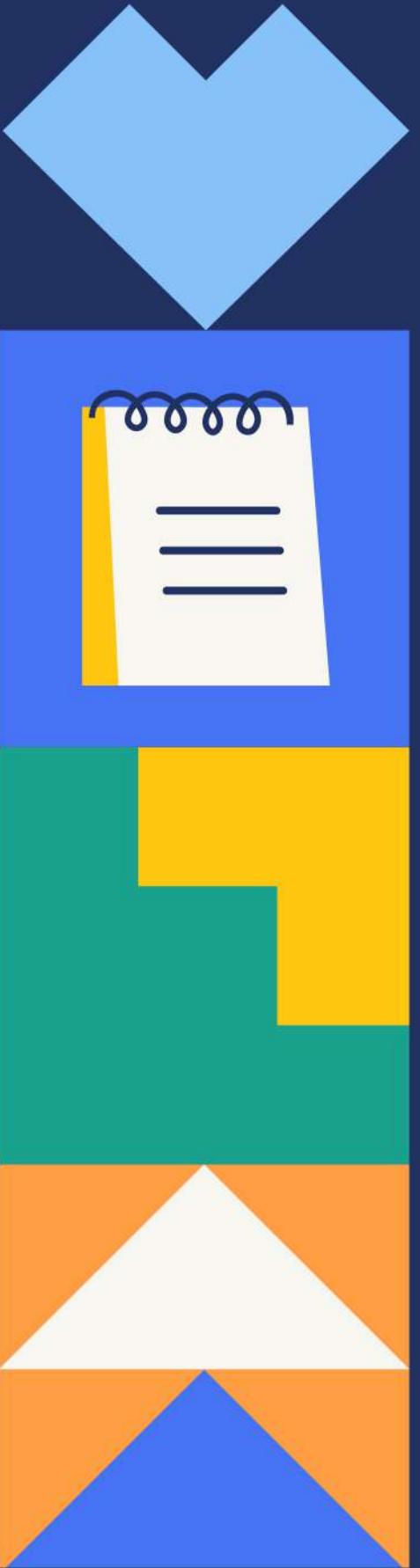


Operation Analytics and Investigating Metric Spike

By Mukesh Dey





Contents



Job Data Analysis

- Jobs Reviewed Over Time
- Throughput Analysis
- Language Share Analysis
- Duplicate Rows Detection



Investigating Metric Spike

- Weekly User Engagement
- User Growth Analysis
- Weekly Retention Analysis
- Weekly Engagement Per Device
- Email Engagement Analysis

Overview

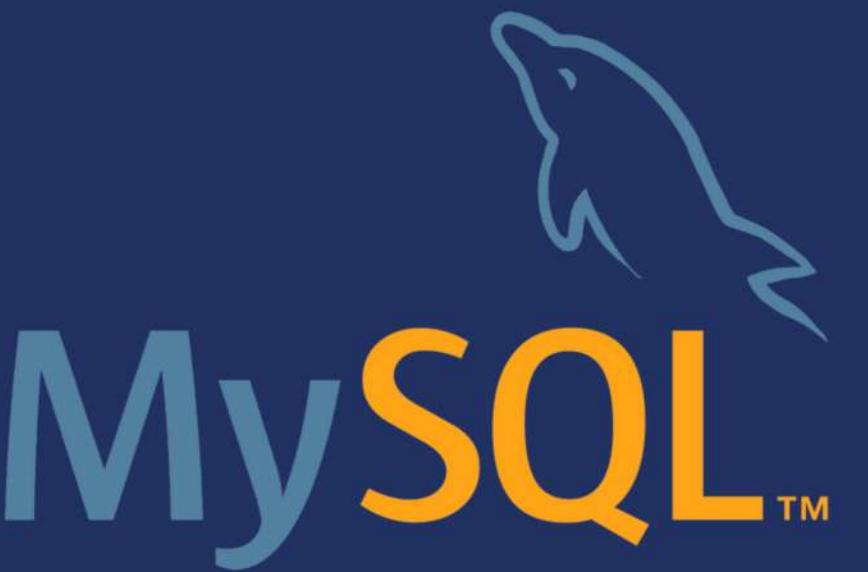
- Operational Analytics is a crucial process that involves analyzing a company's end-to-end operations. This analysis helps identify areas for improvement within the company. As a Data Analyst, I'll be working closely with various teams, such as operations, support, and marketing, helping them derive valuable insights from the data they collect.
- Here, my goal is to use advance 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.



Tech-Stack Used



Microsoft Excel



MySQL



Microsoft
Powerpoint

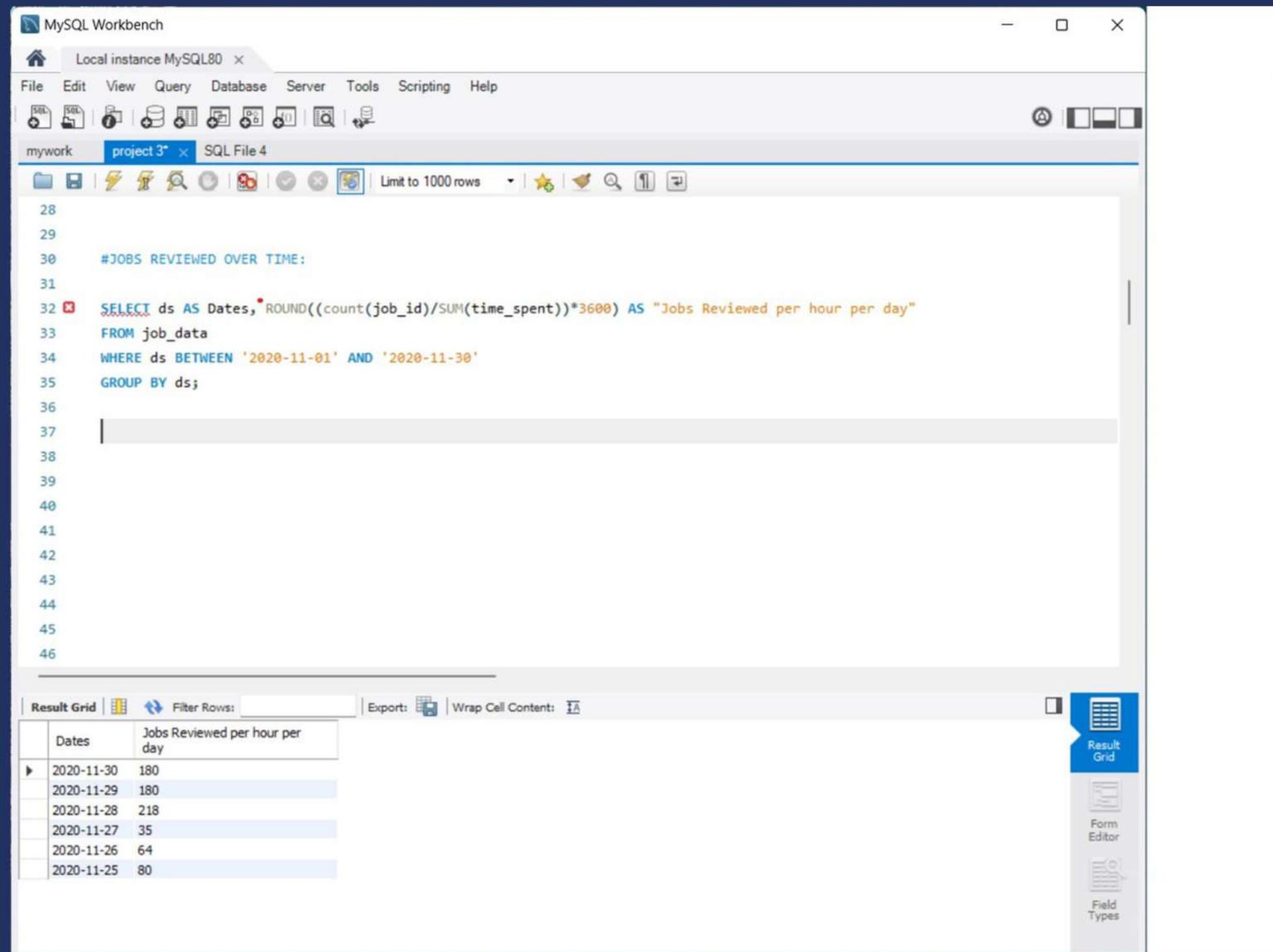


Job Data Analysis



- Jobs Reviewed Over Time

Here, my objective is to calculate the number of jobs reviewed per hour for each day in November 2020.



The screenshot shows the MySQL Workbench interface with a query editor window. The query is:

```
28
29
30    #JOBS REVIEWED OVER TIME:
31
32  SELECT ds AS Dates, ROUND((count(job_id)/SUM(time_spent))*3600) AS "Jobs Reviewed per hour per day"
33  FROM job_data
34  WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'
35  GROUP BY ds;
36
37
38
39
40
41
42
43
44
45
46
```

The result grid shows the following data:

| Dates | Jobs Reviewed per hour per day |
|------------|--------------------------------|
| 2020-11-30 | 180 |
| 2020-11-29 | 180 |
| 2020-11-28 | 218 |
| 2020-11-27 | 35 |
| 2020-11-26 | 64 |
| 2020-11-25 | 80 |



Insights

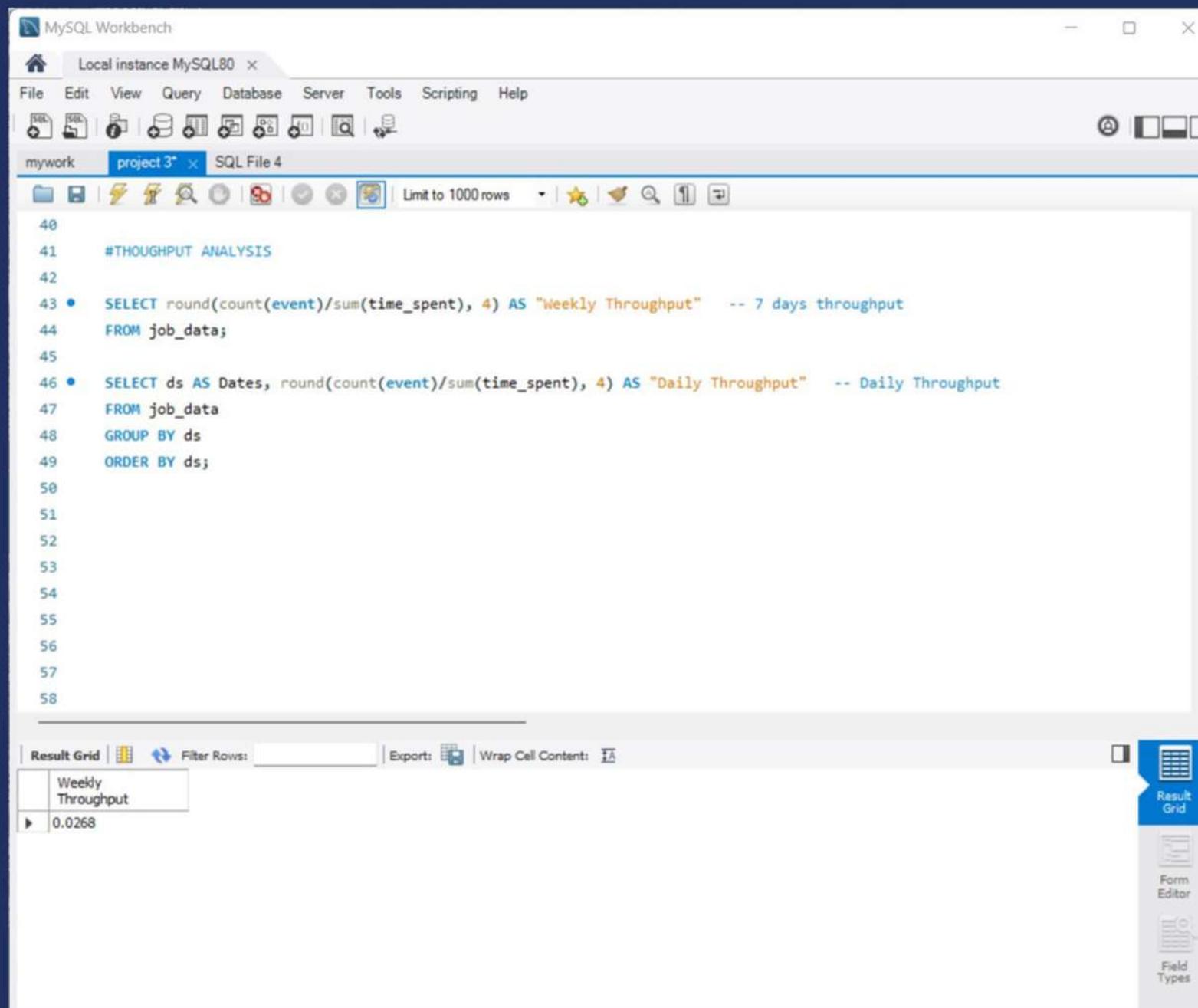
After analyzing the data it can be found that jobs reviewed per hour per day for the month of November 2020.

SQL Query used to find out the given output is mentioned in the previous slide.

| Dates | Jobs Reviewed Per Hour |
|------------|------------------------|
| 2020-11-30 | 180 |
| 2020-11-29 | 180 |
| 2020-11-28 | 218 |
| 2020-11-27 | 35 |
| 2020-11-26 | 64 |
| 2020-11-25 | 80 |

- Throughput Analysis

Here, my objective is to calculate the 7-day Rolling average of throughput analysis(number events per seconds)

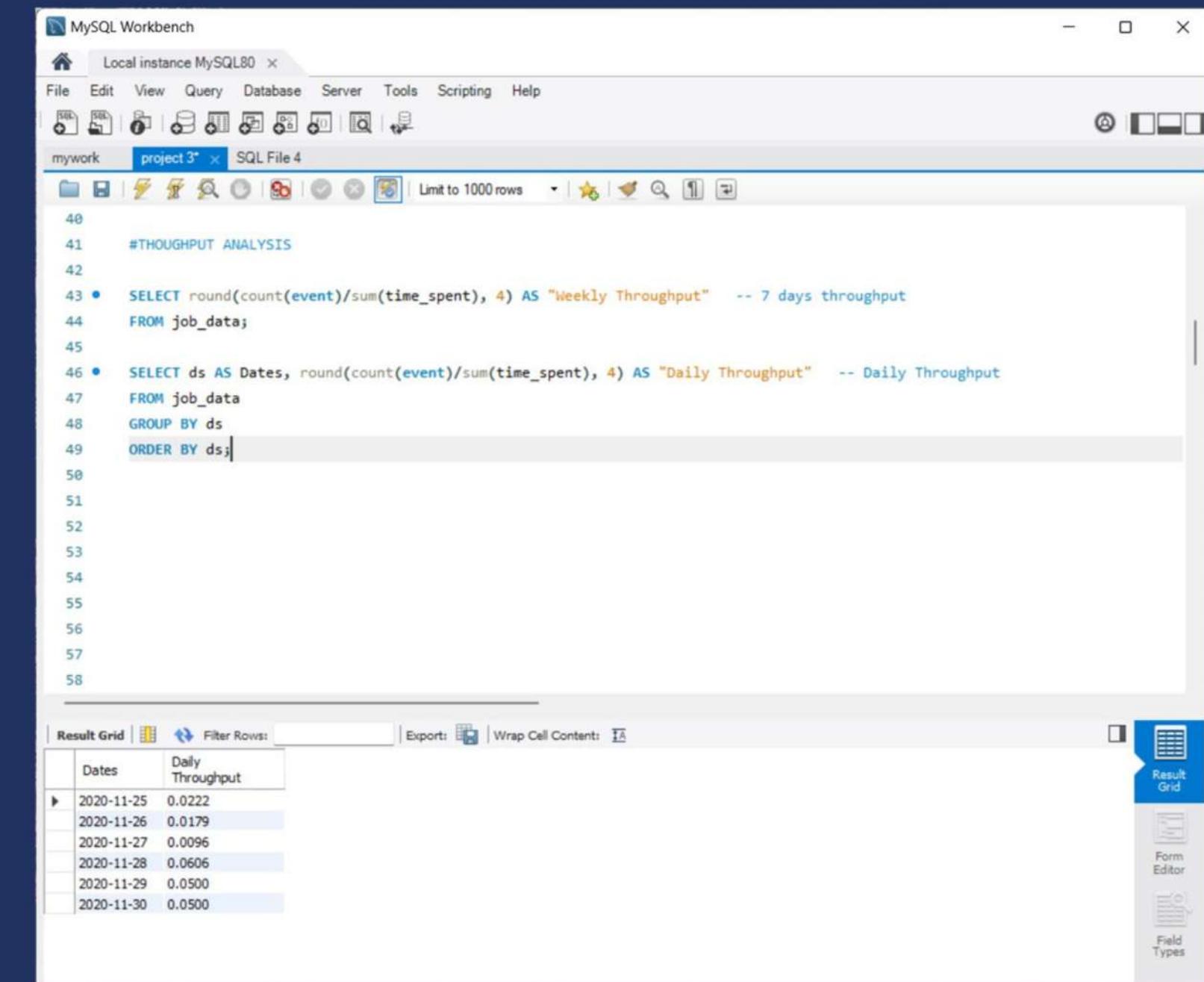


MySQL Workbench window showing SQL code for throughput analysis:

```
40
41  #THROUGHPUT ANALYSIS
42
43 • SELECT round(count(event)/sum(time_spent), 4) AS "Weekly Throughput" -- 7 days throughput
44   FROM job_data;
45
46 • SELECT ds AS Dates, round(count(event)/sum(time_spent), 4) AS "Daily Throughput" -- Daily Throughput
47   FROM job_data
48   GROUP BY ds
49   ORDER BY ds;
50
51
52
53
54
55
56
57
58
```

The Result Grid shows:

| Weekly Throughput |
|-------------------|
| 0.0268 |



MySQL Workbench window showing SQL code for throughput analysis:

```
40
41  #THROUGHPUT ANALYSIS
42
43 • SELECT round(count(event)/sum(time_spent), 4) AS "Weekly Throughput" -- 7 days throughput
44   FROM job_data;
45
46 • SELECT ds AS Dates, round(count(event)/sum(time_spent), 4) AS "Daily Throughput" -- Daily Throughput
47   FROM job_data
48   GROUP BY ds
49   ORDER BY ds;
50
51
52
53
54
55
56
57
58
```

The Result Grid shows:

| Dates | Daily Throughput |
|------------|------------------|
| 2020-11-25 | 0.0222 |
| 2020-11-26 | 0.0179 |
| 2020-11-27 | 0.0096 |
| 2020-11-28 | 0.0606 |
| 2020-11-29 | 0.0500 |
| 2020-11-30 | 0.0500 |



Insights

After analyzing the data it can be found that weekly throughput analysis is **0.0268**.

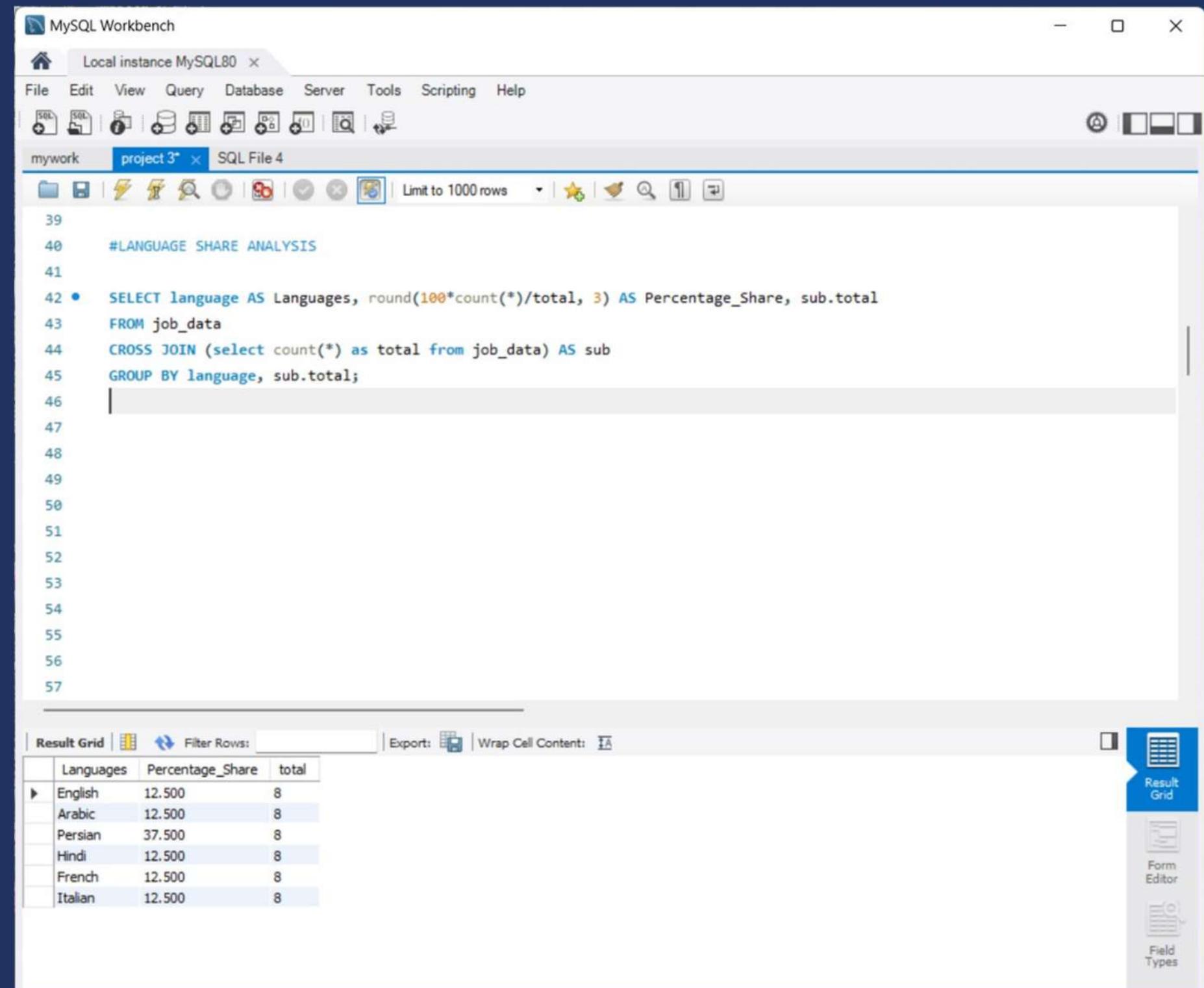
And if we see the daily throughout analysis the value comes as shown:

SQL Query used to find out the given output is mentioned in the previous slide.

| Dates | Daily Throughout |
|------------|------------------|
| 2020-11-25 | 0.0222 |
| 2020-11-26 | 0.0179 |
| 2020-11-27 | 0.0096 |
| 2020-11-28 | 0.0606 |
| 2020-11-29 | 0.0500 |
| 2020-11-30 | 0.0500 |

- Language Share Analysis

Here, my objective is to calculate the percentage share of each language in the last 30 days.



The screenshot shows the MySQL Workbench interface. The main window displays a SQL query for calculating language share:

```
39
40  #LANGUAGE SHARE ANALYSIS
41
42 •  SELECT language AS Languages, round(100*count(*)/total, 3) AS Percentage_Share, sub.total
43  FROM job_data
44  CROSS JOIN (select count(*) as total from job_data) AS sub
45  GROUP BY language, sub.total;
46
47
48
49
50
51
52
53
54
55
56
57
```

The results grid below the query shows the following data:

| Languages | Percentage_Share | total |
|-----------|------------------|-------|
| English | 12.500 | 8 |
| Arabic | 12.500 | 8 |
| Persian | 37.500 | 8 |
| Hindi | 12.500 | 8 |
| French | 12.500 | 8 |
| Italian | 12.500 | 8 |



Insights

The percentage share of each language over the last 30 days is given below:

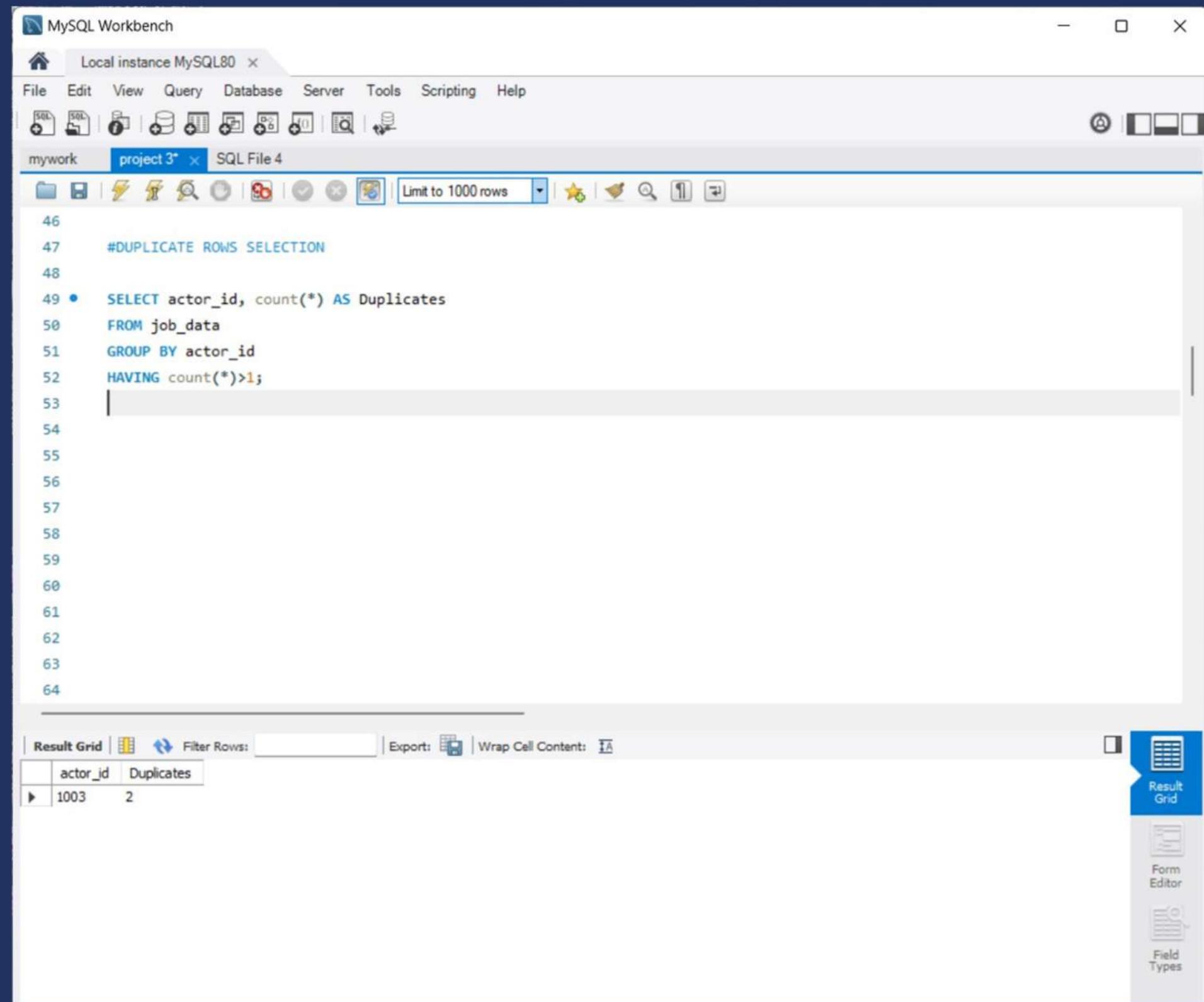
| Languages | Percentage_Share | Total |
|-----------|------------------|-------|
| English | 12.500 | 8 |
| Arabic | 12.500 | 8 |
| Persian | 37.500 | 8 |
| Hindi | 12.500 | 8 |
| French | 12.500 | 8 |
| Italian | 12.500 | 8 |

SQL Query used to find out the given output is mentioned in the previous slide.



- Duplicate Rows Detection

Here, my objective is to identify duplicate rows in the data.



The screenshot shows the MySQL Workbench interface. The title bar reads "MySQL Workbench" and "Local instance MySQL80". The menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The toolbar below the menu has various icons for database management. The main window contains a SQL editor tab with the following code:

```
46
47  #DUPLICATE ROWS SELECTION
48
49 •  SELECT actor_id, count(*) AS Duplicates
50  FROM job_data
51  GROUP BY actor_id
52  HAVING count(*)>1;
53
54
55
56
57
58
59
60
61
62
63
64
```

The result grid below the editor shows the output of the query:

| actor_id | Duplicates |
|----------|------------|
| 1003 | 2 |

A vertical sidebar on the right lists "Result Grid", "Form Editor", and "Field Types".



Insights

After analyzing the data it can be found that there are only **2 duplicate rows with the actor_id as "1003"**

SQL Query used to find out the given output is mentioned in the previous slide.



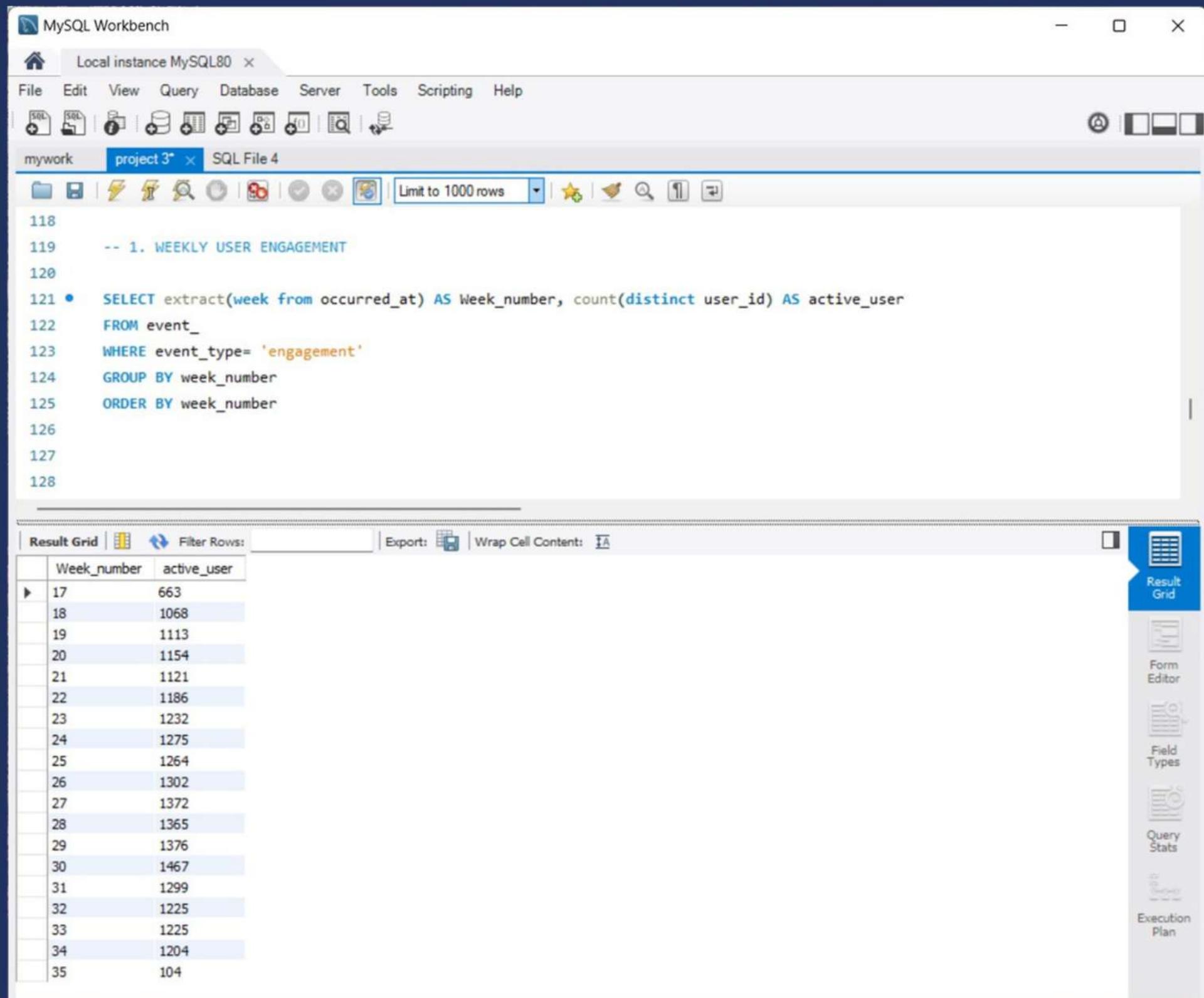


Investigation Metric Spikes



- Weekly User Engagement

Here, my objective is to measure the activeness of users on a weekly basis.



The screenshot shows the MySQL Workbench interface. The title bar says "MySQL Workbench" and "Local instance MySQL80". The menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, Help. The toolbar has various icons for database management. The query editor tab is "project 3*" and the SQL file tab is "SQL File 4". The code in the editor is:

```
118
119 -- 1. WEEKLY USER ENGAGEMENT
120
121 • SELECT extract(week from occurred_at) AS Week_number, count(distinct user_id) AS active_user
122 FROM event_
123 WHERE event_type= 'engagement'
124 GROUP BY week_number
125 ORDER BY week_number
126
127
128
```

The results grid shows the following data:

| Week_number | active_user |
|-------------|-------------|
| 17 | 663 |
| 18 | 1068 |
| 19 | 1113 |
| 20 | 1154 |
| 21 | 1121 |
| 22 | 1186 |
| 23 | 1232 |
| 24 | 1275 |
| 25 | 1264 |
| 26 | 1302 |
| 27 | 1372 |
| 28 | 1365 |
| 29 | 1376 |
| 30 | 1467 |
| 31 | 1299 |
| 32 | 1225 |
| 33 | 1225 |
| 34 | 1204 |
| 35 | 104 |

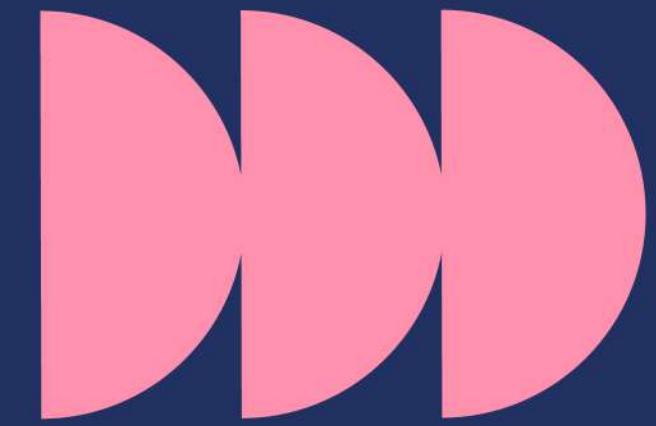
The right sidebar shows icons for Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan.



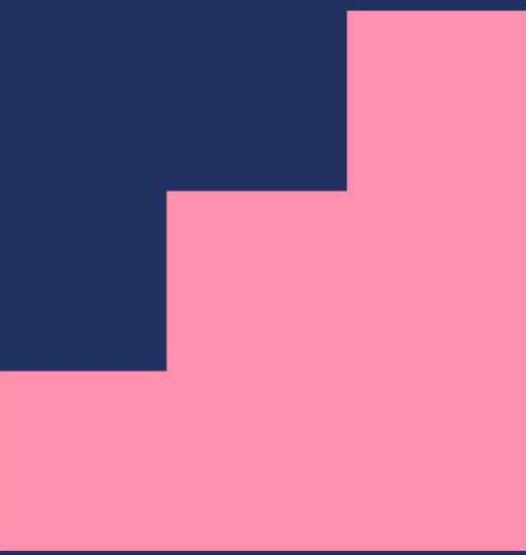
Insights

Week number and no. of active users according to that are mentioned below:

| Week_number | Active_user |
|-------------|-------------|
| 17 | 663 |
| 18 | 1068 |
| 19 | 1113 |
| 20 | 1154 |
| 21 | 1121 |
| 22 | 1186 |



**SQL Query used
to find out the
given output is
mentioned in the
previous slide.**

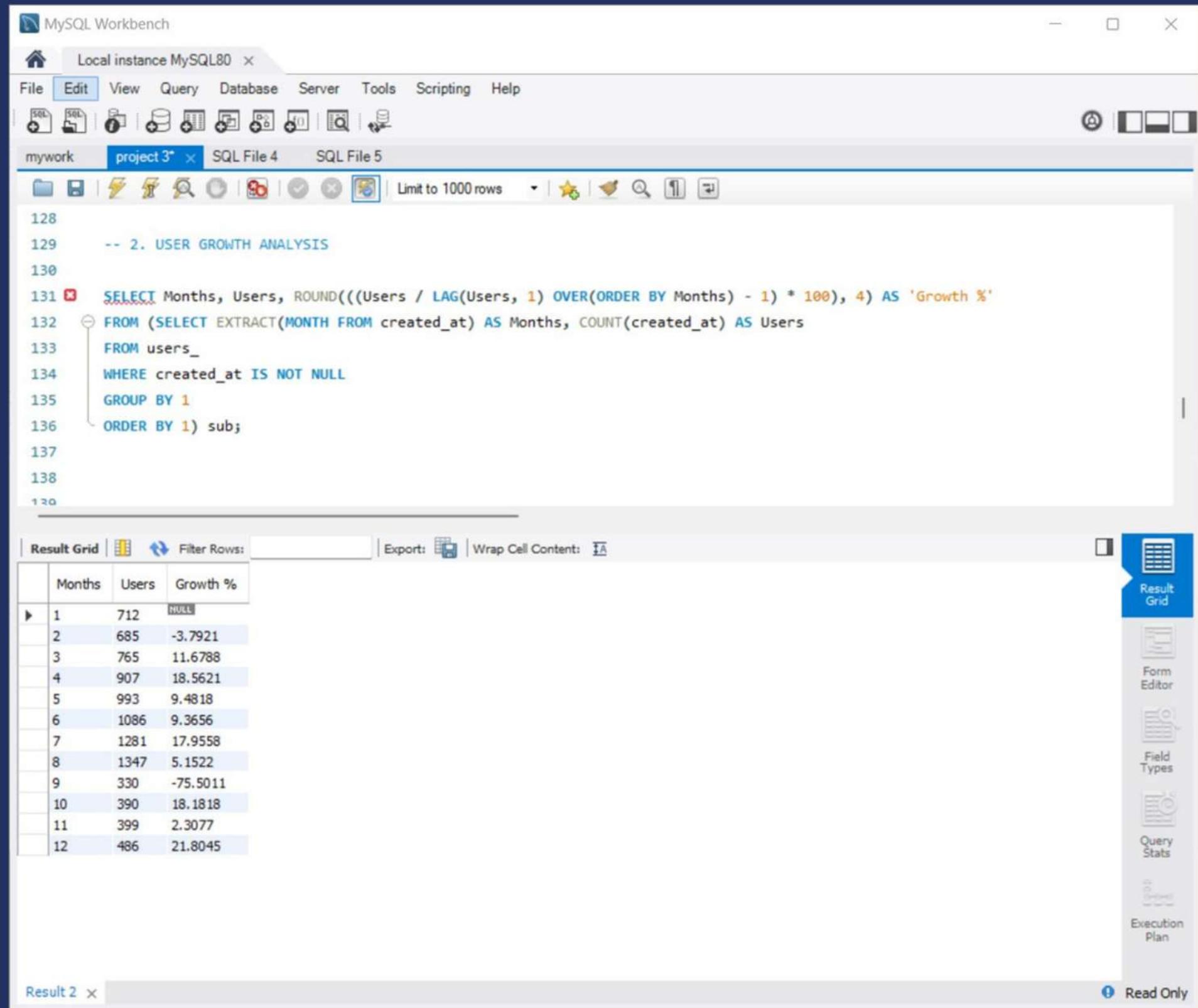


Insights

| Week_number | Active_user | Week_number | Active_user |
|-------------|-------------|-------------|-------------|
| 23 | 1232 | 30 | 1467 |
| 24 | 1275 | 31 | 1299 |
| 25 | 1264 | 32 | 1225 |
| 26 | 1302 | 33 | 1225 |
| 27 | 1372 | 34 | 1204 |
| 28 | 1365 | 35 | 104 |
| 29 | 1376 | | |

- User Growth Analysis:

Here, my objective is to analyze the growth of users over time for a product.



The screenshot shows the MySQL Workbench interface. The main window displays a SQL query for "User Growth Analysis". The results grid shows the output of the query, which includes columns for Months, Users, and Growth %.

```
128
129 -- 2. USER GROWTH ANALYSIS
130
131   SELECT Months, Users, ROUND(((Users / LAG(Users, 1) OVER(ORDER BY Months) - 1) * 100), 4) AS 'Growth %'
132   FROM (SELECT EXTRACT(MONTH FROM created_at) AS Months, COUNT(created_at) AS Users
133     FROM users_
134    WHERE created_at IS NOT NULL
135   GROUP BY 1
136   ORDER BY 1) sub;
137
138
139
```

| Months | Users | Growth % |
|--------|-------|----------|
| 1 | 712 | NULL |
| 2 | 685 | -3.7921 |
| 3 | 765 | 11.6788 |
| 4 | 907 | 18.5621 |
| 5 | 993 | 9.4818 |
| 6 | 1086 | 9.3656 |
| 7 | 1281 | 17.9558 |
| 8 | 1347 | 5.1522 |
| 9 | 330 | -75.5011 |
| 10 | 390 | 18.1818 |
| 11 | 399 | 2.3077 |
| 12 | 486 | 21.8045 |



Insights

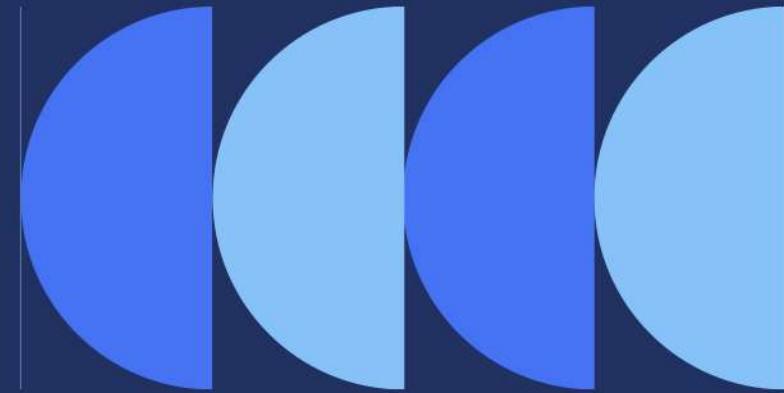
User growth is given according to the month from Jan-Dec according to the growth %. According, to that we can check in which month the growth was more or less.

SQL Query used to find out the given output is mentioned in the previous slide.



| Months | Users | Growth % |
|--------|-------|----------|
| 1 | 712 | NULL |
| 2 | 685 | -3.7921 |
| 3 | 765 | 11.6788 |
| 4 | 907 | 18.5621 |
| 5 | 993 | 9.4818 |
| 6 | 1086 | 9.3656 |
| 7 | 1281 | 17.9558 |
| 8 | 1347 | 5.1522 |

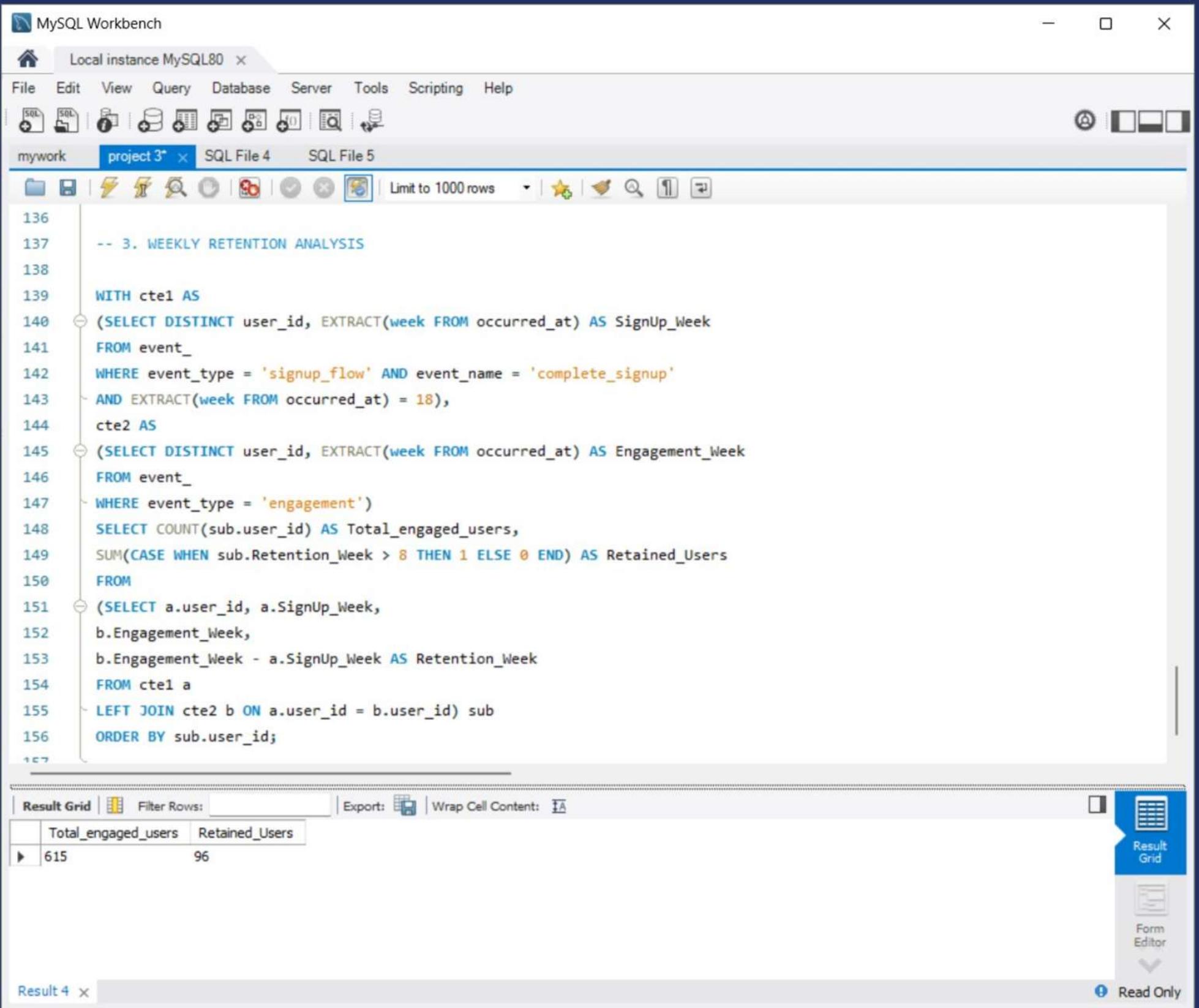
Insights



| Months | Users | Growth % |
|--------|-------|-----------|
| 9 | 330 | -.75.5011 |
| 10 | 390 | 18.1818 |
| 11 | 399 | 2.3077 |
| 12 | 486 | 21.8045 |

- Weekly Retention Analysis:

Here, my objective is to measure the activeness of users on a weekly basis per device.



The screenshot shows the MySQL Workbench application window. The title bar reads "MySQL Workbench" and "Local instance MySQL80". The menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. Below the menu is a toolbar with various icons. The main area contains a SQL editor tab labeled "mywork" with the following query:

```
136
137 -- 3. WEEKLY RETENTION ANALYSIS
138
139 WITH cte1 AS
140 (SELECT DISTINCT user_id, EXTRACT(week FROM occurred_at) AS SignUp_Week
141 FROM event_
142 WHERE event_type = 'signup_flow' AND event_name = 'complete_signup'
143 AND EXTRACT(week FROM occurred_at) = 18),
144 cte2 AS
145 (SELECT DISTINCT user_id, EXTRACT(week FROM occurred_at) AS Engagement_Week
146 FROM event_
147 WHERE event_type = 'engagement')
148 SELECT COUNT(sub.user_id) AS Total_engaged_users,
149 SUM(CASE WHEN sub.Retention_Week > 8 THEN 1 ELSE 0 END) AS Retained_Users
150
151 (SELECT a.user_id, a.SignUp_Week,
152 b.Engagement_Week,
153 b.Engagement_Week - a.SignUp_Week AS Retention_Week
154 FROM cte1 a
155 LEFT JOIN cte2 b ON a.user_id = b.user_id) sub
156 ORDER BY sub.user_id;
```

Below the editor is a "Result Grid" pane showing the results of the query:

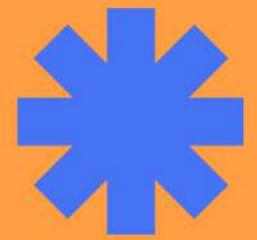
| Total_engaged_users | Retained_Users |
|---------------------|----------------|
| 615 | 96 |

The bottom right corner of the window features a large yellow circular logo with a stylized 'X' or 'A' shape inside.

Insights

SQL Query which is used to analyze the data is given in the previous slide

The number of users calculated after analyzing the weekly data of users is given below:



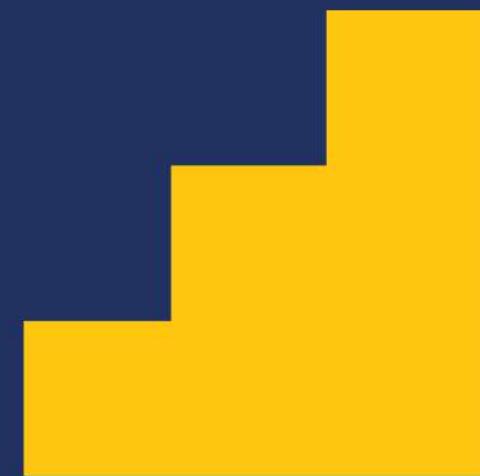
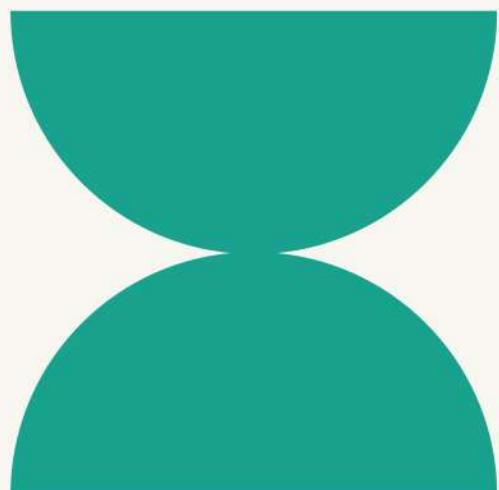
Total engaged users

The total no. of users which where engaged weekly after signing up for a product is
615



Retained users

The no. of users which were retained weekly based on the sign-up cohort is 96.



- Weekly Engagement Per Device:

Here, my objective is to measure the activeness of users on a weekly basis per device.

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

mywork* project 3* x SQL File 4 SQL File 5

163 -- 4 WEEKLY ENGAGEMENT PER DEVICE

164 • WITH cte AS

165 (SELECT EXTRACT(YEAR FROM occurred_at) || '-' || EXTRACT(WEEK FROM occurred_at) AS Week_number, device,

166 COUNT(DISTINCT user_id) AS user_count FROM event_ WHERE event_type = 'engagement' GROUP BY Week_number, device)

167 SELECT Week_number, device, user_count FROM cte

168 ORDER BY Week_numbers;

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

| Week_number | device | user_count |
|-------------|------------------------|------------|
| 1 | acer aspire desktop | 198 |
| 1 | acer aspire notebook | 338 |
| 1 | amazon fire phone | 89 |
| 1 | asus chromebook | 355 |
| 1 | dell inspiron desktop | 360 |
| 1 | dell inspiron notebook | 677 |
| 1 | hp pavilion desktop | 339 |
| 1 | htc one | 196 |
| 1 | ipad air | 478 |
| 1 | ipad mini | 292 |
| 1 | iphone 4s | 409 |
| 1 | iphone 5 | 1025 |
| 1 | iphone 5s | 626 |
| 1 | kindle fire | 205 |
| 1 | lenovo thinkpad | 1309 |
| 1 | mac mini | 150 |
| 1 | macbook air | 950 |
| 1 | macbook pro | 1952 |
| 1 | nexus 10 | 273 |
| 1 | nexus 5 | 621 |
| 1 | nexus 7 | 355 |
| 1 | nokia lumia 635 | 211 |
| 1 | samsung galaxy tablet | 107 |
| 1 | samsung galaxy note | 119 |
| 1 | samsung galaxy s4 | 803 |
| 1 | windows surface | 182 |

Result 1 x



Insights

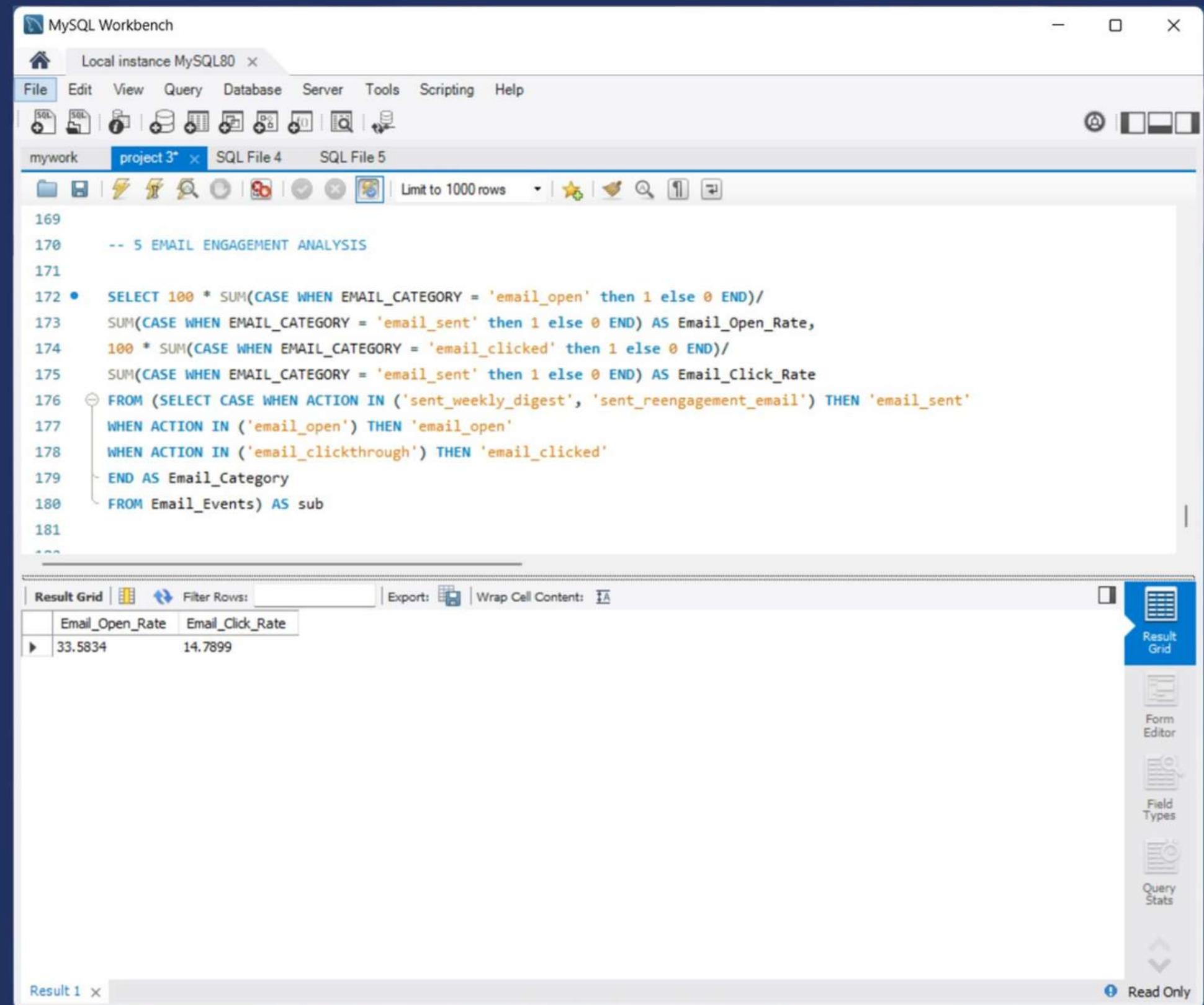
The activeness of users on weekly basis per device is found after analyzing the data from the table. And according to that the name of the device and no. of active users are mentioned below:

SQL query to analyze the find the result for the following objective is given in the previous slide.

Acer aspire notebook-338 ; Amazon fire phone- 89 ; Asus chromebook-355 ; Dell inspiron desktop- 360; Dell inspiron notebook- 677 ; Hp pavilion desktop- 339 ; Htc pne- 196 ; Ipad air- 478 ; Ipad mini- 292 ; Iphone 4s- 409 ; Iphone 5- 1025 ; Kindle fire- 205 ; Lenovo thinkpad- 1309 ; Mac mini-150 ; Macbook air- 950 ; Macbook pro- 1952 ; Nexus 10- 273 ; Nexus 5- 621 ; Nexus 7- 355 ; Nokia lumia 635- 211 ; Samsung galaxy tablet- 107 ; Samsung galaxy note- 119 ; Samsung galaxy s4- 803 ; Window surface-182.

- Email Engagement Analysis:

Here, my objective is to analyze how users are engaged with the email service.



The screenshot shows the MySQL Workbench interface. The main window displays an SQL query for "Email Engagement Analysis". The query uses a CASE WHEN construct to calculate the percentage of emails sent that were opened or clicked. It also includes a subquery to map specific actions to email categories. The results are shown in a grid with two rows: one for the overall statistics and another for the specific category being analyzed.

```
MySQL Workbench - Local instance MySQL80 - project 3* - SQL File 4
```

```
169
170    -- 5 EMAIL ENGAGEMENT ANALYSIS
171
172 •  SELECT 100 * SUM(CASE WHEN EMAIL_CATEGORY = 'email_open' then 1 else 0 END) /
173     SUM(CASE WHEN EMAIL_CATEGORY = 'email_sent' then 1 else 0 END) AS Email_Open_Rate,
174     100 * SUM(CASE WHEN EMAIL_CATEGORY = 'email_clicked' then 1 else 0 END) /
175     SUM(CASE WHEN EMAIL_CATEGORY = 'email_sent' then 1 else 0 END) AS Email_Click_Rate
176   FROM (SELECT CASE WHEN ACTION IN ('sent_weekly_digest', 'sent_reengagement_email') THEN 'email_sent'
177                  WHEN ACTION IN ('email_open') THEN 'email_open'
178                  WHEN ACTION IN ('email_clickthrough') THEN 'email_clicked'
179                  END AS Email_Category
180   FROM Email_Events) AS sub
181
```

| Email_Open_Rate | Email_Click_Rate |
|-----------------|------------------|
| 33.5834 | 14.7899 |

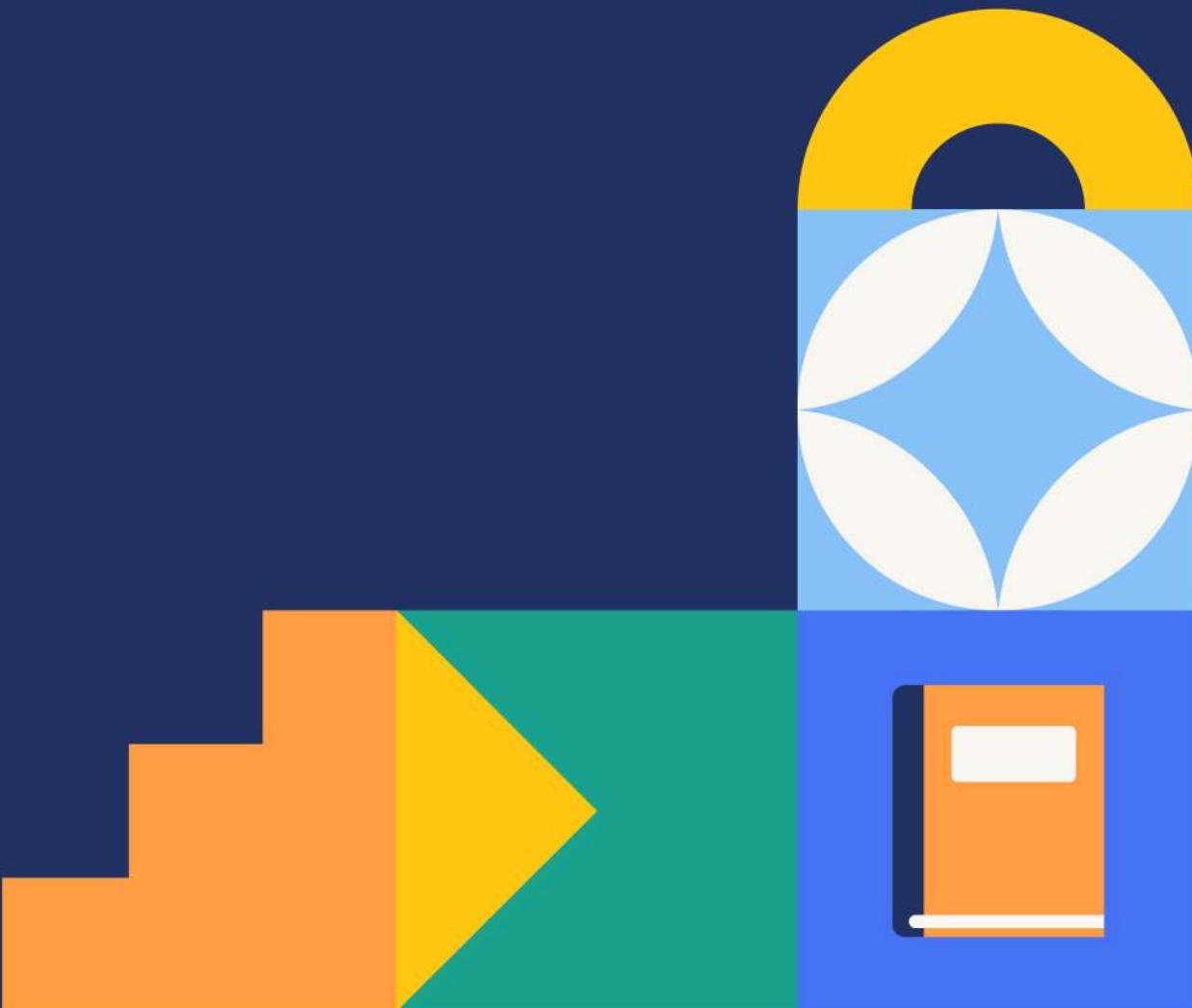
Result Grid | Filter Rows: _____ | Export: | Wrap Cell Content: | Result Grid | Form Editor | Field Types | Query Stats | Read Only

Insights

After analyzing the data from the table Email_events table the activeness of users how they were engaged with email service has been found as follows:

- Email Open Rate :- 33.5834
- Email Click Rate :- 14.7899

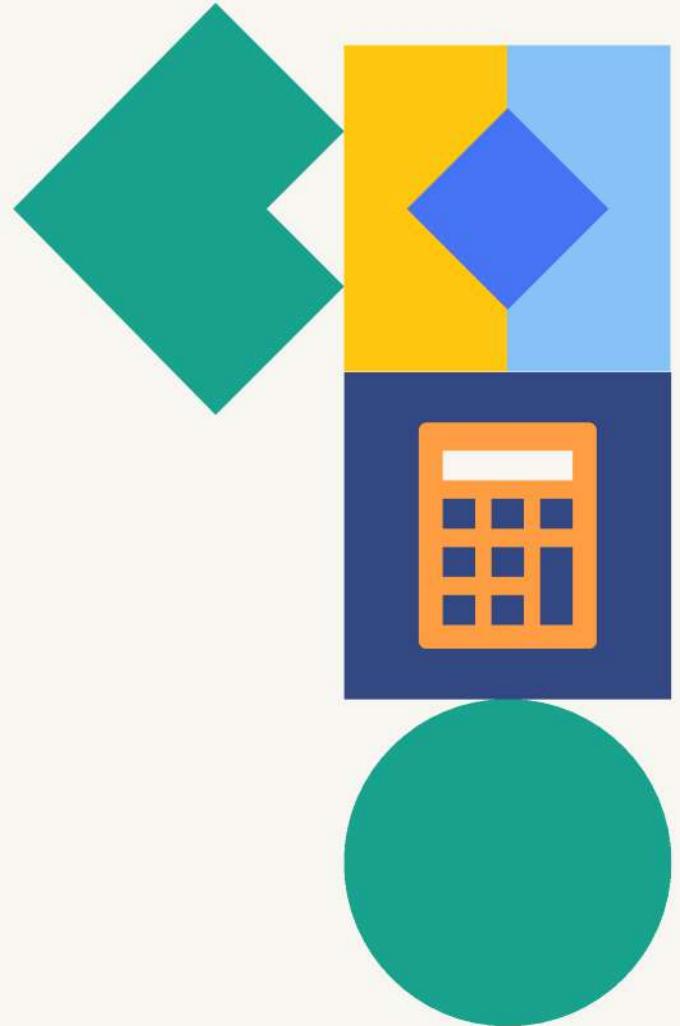
SQL query used to analyze the following objective is mentioned in the previous slide



Result

After analyzing the data from the information given into tables, various analysis has been made based upon which it will be easy to find the data which company wants for their employees as well as their product.

And by analyzing the data I'm able to learn advanced SQL and importing data directly from MS Excel to MySQL.





Thank you

