Operation & Metric Analytics

Project Overview

Project Description: As the Lead Data Analyst, my role is to utilize advanced SQL skills for analyzing operations data. I will investigate metric spikes and sudden changes in key metrics, providing valuable insights to various departments. Through data-driven decision-making, I will optimize operational efficiency and drive continuous improvement, helping our company achieve its business goals.

Approach

My approach towards the project involved a systematic and data-driven methodology to execute the analysis effectively. I started by thoroughly understanding the objectives of the project, which involved operational analytics and investigating metric spikes. I then proceeded with the following steps:

1. **Data Understanding:** Reviewed provided datasets and tables to understand data structure and content.
2. **Query Formulation:** Utilized advanced SQL skills to create specific queries for data extraction.
3. **Data Analysis:** Conducted in-depth analysis to identify patterns and sudden changes in key metrics.
4. **Insight Generation:** Derived valuable insights and root causes behind metric spikes and fluctuations.
5. **Communication:** Presented findings through clear reports to respective departments.
6. **Continuous Improvement:** Emphasized data-driven decision-making and encouraged a culture of ongoing improvement.

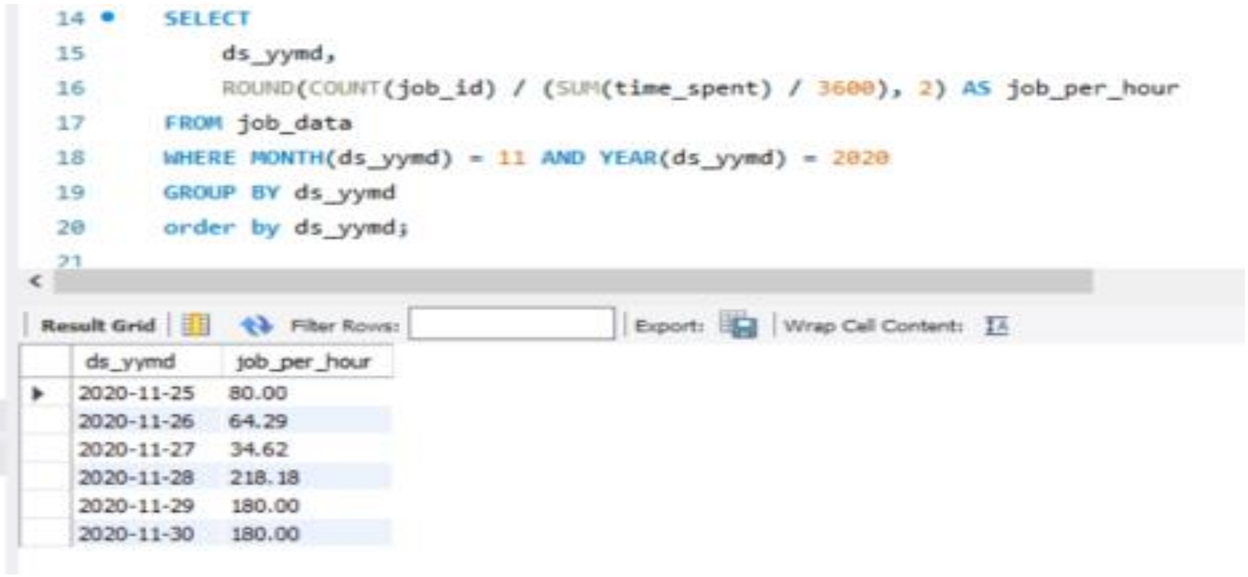
Tech-Stack Used

- **Microsoft Excel:** Employed for data exploration, simple data cleaning, and basic data visualization.
- **MySQL 8:** Used for storing and managing the datasets, providing efficient data storage and retrieval.

By utilizing MySQL 8 and Microsoft Excel, the project achieved efficient data analysis and visualization capabilities, empowering the Data Analyst to derive valuable insights and make informed decisions to improve operational efficiency.

Insights

1. **Jobs Reviewed Over Time:** The following query returns data on how many jobs were reviewed per hour.

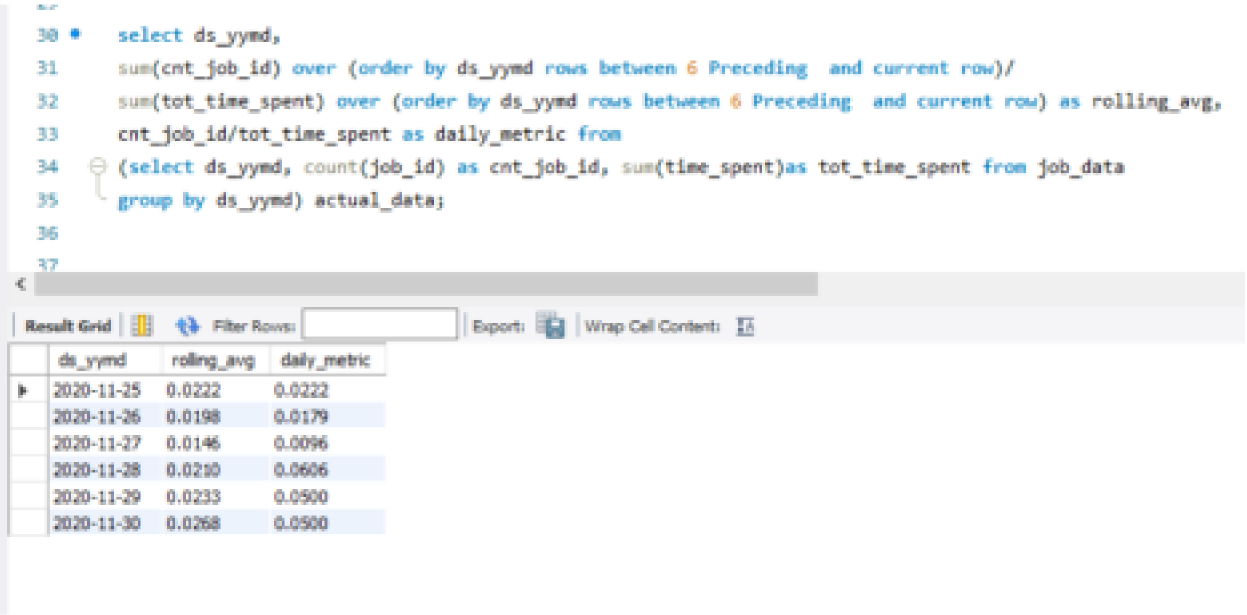


```
SELECT
    ds_yymd,
    ROUND(COUNT(job_id) / (SUM(time_spent) / 3600), 2) AS job_per_hour
FROM job_data
WHERE MONTH(ds_yymd) = 11 AND YEAR(ds_yymd) = 2020
GROUP BY ds_yymd
order by ds_yymd;
```

ds_yymd	job_per_hour
2020-11-25	80.00
2020-11-26	64.29
2020-11-27	34.62
2020-11-28	218.18
2020-11-29	180.00
2020-11-30	180.00

Operation & Metric Analytics

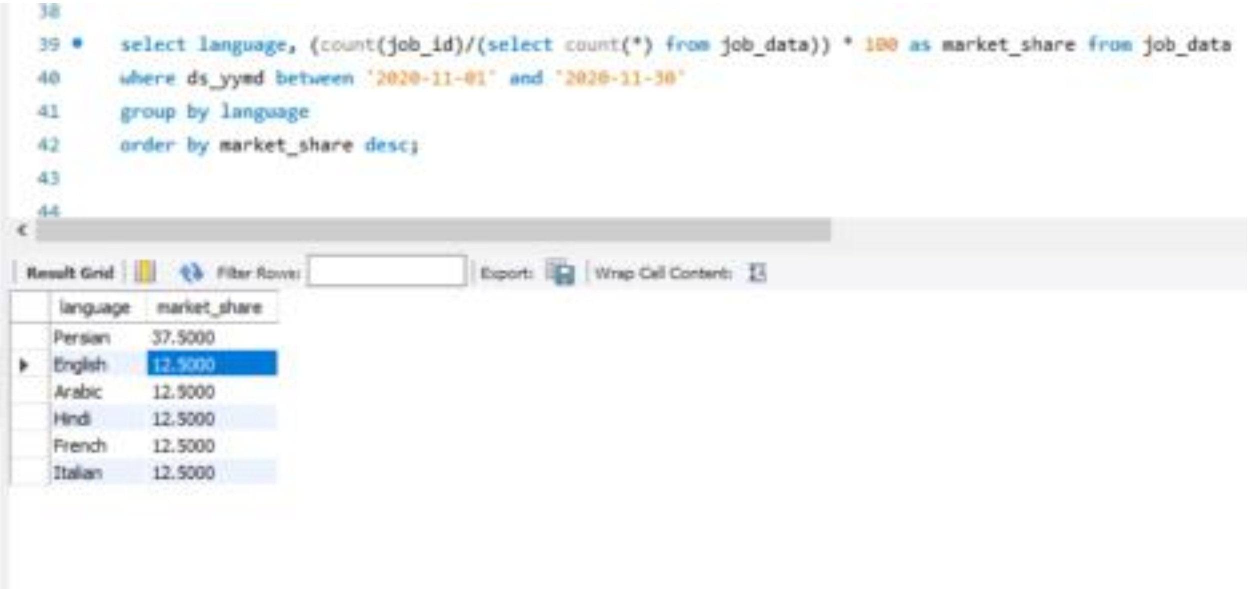
2. **Throughput Analysis:** Both throughput and daily metrics provide valuable insights, focusing on different aspects of the data.



```
select ds_yymd,
sum(cnt_job_id) over (order by ds_yymd rows between 6 Preceding  and current row)/
sum(tot_time_spent) over (order by ds_yymd rows between 6 Preceding  and current row) as rolling_avg,
cnt_job_id/tot_time_spent as daily_metric from
(select ds_yymd, count(job_id) as cnt_job_id, sum(time_spent)as tot_time_spent from job_data
group by ds_yymd) actual_data;
```

ds_yymd	rolling_avg	daily_metric
2020-11-25	0.0222	0.0222
2020-11-26	0.0198	0.0179
2020-11-27	0.0146	0.0096
2020-11-28	0.0210	0.0606
2020-11-29	0.0233	0.0500
2020-11-30	0.0268	0.0500

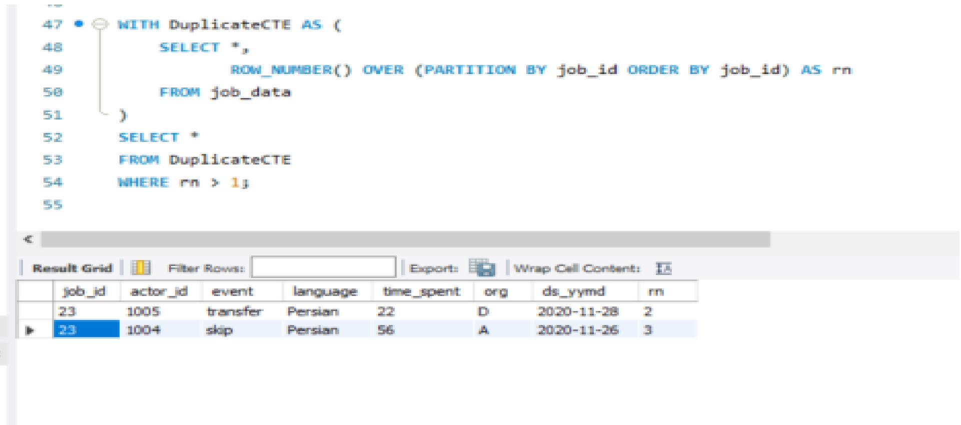
3. **Language Share Analysis:** In the last 30 days, 37.5% of jobs reviewed were in Persian language.



```
select language, (count(job_id)/(select count(*) from job_data)) * 100 as market_share from job_data
where ds_yymd between '2020-11-01' and '2020-11-30'
group by language
order by market_share desc;
```

language	market_share
Persian	37.5000
English	12.5000
Arabic	12.5000
Hindi	12.5000
French	12.5000
Italian	12.5000

4. **Duplicate Rows Detection:** If job ID is unique, it should not be assigned to a new job.



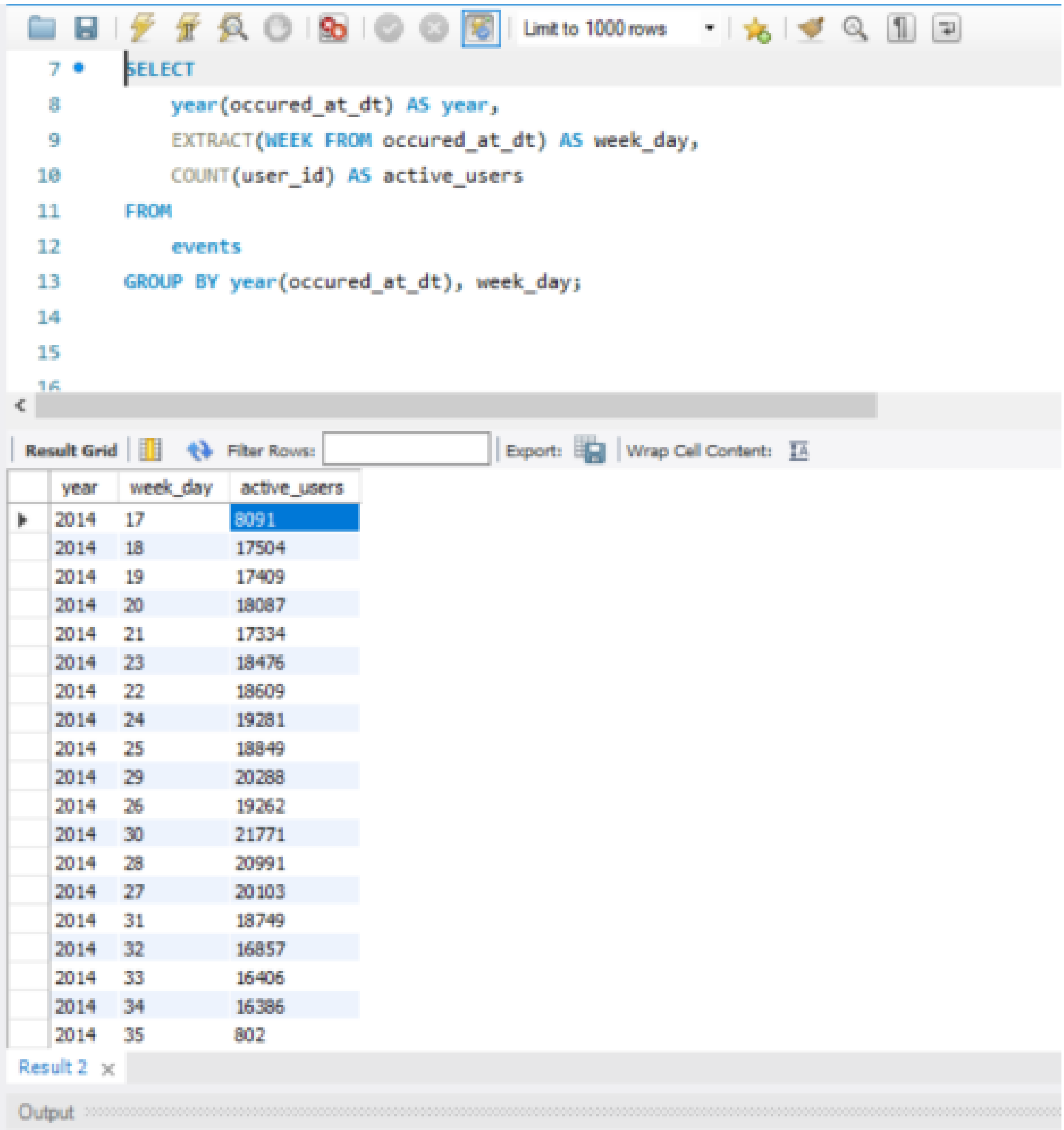
```
WITH DuplicateCTE AS (
    SELECT *,
           ROW_NUMBER() OVER (PARTITION BY job_id ORDER BY job_id) AS rn
    FROM job_data
)
SELECT *
FROM DuplicateCTE
WHERE rn > 1;
```

job_id	actor_id	event	language	time_spent	org	ds_yymd	rn
23	1005	transfer	Persian	22	D	2020-11-28	2
23	1004	skip	Persian	56	A	2020-11-26	3

Operation & Metric Analytics

User Engagement Analysis

1. **Weekly User Engagement:** This result shows weekly user engagement on our platform.

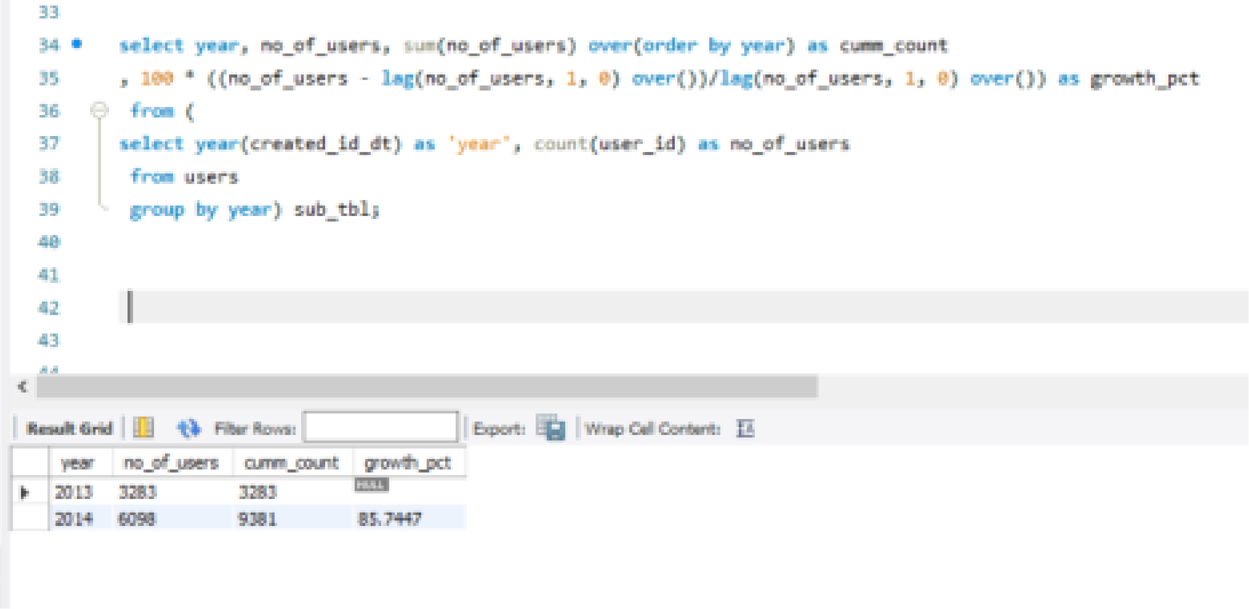


```sql
SELECT
    year(occured_at_dt) AS year,
    EXTRACT(WEEK FROM occured_at_dt) AS week_day,
    COUNT(user_id) AS active_users
FROM
    events
GROUP BY year(occured_at_dt), week_day;
```

year	week_day	active_users
2014	17	8091
2014	18	17504
2014	19	17409
2014	20	18087
2014	21	17334
2014	23	18476
2014	22	18609
2014	24	19281
2014	25	18849
2014	29	20288
2014	26	19262
2014	30	21771
2014	28	20991
2014	27	20103
2014	31	18749
2014	32	16857
2014	33	16406
2014	34	16386
2014	35	802

2. **User Growth Analysis:**
   - Our first query shows year growth in the customer base, increasing by 85.8%.

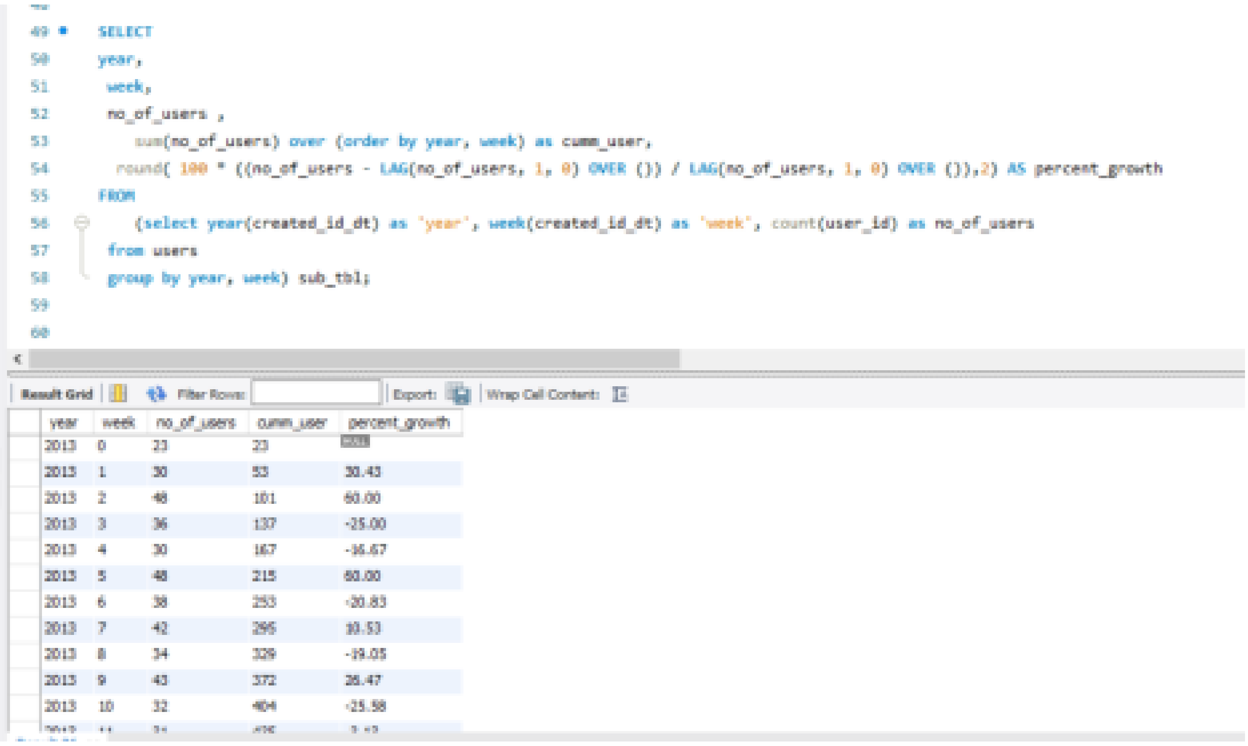
Operation & Metric Analytics



```
select year, no_of_users, sum(no_of_users) over(order by year) as cumm_count
, 100 * ((no_of_users - lag(no_of_users, 1, 0) over())/lag(no_of_users, 1, 0) over()) as growth_pct
 from (
select year(created_id_dt) as 'year', count(user_id) as no_of_users
 from users
 group by year) sub_tbl;
```

year	no_of_users	cumm_count	growth_pct
2013	3283	3283	NULL
2014	6098	9381	85.7447

- ○ The second query shows percent growth on a weekly basis.



```
SELECT
year,
 week,
 no_of_users ,
   sum(no_of_users) over (order by year, week) as cumm_user,
  round( 100 * ((no_of_users - LAG(no_of_users, 1, 0) OVER ()) / LAG(no_of_users, 1, 0) OVER ()),2) AS percent_growth
FROM
   (select year(created_id_dt) as 'year', week(created_id_dt) as 'week', count(user_id) as no_of_users
 from users
 group by year, week) sub_tbl;
```

year	week	no_of_users	cumm_user	percent_growth
2013	0	23	23	NULL
2013	1	30	53	30.43
2013	2	48	101	60.00
2013	3	36	137	-25.00
2013	4	30	167	-16.67
2013	5	48	215	60.00
2013	6	38	253	-20.83
2013	7	42	295	10.53
2013	8	34	329	-19.05
2013	9	43	372	26.47
2013	10	32	404	-25.58

3. **Weekly Retention Analysis:** 4-week retention rate of users on a weekly basis.

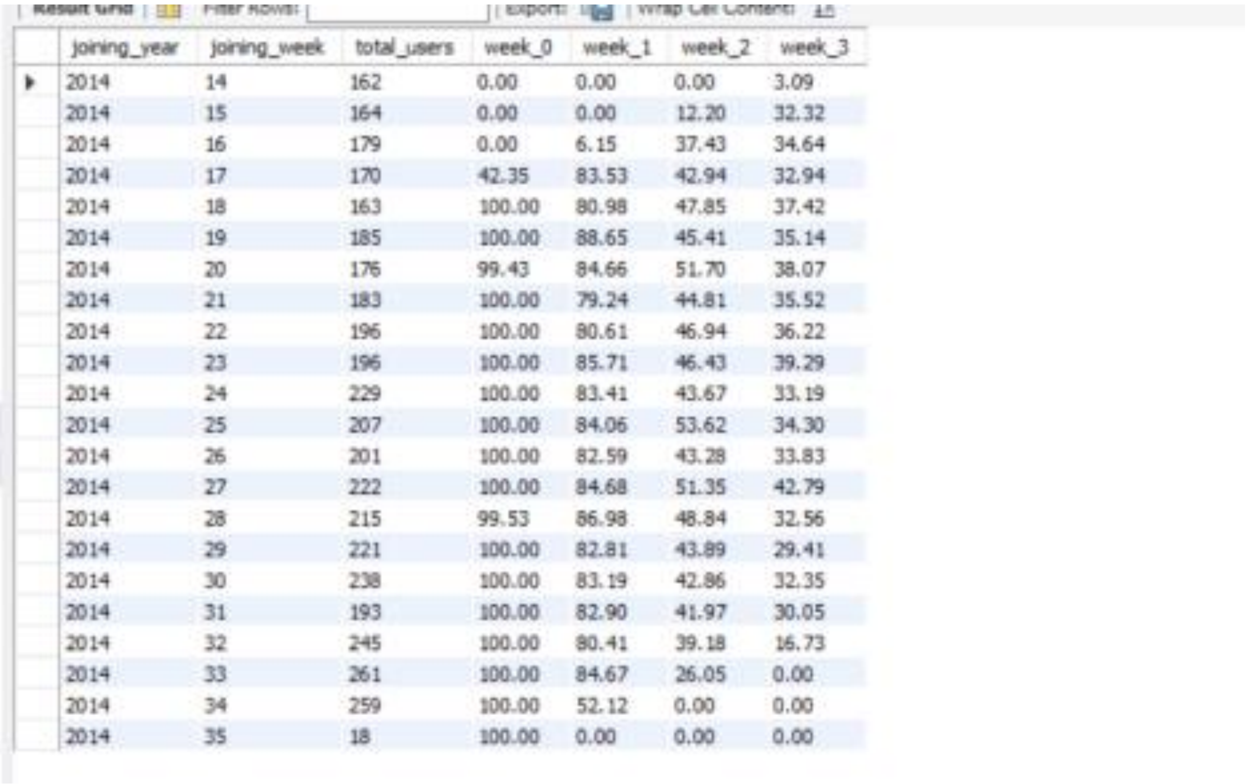
Code:

WITH cohort_week AS (
SELECT
user_id,
WEEK(created_id_dt) AS joining_week,
YEAR(created_id_dt) AS joining_year,
created_id_dt
FROM users
ORDER BY 1
),
user_activities AS (

Operation & Metric Analytics

```
SELECT
e.user_id AS id,
occured_at_dt,
created_id_dt,
DATEDIFF(occured_at_dt, created_id_dt),
CEIL(DATEDIFF(occured_at_dt, created_id_dt) / 7) AS active_diff_week,
joining_week,
joining_year
FROM cohort_week c
LEFT OUTER JOIN events e ON e.user_id = c.user_id
WHERE event_type <> 'signup_flow'
GROUP BY id, active_diff_week
),
cohort_size AS (
SELECT
joining_year,
joining_week,
COUNT(user_id) AS num_users
FROM cohort_week
GROUP BY joining_year, joining_week
ORDER BY joining_year, joining_week
),
retention_table AS (
SELECT
C.joining_year,
C.joining_week,
A.active_diff_week,
COUNT(id) AS num_users
FROM cohort_week C
LEFT JOIN user_activities A ON A.id = C.user_id
GROUP BY joining_year, joining_week, active_diff_week
)
SELECT
joining_year,
joining_week, MAX(total_users) AS total_users,
round(coalesce(MAX(CASE WHEN active_diff_week = 0 THEN percentage END),0), 2) AS week_0,
round(coalesce(MAX(CASE WHEN active_diff_week = 1 THEN percentage END), 0), 2) AS week_1,
round(coalesce(MAX(CASE WHEN active_diff_week = 2 THEN percentage END), 0), 2) AS week_2,
round(coalesce(MAX(CASE WHEN active_diff_week = 3 THEN percentage END), 0), 2) AS week_3
FROM (
SELECT
R.joining_year,
R.joining_week,
S.num_users AS total_users,
R.active_diff_week,
R.num_users * 100 / S.num_users AS percentage
FROM cohort_size S
LEFT JOIN retention_table R ON R.joining_year = S.joining_year AND R.joining_week = S.joining_week
WHERE R.active_diff_week < 4
```

```
) AS pivot_tbl
GROUP BY joining_year, joining_week
ORDER BY joining_year, joining_week;
```

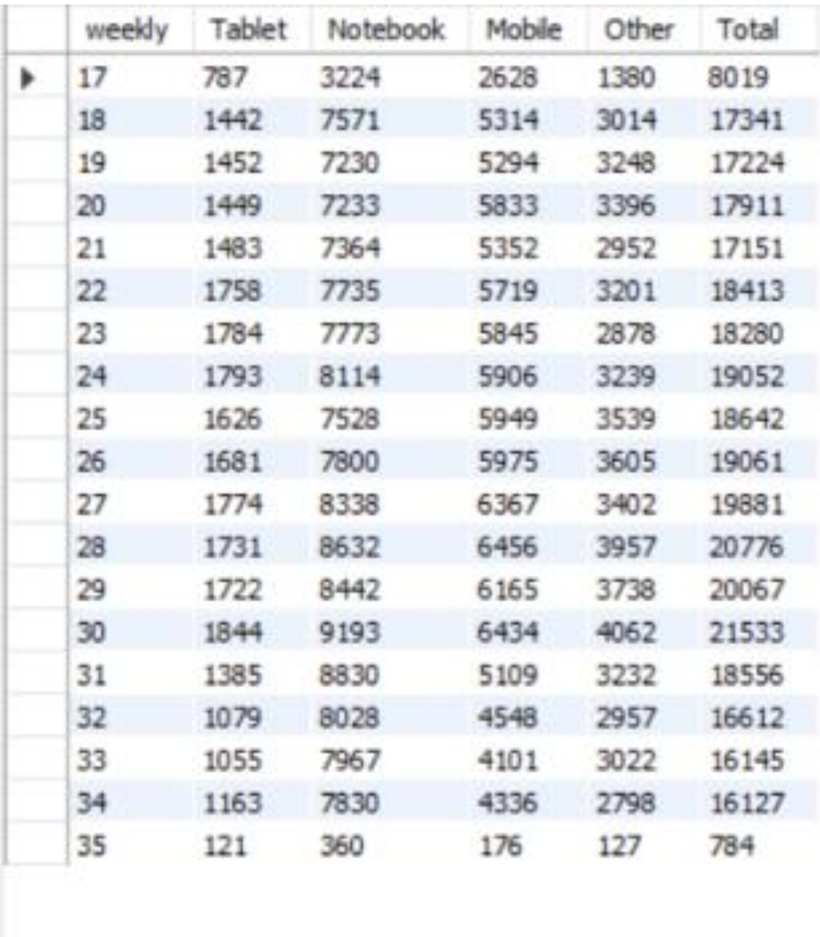


joining_year	joining_week	total_users	week_0	week_1	week_2	week_3
2014	14	162	0.00	0.00	0.00	3.09
2014	15	164	0.00	0.00	12.20	32.32
2014	16	179	0.00	6.15	37.43	34.64
2014	17	170	42.35	83.53	42.94	32.94
2014	18	163	100.00	80.98	47.85	37.42
2014	19	185	100.00	88.65	45.41	35.14
2014	20	176	99.43	84.66	51.70	38.07
2014	21	183	100.00	79.24	44.81	35.52
2014	22	196	100.00	80.61	46.94	36.22
2014	23	196	100.00	85.71	46.43	39.29
2014	24	229	100.00	83.41	43.67	33.19
2014	25	207	100.00	84.06	53.62	34.30
2014	26	201	100.00	82.59	43.28	33.83
2014	27	222	100.00	84.68	51.35	42.79
2014	28	215	99.53	86.98	48.84	32.56
2014	29	221	100.00	82.81	43.89	29.41
2014	30	238	100.00	83.19	42.86	32.35
2014	31	193	100.00	82.90	41.97	30.05
2014	32	245	100.00	80.41	39.18	16.73
2014	33	261	100.00	84.67	26.05	0.00
2014	34	259	100.00	52.12	0.00	0.00
2014	35	18	100.00	0.00	0.00	0.00

4. **Weekly Engagement Per Device:** Analysis of user engagement based on device type.

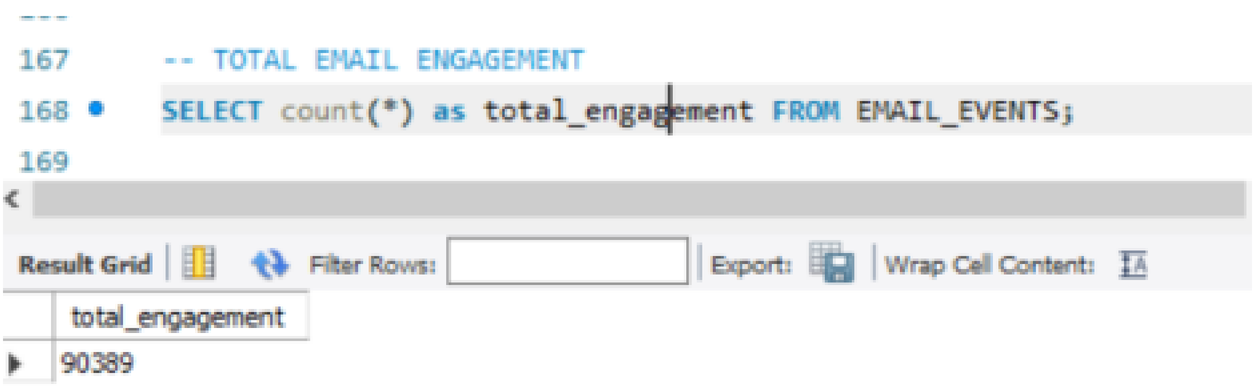
   Code:

```
SELECT

weekly,

SUM(CASE WHEN device_type = 'Tablet' THEN device_count ELSE 0 END) AS Tablet,
SUM(CASE WHEN device_type = 'Notebook' THEN device_count ELSE 0 END) AS Notebook,

SUM(CASE WHEN device_type = 'Mobile' THEN device_count ELSE 0 END) AS Mobile,
SUM(CASE WHEN device_type = 'Other' THEN device_count ELSE 0 END) AS Other,
sum(device_count) as Total

FROM(

select week(occured_at_dt) as weekly, CASE

WHEN device IN ('ipad mini', 'nexus 7', 'samsung galaxy tablet') THEN 'Tablet' WHEN device IN
('dell inspiron notebook', 'macbook air', 'macbook pro', 'acer aspire notebook', 'asus chromebook',
'mac mini', 'hp pavilion desktop', 'acer aspire desktop') THEN 'Notebook'

WHEN device IN ('iphone 5', 'iphone 4s', 'iphone 5s', 'nexus 5', 'samsung galaxy s4', 'htc one',
'amazon fire phone', 'nokia lumia 635') THEN 'Mobile'

WHEN device IN ('windows surface', 'kindle fire', 'nexus 10', 'samsung galaxy note') THEN 'Tablet'

ELSE 'Other'
END AS device_type , count(user_id) as device_count from events

where event_name <> 'complete_signup'

group by weekly, device_type) pivot_tbl

group by weekly

order by weekly;
```



weekly	Tablet	Notebook	Mobile	Other	Total
17	787	3224	2628	1380	8019
18	1442	7571	5314	3014	17341
19	1452	7230	5294	3248	17224
20	1449	7233	5833	3396	17911
21	1483	7364	5352	2952	17151
22	1758	7735	5719	3201	18413
23	1784	7773	5845	2878	18280
24	1793	8114	5906	3239	19052
25	1626	7528	5949	3539	18642
26	1681	7800	5975	3605	19061
27	1774	8338	6367	3402	19881
28	1731	8632	6456	3957	20776
29	1722	8442	6165	3738	20067
30	1844	9193	6434	4062	21533
31	1385	8830	5109	3232	18556
32	1079	8028	4548	2957	16612
33	1055	7967	4101	3022	16145
34	1163	7830	4336	2798	16127
35	121	360	176	127	784

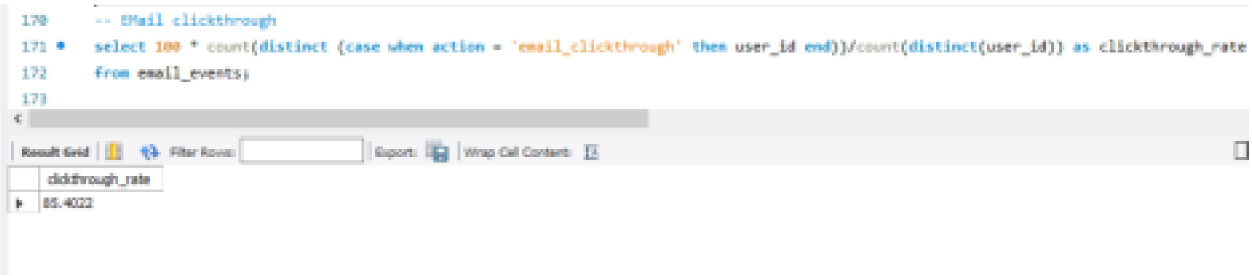
5. **Email Engagement:** Analysis of email engagement metrics.



```
-- TOTAL EMAIL ENGAGEMENT
SELECT count(*) as total_engagement FROM EMAIL_EVENTS;
```

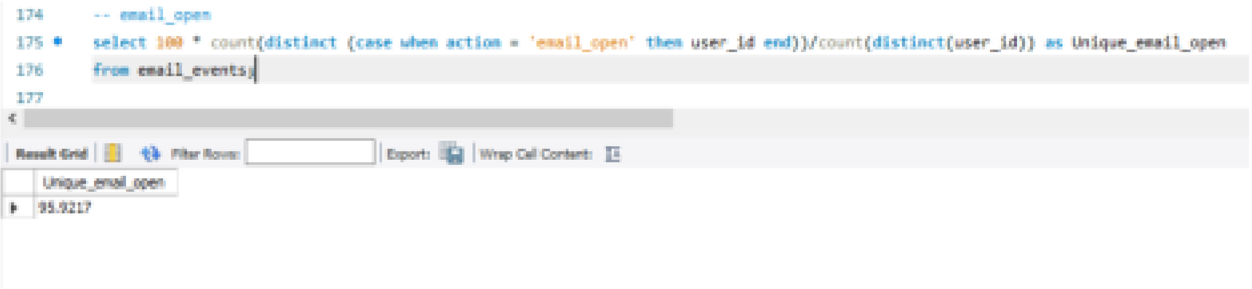
total_engagement
90389

Operation & Metric Analytics



```sql
-- EMail clickthrough
select 100 * count(distinct (case when action = 'email_clickthrough' then user_id end))/count(distinct(user_id)) as clickthrough_rate
from email_events;
```

```
clickthrough_rate
85.4022
```



```sql
-- email_open
select 100 * count(distinct (case when action = 'email_open' then user_id end))/count(distinct(user_id)) as Unique_email_open
from email_events;
```

```
Unique_email_open
95.9217
```

Result: With this analysis, we identified various pros and cons for our company, understanding our performance in reviewing jobs daily and ensuring growth year by year.