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	35	518	44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	36	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

D. Duplicate rows:

OBJECTIVE:

To calculate the weekly user engagement.

Code used to get the result:

```
SELECT extract(week from occurred_at) as Week_number,

COUNT(DISTINCT CASE when device in ('dell inspironnotebook')

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 `table-2 events`
WHERE EVENT_TYPE = 'ENGAGEMENT'
```

GROUP BY Week_number

order by Week_number;

Result:

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

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
20	21	16	17	7	9	6	14	18	27	4	16
33	42	19	33	15	26	13	37	58	52	9	30
41	27	30	23	11	23	18	40	36	55	12	25
40	41	29	22	18	23	26	30	52	59	11	22
47	38	21	25	20	29	18	44	41	51	5	25
41	52	24	25	19	25	25	38	52	58	5	27
43	49	20	31	14	22	18	54	53	41	16	45
40	43	20	35	20	24	29	56	59	57	11	38
47	38	21	37	14	28	21	52	52	57	13	29
35	49	23	42	9	29	11	46	60	56	13	29
49	52	27	31	15	29	15	56	53	55	10	37
49	50	26	35	10	30	28	56	56	54	6	26
53	49	31	43	16	28	31	58	54	52	12	25
60	56	31	34	15	33	23	42	54	70	12	36
55	56	13	28	14	31	24	51	44	55	14	24
55	62	18	28	12	35	20	51	57	48	12	30
46	49	19	27	13	39	32	38	37	40	14	23
63	47	25	17	13	30	30	36	49	39	11	25
3	6	2	2	1	1	2	1	1	0	0	2

E. Duplicate rows:

OBJECTIVE:

To calculate the weekly user engagement.

Code used to get the result:

```
SELECT Week,
  ROUND(sentweeklydigest/total*100, 2) AS `sent weekly digest`,
  ROUND(emailopen/total*100, 2) AS `email open`,
  ROUND(emailclickthrought/total*100, 2) AS `email clickthrough`,
  ROUND(sentreengagementemail/total*100, 2) AS `sent reengagement email`
FROM (
  SELECT EXTRACT(WEEK FROM occurred_at) AS Week,
    COUNT(CASE WHEN action = 'sent_weekly_digest' THEN user_id ELSE NULL END) AS sentweeklydigest,
    COUNT(CASE WHEN action = 'email_open' THEN user_id ELSE NULL END) AS emailopen,
    COUNT(CASE WHEN action = 'email_clickthrough' THEN user_id ELSE NULL END) AS emailclickthrought,
    COUNT(CASE WHEN action = 'sent_reengagement_email' THEN user_id ELSE NULL END) AS
sentreengagementemail,
```

```
  COUNT(user_id) AS total
 FROM email_events
 GROUP BY Week
) sub
GROUP BY Week
ORDER BY Week;
```

Result:

	Week	sent weekly digest	email open	email clickthrought	sent reengagement email
▶	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: This project helped me to understand the importance of operational analytics and how metric spike is been used by the company to find the growth on a daily or weekly basis. I also understood that the some of the code SQL work bench usually used is not found in MS SQL Server, So in some cases they use another function to get the same results