Operation Analytics and Investigating Metric Spike
Advanced SQL

Project Description
- In this project, we will perform operation analytics and investigative metric
- Operation Analytics is the analysis done for the complete end to end operations of a company. Which will help yammers company to find the areas which they have to improve.
- Investigating metric spike will help us to understand about any kind of dip happening in engagement

Approach
Using database of yammers users, SQL queries will be run to analyse. Operation Analytics and Investigating Metric

Tech-Stack Used: MySQL Tutorial
- MySQL is a widely used relational database management system.
- It is free and open-source.
- It is ideal for both small and large application.
- It is very easy to write query in MySQL
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works very quickly and works well even with large data sets.

Insights:
Case Study 1 (Job Data)

A. **Number of jobs reviewed:** Amount of jobs reviewed over time.

Number of jobs reviewed per hour per day for November 2020 are as follow:

Date_	Job_count	Hours_spent
2020-11-30	2	0.0111
2020-11-29	1	0.0056
2020-11-28	2	0.0092
2020-11-27	1	0.0289
2020-11-26	1	0.0156
2020-11-25	1	0.0125

Query used:

```
use jobdata;
select ds as Date_,count(job_id) as Job_count, sum(time_spent)/3600 as Hours_spent
from data_table
where ds >='2020-11-01'  and ds <='2020-11-30'
group by ds
```

B. **Throughput:** It is the no. of events happening per second.

7 day rolling average of throughput as follow:

ds	time_spent	rolling_avg
2020-11-25	45	45.0000
2020-11-26	56	50.5000
2020-11-27	104	68.3333
2020-11-28	22	56.7500
2020-11-28	11	47.6000
2020-11-29	20	43.0000
2020-11-30	15	39.0000
2020-11-30	25	36.1429

Query used:

```
select ds, time_spent,
avg(time_spent) over(order by ds rows between 6 preceding and current row) as rolling_avg
FROM data_table
order by ds;
```

If a business is small then daily metric is preferred whereas if the business is large the prefer 7-day rolling is preferred. This analysis will help them to grow their business.

C. **Percentage share of each language:** Share of each language for different contents.
Percentage share of each language in the last 30 days is as follow:

	language	total_time_per_language	total_time	percentage share of language
▶	Arabic	25	220	11.36364
	English	15	220	6.81818
	French	104	220	47.27273
	Hindi	11	220	5.00000
	Italian	45	220	20.45455
	Persian	98	220	44.54545

Query used:

```
SELECT language,
sum(time_spent) as total_time_per_language,
sum(time_spent) over(order by language rows between unbounded preceding and unbounded following ) as total_time,
sum(time_spent)* 100.0 / sum(time_spent) Over() as 'percentage share of language'
FROM data_table
where ds >='2020-11-01'  and ds <='2020-11-30'
group by language;
```

D. **Duplicate rows:** Rows that have the same value present in them.

As there is no two rows same in given data the result is null.

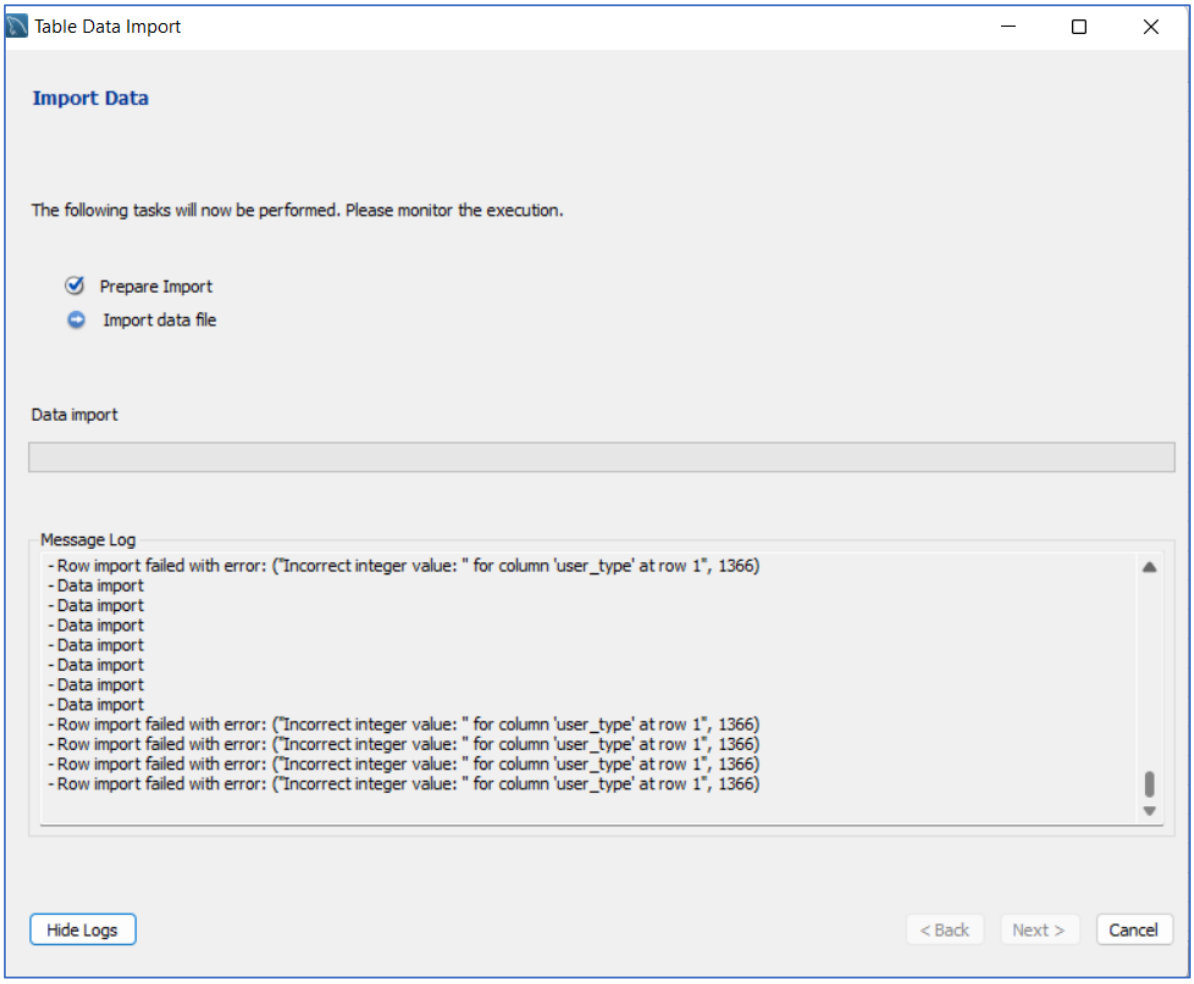
	ds	job_id	actor_id	event	language	time_spent	org	rownum

Query used:

```
with cte as
( select *,
row_number() over (partition by ds, job_id, actor_id) as rownum from
data_table)
select*
from cte
where rownum > 1
```

Case Study 2 (Investigating metric spike)

I tried for more than 3 hours to import tables correctly but it always showed same error so output can vary than actual one.



A. **User Engagement:** To measure the activeness of a user. Measuring if the user finds quality in a product/service.

Weekly 2882 users engage with product

week_	active_users
NULL	2882

Query used:

```
select extract(week from occurred_at) as week_ , count(distinct user_id) as active_users
from events
where event_type = 'engagement' and event_name = 'login'
group by 1
order by 1;
```

B. **User Growth:** Amount of users growing over time for a product.

Query used:

```
select extract(day from created_at) as day_,
count(*) as user_count,
count(case when activated_at is not null then user_id else null end) as activated_user_count
from users
where created_at >= '2012-04-01'and created_at < '2012-04-30'
group by 1
order by 1;
```

C. **Weekly Retention:** Users getting retained weekly after signing-up for a product.

Query used:

```
select extract(week from a.occurred_at) as week_, avg (a.age_at_event) as avg_age_durig_week,
count(distinct case when a.user_age > 70 then a.user_id else null end) as '10+ weeks',
count(distinct case when a.user_age < 70 and a.user_age >=63 then a.user_id else null end) as '9 weeks',
count(distinct case when a.user_age < 63 and a.user_age >-56 then a.user_id else null end) as '8 weeks',
count(distinct case when a.user_age < 56 and a.user_age >-49 then a.user_id else null end) as '17 weeks',
count(distinct case when a.user_age < 49 and a.user_age >=42 then a.user_id else null end) as '16 weeks',
count(distinct case when a.user_age < 42 and a.user_age >=35 then a.user_id else null end) as '5 weeks',
count(distinct case when a.user_age < 35 and a.user_age >=28 then a.user_id else null end) as '4 weeks',
count(distinct case when a.user_age < 28 and a.user_age >=21 then a.user_id else null end) as '3 weeks',
count(distinct case when a.user_age < 21 and a.user_age >-14 then a.user_id else null end) as '12 weeks',
count(distinct case when a.user_age < 14 and a.user_age >=7 then a.user_id else null end) as '1 weeks',
count(distinct case when a.user_age < 7 and a.user_age >=63 then a.user_id else null end) as 'less than a week'
from
(select b.occurred_at, c.user_id,
extract(week from c.activated_at) as activation_week,
extract(day from b.occurred_at - c.activated_at) as age_at_event,
extract(day from '2014-09-01' - c.activated_at) as user_age
from case2.users c
join case2.events b
on b.user_id = c.user_id
and b.event_type = 'engagement'
and b.event_name= 'login'
and b.occurred_at > 2014-05-01 and occurred_at <2014-09-01
where c.activated_at is not null) a
group by 1
order by 1;
```

D. **Weekly Engagement:** To measure the activeness of a user. Measuring if the user finds quality in a product/service weekly.

Weekly engagement per device is as follow

Total weekly users: 2882
Computer: 1513
Phone: 1042
Tablet: 417

week_	weekly_active_users	computer	phone	tablet
NULL	2882	1513	1042	417

Query used:

```
select extract(week from occurred_at) as week_,
count(distinct user_id) as weekly_active_users,

count(distinct case when device in
('macbook pro', 'lenovo thinkpad', 'macbook air',' dell inspiron notebook',
'asus chromebook', 'dell inspiron desktop', 'acer aspire notebook',
'hp pavilion desktop', 'acer aspire desktop', 'mac mini')
then user_id else null end) as computer,

count(distinct case when device in('iphone 5','samsung galaxy s4',
'nexus 5','iphone 5s', 'iphone 4s', 'nokia lumia 635',
 'htc one','samsung galaxy note', 'amazon fire phone')
 then user_id else null end) as phone,

count(distinct case when device in('ipad air', 'nexus 7','ipad mini',
'nexus 10', 'kindle fire', 'windows surface', 'samsung galaxy tablet')
then user_id else null end) as tablet

from events
where event_type = 'engagement'
and event_name = 'login'
group by 1
order by 1;
```

E. **Email Engagement:** Users engaging with the email service.

Email engagement metrics as follow:

For weekly sent count: 57267
Email opened count: 20459
Clicked links count: 9010

	week_	weekly_email_count	email_open_count	email_clickthroughs_count
▶	NULL	57267	20459	9010

Query used:

```
select extract(week from occurred_at) as week_,
count(case when action = 'sent_weekly_digest' then user_id else null end) as weekly_email_count,
count(case when action= 'email_open' then user_id else null end) as email_open_count,
count(case when action = 'email_clickthrough' then user_id else null end) as email_clickthroughs_count
from email_events
group by 1;
```

Result:

- In this project I have gain practical hands on knowledge advanced SQL queries to perform operation analytics and investigative metrics spike.
- This will further help me to perform data analysis in real world scenarios.