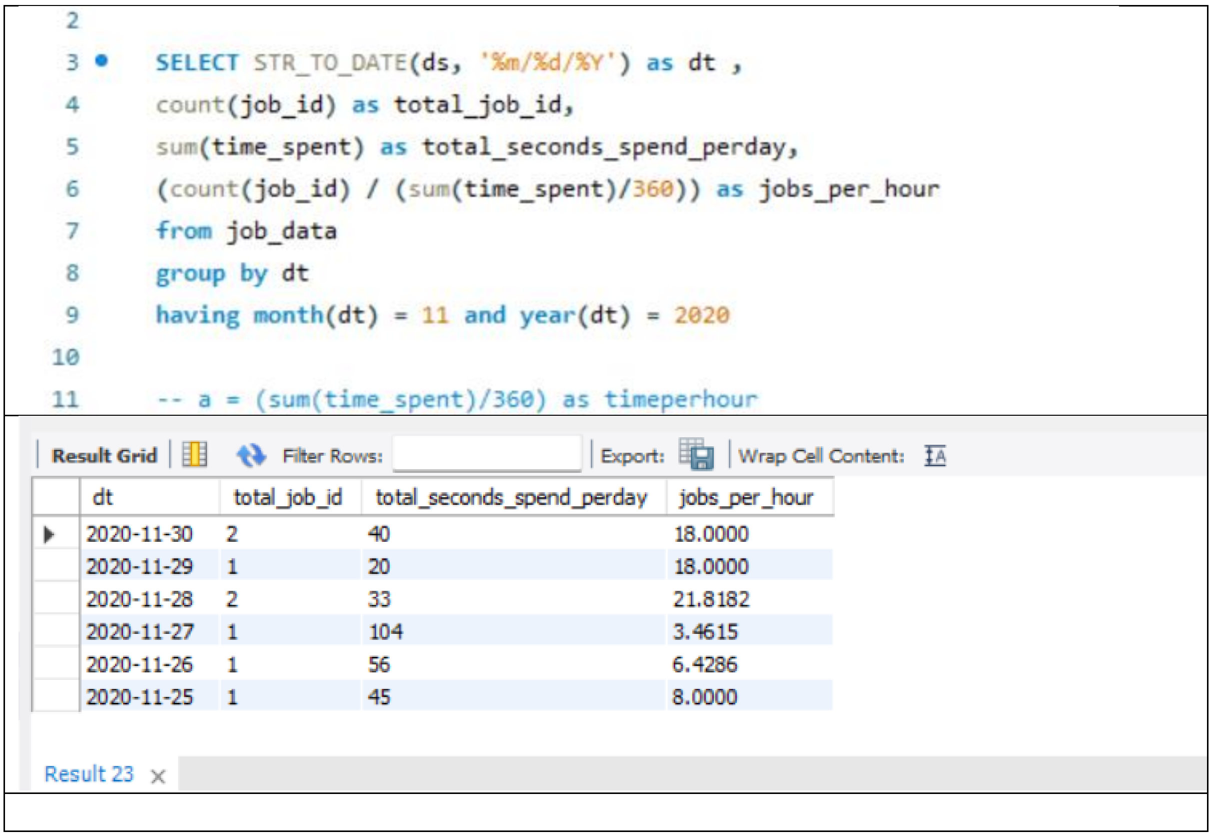
Case Study 1: Job Data Analysis

You will be working with a table named job_data with the following columns:

- **job_id:** Unique identifier of jobs
- **actor_id:** Unique identifier of actor
- **event:** The type of event (decision/skip/transfer).
- **language:** The Language of the content
- **time_spent:** Time spent to review the job in seconds.
- **org:** The Organization of the actor
- **ds:** The date in the format yyyy/mm/dd (stored as text).

Tasks:

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



```

SELECT STR_TO_DATE(ds, '%m/%d/%Y') as dt ,
count(job_id) as total_job_id,
sum(time_spent) as total_seconds_spend_perday,
(count(job_id) / (sum(time_spent)/360)) as jobs_per_hour
from job_data
group by dt
having month(dt) = 11 and year(dt) = 2020

-- a = (sum(time_spent)/360) as timeperhour
```

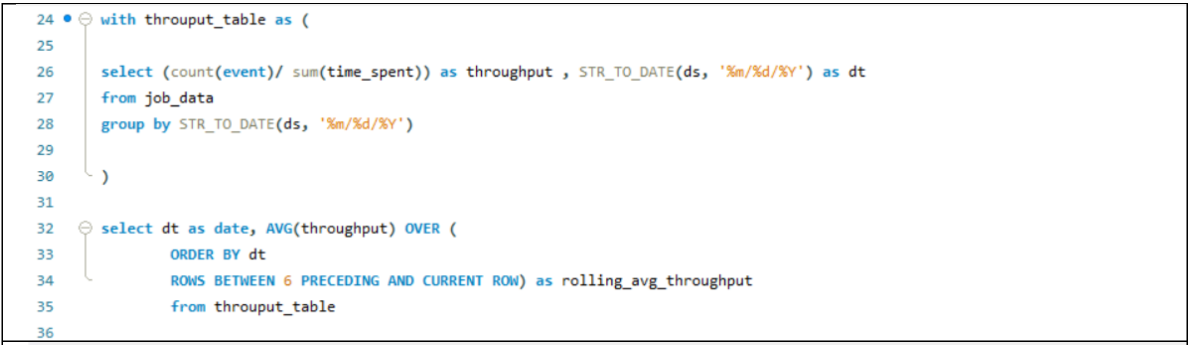
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

dt	total_job_id	total_seconds_spend_perday	jobs_per_hour
2020-11-30	2	40	18.0000
2020-11-29	1	20	18.0000
2020-11-28	2	33	21.8182
2020-11-27	1	104	3.4615
2020-11-26	1	56	6.4286
2020-11-25	1	45	8.0000

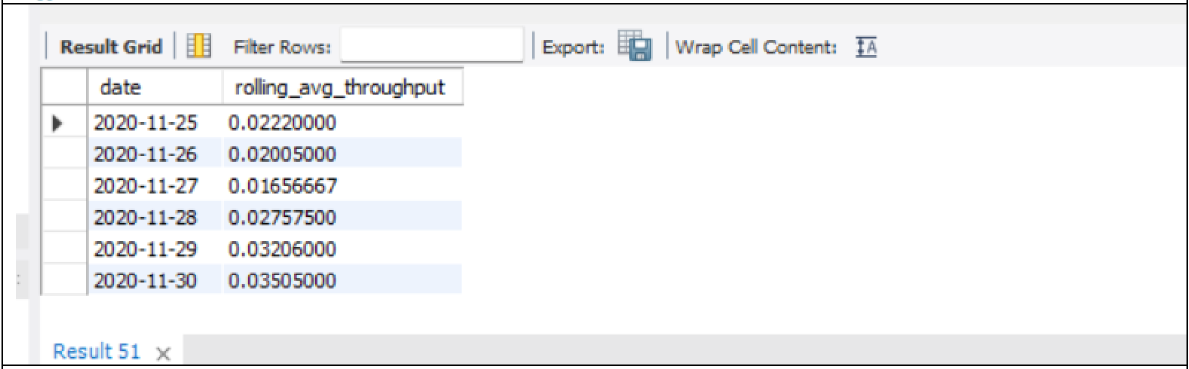
Result 23

B. **Throughput Analysis:**
   - Objective: Calculate the 7-day rolling average of throughput (number of events per second).

- Your 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.



```sql
with througput_table as (

select (count(event)/ sum(time_spent)) as throughput , STR_TO_DATE(ds, '%m/%d/%Y') as dt
from job_data
group by STR_TO_DATE(ds, '%m/%d/%Y')

)

select dt as date, AVG(throughput) OVER (
        ORDER BY dt
        ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) as rolling_avg_throughput
        from througput_table
```

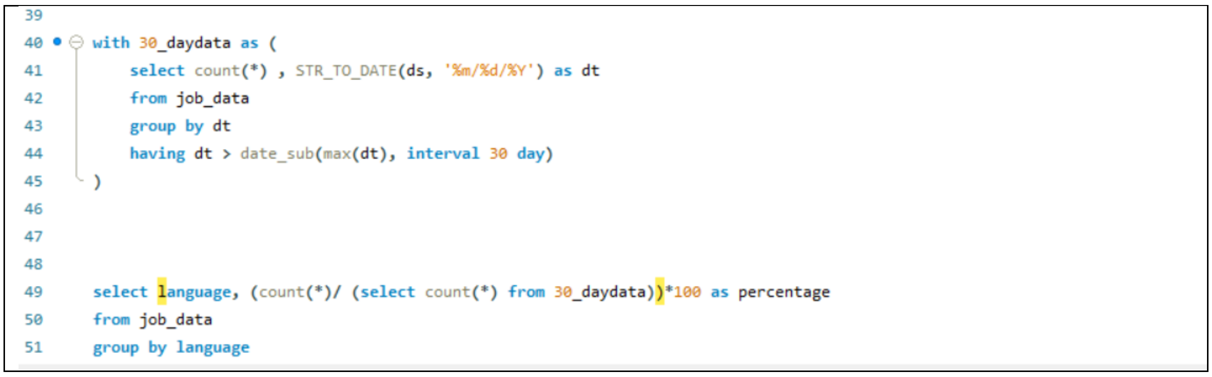


date	rolling_avg_throughput
2020-11-25	0.02220000
2020-11-26	0.02005000
2020-11-27	0.01656667
2020-11-28	0.02757500
2020-11-29	0.03206000
2020-11-30	0.03505000

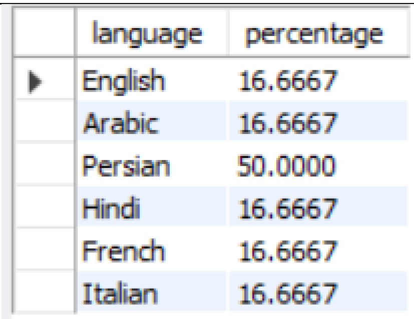
The 7 day rolling average gives a better insight as daily metric is subjected to fluctuations which is taken care in 7 day rolling average. Also for the future prediction, analysis the 7 day rolling average could give better insights for the same reason.

C. Language Share Analysis:

- Objective: Calculate the percentage share of each language in the last 30 days.
- Your Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.



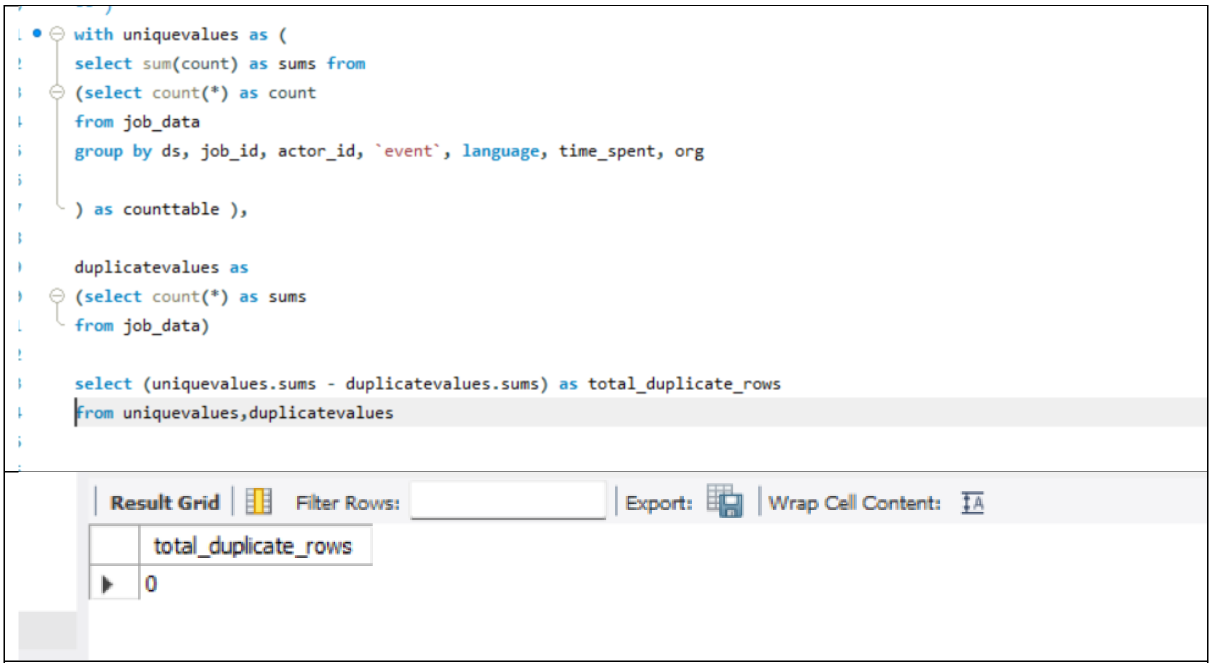
```sql
with 30_daydata as (
    select count(*) , STR_TO_DATE(ds, '%m/%d/%Y') as dt
    from job_data
    group by dt
    having dt > date_sub(max(dt), interval 30 day)
)



select language, (count(*)/ (select count(*) from 30_daydata))*100 as percentage
from job_data
group by language
```



language	percentage
English	16.6667
Arabic	16.6667
Persian	50.0000
Hindi	16.6667
French	16.6667
Italian	16.6667

D. **Duplicate Rows Detection:**

- Objective: Identify duplicate rows in the data.
- Your Task: Write an SQL query to display duplicate rows from the job_data table.



```
with uniquevalues as (
select sum(count) as sums from
(select count(*) as count
from job_data
group by ds, job_id, actor_id, `event`, language, time_spent, org

) as counttable ),

duplicatevalues as
(select count(*) as sums
from job_data)

select (uniquevalues.sums - duplicatevalues.sums) as total_duplicate_rows
from uniquevalues,duplicatevalues
```

```
total_duplicate_rows
0
```

There is no duplicate rows

Case Study 2: Investigating Metric Spike

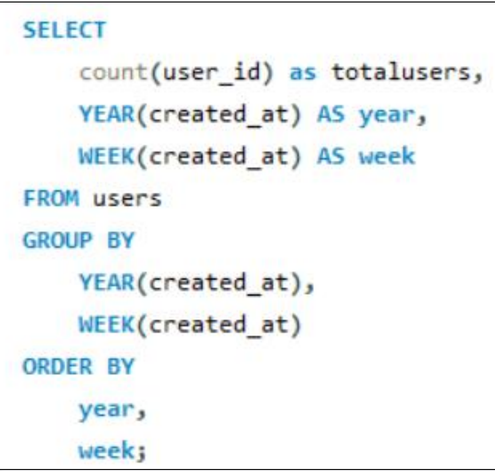
You will be working with three tables:

- **users**: Contains one row per user, with descriptive information about that user's account.
- **events**: Contains one row per event, where an event is an action that a user has taken (e.g., login, messaging, search).

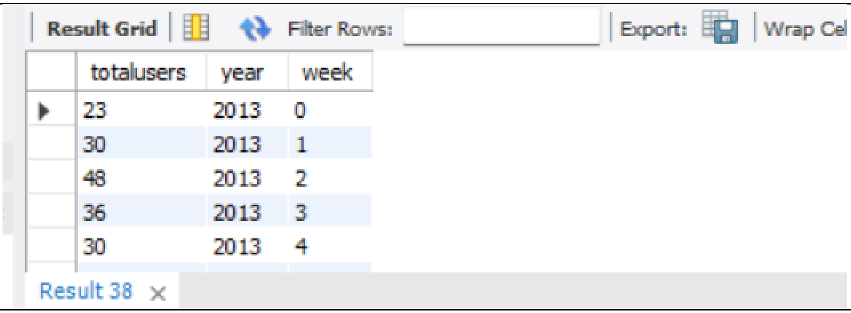
- **email_events**: Contains events specific to the sending of emails.

Tasks:

A. **Weekly User Engagement:**
   - Objective: Measure the activeness of users on a weekly basis.
   - Your Task: Write an SQL query to calculate the weekly user engagement.



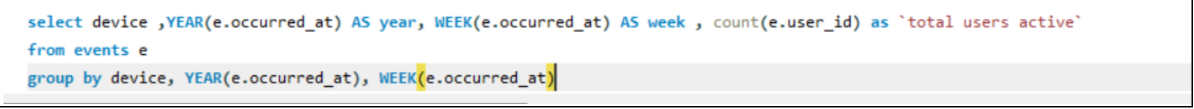
```sql
SELECT
    count(user_id) as totalusers,
    YEAR(created_at) AS year,
    WEEK(created_at) AS week
FROM users
GROUP BY
    YEAR(created_at),
    WEEK(created_at)
ORDER BY
    year,
    week;
```



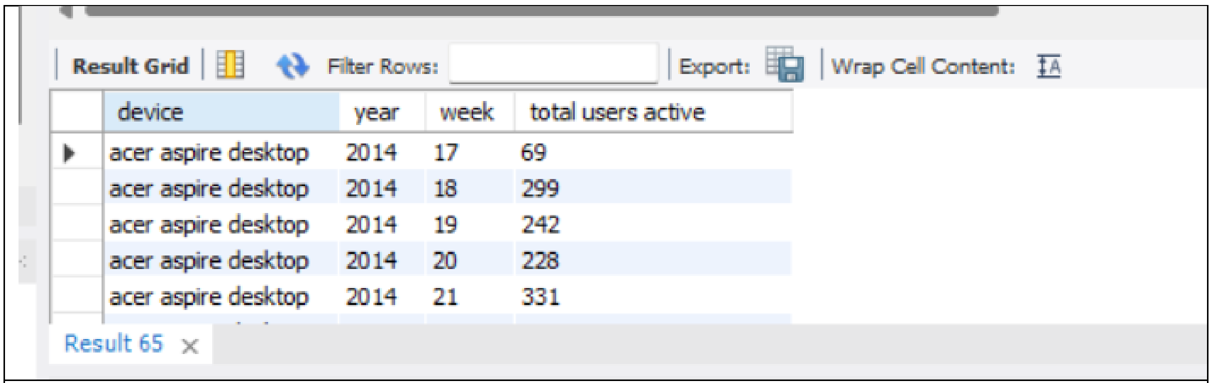
```
  totalusers  year  week
▶ 23          2013  0
  30          2013  1
  48          2013  2
  36          2013  3
  30          2013  4
```

The weekly user engagement is as above

B. **User Growth Analysis:**
   - Objective: Analyze the growth of users over time for a product.
   - Your Task: Write an SQL query to calculate the user growth for the product.



```sql
select device ,YEAR(e.occurred_at) AS year, WEEK(e.occurred_at) AS week , count(e.user_id) as `total users active`
from events e
group by device, YEAR(e.occurred_at), WEEK(e.occurred_at)
```

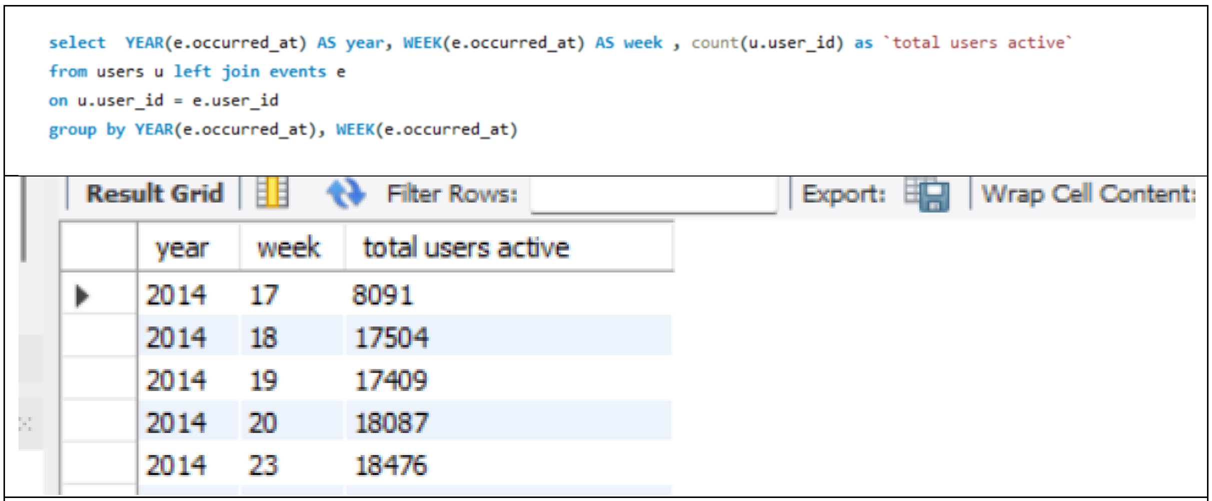

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	device	year	week	total users active
▶	acer aspire desktop	2014	17	69
	acer aspire desktop	2014	18	299
	acer aspire desktop	2014	19	242
	acer aspire desktop	2014	20	228
	acer aspire desktop	2014	21	331

Result 65

The growth of usage over each week is as follows

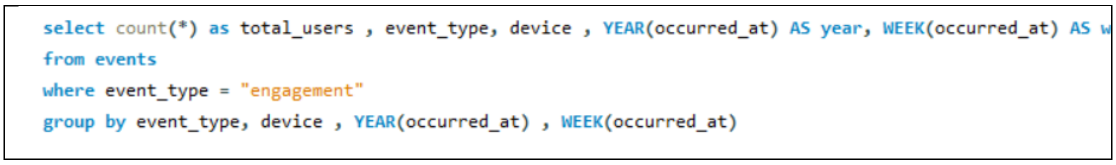
C. **Weekly Retention Analysis:**
- Objective: Analyze the retention of users on a weekly basis after signing up for a product.
- Your Task: Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

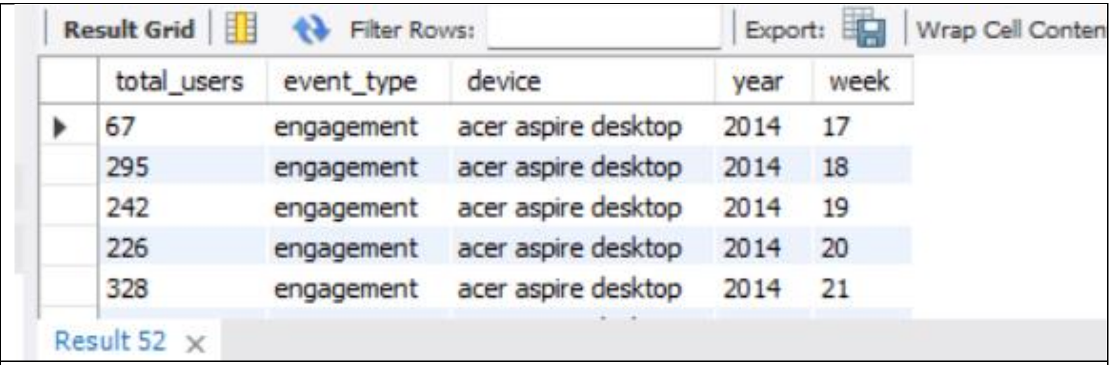

```
select  YEAR(e.occurred_at) AS year, WEEK(e.occurred_at) AS week , count(u.user_id) as `total users active`
from users u left join events e
on u.user_id = e.user_id
group by YEAR(e.occurred_at), WEEK(e.occurred_at)
```

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

	year	week	total users active
▶	2014	17	8091
	2014	18	17504
	2014	19	17409
	2014	20	18087
	2014	23	18476

D. **Weekly Engagement Per Device:**
- Objective: Measure the activeness of users on a weekly basis per device.
- Your Task: Write an SQL query to calculate the weekly engagement per device.


```
select count(*) as total_users , event_type, device , YEAR(occurred_at) AS year, WEEK(occurred_at) AS w
from events
where event_type = "engagement"
group by event_type, device , YEAR(occurred_at) , WEEK(occurred_at)
```

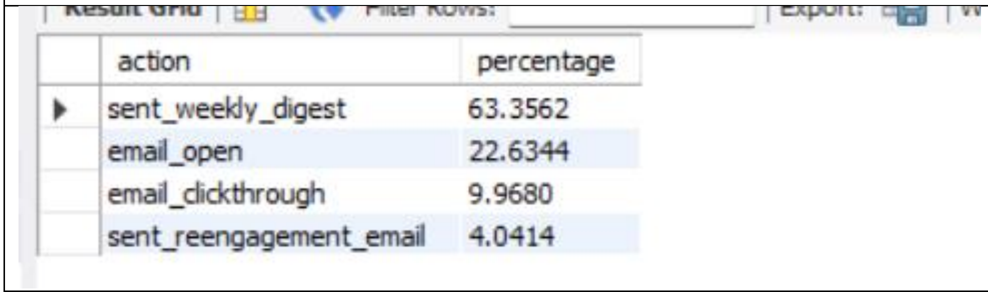


total_users	event_type	device	year	week
67	engagement	acer aspire desktop	2014	17
295	engagement	acer aspire desktop	2014	18
242	engagement	acer aspire desktop	2014	19
226	engagement	acer aspire desktop	2014	20
328	engagement	acer aspire desktop	2014	21

The weekly engagement per device is as follows.

E. **Email Engagement Analysis:**
   - Objective: Analyze how users are engaging with the email service.
   - Your Task: Write an SQL query to calculate the email engagement metrics.

```
select action, ((count(user_id) / (select count(*) from emailevents)) *100) as percentage
from emailevents
group by action;
```



action	percentage
sent_weekly_digest	63.3562
email_open	22.6344
email_clickthrough	9.9680
sent_reengagement_email	4.0414

22 percent users open email, approx. 10 percent click the email and 63 percent are subscribed to weekly digerst

Please note that for each task, you should also provide insights and interpretations of the results obtained from your queries.