Operation Analytics and Investigating Metric Spike

Presented by Aditya Palande

email: adityap.works@gmail.com

TRAINITY PROJECT
TASK 3

Content

01. **CASE Study 1:**
Job data Analysis

- **a.** Jobs Reviewed Over Time
- **b.** Throughput Analysis
- **c.** Language Share Analysis
- **d.** Duplicate Rows Detection

02. **CASE Study 2:**
Investigating metric spike

- **a.** Weekly User Engagement
- **b.** User Growth Analysis
- **c.** Weekly Retention Analysis
- **d.** Weekly Engagement Per Device
- **e.** Email Engagement Analysis

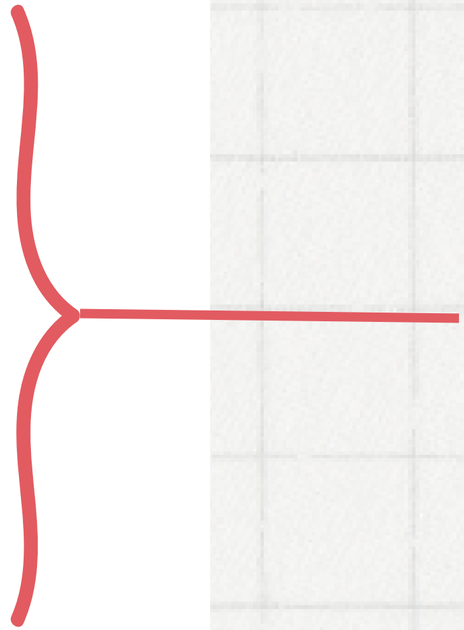
```sql
show databases;

create database Job_data_Analysis;

use Job_data_Analysis;

create table job_data(
    ds varchar(50),
    job_id int NOT NULL,
    actor_id int NOT NULL,
    `event` varchar(50),
    `language` varchar(50),
    time_spent int,
    org char(2)
);

show variables like 'secure_file_priv';
```

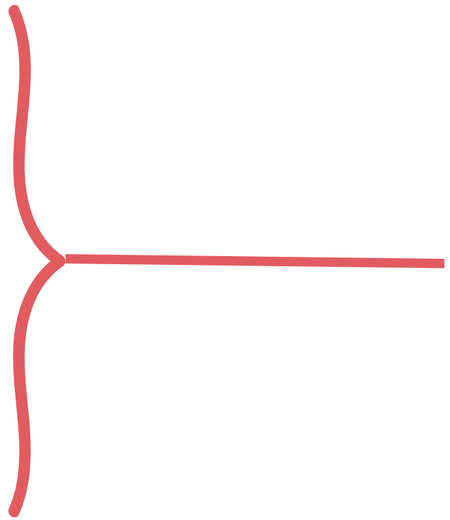


creating database and table

```sql
load data infile 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/job_data.csv'
into table job_data
fields terminated by ','
enclosed by '"'
lines terminated by '\n'
ignore 1 rows;

SET SQL_SAFE_UPDATES = 0;

alter table job_data add column `date` date after ds;
update job_data set `date` = str_to_date(ds, '%m/%d/%Y');
alter table job_data drop column ds;
```



loading data from csv file into the table and changing datatype where needed

```sql
create table users(
    user_id int not null primary key,
    created_at  varchar(50),
    company_id  int not null,
    `language`  varchar(20),
    activated_at varchar(50),
    state varchar(10)
);
```

creating table and loading data from csv file into the table

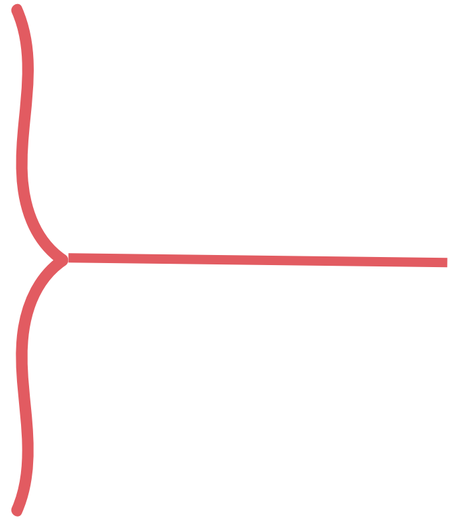
```sql
load data infile 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users.csv'
into table users
fields terminated by ','
enclosed by '"'
lines terminated by '\n'
ignore 1 rows;
```

```sql
alter table users add column activation_date date after `language`;
update users set activation_date = str_to_date(activated_at, '%d-%m-%Y %H:%i');
alter table users drop column activated_at;

alter table users add column creation_date date after user_id;
update users set creation_date = str_to_date(created_at, '%d-%m-%Y %H:%i');
alter table users drop column created_at;

create table `events`(
    user_id int not null,
    occured_at_date varchar(50),
    event_type varchar(20),
    event_name varchar(20),
    location varchar(20),
    device varchar(50),
    user_type int

);
```

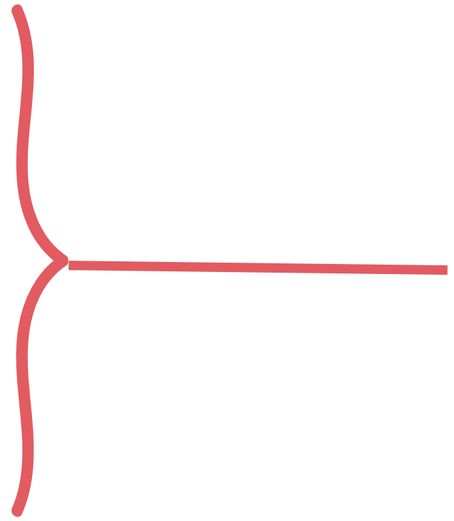


changing datatype and creating “events” table

```sql
alter table events modify column event_name varchar(50);

load data infile 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/events.csv'
into table `events`
fields terminated by ','
enclosed by '"'
lines terminated by '\n'
ignore 1 rows;


create table email_events(
    user_id int not null,
    occured_at_date varchar(50),
    `action` varchar(30),
    user_type int
);
```



loading data into table and creating “email_events” table

```sql
load data infile 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/email_events.csv'
into table `email_events`
fields terminated by ','
enclosed by '"'
lines terminated by '\n'
ignore 1 rows;

select * from email_events;

alter table email_events add column occured_at_date_date datetime after user_id;
update email_events set occured_at_date_date = str_to_date(occured_at_date, '%d-%m-%Y %H:%i');
alter table email_events drop column occured_at_date;
```

loading data and changing datatype

CASE Study 1

Job data Analysis

Jobs Reviewed Over Time:

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

QUERY:

```
SELECT
count(job_id)/sum(time_spent)/3600 as
no_of_jobs_reviewed_per_hour
FROM
job_data;
```

```
36 •    select count(job_id)/sum(time_spent)/3600 as no_of_jobs_reviewed_per_hour from job_data;
```

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

	no_of_jobs_reviewed_per_hour
▶	0.00000746

To find the number of jobs reviewed per hour, we make use of count() and sum() functions, divide them and further convert seconds to hours by dividing the quotient by 3600(60*60)

Throughput Analysis:

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:

```
SELECT
`date`, count(job_id) as no_of_jobs,
sum(time_spent) as total_time_spent,
count(job_id)/sum(time_spent) as
throughput  FROM
job_data
GROUP BY `date`
ORDER BY date;
```

```sql
38    -- 2nd task
39    select
40    `date`, count(job_id) as no_of_jobs, sum(time_spent) as total_time_spent, count(job_id)/sum(time_spent) as throughput
41    from job_data group by `date` order by date;
42
```

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

date	no_of_jobs	total_time_spent	throughput
2020-11-25	1	45	0.0222
2020-11-26	1	56	0.0179
2020-11-27	1	104	0.0096
2020-11-28	2	33	0.0606
2020-11-29	1	20	0.0500
2020-11-30	2	40	0.0500

The throughput is calculated by divided the no_of_jobs by total_time_spent. This gives us the daily metric. For 7-day rolling average of throughput, we simply find the average of throughput for a duration of a week.

The choice between a daily metric and a 7-day rolling average for measuring throughput depends on the specific context of your analysis. For this particular table, I feel considering 7-day rolling average is beneficial as it is better for Long-term Planning and for making predictions. It also removes the impact of outliners in the data.

Language Share Analysis:

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

QUERY:

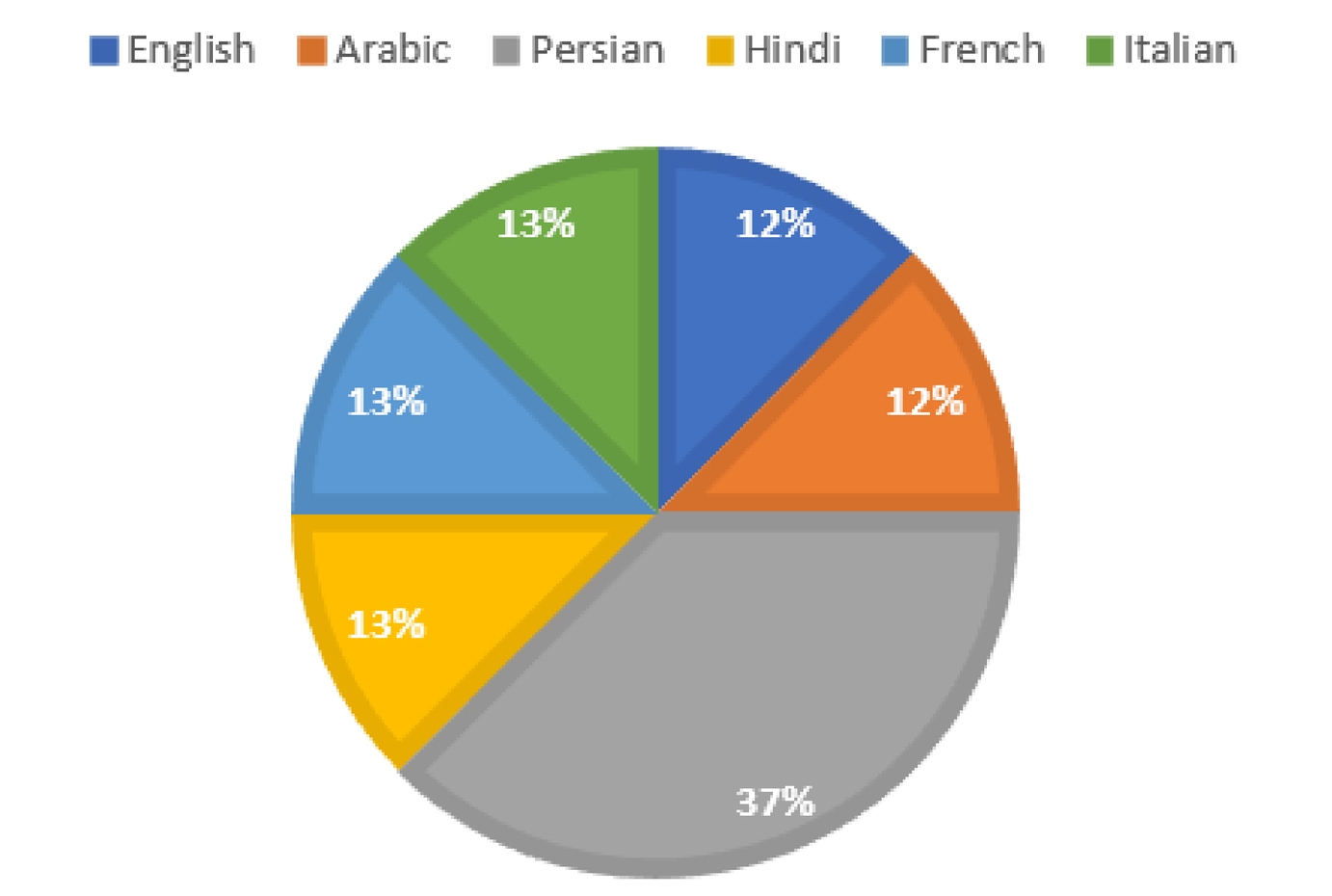
```
SELECT
`language`, count(*) as count,
count(*)*100/(SELECT count(*) FROM
job_data) as percentage
FROM job_data
GROUP BY language;
```

```sql
-- 3nd task
-- Percentage share of each language
select `language`, count(*) as count, count(*)*100/(select count(*) from job_data) as percentage
from job_data
group by language;
```

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

language	count	percentage
English	1	12.5000
Arabic	1	12.5000
Persian	3	37.5000
Hindi	1	12.5000
French	1	12.5000
Italian	1	12.5000

The above table shows the percentage of each language. It can be seen that Persian language has the maximum percentage share



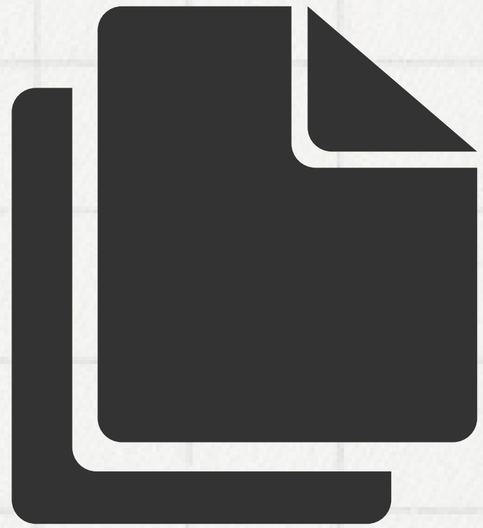
The pie chart can graphically represent percentage share of each language.

Duplicate Rows Detection:

Task: Write an SQL query to display duplicate rows FROM the job_data table.

QUERY:

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



```
49      -- 4th task
50      -- duplicate rows
51 •    select *, count(*) as count
52      from job_data group by date, job_id, actor_id, event, language, time_spent, org
53      having count > 1;
```

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

	date	job_id	actor_id	event	language	time_spent	org	count

Since, there are no duplicate rows in the table, the query returns zero rows.

CASE Study 2

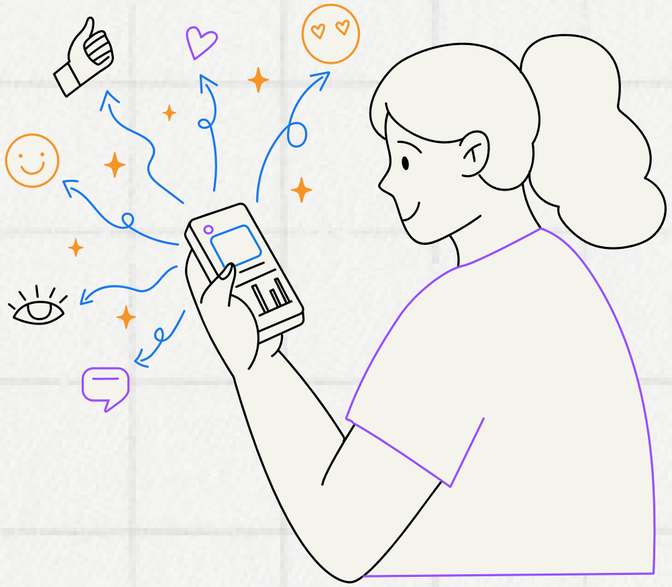
Investigating Metric Spike

Weekly User Engagement:

Task: Write an SQL query to calculate the weekly user engagement.

QUERY:

```
SELECT
extract(year FROM activation_date) as
`year`, extract(week FROM
activation_date) as week_number,
count(*) as
weekly_measure_of_activeness
FROM users
 GROUP BY `year`,week_number;
```



```sql
-- 1st task
select extract(year from activation_date) as `year`, extract(week from activation_date) as week_number,
count(*) as weekly_measure_of_activeness
from users group by `year`,week_number;
```

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

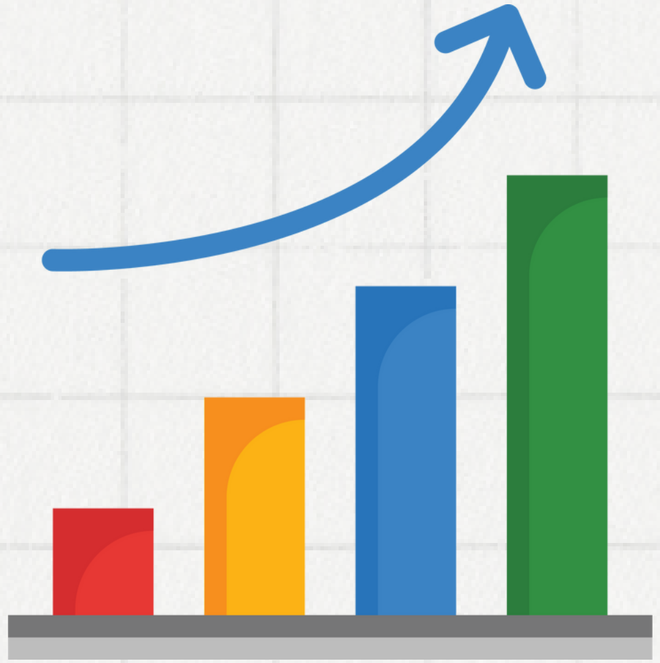
year	week_number	weekly_measure_of_activeness
2013	0	23
2013	1	30
2013	2	48
2013	3	36
2013	4	30
2013	5	48
2013	6	38
2013	7	42
2013	8	34
2013	9	43
2013	10	32
2013	11	31
2013	12	33

User Growth Analysis:

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

QUERY:

```
SELECT
extract(year FROM creation_date) as
year, extract(week FROM creation_date)
as `week`, count(state) as active_state
FROM users
GROUP BY `week`, `year`;
```



```sql
-- 2nd task
-- users growth = number of activ users per week
-- select * from users;
select extract(year from creation_date) as year, extract(week from creation_date) as `week`,
count(state) as active_state
from users group by `week`, `year`;
```

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

year	week	active_state
2013	0	23
2013	1	30
2013	2	48
2013	3	36
2013	4	30
2013	5	48
2013	6	38
2013	7	42
2013	8	34
2013	9	43
2013	10	32
2013	11	31
2013	12	33
2013	13	39
2013	14	35
2013	15	43

Weekly Retention Analysis:

Task: Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

OUTPUT:

user_id	no_of_user	per_week_retention
11768	1	0
11770	1	0
11775	1	0
11778	1	0
11779	1	0
11780	1	0
11785	1	0
11787	1	0
11791	1	0
11793	1	0
11795	1	0
11798	1	0
11799	1	0

QUERY:

```
SELECT distinct user_id, count(user_id)
as no_of_user,
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 occurred_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 occurred_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;
```

Weekly Engagement Per Device:

Task: Write an SQL query to calculate the weekly engagement per device.

QUERY:

```
SELECT
extract(year FROM users.creation_date) as
year, extract(week FROM users.creation_date)
as `week`, device, count(distinct users.user_id)
as activity FROM users
INNER JOIN events ON users.user_id =
events.user_id
GROUP BY device, week, year
ORDER BY year,week,device;
```

```sql
-- 4th task
-- select * from events;
select extract(year from users.creation_date) as year, extract(week from users.creation_date) as `week`, device,
count(distinct users.user_id) as activity
from users inner join events on users.user_id = events.user_id
group by device, week, year order by year,week,device;
```

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

year	week	device	activity
2013	0	asus chromebook	1
2013	0	dell inspiron desktop	3
2013	0	hp pavilion desktop	1
2013	0	ipad air	1
2013	0	iphone 4s	1
2013	0	iphone 5	1
2013	0	iphone 5s	1
2013	0	kindle fire	1
2013	0	lenovo thinkpad	4
2013	0	macbook pro	4
2013	0	nexus 5	2
2013	0	nexus 7	3
2013	0	samsung galaxy s4	2
2013	1	acer aspire desktop	1
2013	1	acer aspire notebook	2
2013	1	amazon fire phone	1

Email Engagement Analysis

Task: Write an SQL query to calculate the email engagement metrics.

OUTPUT:

Result Grid | Filter Rows:

	opening_rate	clicking_rate
▶	33.5834	14.7899

QUERY:

```
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 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 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
job_data_analysis.email_events
) a;
```

Conclusion

The task was performed using basic and advanced topics in Sql. Mysql workbench,a graphical tool, was used to write queries and output the desired results.The Task helped to provide deeper insights FROM the given dataset.