

PROJECT DESCRIPTION:

This project deals with the process of analyzing a company's end to end operations which helps identify areas for improvement within the company. This involved understanding and explaining sudden change in key metrics, such as a dip in daily user engagement. Here, I'm supposed to work as a lead data analyst in a company like Microsoft. This project involves two tasks having two case studies where analysis is to be done .

APPROACH:

I have used MYSQL Workbench 8.0 CE for the database creation and finding the answers for the tasks using the SQL queries in an appropriate way to get the best answers for the questions asked. I have did separate analysis for both the test cases using various SQL queries to get analysis done for the various tasks assigned. The dataset provided has been involved using the 'load data infile' command of SQL.

TECH-STACK USED:

I have used SQL WORKBENCH and various SQL DML and other SQL commands to get the desired output . I have used the 8.0 CE version of the workbench. Also , I used excel to check for any empty data in tn the datasheet provided to prevent any issues while running the SQL commands.

INSIGHTS:

Here, I will be providing insights about the data regarding the tasks assigned along with the SQL query I used as well as outputs I received for those queries.

CASE STUDY 1:

SQL QUERY for job_data table creation

CREATE TABLE job_data (
ds DATE,

job_id INT NOT NULL,
actor_id INT NOT NULL,
event VARCHAR(10) NOT NULL,
language VARCHAR(10) NOT NULL,
time_spent INT NOT NULL,
org CHAR(2)

);

**TASK A: JOBS REVIEWED PER HOUR FOR EACH DAY IN NOVEMBER
2020**

SQL QUERY:

```
SELECT ds AS DATE,  
       COUNT(job_id) AS Joint_Job_Id,  
       ROUND((SUM(time_spent)/3600),2) AS Time_sp_hr,  
       ROUND((COUNT(job_id)/(SUM(time_spent)/3600)),2) AS job_review_phr_pday  
FROM job_data WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'  
GROUP BY ds ORDER BY ds;
```

SQL OUTPUT:

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

project3part1 job_data

Limit to 1000 rows

21 ('2020-11-26',23,1004,'skip','Persian',56,'A'),

22 ('2020-11-25',20,1003,'transfer','Italian',45,'C'))

23

24 # CASE STUDY 1---TASK A

25 # jobs reviewed per hour for each day in nov 2020

26 SELECT ds AS DATE,

27 COUNT(job_id) AS Joint_Job_Id,

28 ROUND((SUM(time_spent)/3600),2) AS Time_sp_hr,

29 ROUND((COUNT(job_id)/(SUM(time_spent)/3600)),2) AS job_review_phr_pday

30 FROM job_data WHERE ds BETWEEN '2020-11-01' AND '2020-11-30'

31 GROUP BY ds ORDER BY ds

32

33

Result Grid

DATE	Joint_Job_Id	Time_sp_hr	job_review_phr_pday
2020-11-25	2	0.03	80.00
2020-11-26	2	0.03	64.29
2020-11-27	2	0.06	34.62
2020-11-28	4	0.02	218.18
2020-11-29	2	0.01	180.00

Administration Schemas

Information

Schema: project3

Output

Action Output

#	Time	Action	Message	Duration / Fetch
14	19:01:49	insert into job_data (ds,job_id,actor_id,event,language,time_spent,org) values('2020/11/30',21,1001,'skip','En...	Error Code: 1292. Incorrect date value: '11/30/2020' for column 'ds' at row 2	0.359 sec
15	19:03:34	insert into job_data (ds,job_id,actor_id,event,language,time_spent,org) values('2020/11/30',21,1001,'skip','En...	8 row(s) affected, 8 warning(s): 4095 Delimiter '/' in position 4 in datetime value '2020/11/30' at row 1 is deprec...	0.296 sec
16	19:04:52	insert into job_data (ds,job_id,actor_id,event,language,time_spent,org) values('2020-11-30',21,1001,'skip','Engl...	8 row(s) affected Records: 8 Duplicates: 0 Warnings: 0	0.188 sec
17	19:08:58	SELECT * FROM project3.job_data LIMIT 0, 1000	16 row(s) returned	0.000 sec / 0.000 sec
18	19:44:35	SELECT ds AS DATE, COUNT(job_id) AS Joint_Job_Id, ROUND((SUM(time_spent)/3600),2) AS Time...	6 row(s) returned	0.219 sec / 0.000 sec

Object Info Session

76°F Mostly cloudy

Search

ENG IN

7:45 PM 12/25/2024

RESULT :

The above screenshot shows the tabulated data about the number of jobs reviewed per hour in November 2020.

TASK B : To calculate 7-day rolling average of throughput.

SQL QUERY:

SELECT ROUND(COUNT(event)/SUM(time_spent),2) AS weekly_avg_throughput

FROM job_data;

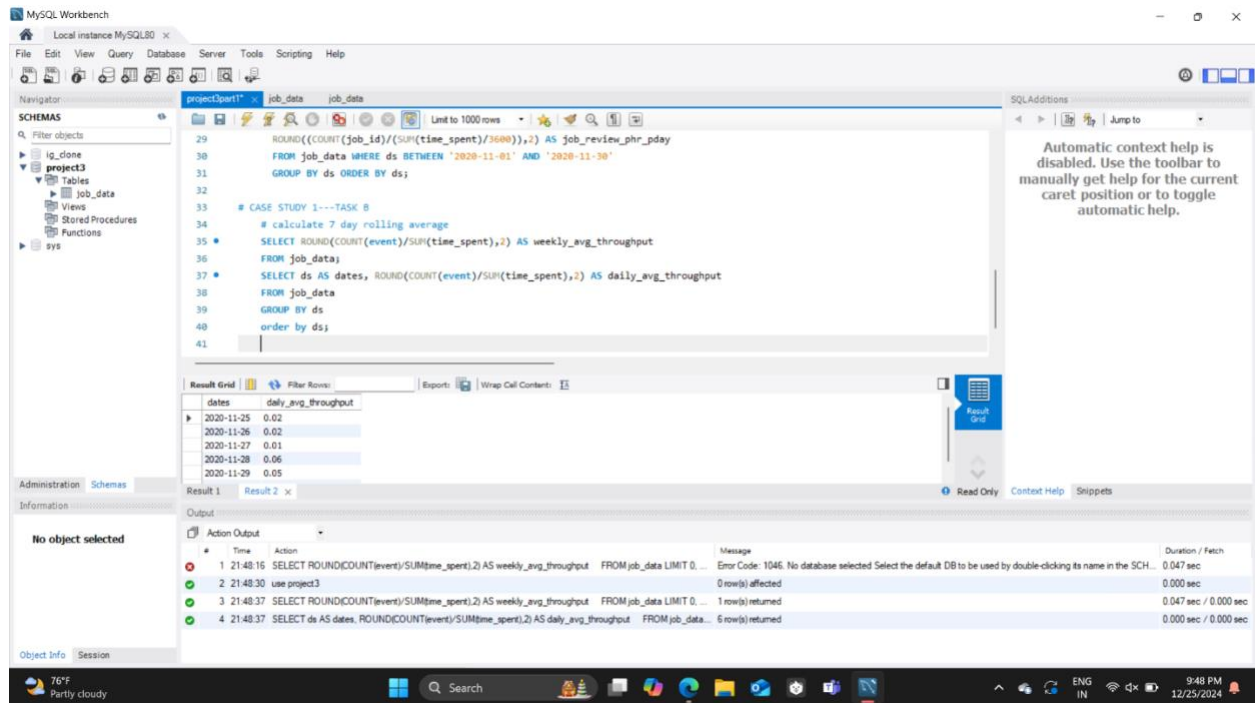
SELECT ds AS dates, ROUND(COUNT(event)/SUM(time_spent),2) AS daily_avg_throughput

FROM job_data

GROUP BY ds

order by ds;

SQL OUTPUT:



RESULT: The above output shows the daily average throughput for the seven days of the week. I would prefer the daily average throughput in order to get the exact daily average to avoid the data misconception that could have happened taking the weekend offs as consideration in calculating the weekly average throughput.

TASK C :

To calculate the percentage share of each language in the last 30 days

SQL QUERY:

SELECT language, ROUND(100*COUNT(*)/total,2) AS percentage,

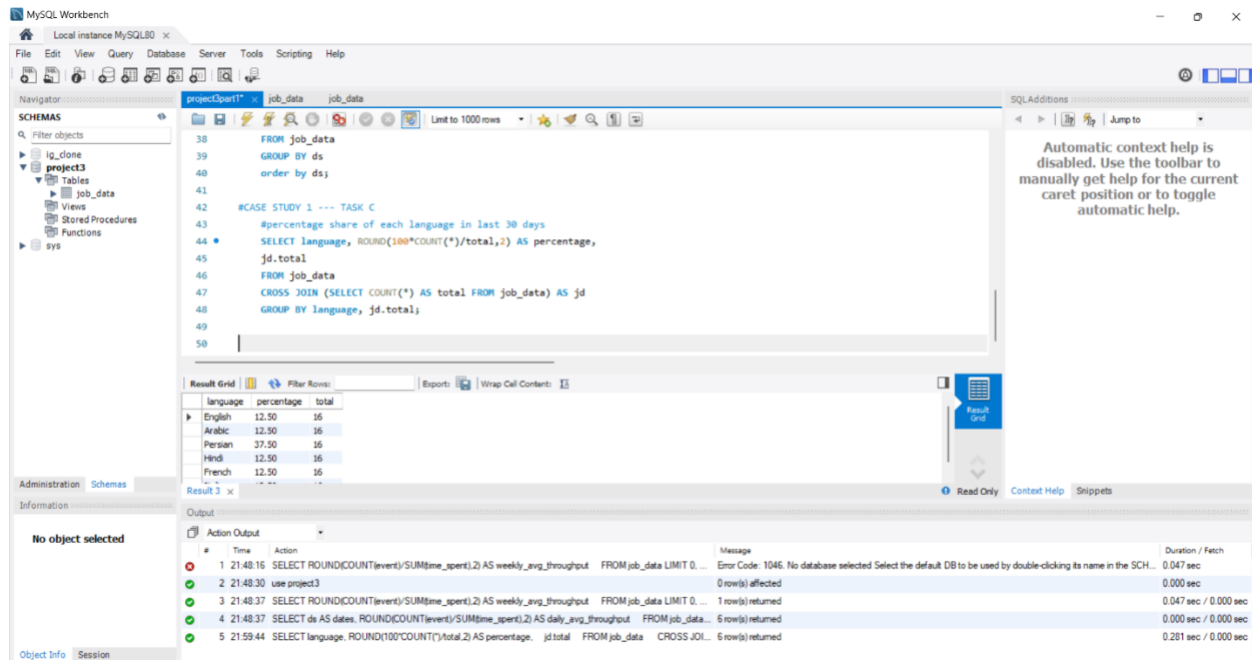
jd.total

FROM job_data

CROSS JOIN (SELECT COUNT(*) AS total FROM job_data) AS jd

GROUP BY language, jd.total;

SQL OUTPUT:



RESULT : The above screenshot shows the SQL output about the percentage share of each language such as 12.50% for ENGLISH and ARABIC and so on for other languages.

TASK D : Display duplicate rows from the job_data table:

SQL QUERY:

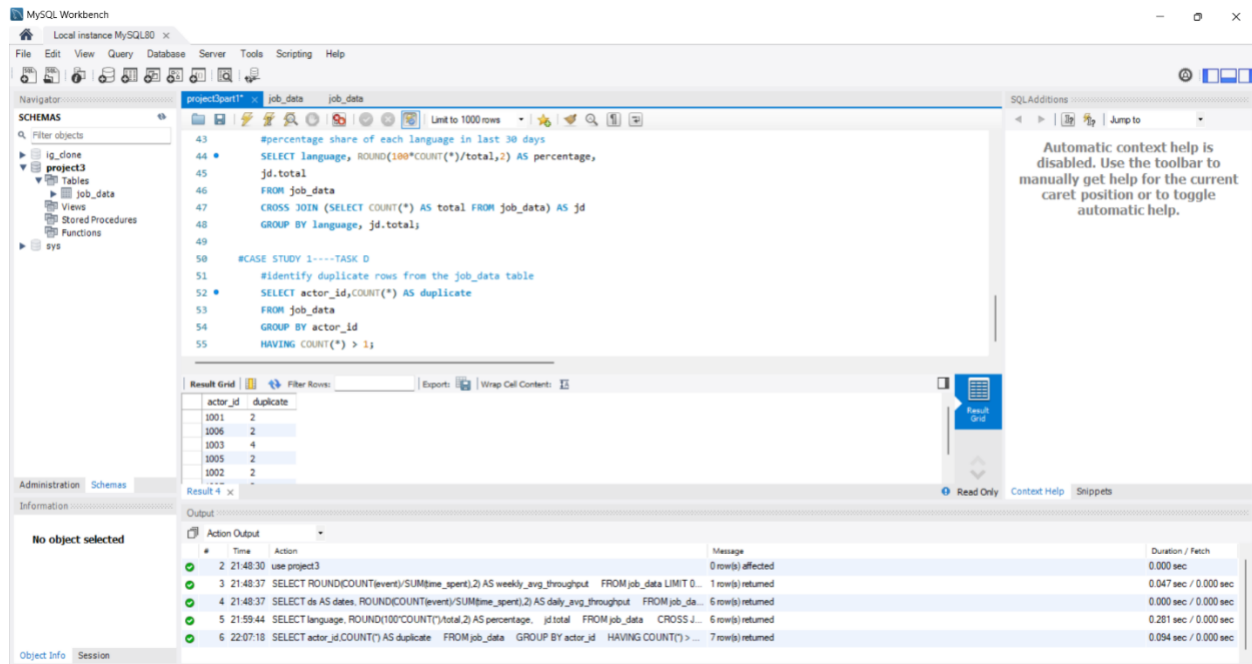
SELECT actor_id,COUNT(*) AS duplicate

FROM job_data

GROUP BY actor_id

HAVING COUNT(*) > 1;

SQL OUTPUT:



RESULT: The above screenshot shows the output for the given SQL query about the number of duplicate rows in the job_data table.

CASE STUDY 2: Investigating metric spike

It involved creating three tables namely users,events,email_events.

SQL QUERY for creating the three tables are as follows:

users table

```

create table users(
  user_id int,
  created_at varchar(100),
  company_id int,
  language varchar(50),
  activated_at varchar(100),
  state varchar(50));

```

events table

```

CREATE TABLE events(

```

```
user_id INT,  
occurred_at VARCHAR(100),  
event_type VARCHAR(50),  
event_name VARCHAR(100),  
location VARCHAR(50),  
device VARCHAR(100),  
user_type INT);
```

email-events table

```
CREATE TABLE email_events(  
user_id INT,  
occurred_at VARCHAR(100),  
action_type VARCHAR(100),  
user_type INT);
```

TASK A: Calculate weekly user engagement

SQL QUERY:

```
SELECT EXTRACT(WEEK FROM occurred_at) AS week_num,  
COUNT(DISTINCT user_id) AS active_users  
FROM events WHERE event_type = 'engagement' AND occurred_at IS NOT NULL  
GROUP BY week_num  
ORDER BY week_num;
```

SQL OUTPUT:

Result Grid				
		Filter Rows:		
				Export:
	year	week_number	num_of_users	cumulative_users
▶	2014	33	261	9104
	2014	34	259	9363
	2014	32	245	8843
	2014	30	238	8405
	2014	24	229	7101
	2014	27	222	7731
	2014	29	221	8167

RESULT: The above screenshot shows the activeness of users on a weekly basis according to the week numbers. The screenshot includes a part of the output to show how the query works and gives the desired results.

TASK B: Calculating user growth for the product

SQL QUERY:

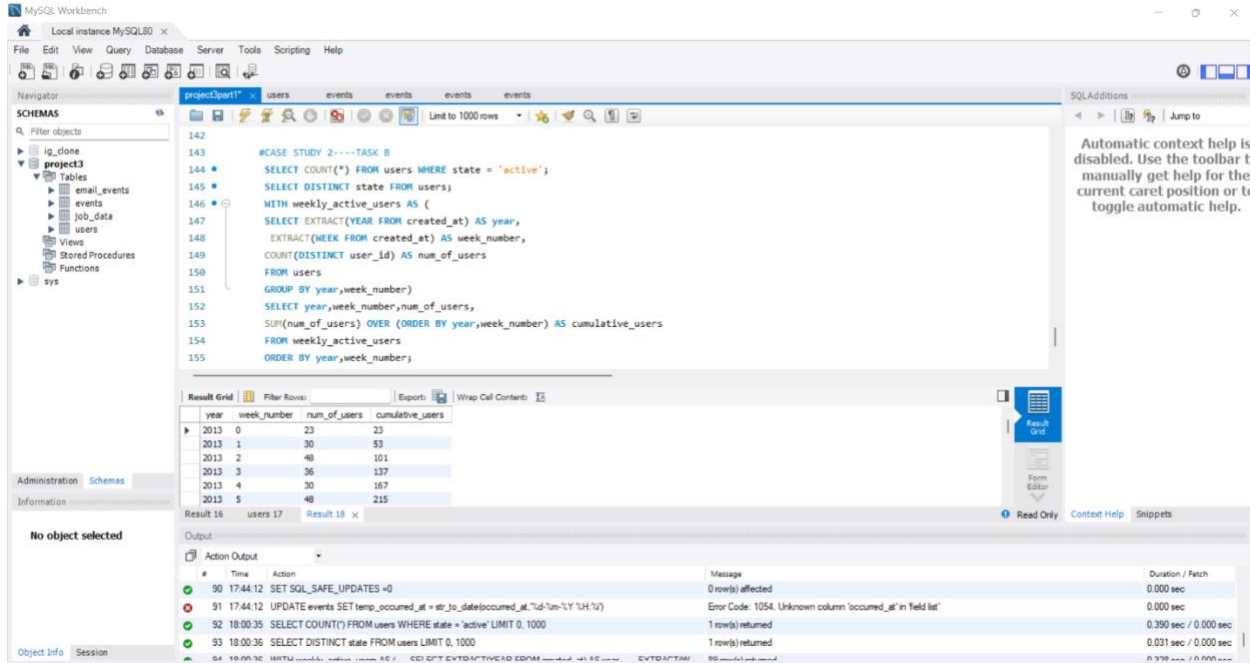
```
SELECT COUNT(*) FROM users WHERE state = 'active';

SELECT DISTINCT state FROM users;

WITH weekly_active_users AS (
  SELECT EXTRACT(YEAR FROM created_at) AS year,
    EXTRACT(WEEK FROM created_at) AS week_number,
    COUNT(DISTINCT user_id) AS num_of_users
  FROM users
  GROUP BY year, week_number)
SELECT year, week_number, num_of_users,
  SUM(num_of_users) OVER (ORDER BY year, week_number) AS cumulative_users
FROM weekly_active_users
```


ORDER BY year,week_number;

SQL OUTPUT:



The screenshot shows the MySQL Workbench interface. The SQL editor contains a query for a case study task. The query counts active users, extracts year and week from the created_at date, and calculates cumulative users over time. The results grid shows data for the year 2013 across weeks 0 to 5.

```
#CASE STUDY 2----TASK B
SELECT COUNT(*) FROM users WHERE state = 'active';
SELECT DISTINCT state FROM users;
WITH weekly_active_users AS (
SELECT EXTRACT(YEAR FROM created_at) AS year,
EXTRACT(WEEK FROM created_at) AS week_number,
COUNT(DISTINCT user_id) AS num_of_users
FROM users
GROUP BY year,week_number)
SELECT year,week_number,num_of_users,
SUM(num_of_users) OVER (ORDER BY year,week_number) AS cumulative_users
FROM weekly_active_users
ORDER BY year,week_number;
```

year	week_number	num_of_users	cumulative_users
2013	0	23	23
2013	1	30	53
2013	2	48	101
2013	3	36	137
2013	4	30	167
2013	5	48	215

RESULT: The above screenshot show the output for the above applied query about the growth of users over time for a product. It tabulates the data regarding the week number and number of users along with the cumulative users.

TASK C: Calculate weekly retention of users based on their sign-up cohort

SQL QUERY:

SELECT first AS "week_numbers",

SUM(CASE WHEN week_number=0 THEN 1 ELSE 0 END) AS "week_0",

SUM(CASE WHEN week_number=1 THEN 1 ELSE 0 END) AS "week_1",

SUM(CASE WHEN week_number=2 THEN 1 ELSE 0 END) AS "week_2",

SUM(CASE WHEN week_number=3 THEN 1 ELSE 0 END) AS "week_3",

SUM(CASE WHEN week_number=4 THEN 1 ELSE 0 END) AS "week_4",

SUM(CASE WHEN week_number=5 THEN 1 ELSE 0 END) AS "week_5",

```

SUM(CASE WHEN week_number=6 THEN 1 ELSE 0 END) AS "week_6",
SUM(CASE WHEN week_number=7 THEN 1 ELSE 0 END) AS "week_7",
SUM(CASE WHEN week_number=8 THEN 1 ELSE 0 END) AS "week_8",
SUM(CASE WHEN week_number=9 THEN 1 ELSE 0 END) AS "week_9",
SUM(CASE WHEN week_number=10 THEN 1 ELSE 0 END) AS "week_10",
SUM(CASE WHEN week_number=11 THEN 1 ELSE 0 END) AS "week_11",
SUM(CASE WHEN week_number=12 THEN 1 ELSE 0 END) AS "week_12",
SUM(CASE WHEN week_number=13 THEN 1 ELSE 0 END) AS "week_13",
SUM(CASE WHEN week_number=14 THEN 1 ELSE 0 END) AS "week_14",
SUM(CASE WHEN week_number=15 THEN 1 ELSE 0 END) AS "week_15",
SUM(CASE WHEN week_number=16 THEN 1 ELSE 0 END) AS "week_16",
SUM(CASE WHEN week_number=17 THEN 1 ELSE 0 END) AS "week_17",
SUM(CASE WHEN week_number=18 THEN 1 ELSE 0 END) AS "week_18"
FROM ( SELECT m.user_id,m.login_week,n.first,m.login_week-n.first AS week_nnumber
FROM ( SELECT user_id,EXTRACT(WEEK FROM occurred_at) AS login_week
FROM events
GROUP BY user_id,login_week)m
JOIN (SELECT user_id,MIN(EXTRACT(WEEK FROM occurred_at)) AS first
FROM events
GROUP BY user_id)n
ON m.user_id=n.user_id
)sub
GROUP BY first
ORDER BY first;

```

SQL OUTPUT:

Result Grid		Filter Rows:		Export:		Wrap Cell Contents:									
	week_numbers	week_0	week_1	week_2	week_3	week_4	week_5	week_6	week_7	week_8	week_9	week_10	week_11	week_12	
17		663	472	324	251	205	187	167	146	145	145	136	131	132	
18		596	362	261	203	168	147	144	127	113	122	106	118	127	
19		427	284	173	153	114	95	91	81	95	82	68	65	63	
20		358	223	165	121	91	72	63	67	63	65	67	41	40	
21		317	187	131	91	74	63	75	72	58	48	45	39	35	
22		326	224	150	107	87	73	63	60	55	48	41	39	31	

RESULT: The above screenshot shows the output of the query about the weekly retention of users according to their sign-up cohort.

TASK D: Calculating the weekly engagement per device

SQL QUERY:

SELECT

EXTRACT(WEEK FROM occurred_at) AS week_number,

COUNT(DISTINCT CASE WHEN device = 'dell inspiron notebook' THEN user_id ELSE NULL END) AS dell_inspiron_notebook,

COUNT(DISTINCT CASE WHEN device = 'iphone 5' THEN user_id ELSE NULL END) AS iphone_5,

COUNT(DISTINCT CASE WHEN device = 'iphone 4s' THEN user_id ELSE NULL END) AS iphone_4s,

COUNT(DISTINCT CASE WHEN device = 'iphone 5s' THEN user_id ELSE NULL END) AS iphone_5s,

COUNT(DISTINCT CASE WHEN device = 'ipad air' THEN user_id ELSE NULL END) AS ipad_air,

COUNT(DISTINCT CASE WHEN device = 'windows surface' THEN user_id ELSE NULL END) AS windows_surface,

COUNT(DISTINCT CASE WHEN device = 'macbook air' THEN user_id ELSE NULL END) AS macbook_air,

COUNT(DISTINCT CASE WHEN device = 'macbook pro' THEN user_id ELSE NULL END) AS macbook_pro,

COUNT(DISTINCT CASE WHEN device = 'ipad mini' THEN user_id ELSE NULL END) AS ipad_mini,

COUNT(DISTINCT CASE WHEN device = 'kindle fire' THEN user_id ELSE NULL END) AS kindle_fire,

```

COUNT(DISTINCT CASE WHEN device = 'amazon fire phone' THEN user_id ELSE
NULL END) AS amazon_fire_phone,

COUNT(DISTINCT CASE WHEN device = 'nexus 5' THEN user_id ELSE NULL END) AS
nexus_5,

COUNT(DISTINCT CASE WHEN device = 'nexus 7' THEN user_id ELSE NULL END) AS
nexus_7,

COUNT(DISTINCT CASE WHEN device = 'nexus 10' THEN user_id ELSE NULL END)
AS nexus_10,

COUNT(DISTINCT CASE WHEN device = 'samsung galaxy s4' THEN user_id ELSE
NULL END) AS samsung_galaxy_s4,

COUNT(DISTINCT CASE WHEN device = 'samsung galaxy tablet' THEN user_id ELSE
NULL END) AS samsung_galaxy_tablet,

COUNT(DISTINCT CASE WHEN device = 'samsung galaxy note' THEN user_id ELSE
NULL END) AS samsung_galaxy_note,

COUNT(DISTINCT CASE WHEN device = 'lenovo thinkpad' THEN user_id ELSE NULL
END) AS lenovo_thinkpad,

COUNT(DISTINCT CASE WHEN device = 'acer aspire notebook' THEN user_id ELSE
NULL END) AS acer_aspire_notebook,

COUNT(DISTINCT CASE WHEN device = 'asus chromebook' THEN user_id ELSE NULL
END) AS asus_chromebook,

COUNT(DISTINCT CASE WHEN device = 'htc one' THEN user_id ELSE NULL END) AS
htc_one,

COUNT(DISTINCT CASE WHEN device = 'nokia lumia 635' THEN user_id ELSE NULL
END) AS nokia_lumia635,

COUNT(DISTINCT CASE WHEN device = 'mac mini' THEN user_id ELSE NULL END)
AS mac_mini,

COUNT(DISTINCT CASE WHEN device = 'hp pavilion desktop' THEN user_id ELSE
NULL END) AS hp_pavilion_desktop,

COUNT(DISTINCT CASE WHEN device = 'dell inspiron desktop' THEN user_id ELSE
NULL END) AS dell_inspiron_desktop

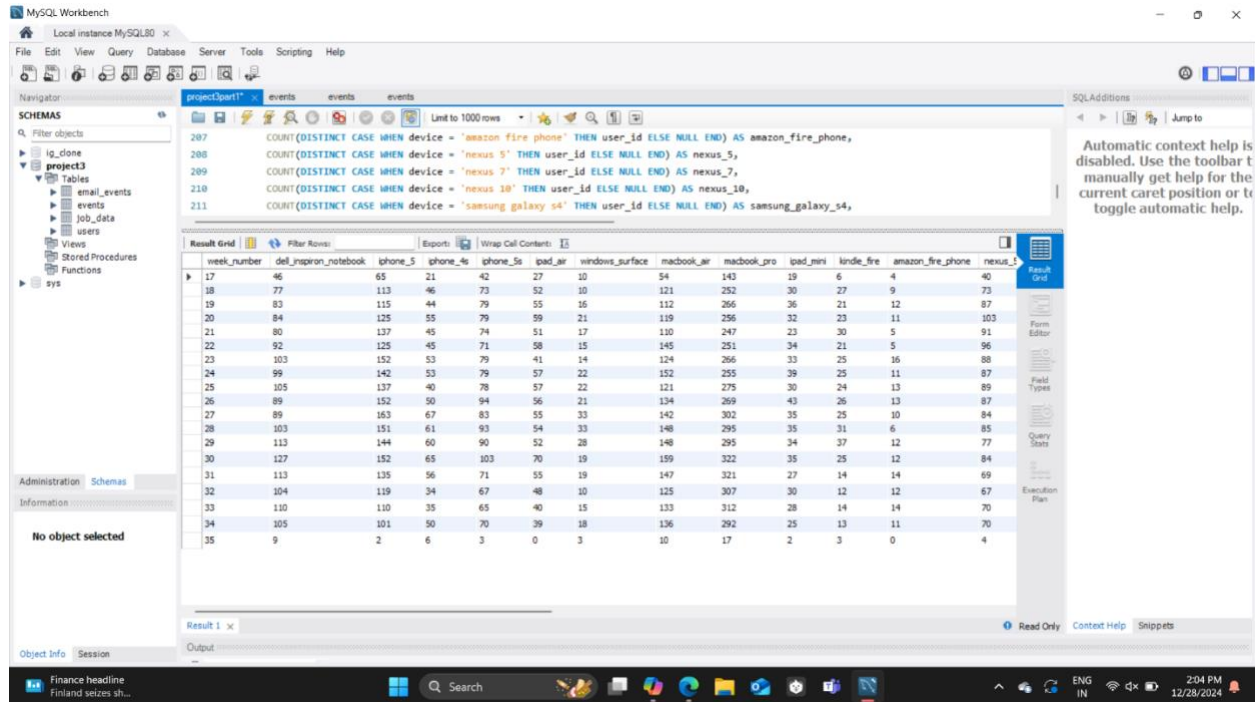
FROM events WHERE event_type = 'engagement'

GROUP BY week_number

```

ORDER BY week_number;

SQL OUTPUT:



The screenshot shows the MySQL Workbench interface. The SQL editor contains a query that counts distinct user IDs for various devices. The results grid displays the output of this query, showing counts for 12 different devices across 35 weeks. The devices listed in the columns are: del_inspiron_notebook, phone_5, phone_4s, phone_5s, ipad_ar, windows_surface, macbook_ar, macbook_pro, ipad_mini, kindle_fire, amazon_fire_phone, and nexus_5.

week_number	del_inspiron_notebook	phone_5	phone_4s	phone_5s	ipad_ar	windows_surface	macbook_ar	macbook_pro	ipad_mini	kindle_fire	amazon_fire_phone	nexus_5
17	46	65	21	42	27	10	54	143	19	6	4	40
18	77	113	46	73	52	10	121	252	30	27	9	73
19	83	115	44	79	55	16	112	266	36	21	12	87
20	84	125	55	79	59	21	119	296	32	23	11	103
21	80	137	45	74	51	17	110	247	23	30	5	91
22	92	125	45	71	58	15	145	251	34	21	5	96
23	103	152	53	79	41	14	124	266	33	25	16	88
24	99	142	53	79	57	22	152	255	39	25	11	87
25	105	137	40	78	57	22	121	275	30	24	13	89
26	89	152	50	94	56	21	134	269	43	26	13	87
27	89	163	67	83	55	33	142	302	35	25	10	84
28	103	151	61	93	54	33	148	295	35	31	6	85
29	113	144	60	90	52	28	148	295	34	37	12	77
30	127	152	65	103	70	19	159	322	35	25	12	84
31	113	135	56	71	55	19	147	321	27	14	14	69
32	104	119	34	67	48	10	125	307	30	12	12	67
33	110	110	35	65	40	15	133	312	28	14	14	70
34	105	101	50	70	39	18	136	292	25	13	11	70
35	9	2	6	3	0	3	10	17	2	3	0	4

RESULT: The above screenshot shows the output about the query for the activeness of users on a weekly basis per device.

TASK E : Calculate the email engagement metrics

SQL QUERY:

SELECT

100 * SUM(CASE WHEN email_action = 'email_open' THEN 1 ELSE 0 END) /

SUM(CASE WHEN email_action = 'email_sent' THEN 1 ELSE 0 END) AS email_open_rate,

100 * SUM(CASE WHEN email_action = 'email_clicked' THEN 1 ELSE 0 END) /

SUM(CASE WHEN email_action = 'email_sent' THEN 1 ELSE 0 END) AS
email_clicked_rate

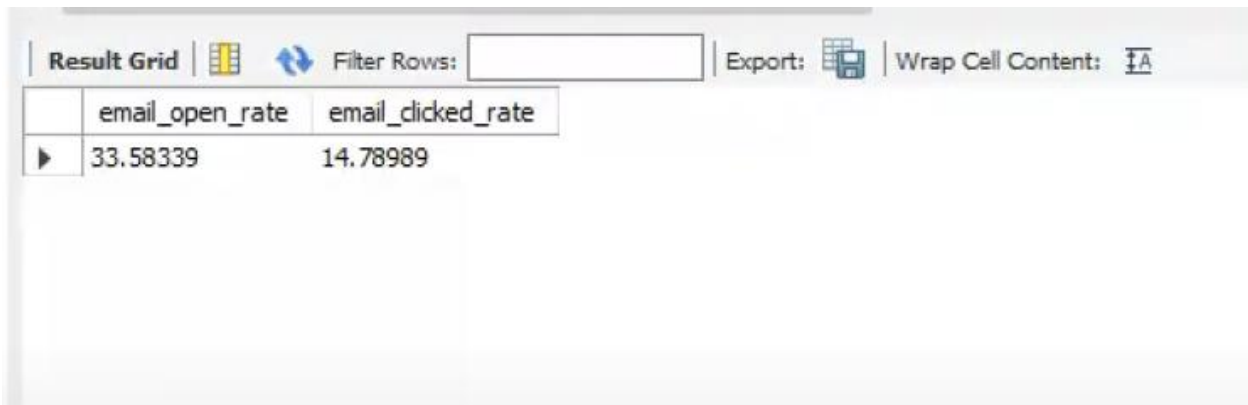
FROM (

```

SELECT *,
CASE
    WHEN action IN ('sent_weekly_digest', 'sent_reengagement_email') THEN 'email_sent'
    WHEN action = 'email_open' THEN 'email_open'
    WHEN action = 'email_clickthrough' THEN 'email_clicked'
    ELSE NULL
END AS email_action
FROM project3.email_events
) a;

```

SQL OUTPUT:



The screenshot shows a SQL query result grid with two columns: 'email_open_rate' and 'email_clicked_rate'. The first row displays the values 33.58339 and 14.78989 respectively. The grid includes a 'Filter Rows' search bar, an 'Export' button, and a 'Wrap Cell Content' toggle.

	email_open_rate	email_clicked_rate
▶	33.58339	14.78989

RESULT: The above screenshot shows the email engagement metrics in the form of email_clicked_rate and email_open_rate. It helped to analyze how users are engaging with the email service.

RESULT OF THE PROJECT:

I improved my learning of MYSQL and EXCEL through this project while working as a data lead analyst. I explored how to tackle data to get various answers about your queries using SQL. I also learned to create databases and am confident enough to work as a lead data analyst in such firms. We got to know the various insights of the data such as user engagement as well as use of services as email by the users. It also involved job data analysis including the throughput analysis as

well as cleaning the dataset by identifying any duplicate rows. I would like to thank trainity team to provide me such a nice project to work on and gain experience.