OPERATION ANALYTICS AND INVESTIGATING METRIC SPIKE

-YAADHAV R

PROJECT DESCRIPTION :

THE OBJECTIVE OF THIS PROJECT IS TO TAKE THE ROLE AS A LEAD DATA ANALYST AND DERIVE USEEFUL INSIGHTS FROM THE DATASETS. ALSO TO UNDERSTAND ANALYZE THE DIFFERENT METRICS SPIKES AND ANSWER THE VARIOUS QUESTION POSTED BY THE COMPANY.

TECH-STACK USED :

MYSQL WORKBENCH - VERSION 8.0.34 IS USED IN THIS PROJECT AS IT IS :

- IT A SIMPLE AND EASY TO USE SOFTWARE.
- IT IS FASTER COMPARED TO OTHER SOFTWARES DUE TO ITS SIMPLICITY.

CONCEPTS APPLIED:

DDL COMMAND USED:

- CREATE

DML COMMANDS USED :

- SELECT

OTHER COMMANDS USED:

- FROM CLAUSE.
- WHERE CLAUSE.
- ORDER BY CLAUSE.
- GROUP BY CLAUSE.
- HAVING CLAUSE

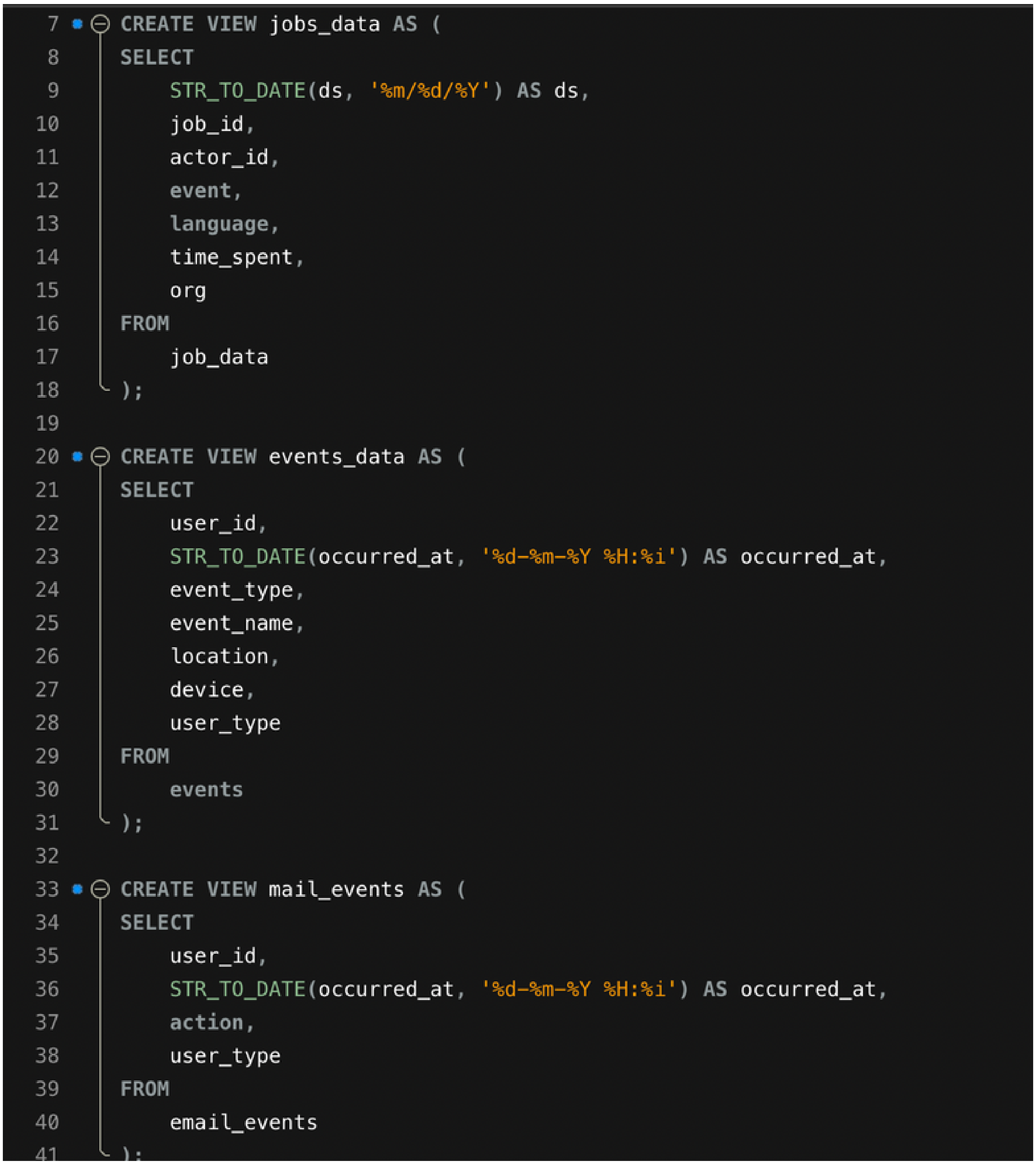
FUNCTIONS USED:

- SUM()
- COUNT()
- OVER()
- STR_TO_DATE()
- WEEKOFYEAR()
- MOTH()
- YEAR()

OTHER CONCEPTS:

- SUB-QURIES
- COMMON TABLE EXPRESSIONS
- DERIVED TABLES

DATA CLEANING/PREPARATION :



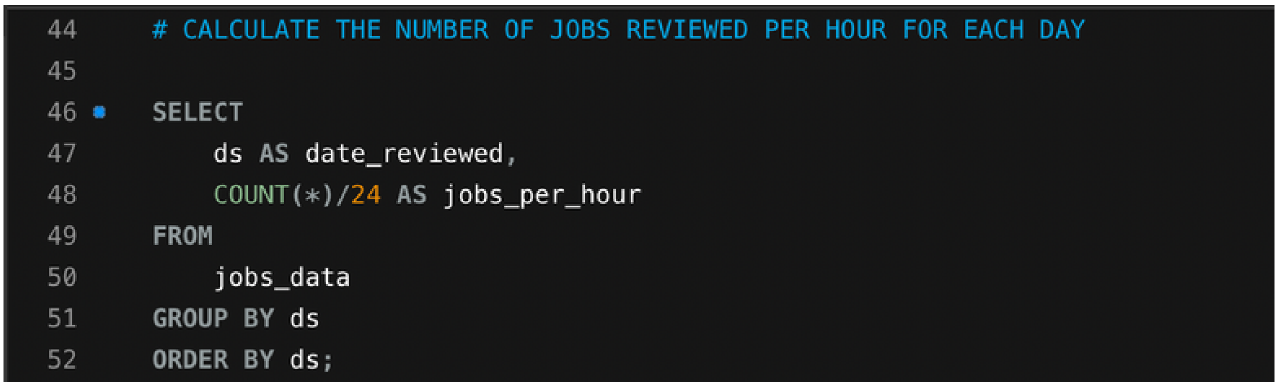
```sql
CREATE VIEW jobs_data AS (
SELECT
    STR_TO_DATE(ds, '%m/%d/%Y') AS ds,
    job_id,
    actor_id,
    event,
    language,
    time_spent,
    org
FROM
    job_data
);

CREATE VIEW events_data AS (
SELECT
    user_id,
    STR_TO_DATE(occurred_at, '%d-%m-%Y %H:%i') AS occurred_at,
    event_type,
    event_name,
    location,
    device,
    user_type
FROM
    events
);

CREATE VIEW mail_events AS (
SELECT
    user_id,
    STR_TO_DATE(occurred_at, '%d-%m-%Y %H:%i') AS occurred_at,
    action,
    user_type
FROM
    email_events
);
```

- SINCE THE GIVEN DATA CANNOT BE USED DIRECTLY WITHOUT PREPARATION AS THE TABLES CONTAINS A FEW FIELDS IN STRING DATATYPE THAT ARE SUPPOSED TO BE IN TIMESTAMP/DATE DATATYPE .
- SO THOSE FIELDS ARE CONVERT INTO DATE DATATYPE WITH THE HELP OF THE STR_TO_DATE FUNCTION().
- THESE FIELDS ARE ADDED TO NEW VIEWS, WHICH WILL SERVE THE TEMPORARY SPACE TO WORK WITH THE DATA.

TASKS

CASE STUDY-1 :

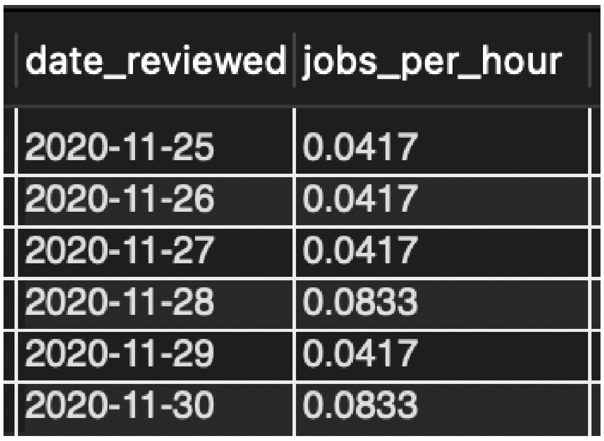
A) JOBS REVIEWED OVER TIME:



```
# CALCULATE THE NUMBER OF JOBS REVIEWED PER HOUR FOR EACH DAY

SELECT
    ds AS date_reviewed,
    COUNT(*)/24 AS jobs_per_hour
FROM
    jobs_data
GROUP BY ds
ORDER BY ds;
```

APPROACH

- COUNT() FUNCTION IS USED TO CALCULATE THE NUMBER OF JOBS REVIEWED AND IT IS DIVIDED BY 24 TO GET THE NUMBERS OF JOBS REVIEWED PER HOUR.
- GROUP BY CLAUSE DIVIDES THE COUNT() ACROSS DATES AND ORDER BY DISPLAYS THE OUTPUT IN INCREASING ORDER OF DATES.

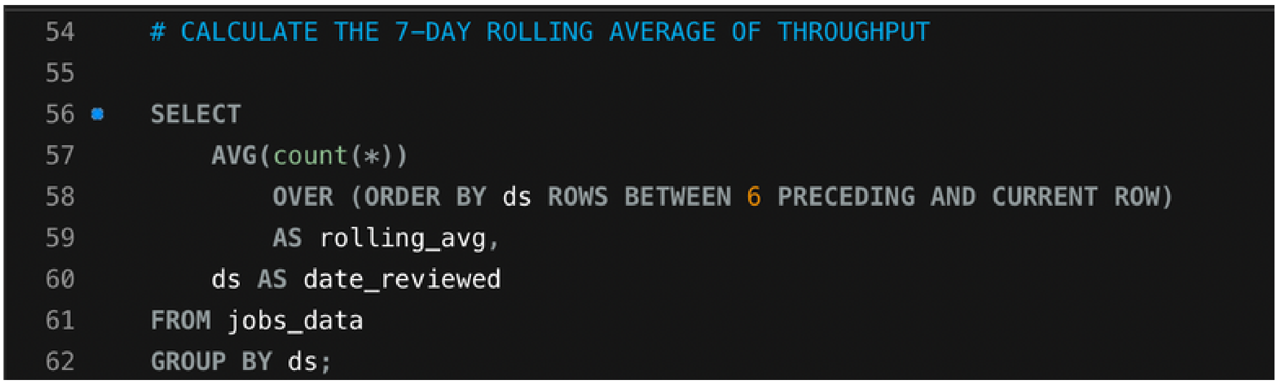


date_reviewed	jobs_per_hour
2020-11-25	0.0417
2020-11-26	0.0417
2020-11-27	0.0417
2020-11-28	0.0833
2020-11-29	0.0417
2020-11-30	0.0833

INSIGHTS

- THE RATE AT WHICH THE JOBS ARE REVIEWED IS LOW AS ONLY 8 JOBS HAS BEEN REVIEWED FOR 6 DAYS.
- THERE IS A CONSTANT LAG IN REVIEWING THE JOBS.

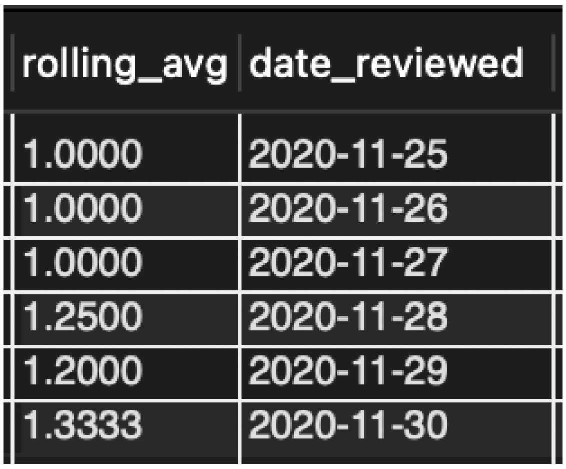
B) THROUGHPUT ANALYSIS :



```
# CALCULATE THE 7-DAY ROLLING AVERAGE OF THROUGHPUT

SELECT
    AVG(count(*))
        OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW)
        AS rolling_avg,
    ds AS date_reviewed
FROM jobs_data
GROUP BY ds;
```

APPROACH

- ROLLING AVERAGE HAS BEEN CALCULATED WITH HELP OF AVG() OVER()
- TO CALCULATE 7 DAY ROLLING AVERAGE, THE CURRENT ROW AND THE PRECEDING 6 VALUES ARE TAKEN INTO CONSIDERATION.
- GROUP BY CLAUSE DIVIDES THE COUNT() ACROSS DATES.

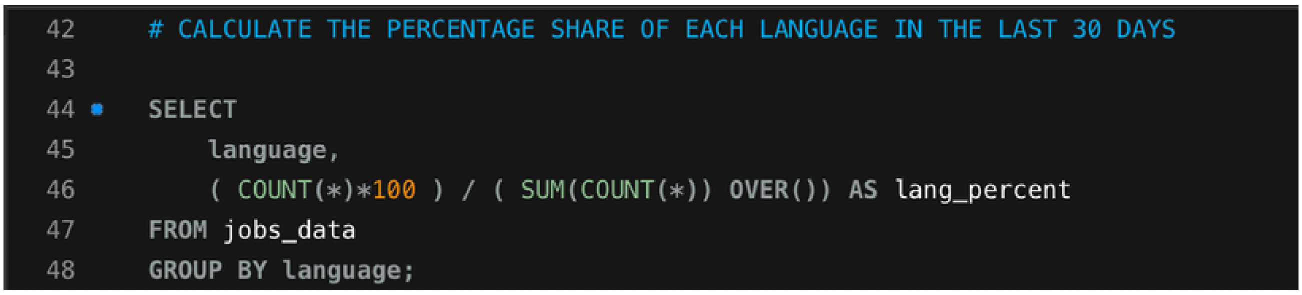


rolling_avg	date_reviewed
1.0000	2020-11-25
1.0000	2020-11-26
1.0000	2020-11-27
1.2500	2020-11-28
1.2000	2020-11-29
1.3333	2020-11-30

INSIGHTS

- THE FIRST 3 DAYS FROM THE GIVEN DATA HAD A CONSTANT AVERAGE OF 1.0.
- THEN IT RAISED IN THE NEXT 3DAYS AND RECORDED THE HIGHEST AVERAGE ON THE LAST DAY.
- ROLLING AVERAGE ARE IS CALCULATED WITH LIMITED DATA AS THE DATA PROVIDED WAS LIMITED AS WELL.

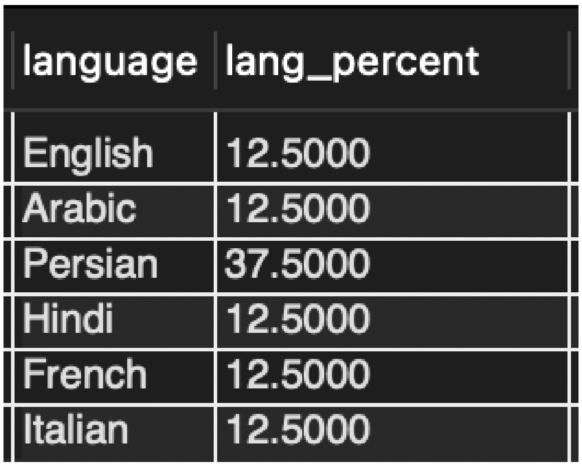
C) LANGUAGE SHARE ANALYSIS :



```
# CALCULATE THE PERCENTAGE SHARE OF EACH LANGUAGE IN THE LAST 30 DAYS

SELECT
    language,
    ( COUNT(*)*100 ) / ( SUM(COUNT(*)) OVER()) AS lang_percent
FROM jobs_data
GROUP BY language;
```

APPROACH

- .COUNT() IS USED TO CALCULATE THE TOTAL OF THAT PARTICULAR LANGUAGE AND THE SUM(COUNT ()) CALCULATES THE TOTAL NUMBER OF LANGUAGES.
- THE OVER() FUNCTIONS IS USED TO APPLY THE TOTAL IN CALCULATIONS FOR THE PERCENTAGE SHARE.
- FINALLY THE GROUP BY DIVIDES THE COUNT ACROSS LANGUAGES.

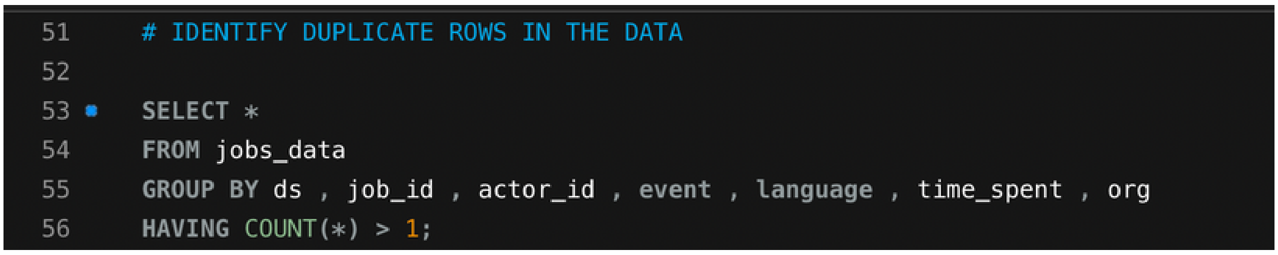


language	lang_percent
English	12.5000
Arabic	12.5000
Persian	37.5000
Hindi	12.5000
French	12.5000
Italian	12.5000

INSIGHTS

- PERSIAN LANGUAGE IS THE MOST USED LANGUAGE WHICH CONSTITUTES AROUND 38% OF THE TOTAL SHARE WHICH STANDS APART.
- AS THE REST OF THE LANGUAGES ARE USED THE SAME AND CONSTITUTED 12.5% EACH..

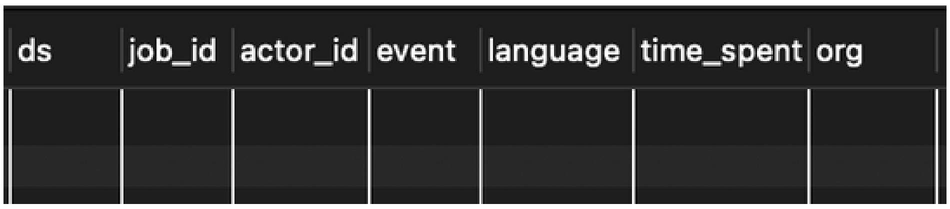
D) DUPLICATE ROWS DETECTION :



```
# IDENTIFY DUPLICATE ROWS IN THE DATA

SELECT *
FROM jobs_data
GROUP BY ds , job_id , actor_id , event , language , time_spent , org
HAVING COUNT(*) > 1;
```

APPROACH

- GROUP BY CLAUSE IS USED ALONG WITH HAVING CLAUSE THAT USES A COUNT() FUNCTION TO COUNT THE OCCURRENCES OF EACH ROW AND THE ROWS THAT HAVE MORE THAN ONE OCCURRENCE ARE FILTERED.
- THESE ARE DUPLICATE ROWS IN THE TABLE AS THE SAME VALUES ARE REOEATED AGAIN.



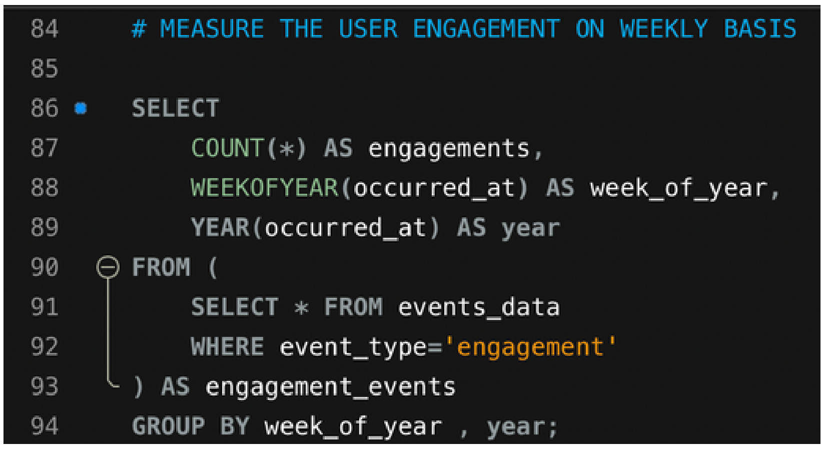
ds	job_id	actor_id	event	language	time_spent	org

INSIGHTS

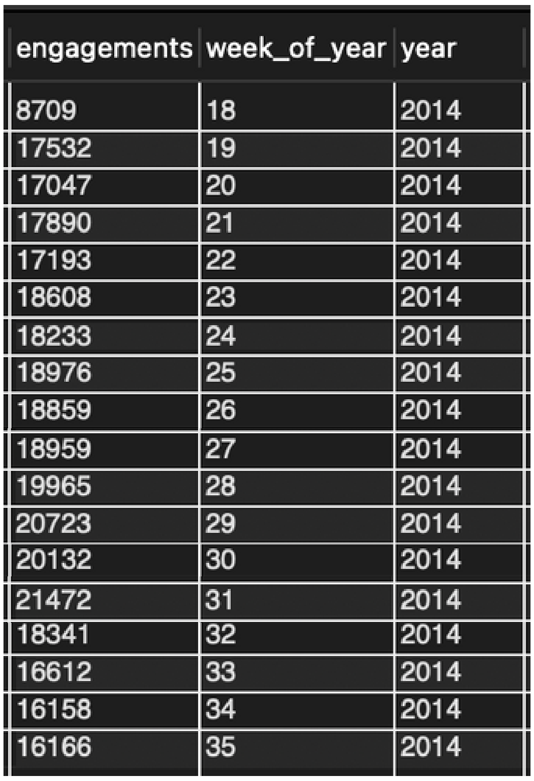
- THERE IS NO PRESENCE OF DUPLICATE ROWS IN THE GIVEN DATA.

CASE STUDY-2 :

A)WEEKLY USER ENGAGEMENT :



```sql
# MEASURE THE USER ENGAGEMENT ON WEEKLY BASIS

SELECT
    COUNT(*) AS engagements,
    WEEKOFYEAR(occurred_at) AS week_of_year,
    YEAR(occurred_at) AS year
FROM (
    SELECT * FROM events_data
    WHERE event_type='engagement'
) AS engagement_events
GROUP BY week_of_year , year;
```



engagements	week_of_year	year
8709	18	2014
17532	19	2014
17047	20	2014
17890	21	2014
17193	22	2014
18608	23	2014
18233	24	2014
18976	25	2014
18859	26	2014
18959	27	2014
19965	28	2014
20723	29	2014
20132	30	2014
21472	31	2014
18341	32	2014
16612	33	2014
16158	34	2014
16166	35	2014

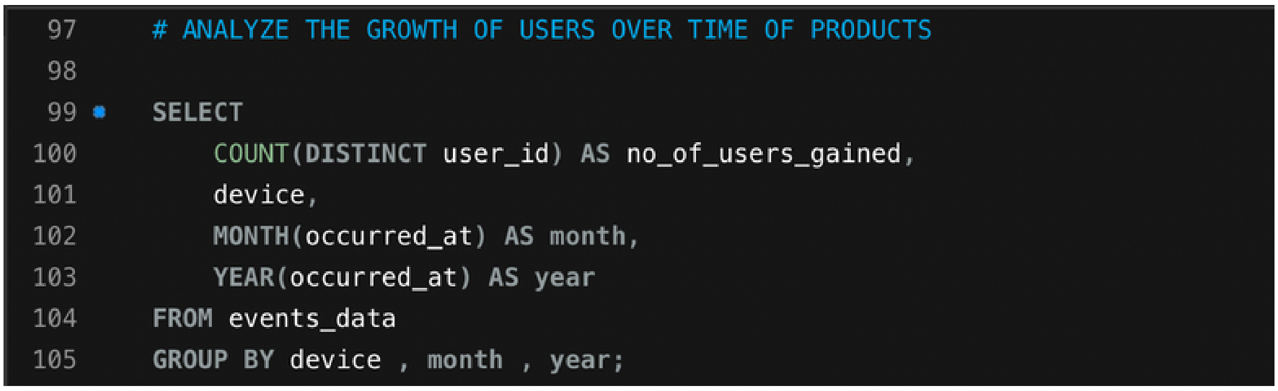
APPROACH

- COUNT() FUNCTION IS USED TO COUNT NUMBER OF ENGAGEMENTS FROM THE ‘ENGAGEMENT_EVENTS’ TABLE WHICH CONTAINS ONLY THE EVENT_TYPE WITH ENGAGEMENT.
- THE GROUP BY CLAUSE DIVIDES THE COUNT() ACROSS WEEK_OF_YEAR AND YEAR WHICH ARE CALCULATED BY WEEKOFYEAR() AND YEAR() FUNCTIONS RESPECTIVELY.

INSIGHTS

- AFTER THE INITIAL WEEK, THERE WAS A SIGNIFICANT SPIKE IN USER ENGAGEMENTS DURING THE SECOND WEEK, SHOWING AN INCREASE OF OVER 100% COMPARED TO THE PREVIOUS WEEK.
- WHILE THE SECOND-HIGHEST SPIKE WAS JUST AN 8% RISE.
- USER ENGAGEMENTS WAS LOWEST DURING THE 18TH WEEK OF 2014 WHICH WAS THE FIRST WEEK OF USER ENGAGEMENTS AS WELL.
- THE HISHEST WAS RECORDED DURING THE 31ST WEEK OF 2014, BUT SUBSEQUENTLY ENGAGEMENTS STARTED TO DECLINE EVERY WEEK FOLLOWED.

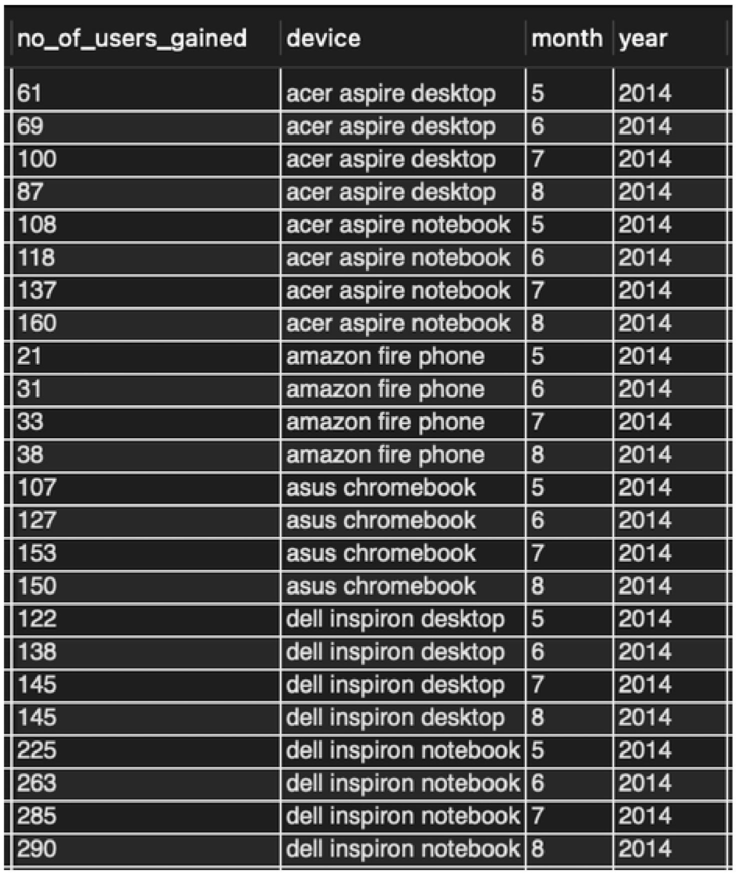
B) USER GROWTH ANALYSIS :



```
# ANALYZE THE GROWTH OF USERS OVER TIME OF PRODUCTS

SELECT
    COUNT(DISTINCT user_id) AS no_of_users_gained,
    device,
    MONTH(occurred_at) AS month,
    YEAR(occurred_at) AS year
FROM events_data
GROUP BY device , month , year;
```

APPROACH

- COUNT() FUNCTION IS USED TO CALCULATE THE NUMBER OF USERS AND THE DISTINCT CLAUSE FILTERS OUT THE DUPLICATE USER_ID.
- GROUP BY CLAUSE DIVIDES THE COUNT ACROSS DEVICE, MONTH AND YEAR.

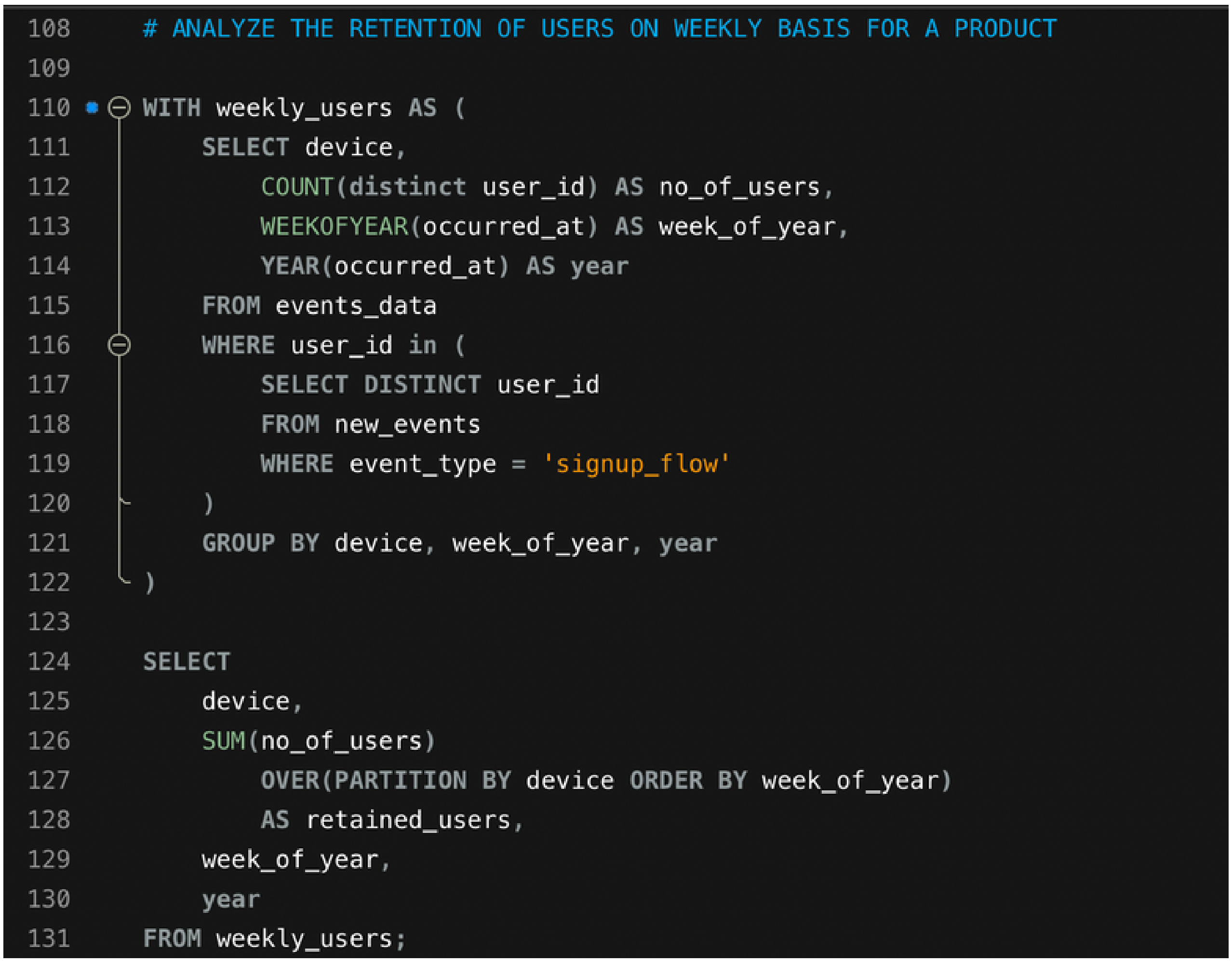


no_of_users_gained	device	month	year
61	acer aspire desktop	5	2014
69	acer aspire desktop	6	2014
100	acer aspire desktop	7	2014
87	acer aspire desktop	8	2014
108	acer aspire notebook	5	2014
118	acer aspire notebook	6	2014
137	acer aspire notebook	7	2014
160	acer aspire notebook	8	2014
21	amazon fire phone	5	2014
31	amazon fire phone	6	2014
33	amazon fire phone	7	2014
38	amazon fire phone	8	2014
107	asus chromebook	5	2014
127	asus chromebook	6	2014
153	asus chromebook	7	2014
150	asus chromebook	8	2014
122	dell inspiron desktop	5	2014
138	dell inspiron desktop	6	2014
145	dell inspiron desktop	7	2014
145	dell inspiron desktop	8	2014
225	dell inspiron notebook	5	2014
263	dell inspiron notebook	6	2014
285	dell inspiron notebook	7	2014
290	dell inspiron notebook	8	2014

INSIGHTS

- ONLY THE DATA OF 6 DEVICES ARE SHOWN DUE TO THE SIZE CONSTRAINT.
- THE NUMBER OF USERS GAINED ONLY KEPT ON INCREASING EVERY MONTH FOR MOST OF THE DEVICES.
- NO SUDDEN SPIKES WERE OBSERVED FROM THE OUTPUT.

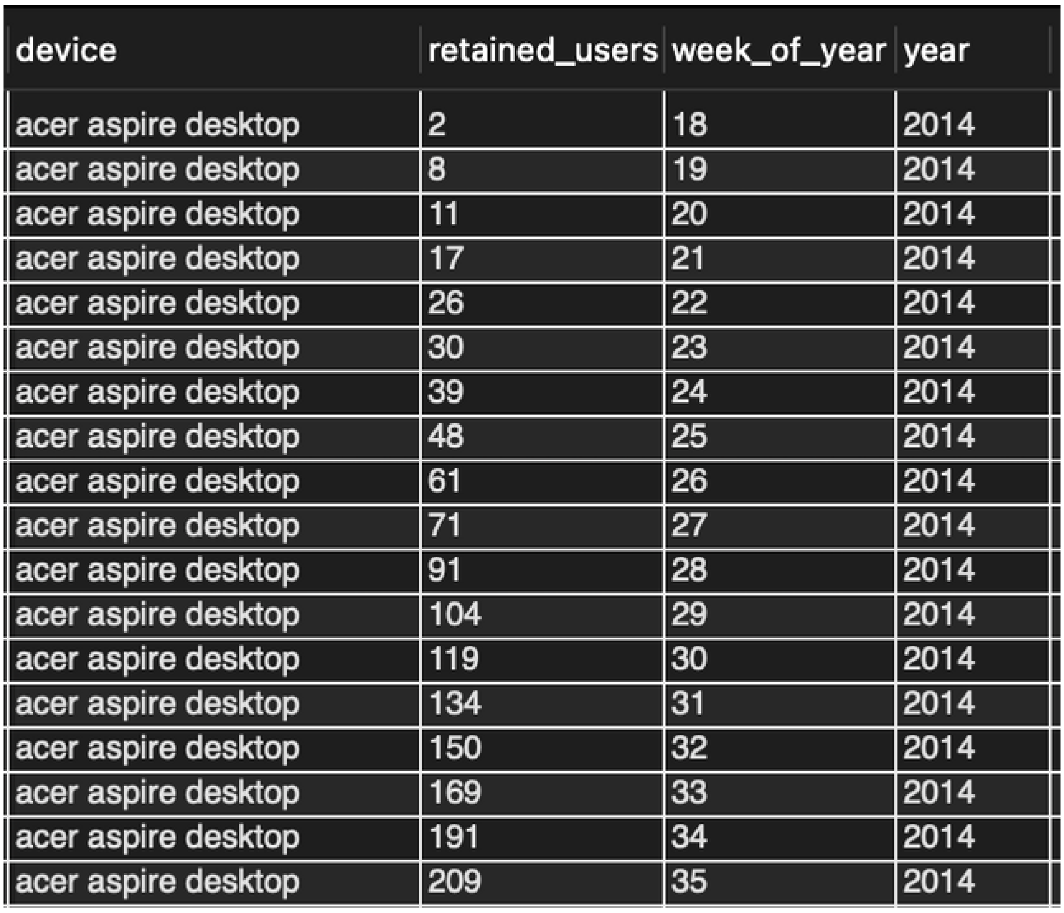
C) WEEKLY RETENTION ANALYSIS :



```
# ANALYZE THE RETENTION OF USERS ON WEEKLY BASIS FOR A PRODUCT

WITH weekly_users AS (
    SELECT device,
        COUNT(distinct user_id) AS no_of_users,
        WEEKOFYEAR(occurred_at) AS week_of_year,
        YEAR(occurred_at) AS year
    FROM events_data
    WHERE user_id in (
        SELECT DISTINCT user_id
        FROM new_events
        WHERE event_type = 'signup_flow'
    )
    GROUP BY device, week_of_year, year
)

SELECT
    device,
    SUM(no_of_users)
        OVER(PARTITION BY device ORDER BY week_of_year)
        AS retained_users,
    week_of_year,
    year
FROM weekly_users;
```

APPROACH

- COUNT() IS USED TO CALCULATE THE NUMBER OF USERS FROM THE FILTERED USERS WHO COMPLETED SIGN UP THROUGH A SUB-QUERY.
- THEN GROUP BY CLAUSE DIVIDES THE COUNT ACROSS DEVICE, WEEK_OF_YEAR, YEAR AND IT IS USED AS A CTE WIHT ALIASE WEEKLY_USERS.
- WITH THIS CTE, THE SUM() OVER() CALCULATES THE TOTAL OF OLD AND NEW USERS WHICH GIVES US THE RETENTAINED USERS COUNT.

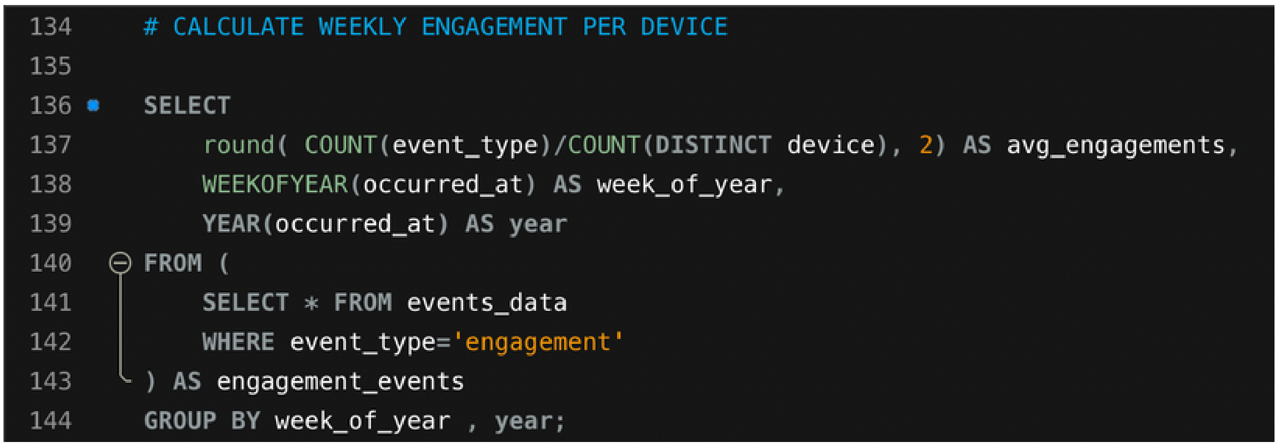


device	retained_users	week_of_year	year
acer aspire desktop	2	18	2014
acer aspire desktop	8	19	2014
acer aspire desktop	11	20	2014
acer aspire desktop	17	21	2014
acer aspire desktop	26	22	2014
acer aspire desktop	30	23	2014
acer aspire desktop	39	24	2014
acer aspire desktop	48	25	2014
acer aspire desktop	61	26	2014
acer aspire desktop	71	27	2014
acer aspire desktop	91	28	2014
acer aspire desktop	104	29	2014
acer aspire desktop	119	30	2014
acer aspire desktop	134	31	2014
acer aspire desktop	150	32	2014
acer aspire desktop	169	33	2014
acer aspire desktop	191	34	2014
acer aspire desktop	209	35	2014

INSIGHTS

- ONLY THE RETAINED USERS OF 1 DEVICE IS SHOWN DUE TO THE SIZE CONSTRAINT.
- AS NEW USERS KEPT ON SIGNING UP EVERY WEEK, THE RETENTION OF USERS WAS ONLY INCREASING EVERY WEEK FOR ALL THE DEVICES.
- HERE TOO, NO SUDDEN SPIKES WERE OBSERVED.

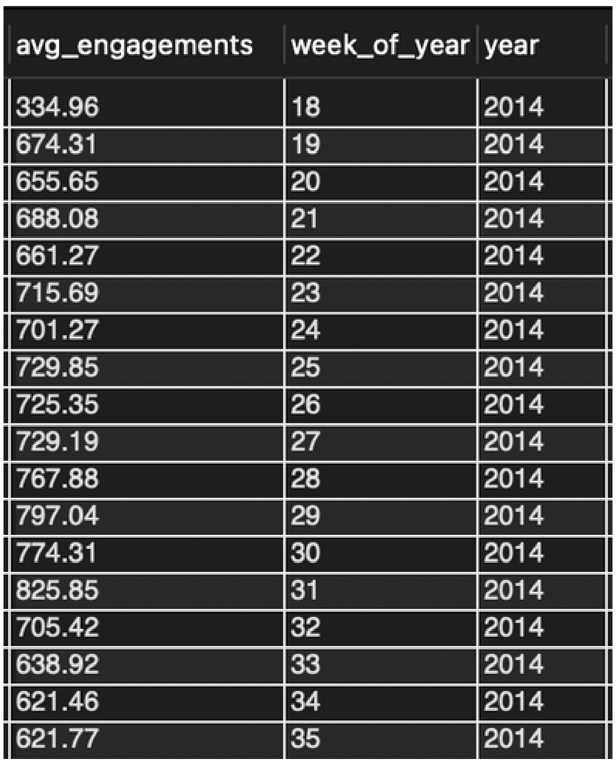
D) WEEKLY ENGAGEMENT PER DEVICE:



```
# CALCULATE WEEKLY ENGAGEMENT PER DEVICE

SELECT
    round( COUNT(event_type)/COUNT(DISTINCT device), 2) AS avg_engagements,
    WEEKOFYEAR(occurred_at) AS week_of_year,
    YEAR(occurred_at) AS year
FROM (
    SELECT * FROM events_data
    WHERE event_type='engagement'
) AS engagement_events
GROUP BY week_of_year , year;
```

APPROACH

- COUNT() CALCULATES THE NUMBER OF ENGAGEMENTS AND IT IS DIVIDED BY THE COUNT OF DISTINCT DEVICES TO GET ENGAGEMENTS PER DEVICE.
- ROUND() FUNCTION ROUNDS OFF THE RESULT TO 2 DECIMAL PLACES.
- THESE CALCULATIONS ARE MADE OVER THE FILTERED TABLE WITH EVENT_TYPE AS ENGAGEMENT THROUGH A SUB-QUERY.
- GROUP BY DIVIDES THE RESULT ACROSS WEEK_OF_YEAR AND YEAR.

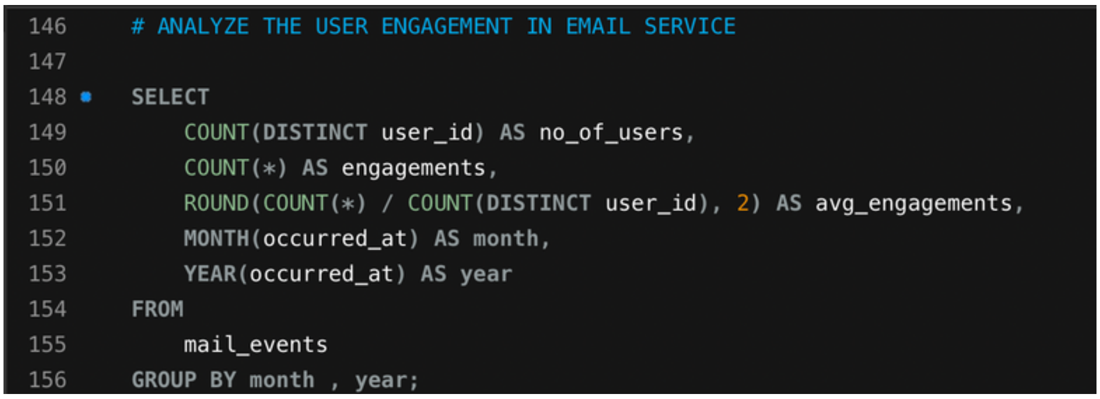


avg_engagements	week_of_year	year
334.96	18	2014
674.31	19	2014
655.65	20	2014
688.08	21	2014
661.27	22	2014
715.69	23	2014
701.27	24	2014
729.85	25	2014
725.35	26	2014
729.19	27	2014
767.88	28	2014
797.04	29	2014
774.31	30	2014
825.85	31	2014
705.42	32	2014
638.92	33	2014
621.46	34	2014
621.77	35	2014

INSIGHTS

- THE WEEKLY ENGAGEMENT PER DEVICE HAS EXACTLY THE SAME OBSERVATIONS AS THAT OF THE WEEKLY USER ENGAGEMENTS.

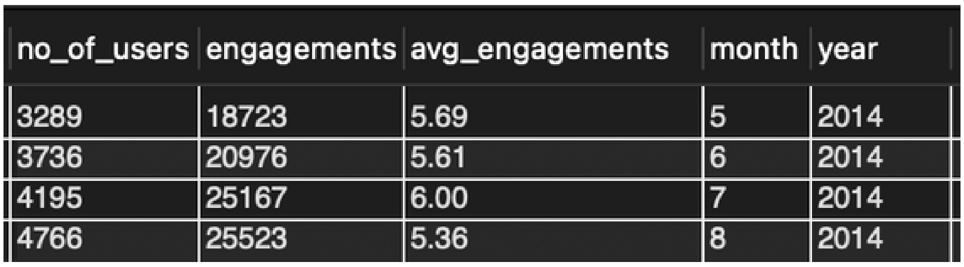
E) EMAIL ENGAGEMENT ANALYSIS:



```
# ANALYZE THE USER ENGAGEMENT IN EMAIL SERVICE

SELECT
    COUNT(DISTINCT user_id) AS no_of_users,
    COUNT(*) AS engagements,
    ROUND(COUNT(*) / COUNT(DISTINCT user_id), 2) AS avg_engagements,
    MONTH(occurred_at) AS month,
    YEAR(occurred_at) AS year
FROM
    mail_events
GROUP BY month , year;
```

APPROACH

- COUNT() CALCULATES THE NUMBER OF USERS AND ENGAGEMENTS.
- ENGAGEMENTS IS DIVIDED BY NUMBER OF USERS TO GET THE AVERAGE ENGAGEMENTS PER USER.
- GROUP BY CLAUSE DIVIDES THE RESULTS ACROSS MONTH AND YEAR.



no_of_users	engagements	avg_engagements	month	year
3289	18723	5.69	5	2014
3736	20976	5.61	6	2014
4195	25167	6.00	7	2014
4766	25523	5.36	8	2014

INSIGHTS

- EVEN THOUGH THE HIGHEST NUMBER OF USERS WERE ENGAGED DURING AUGUST 2014 , IT RECORDED THE LOWEST ENGAGEMENTS TO USER RATIO.
- DESPITE HAVING AROUND 500 USERS FEWER THAN AUGUST 2014, JULY 2014 FELL JUST OF SHORT IN TOTAL NUMBER OF ENGAGEMENTS.
- BUT THE HIGHEST ENGAGEMENTS TO USER WAS RECORDED DURING THIS MONTH .
- NO SUDDEN SPIKES WERE OBSERVED HERE AS WELL.

RESULT:

- FIRST OF ALL THIS PROJECT HELPED TO UNDERSTAND THE CONCEPTS I LEARNED IN BETTER AND INTRESTING WAY.
- SO NOW FEEL CONFIDENT IN APPLYING MY SQL SKILLS AS I WAS ABLE TO COMPLETE ALL THE GIVEN TASKS.
- IT WAS SO EXICTING TO ANALYZE THE OUTPUTS AND DERIVE INSIGHTS FROM IT.
- OVERALL IT WAS GREAT TO EXPERIENCE TO APPLY THE SKILLS AND LEARN ALONG THE WAY.
- LOOKING FORWARD TO FACE THE UPCOMING CHALLENGES WITH CONFIDENCE AND EXICTMENT.

DRIVE LINK OF SQL FILE:

HTTPS://DRIVE.GOOGLE.COM/FILE/D/19R9QGFEEMO3EKJKKP_QZBKLL2UAPMNOD/VIEW?USP=SHARE_LINK