Operation Analytics and Investigating Metric Spike

Project Description:

Operational Analytics is a crucial process that involves analyzing a company's end-to-end operations. In this project, You'll be provided with various datasets and tables, and your task will be to derive insights from this data to answer questions posed by different departments within the company. Your goal is to use your advanced SQL skills to analyze the data and provide valuable insights that can help improve the company's operations and understand sudden changes in key metrics.

Approach: Firstly, I try to understand the data provided in the database like no. of tables, column names and the relationship between them. Then I started understanding the questions asked and run the suitable queries in MySQL which I've learnt, to get the possible outcomes.

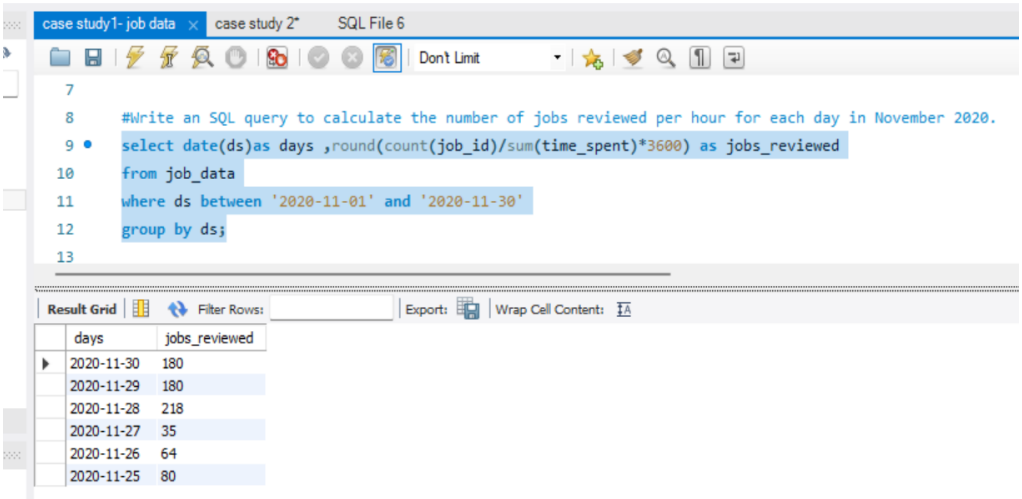
Tech-Stack Used: I've used MySQL Workbench 8.0 CE to get the desired results by querying the data present on the database. I've used this because MySQL is one of the most used database management software and is user friendly.

Insights: Below are the insights and the answers of the questions asked by the team

Case Study 1: Job Data Analysis

Tasks:

A. **Jobs Reviewed Over Time:** Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.



days	jobs_reviewed
2020-11-30	180
2020-11-29	180
2020-11-28	218
2020-11-27	35
2020-11-26	64
2020-11-25	80

B. **Throughput Analysis:** 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.

```
#Write an SQL query to calculate the 7-day rolling average of throughput.
SELECT ROUND(COUNT(event)/SUM(time_spent), 2) AS "Weekly Throughput" FROM job_data;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Weekly Throughput
0.03

*Weekly throughput is 0.03.

case study1- job data* | case study 2* | SQL File 6

Don't Limit

```
#Write an SQL query to calculate the 7-day rolling average of throughput.
SELECT ROUND(COUNT(event)/SUM(time_spent), 2) AS "Weekly Throughput" FROM job_data;
#The weekly throughput is 0.03.
SELECT ds AS Dates, ROUND(COUNT(event)/SUM(time_spent), 2) AS "Daily Throughput" FROM job_data
GROUP BY ds
ORDER BY ds;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

*On daily basis highest throughput is 0.06.

***Metrics will always go up and down on a weekly and daily basis. You'll get numbers faster every day or minute if you want. As a result, rolling metrics are superb at showing if your metrics are trending up or down on a daily level.

C. **Language Share Analysis:** Write an SQL query to calculate the percentage share of each language over the last 30 days.

case study1- job data | case study 2* | SQL File 6

```sql
#Write an SQL query to calculate the percentage share of each language over the last 30 days.
select language, round(100*count(*)/total,2) as per_share,sub.total,count(*)as t2
from job_data
cross join(
    select count(*) as total from job_data) as sub
group by language;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

language	per_share	total	t2
English	12.50	8	1
Arabic	12.50	8	1
Persian	37.50	8	3
Hindi	12.50	8	1
French	12.50	8	1
Italian	12.50	8	1

D. **Duplicate Rows Detection:** Write an SQL query to display duplicate rows from the job_data table.

case study1- job data* | case study 2* | SQL File 6

```sql
#Write an SQL query to display duplicate rows from the job_data table.
select actor_id, count(*) as duplicate_row
from job_data
group by actor_id
having count(actor_id)>1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

actor_id	duplicate_row
1003	2

Case Study 2: Investigating Metric Spike

Tasks:

A. **Weekly User Engagement:** Write an SQL query to calculate the weekly user engagement.

```
# Write an SQL query to calculate the weekly user engagement.
select extract(week from occurred_at) as week_numbers , count(distinct user_id) as weekly_active_users
from events
where event_type='engagement'
group by 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

Result Grid | Filter Rows:

week_numbers	weekly_active_users
24	1275
25	1264
26	1302
27	1372
28	1365
29	1376
30	1467
31	1299
32	1225
33	1225
34	1204
35	104

B. **User Growth Analysis:** Write an SQL query to calculate the user growth for the product.

```
case study1- job data*   case study 2*   SQL File 6
Don't Limit

#Write an SQL query to calculate the user growth for the product.
 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 NOT IN("")
GROUP BY 1
ORDER BY 1
) sub;
```

C. **Weekly Retention Analysis:** 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, EXTRACT(WEEK FROM occurred_at) AS login_week
FROM events GROUP BY 1, 2) m,

(SELECT user_id, MIN(EXTRACT(WEEK FROM occurred_at)) AS first
FROM events

GROUP BY 1) n

WHERE m.user_id = n.user_id

) sub

GROUP BY first

ORDER BY first;
```

Output:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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	663	472	324	251	205	187	167	146	145	145	136	131	132	143	116	91	82	77	5
18	596	362	261	203	168	147	144	127	113	122	106	118	127	110	97	85	67	4	0
19	427	284	173	153	114	95	91	81	95	82	68	65	63	42	51	49	2	0	0
20	358	223	165	121	91	72	63	67	63	65	67	41	40	33	40	0	0	0	0
21	317	187	131	91	74	63	75	72	58	48	45	39	35	28	2	0	0	0	0
22	326	224	150	107	87	73	63	60	55	48	41	39	31	1	0	0	0	0	0
23	328	219	138	101	90	79	69	61	54	47	35	30	0	0	0	0	0	0	0
24	339	205	143	102	81	63	65	61	38	39	29	0	0	0	0	0	0	0	0
25	305	218	139	101	75	63	50	46	38	35	2	0	0	0	0	0	0	0	0
26	288	181	114	83	73	55	47	43	29	0	0	0	0	0	0	0	0	0	0
27	292	199	121	106	68	53	40	36	1	0	0	0	0	0	0	0	0	0	0
28	274	194	114	69	46	30	28	3	0	0	0	0	0	0	0	0	0	0	0
29	270	186	102	65	47	40	1	0	0	0	0	0	0	0	0	0	0	0	0
30	294	202	121	78	53	3	0	0	0	0	0	0	0	0	0	0	0	0	0
31	215	145	76	57	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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
22	326	224	150	107	87	73	63	60	55	48	41	39	31	1	0	0	0	0	0
23	328	219	138	101	90	79	69	61	54	47	35	30	0	0	0	0	0	0	0
24	339	205	143	102	81	63	65	61	38	39	29	0	0	0	0	0	0	0	0
25	305	218	139	101	75	63	50	46	38	35	2	0	0	0	0	0	0	0	0
26	288	181	114	83	73	55	47	43	29	0	0	0	0	0	0	0	0	0	0
27	292	199	121	106	68	53	40	36	1	0	0	0	0	0	0	0	0	0	0
28	274	194	114	69	46	30	28	3	0	0	0	0	0	0	0	0	0	0	0
29	270	186	102	65	47	40	1	0	0	0	0	0	0	0	0	0	0	0	0
30	294	202	121	78	53	3	0	0	0	0	0	0	0	0	0	0	0	0	0
31	215	145	76	57	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	267	188	94	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	286	202	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	279	44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

D. **Weekly Engagement Per Device:** Write an SQL query to calculate the weekly engagement per device.

<u>Query:</u>

SELECT EXTRACT(WEEK FROM occurred_at) AS "Week Numbers",

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'

GROUP BY 1

ORDER BY 1;
```

<u>Output:</u>

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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	Samsu 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

Result 7

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

Result 7

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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	Samsu Tablet
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

Result 7

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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
49	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

Result 7

E. **Email Engagement Analysis:** Write an SQL query to calculate the email engagement metrics.

<u>Query:</u>

```
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:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

Result 8

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Week	Weekly Digest Rate	Email Open Rate	Email Clickthrough Rate	Reengagement Email Rate
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

Results:

This project helps me to understand the importance of operation analytics. Through this project I am able to understand how the companies use metric spike as a secret weapon. With an informed and proactive approach, they can leverage insights to make data-backed decisions that optimize their strategy and boost ROI.

Operational Analytics tackles the problem by synchronizing real-time data. Operational Analytics has the capability to aggregate data from multiple data sources into a cumulative, organized, actionable solution capable of delivering analytical models in real-time to create individual customer profiles and a holistic view of operations for a company. This guarantees that your operational routines and systems are used efficiently. Whenever utilized correctly, operational analytics can achieve a significant positive effect on our general public and world everywhere and increment the general efficiency of specific areas.