Project 3

Operation Analytics and Investigating Metric Spike

Sri Harshitha Gaddamanugu

Project Description

- This Project is about understanding the working of Operational Analytics of an organization that helps them identify areas which need improvements.
- It also involves investigating Metric Spikes which can explain sudden rise or drop in key metrics.
- These analysis are to be done using Data Analytics tools and processes.
- As a Lead Data Analyst in Microsoft, we received datasets and queries from various departments and we are required to provide insights to those queries.
- To do this, we will use SQL to extract and filter data from the datasets provided and analyse them to get the required insights.

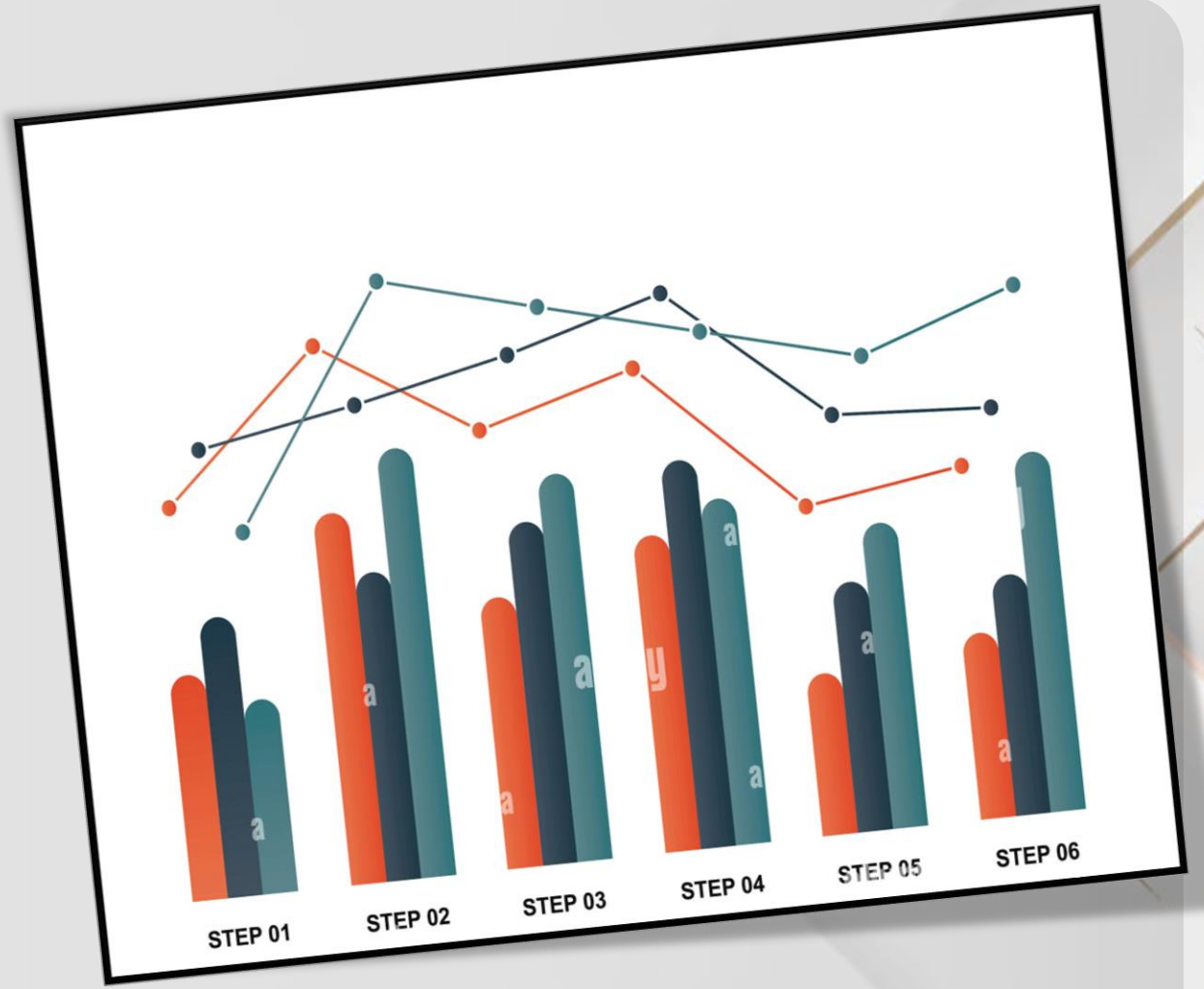


Image Source: Google Images

Approach

- This project has two Case Studies. Case Study 1 is related to job reviews of candidates and Case Study 2 is related to events that users have performed in online space.

- For **Case Study 1**, we had one table named **job_data** which has information about job id, actor id, time to review, date of review, language and outcome of review.

- For **Case Study 2**, we had three tables, **users** with information in each row, about a user's account, **events** with information in each row, about any event in online space a user has performed and **email_events** with information in each row, about any event related to emails the user has performed.

Image Source: Google Images

Approach

- We first studied and tried to understand the data for each case study. Then we started extracting information from the data as per the questions we had (we would have processed the data like cleaning, transforming etc. if required).

- Extraction of data was done using **pandas** and **pymysql** libraries in **python**. We also plotted different types of graphs for better understanding of the data. Then from extracted data and graphs, we were able to drew conclusions and insights. Now, since we got all the insights required, we will be able to make a detailed report about it.

Image Source: Google Images

Tech-Stack Used

- **MySQL Workbench 8.0.33 Community Version** – A relational database management system used to create the database. It's connections are used to connect the database with Python.
- **Python 3.10.9 –** Programming language used to extract and filter data as per requirements using SQL Queries.
- **Jupyter Notebook 6.5.2 –** Interactive platform to write and execute codes in various programming languages (in this case Python).

Image Source: Google Images

Insights

Case Study 1: Job Data Analysis

Table: **job_data**
Table name used in project: **cs_1**
Description: **Contains details about job reviews**

P.S.: The original table had only 8 rows. Analyzing this small table wouldn't had provided much accurate insights to us. So I had added extra rows in the table. The modified table can be found in the given link. (**https://drive.google.com/file/d/1-F_bwHDe5GDzNiGBgFMg9iJlGtA0rPG_/view?usp=sharing**)

Insights

Case Study 1: Job Data Analysis

A. Jobs Reviewed Over Time:

Objective: Calculate the number of jobs reviewed per hour for each day in November 2020.

Task: Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.

Query:-

```
query='''SELECT ds AS Date, COUNT(job_id) AS Cnt_JID, ROUND((SUM(time_spent)/3600),2) AS Tot_Time_Sp_Hr,
            ROUND((COUNT(job_id)/(SUM(time_spent)/3600)),2) AS Job_Rev_PHr_PDy
            FROM cs_1
            WHERE ds BETWEEN \'01-11-2020\' AND \'30-11-2020\'
            GROUP BY ds
            ORDER BY ds'''

df1=pd.read_sql_query(query, conn)
```

Insights

Case Study 1: Job Data Analysis

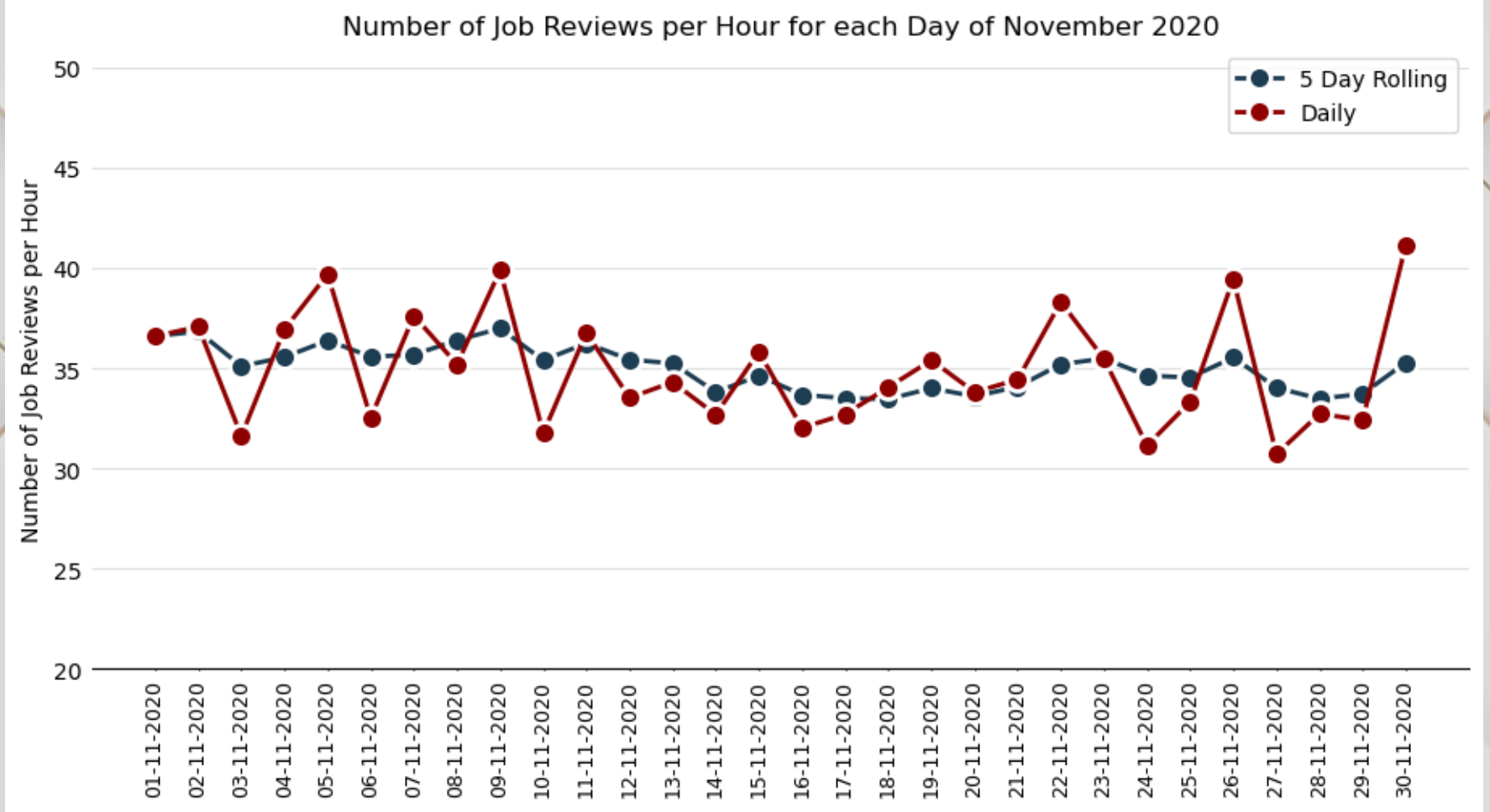
Result:-

	Date	Cnt_JID	Tot_Time_Sp_Hr	Job_Rev_PHr_PDy		Date	Cnt_JID	Tot_Time_Sp_Hr	Job_Rev_PHr_PDy
0	01-11-2020	39	1.07	36.60	**15**	16-11-2020	44	1.37	32.03
1	02-11-2020	40	1.08	37.08	**16**	17-11-2020	49	1.50	32.71
2	03-11-2020	25	0.79	31.63	**17**	18-11-2020	36	1.06	34.05
3	04-11-2020	34	0.92	36.95	**18**	19-11-2020	32	0.90	35.41
4	05-11-2020	36	0.91	39.69	**19**	20-11-2020	31	0.92	33.81
5	06-11-2020	32	0.98	32.53	**20**	21-11-2020	27	0.78	34.44
6	07-11-2020	42	1.12	37.62	**21**	22-11-2020	41	1.07	38.31
7	08-11-2020	29	0.82	35.20	**22**	23-11-2020	46	1.30	35.48
8	09-11-2020	36	0.90	39.89	**23**	24-11-2020	42	1.35	31.15
9	10-11-2020	41	1.29	31.77	**24**	25-11-2020	38	1.14	33.33
10	11-11-2020	32	0.87	36.77	**25**	26-11-2020	32	0.81	39.45
11	12-11-2020	33	0.98	33.53	**26**	27-11-2020	45	1.46	30.72
12	13-11-2020	23	0.67	34.31	**27**	28-11-2020	38	1.16	32.73
13	14-11-2020	37	1.13	32.69	**28**	29-11-2020	38	1.17	32.42
14	15-11-2020	36	1.00	35.83	**29**	30-11-2020	41	1.00	41.15

Insights

Case Study 1: Job Data Analysis

Result:-



Insights

Case Study 1: Job Data Analysis

Insights:-

- From the table, we can see that the number of **job reviews** done for **most** days of November 2020 was between **30** and **45**.

- From the table, we can observe that the **total time spent** for job reviews on **most** days of November 2020 was between **0.9** and **1.4 hours**.

- From the plot, we cannot observe any particular trend, but **hourly rate of job reviews** for each day is between **30** and **42**.

- From the plot, if we observe the last four days, which were Friday, Saturday, Sunday and Monday, the **low hourly review rate** of **Friday**, **Saturday** and **Sunday** can be contributed to **high lethargy** of employees due to **weekend** time. The **high hourly review rate** on **Monday** can be contributed to the **last working day** of the month and also to the **pile up of job reviews** of Friday, Saturday and Sunday.

- From the plot, we can observe that the **5 day rolling average** (we calculated this metric using pandas library of python programming language) has an **overall slight downward** trend. This signifies that the hourly rate of job reviews is slowly decreasing.

Insights

Case Study 1: Job Data Analysis

B. Throughput Analysis:

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

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.

Query:-

```
query="""WITH d AS (
            SELECT ds, CAST(COUNT(job_id) AS FLOAT)/CAST(SUM(time_spent) AS FLOAT) AS c_by_s
            FROM cs_1
            WHERE ds BETWEEN \'01-11-2020\' AND \'30-11-2020\'
            GROUP BY 1 )

            SELECT ds AS Date, c_by_s AS Job_Rev_PSec_PDy, AVG(c_by_s) OVER(ORDER BY ds ROWS BETWEEN 6 PRECEDING AND
            CURRENT ROW) AS 7_Dy_Roll_Avg
            FROM d"""

df2=pd.read_sql_query(query, conn)
```

Insights

Case Study 1: Job Data Analysis

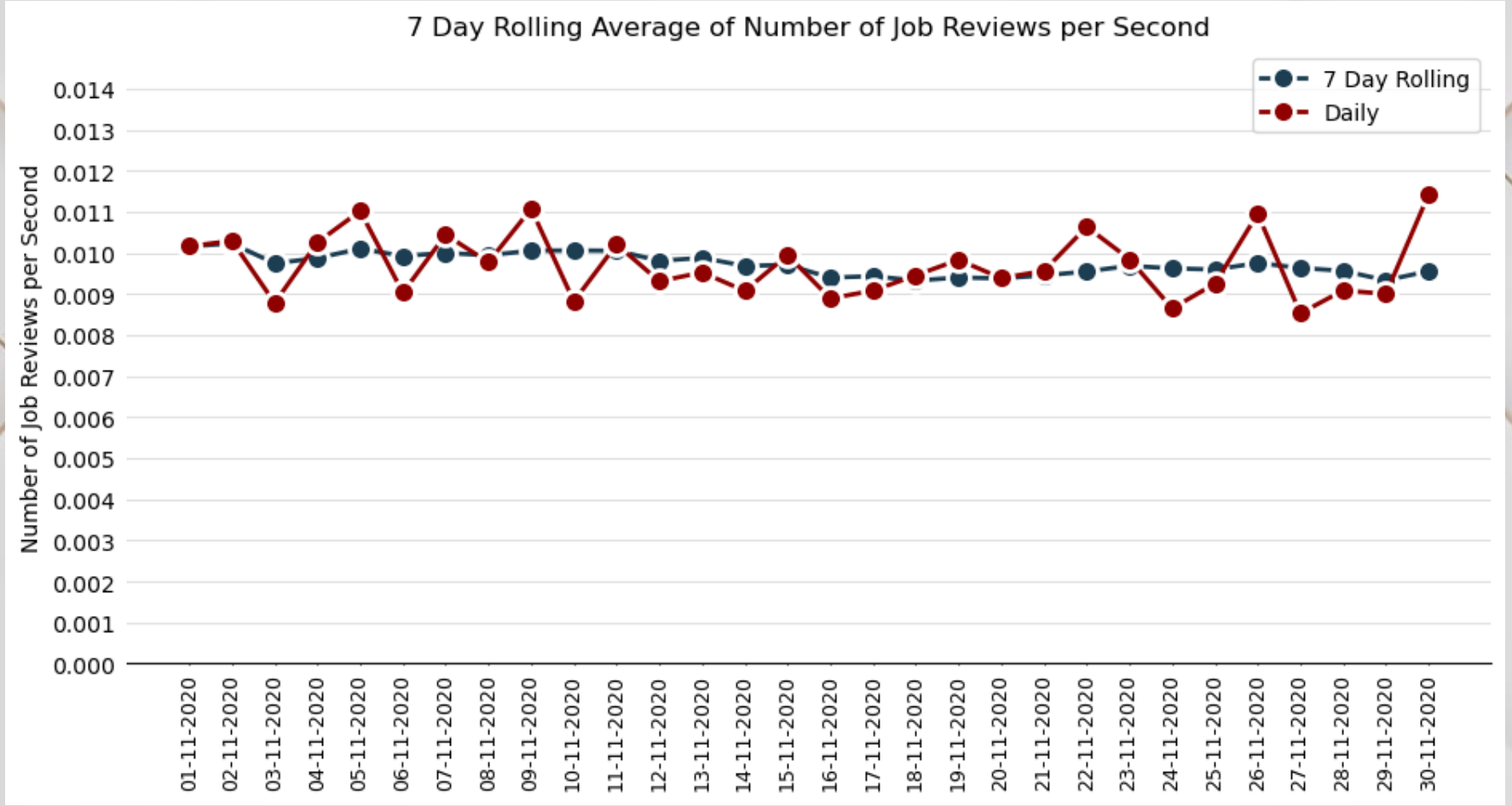
Result:-

	Date	Job_Rev_PSec_PDy	7_Dy_Roll_Avg		Date	Job_Rev_PSec_PDy	7_Dy_Roll_Avg
0	01-11-2020	0.010167	0.010167	**15**	16-11-2020	0.008896	0.009402
1	02-11-2020	0.010301	0.010234	**16**	17-11-2020	0.009086	0.009439
2	03-11-2020	0.008787	0.009752	**17**	18-11-2020	0.009459	0.009331
3	04-11-2020	0.010263	0.009880	**18**	19-11-2020	0.009837	0.009406
4	05-11-2020	0.011026	0.010109	**19**	20-11-2020	0.009391	0.009386
5	06-11-2020	0.009037	0.009930	**20**	21-11-2020	0.009568	0.009456
6	07-11-2020	0.010450	0.010004	**21**	22-11-2020	0.010641	0.009554
7	08-11-2020	0.009777	0.009949	**22**	23-11-2020	0.009854	0.009691
8	09-11-2020	0.011080	0.010060	**23**	24-11-2020	0.008653	0.009629
9	10-11-2020	0.008825	0.010066	**24**	25-11-2020	0.009257	0.009600
10	11-11-2020	0.010214	0.010059	**25**	26-11-2020	0.010959	0.009760
11	12-11-2020	0.009314	0.009814	**26**	27-11-2020	0.008534	0.009638
12	13-11-2020	0.009532	0.009885	**27**	28-11-2020	0.009091	0.009570
13	14-11-2020	0.009080	0.009689	**28**	29-11-2020	0.009007	0.009336
14	15-11-2020	0.009953	0.009714	**29**	30-11-2020	0.011430	0.009562

Insights

Case Study 1: Job Data Analysis

Result:-



Insights

Case Study 1: Job Data Analysis

Insights:-

- From the daily line plot, we cannot observe any particular trend, but **job reviews per second** for each day is between **0.008** and **0.012**.

- From the 7-day rolling average plot, we can observe that the **7 day rolling average** has an **overall slight downward** trend. This signifies that the rate of job reviews is slowly decreasing.

- We should prefer the **7-Day Rolling Average** over daily metric for thoroughput. The reason being daily metrics can go up or down on a daily basis for **factors that cannot be controlled by the organizations** like seasonality, major events etc. Sudden spikes caused by these factors can give a **false signal** which can prompt the organization to take steps which may **prove to be harmful**. So to get a real sense of the thoroughput data, we should use the 7-day rolling average as they are **very less impacted** by above mentioned factors and can **give a realistic sense** of the data.

Insights

Case Study 1: Job Data Analysis

C. Language Share Analysis:

Objective: Calculate the percentage share of each language in the last 30 days.

Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.

Query:-

```
query="""WITH d AS (
            SELECT language, COUNT(language) AS cnt
            FROM cs_1
            WHERE ds BETWEEN \'01-11-2020\' AND \'30-11-2020\'
            GROUP BY language )

            SELECT language AS Language, cnt AS Tot_Cnt_Lang, ROUND((100*cnt/SUM(cnt) OVER()), 2) AS Perc_Share_Lang
            FROM d
            ORDER BY Perc_Share_Lang DESC"""

df3=pd.read_sql_query(query, conn)
```

Insights

Case Study 1: Job Data Analysis

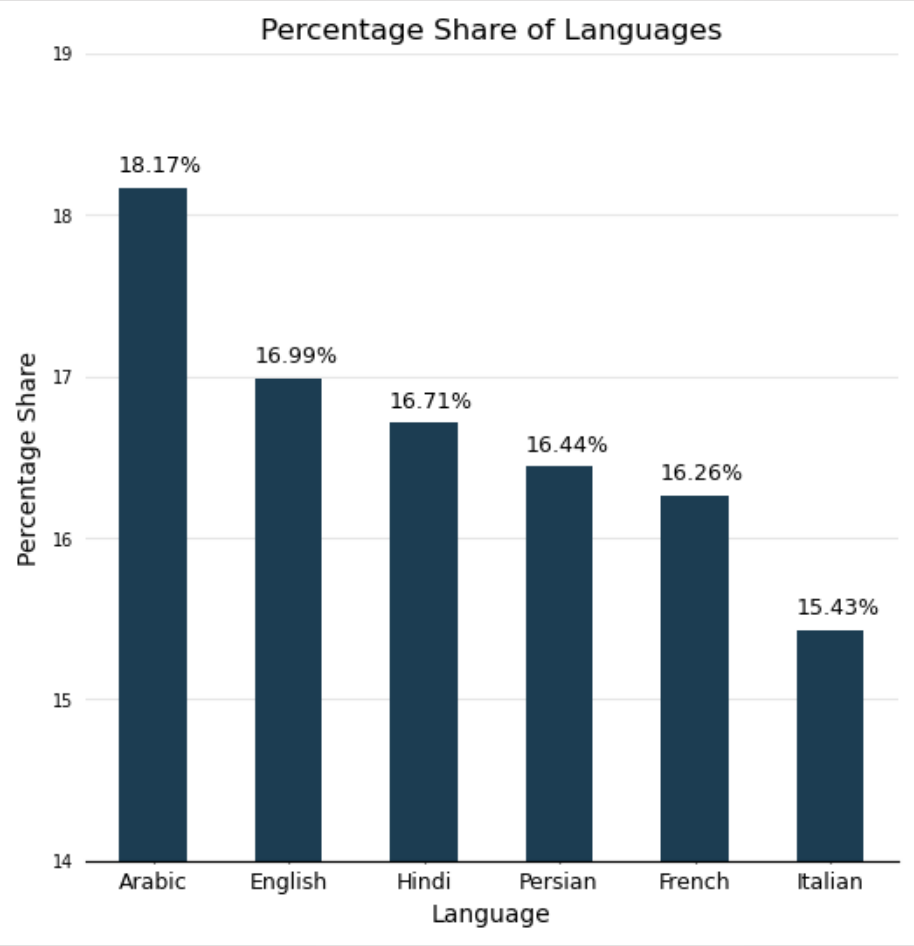
Result:-

	Language	Tot_Cnt_Lang	Perc_Share_Lang
0	Arabic	199	18.17
1	English	186	16.99
2	Hindi	183	16.71
3	Persian	180	16.44
4	French	178	16.26
5	Italian	169	15.43

Insights

Case Study 1: Job Data Analysis

Result:-



Insights

Case Study 1: Job Data Analysis

Insights:-

- From the table and the plot, we can observe that **Arabic** language is the most used language with percentage share of **18.17%** followed by **English** with **16.99%** and **Hindi** with **16.71%**.

- Although Arabic is the most used language, the percentage share of languages is **not skewed** in any one language's favour.

Insights

Case Study 1: Job Data Analysis

D. Duplicate Rows Detection:

Objective: Identify duplicate rows in the data.

Task: Write an SQL query to display duplicate rows from the job_data table.

Query:-

>> First we will check whether there are rows which are entirely duplicate.

```
query="""SELECT *
            FROM cs_1
            GROUP BY ds, job_id, actor_id, event, language,
            time_spent, org
            HAVING COUNT(*)>1"""

df4a=pd.read_sql_query(query, conn)
```

Result:-

ds	job_id	actor_id	event	language	time_spent	org

Insights:-

- All the rows are unique when considered in its entirety

Insights

Case Study 1: Job Data Analysis

D. Duplicate Rows Detection:

Objective: Identify duplicate rows in the data.

Task: Write an SQL query to display duplicate rows from the job_data table.

Query:-

>> Second we will check whether there are rows which have duplicate **job_id** values. The below query gives all the **job_id**s with duplicate entries along with their counts.

```
query1a="""SELECT job_id, COUNT(job_id) AS Cnt_JID FROM cs_1
            GROUP BY job_id
            HAVING Cnt_JID>1
            ORDER BY job_id"""

df4b=pd.read_sql_query(query1a, conn)
```

Insights

Case Study 1: Job Data Analysis

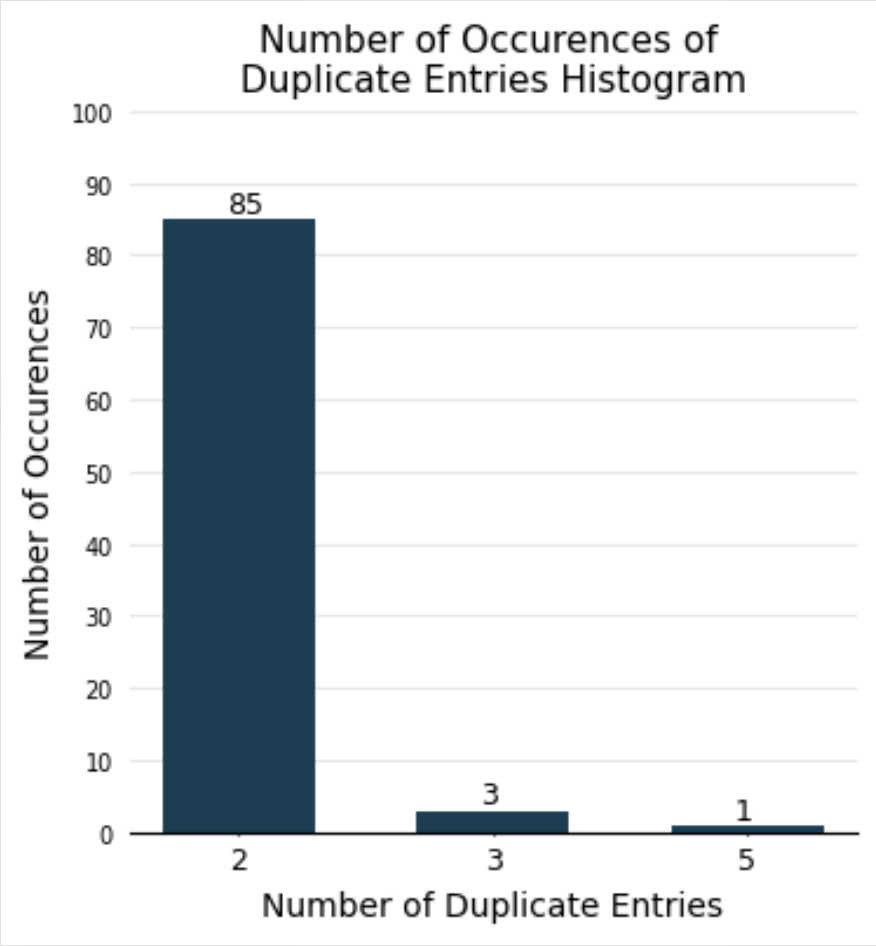
Result:-

	job_id	Cnt_JID		job_id	Cnt_JID		job_id	Cnt_JID		job_id	Cnt_JID		job_id	Cnt_JID		job_id	Cnt_JID		job_id	Cnt_JID		job_id	Cnt_JID
0	1	2	**11**	12	2	**22**	23	5	**33**	34	2	**44**	45	2	**55**	56	2	**66**	67	2	**77**	78	2
1	2	2	**12**	13	2	**23**	24	2	**34**	35	2	**45**	46	2	**56**	57	2	**67**	68	2	**78**	79	2
2	3	2	**13**	14	2	**24**	25	3	**35**	36	2	**46**	47	2	**57**	58	2	**68**	69	2	**79**	80	2
3	4	2	**14**	15	2	**25**	26	2	**36**	37	2	**47**	48	2	**58**	59	2	**69**	70	2	**80**	81	2
4	5	2	**15**	16	2	**26**	27	2	**37**	38	2	**48**	49	2	**59**	60	2	**70**	71	2	**81**	82	2
5	6	2	**16**	17	2	**27**	28	2	**38**	39	2	**49**	50	2	**60**	61	2	**71**	72	2	**82**	83	2
6	7	2	**17**	18	2	**28**	29	2	**39**	40	2	**50**	51	2	**61**	62	2	**72**	73	2	**83**	84	2
7	8	2	**18**	19	2	**29**	30	2	**40**	41	2	**51**	52	2	**62**	63	2	**73**	74	2	**84**	85	2
8	9	2	**19**	20	3	**30**	31	2	**41**	42	2	**52**	53	2	**63**	64	2	**74**	75	2	**85**	86	2
9	10	2	**20**	21	2	**31**	32	2	**42**	43	2	**53**	54	2	**64**	65	2	**75**	76	2	**86**	87	2
10	11	3	**21**	22	2	**32**	33	2	**43**	44	2	**54**	55	2	**65**	66	2	**76**	77	2	**87**	88	2
																					88	89	2

Insights

Case Study 1: Job Data Analysis

Result:-



Insights

Case Study 1: Job Data Analysis

D. Duplicate Rows Detection:

Objective: Identify duplicate rows in the data.

Task: Write an SQL query to display duplicate rows from the job_data table.

Query:-

>> The below query gives all the row entries whose **job_id**s are given above.

```
query1b="""SELECT cs_1.*
            FROM cs_1
            JOIN ({}) AS q1
            ON cs_1.job_id=q1.job_id
            ORDER BY cs_1.job_id""".format(query1a)

df4c=pd.read_sql_query(query1b, conn)
```

Insights

Case Study 1: Job Data Analysis

Result:-

	ds	job_id	actor_id	event	language	time_spent	org		ds	job_id	actor_id	event	language	time_spent	org		ds	job_id	actor_id	event	language	time_spent	org		ds	job_id	actor_id	event	language	time_spent	org
0	02-11-2020	1	1819	decision	Persian	32	A	46	29-11-2020	23	1003	decision	Persian	20	C	92	29-11-2020	44	1223	transfer	English	95	B	138	21-11-2020	67	1920	decision	Hindi	42	A
1	01-11-2020	1	1509	transfer	Arabic	20	D	47	28-11-2020	23	1005	transfer	Persian	22	D	93	06-11-2020	44	1994	transfer	Italian	141	B	139	16-11-2020	67	1707	decision	Persian	198	C
2	10-11-2020	2	1270	transfer	Italian	175	D	48	26-11-2020	23	1004	skip	Persian	56	A	94	18-11-2020	45	1701	transfer	English	108	D	140	20-11-2020	68	1936	skip	Arabic	154	C
3	24-11-2020	2	2011	decision	Persian	169	C	49	29-11-2020	23	1791	decision	Hindi	137	C	95	15-11-2020	45	1085	skip	French	65	C	141	25-11-2020	68	1559	transfer	Arabic	107	B
4	15-11-2020	3	1266	skip	Hindi	48	D	50	02-11-2020	23	2040	skip	Persian	110	C	96	30-11-2020	46	2062	decision	Persian	34	D	142	29-11-2020	69	1353	transfer	French	190	B
5	18-11-2020	3	2024	decision	French	60	A	51	17-11-2020	24	1542	decision	Italian	93	C	97	09-11-2020	46	1811	skip	Hindi	16	C	143	16-11-2020	69	1138	transfer	Arabic	84	C
6	29-11-2020	4	1387	decision	Hindi	180	C	52	16-11-2020	24	1585	transfer	Arabic	72	B	98	20-11-2020	47	1416	decision	Italian	166	A	144	18-11-2020	70	1408	skip	Persian	163	A
7	08-11-2020	4	1512	skip	English	200	A	53	28-11-2020	25	1002	decision	Hindi	11	B	99	05-11-2020	47	1634	decision	English	58	A	145	04-11-2020	70	1891	decision	French	98	D
8	23-11-2020	5	1966	skip	Hindi	52	B	54	26-11-2020	25	1525	decision	French	23	B	100	04-11-2020	48	1751	skip	Arabic	179	C	146	08-11-2020	71	1957	skip	Italian	64	A
9	23-11-2020	5	1316	decision	Persian	143	D	55	08-11-2020	25	1268	transfer	Arabic	19	D	101	02-11-2020	48	1800	transfer	English	150	A	147	30-11-2020	71	1439	decision	English	133	C
10	06-11-2020	6	1145	transfer	English	136	C	56	23-11-2020	26	1768	decision	Persian	169	D	102	13-11-2020	49	1750	transfer	Persian	66	C	148	06-11-2020	72	1782	decision	Persian	151	D
11	03-11-2020	6	2067	skip	English	119	B	57	08-11-2020	26	1452	decision	Arabic	14	D	103	12-11-2020	49	1665	skip	Arabic	139	B	149	04-11-2020	72	1656	skip	Persian	94	B
12	07-11-2020	7	1949	skip	Persian	175	C	58	27-11-2020	27	1753	skip	Italian	91	A	104	26-11-2020	50	1982	decision	Arabic	89	C	150	04-11-2020	73	1481	transfer	French	82	C
13	07-11-2020	7	1779	decision	Italian	115	B	59	28-11-2020	27	1526	skip	Persian	178	D	105	29-11-2020	50	1431	transfer	Arabic	34	C	151	20-11-2020	73	1739	skip	English	11	C
14	14-11-2020	8	1621	skip	Italian	42	A	60	03-11-2020	28	1207	decision	Arabic	165	C	106	14-11-2020	51	1799	skip	Hindi	178	C	152	28-11-2020	74	1805	skip	Hindi	62	B
15	09-11-2020	8	1399	skip	Hindi	56	C	61	16-11-2020	28	1757	skip	English	185	D	107	16-11-2020	51	1034	skip	Hindi	58	A	153	25-11-2020	74	1190	skip	Hindi	10	D
16	11-11-2020	9	1262	decision	Hindi	37	C	62	18-11-2020	29	1862	transfer	Persian	12	D	108	16-11-2020	52	1297	skip	Italian	136	B	154	17-11-2020	75	2043	decision	Arabic	169	B
17	14-11-2020	9	1892	decision	Italian	187	A	63	12-11-2020	29	1650	skip	Arabic	24	C	109	27-11-2020	52	1258	decision	Persian	24	B	155	06-11-2020	75	1999	transfer	French	92	A
18	28-11-2020	10	1586	transfer	French	117	D	64	24-11-2020	30	1057	transfer	French	187	D	110	15-11-2020	53	1636	decision	English	80	C	156	04-11-2020	76	1090	transfer	Persian	81	C
19	30-11-2020	10	1492	transfer	English	76	B	65	13-11-2020	30	1017	skip	French	35	B	111	14-11-2020	53	1209	decision	Italian	46	A	157	10-11-2020	76	1868	decision	Italian	133	C
20	27-11-2020	11	1007	decision	French	104	D	66	07-11-2020	31	1839	transfer	French	74	D	112	12-11-2020	54	1508	skip	Italian	168	B	158	30-11-2020	77	1099	decision	Italian	32	A
21	02-11-2020	11	1499	decision	Persian	151	D	67	26-11-2020	31	2042	decision	English	53	D	113	25-11-2020	54	1903	decision	French	62	C	159	23-11-2020	77	1963	decision	French	31	A
22	28-11-2020	11	1489	skip	Hindi	190	D	68	04-11-2020	32	1619	decision	Hindi	28	B	114	22-11-2020	55	2038	decision	French	190	A	160	19-11-2020	78	1203	skip	Hindi	28	B
23	07-11-2020	12	1833	skip	Arabic	99	B	69	28-11-2020	32	1165	decision	English	181	A	115	22-11-2020	55	1036	decision	Hindi	47	C	161	23-11-2020	78	1567	transfer	Italian	62	C
24	04-11-2020	12	1709	skip	Persian	165	D	70	09-11-2020	33	1817	transfer	Hindi	29	B	116	04-11-2020	56	1597	decision	Hindi	87	C	162	27-11-2020	79	2025	skip	English	155	B
25	29-11-2020	13	1976	skip	French	142	B	71	16-11-2020	33	1703	decision	English	110	C	117	02-11-2020	56	1127	transfer	French	146	B	163	09-11-2020	79	1141	transfer	French	112	D
26	27-11-2020	13	1483	skip	French	199	D	72	07-11-2020	34	1977	decision	French	172	A	118	15-11-2020	57	1911	transfer	French	159	B	164	21-11-2020	80	1260	transfer	Hindi	173	A
27	02-11-2020	14	1043	skip	Persian	94	D	73	23-11-2020	34	1886	transfer	Persian	103	A	119	11-11-2020	57	1181	skip	Italian	199	C	165	11-11-2020	80	1992	skip	English	41	A
28	14-11-2020	14	1565	decision	Arabic	12	D	74	17-11-2020	35	1577	decision	French	87	D	120	24-11-2020	58	1616	decision	Persian	152	D	166	27-11-2020	81	1940	decision	Hindi	162	A
29	25-11-2020	15	1116	decision	Hindi	45	A	75	05-11-2020	35	1500	skip	Persian	86	C	121	25-11-2020	58	1351	skip	Hindi	173	D	167	20-11-2020	81	1235	skip	Italian	66	B
30	16-11-2020	15	1959	transfer	Italian	183	B	76	29-11-2020	36	1735	skip	French	84	D	122	21-11-2020	59	1899	transfer	English	110	C	168	18-11-2020	82	1845	skip	Hindi	145	D
31	02-11-2020	16	1084	skip	Hindi	170	A	77	07-11-2020	36	1362	transfer	French	98	D	123	05-11-2020	59	1938	transfer	Persian	139	A	169	07-11-2020	82	1022	transfer	Arabic	63	A
32	10-11-2020	16	1035	transfer	Hindi	19	A	78	02-11-2020	37	1717	skip	Arabic	49	D	124	06-11-2020	60	1097	transfer	Arabic	61	D	170	30-11-2020	83	1276	skip	Arabic	64	D
33	13-11-2020	17	1040	skip	French	44	D	79	29-11-2020	37	1880	skip	Hindi	65	C	125	03-11-2020	60	1657	skip	Arabic	191	B	171	17-11-2020	83	1252	skip	French	91	B
34	17-11-2020	17	1132	skip	Hindi	76	A	80	30-11-2020	38	1692	transfer	Hindi	160	A	126	26-11-2020	61	1651	decision	Italian	22	A	172	03-11-2020	84	1945	transfer	English	165	A
35	19-11-2020	18	1008	decision	Italian	89	C	81	09-11-2020	38	1309	decision	Arabic	114	B	127	30-11-2020	61	1184	decision	Arabic	43	B	173	15-11-2020	84	1369	transfer	Italian	198	B
36	15-11-2020	18	1677	transfer	Hindi	37	B	82	16-11-2020	39	1334	skip	Italian	141	D	128	10-11-2020	62	1822	transfer	Italian	134	A	174	22-11-2020	85	1852	decision	French	179	C
37	02-11-2020	19	1345	transfer	Persian	160	C	83	03-11-2020	39	1541	decision	Arabic	30	A	129	01-11-2020	62	1678	decision	Persian	145	B	175	29-11-2020	85	2054	skip	Hindi	84	B
38	27-11-2020	19	1767	transfer	Arabic	33	A	84	13-11-2020	40	1697	decision	French	170	A	130	09-11-2020	63	1243	transfer	English	51	D	176	09-11-2020	86	1082	transfer	English	29	D
39	25-11-2020	20	1003	transfer	Italian	45	C	85	01-11-2020	40	1104	decision	Hindi	47	C	131	26-11-2020	63	1558	skip	Arabic	129	B	177	25-11-2020	86	1548	decision	Arabic	177	D
40	06-11-2020	20	1280	transfer	Persian	114	D	86	15-11-2020	41	1402	skip	Hindi	26	D	132	11-11-2020	64	1795	transfer	Italian	147	B	178	19-11-2020	87	1700	transfer	English	125	D
41	26-11-2020	20	2032	skip	English	22	B	87	11-11-2020	41	1926	decision	Arabic	108	D	133	25-11-2020	64	1536	decision	Hindi	180	B	179	10-11-2020	87	1560	skip	Italian	171	B
42	30-11-2020	21	1001	skip	English	15	A	88	07-11-2020	42	1618	decision	Hindi	107	D	134	05-11-2020	65	1341	transfer	Persian	160	C	180	22-11-2020	88	1910	decision	French	26	D
43	27-11-2020	21	1929	transfer	English	197	D	89	21-11-2020	42	1123	skip	Arabic	80	D	135	16-11-2020	65	1476	decision	Hindi	56	B	181	12-11-2020	88	1583	skip	Arabic	175	B
44	30-11-2020	22	1006	transfer	Arabic	25	B	90	03-11-2020	43	1725	skip	English	194	B	136	05-11-2020	66	1506	transfer	English	147	B	182	24-11-2020	89	2029	transfer	Hindi	123	A
45	10-11-2020	22	1149	decision	French	139	D	91	06-11-2020	43	1587	skip	French	160	D	137	16-11-2020	66	1978	decision	English	26	C	183	18-11-2020	89	1871	skip	Arabic	186	B

Insights:-

- There are **89** number of rows with duplicate values of **job_id**.
- Out of those 89 number of **job_id**s, **85** have **2** number of duplicate rows, **3** have **3** number of duplicate rows and **1** have **5** number of duplicate rows.
- These rows needs to be either removed or find the correct **job_id** and replace them here.

Insights

Case Study 1: Job Data Analysis

D. Duplicate Rows Detection:

Objective: Identify duplicate rows in the data.

Task: Write an SQL query to display duplicate rows from the job_data table.

Query:-

>> Third we will check whether there are rows which have duplicate **actor_id** values. The below left query gives all the **actor_id**s with duplicate entries along with their counts and the below right query gives all the row entries whose **actor_id**s are given above.

```
query2a="""SELECT actor_id, COUNT(actor_id) AS Cnt_AID
            FROM cs_1
            GROUP BY actor_id
            HAVING Cnt_AID>1
            ORDER BY actor_id"""

df4d=pd.read_sql_query(query2a, conn)
```

```
query2b="""SELECT cs_1.*
            FROM cs_1
            JOIN ({}) AS q1
            ON cs_1.actor_id=q1.actor_id
            ORDER BY cs_1.actor_id""".format(query2a)

df4e=pd.read_sql_query(query2b, conn)
```

Insights

Case Study 1: Job Data Analysis

Result:-

Query2a:-

	actor_id	Cnt_AID
0	1001	2
1	1002	2
2	1003	3
3	1004	2
4	1005	2
5	1006	2
6	1007	2

Query2b:-

	ds	job_id	actor_id	event	language	time_spent	org
0	30-11-2020	21	1001	skip	English	15	A
1	04-11-2020	734	1001	skip	English	75	A
2	28-11-2020	25	1002	decision	Hindi	11	B
3	17-11-2020	365	1002	transfer	Italian	138	C
4	29-11-2020	23	1003	decision	Persian	20	C
5	25-11-2020	20	1003	transfer	Italian	45	C
6	18-11-2020	325	1003	transfer	Persian	79	B
7	26-11-2020	23	1004	skip	Persian	56	A
8	21-11-2020	919	1004	skip	Persian	73	D
9	28-11-2020	23	1005	transfer	Persian	22	D
10	02-11-2020	575	1005	skip	Persian	51	A
11	30-11-2020	22	1006	transfer	Arabic	25	B
12	26-11-2020	606	1006	transfer	French	94	C
13	27-11-2020	11	1007	decision	French	104	D
14	12-11-2020	905	1007	decision	English	108	B

Insights

Case Study 1: Job Data Analysis

Insights:-

- There are **7** number of rows with duplicate values of **actor_id**.
- Out of those 89 number of **actor_id**s, **6** have **2** number of duplicate rows and **1** have **3** number of duplicate rows.
- These rows needs to be either removed or find the correct **actor_id** and replace them here.

Insights

Case Study 2: Investigating Metric Spike

1. Table: **users**
   Table name used in project: **cs_2_t_1**
   Description: **Contains descriptive information about a user's account**

2. Table: **events**
   Table name used in project: **cs_2_t_2**
   Description: **Contains information about an event that a user has taken**

3. Table: **email_events**
   Table name used in project: **cs_2_t_3**
   Description: **Contains information about events specific to sending of emails**

Insights

Case Study 2: Investigating Metric Spike

A. Weekly User Engagement:

Objective: Measure the activeness of users on a weekly basis.

Task: Write an SQL query to calculate the weekly user engagement.

Query:-

```
query1='''SELECT user_id, TIMESTAMPDIFF(WEEK, \'2013-01-01 04:40:10\', DATE_FORMAT(STR_TO_DATE(occurred_at, \'%d-%m-%Y
            %H:%i\'), \'%Y-%m-%d %H:%i:%S\')) AS wk
            FROM cs_2_t_2
            WHERE event_type = \'engagement\''''

query2='''SELECT user_id, TIMESTAMPDIFF(WEEK, \'2013-01-01 04:40:10\', occurred_at) AS wk
            FROM cs_2_t_3'''

query3='''SELECT *
            FROM ({}) AS q1

            UNION ALL

            SELECT * FROM ({}) AS q2
            ORDER BY user_id'''.format(query1, query2)

query4='''SELECT user_id, wk, COUNT(user_id) AS Count
            FROM ({}) AS q3
            GROUP BY user_id, wk
            ORDER BY user_id'''.format(query3)

query5='''SELECT ROUND(AVG(Count),2) AS Avg_Weekly_User_Eng
            FROM ({}) AS q4 '''.format(query4)

df1=pd.read_sql_query(query5, conn)
```

Insights

Case Study 2: Investigating Metric Spike

Result:-

	Avg_Weekly_User_Eng
0	6.43

Insights:-

- From our calculations above, the Average Weekly User Engagement is **6.43**. That is, on an average, users engage in **6.43** events on weekly basis.

Insights

Case Study 2: Investigating Metric Spike

B. User Growth Analysis:

Objective: Analyze the growth of users over time for a product.

Task: Write an SQL query to calculate the user growth for the product.

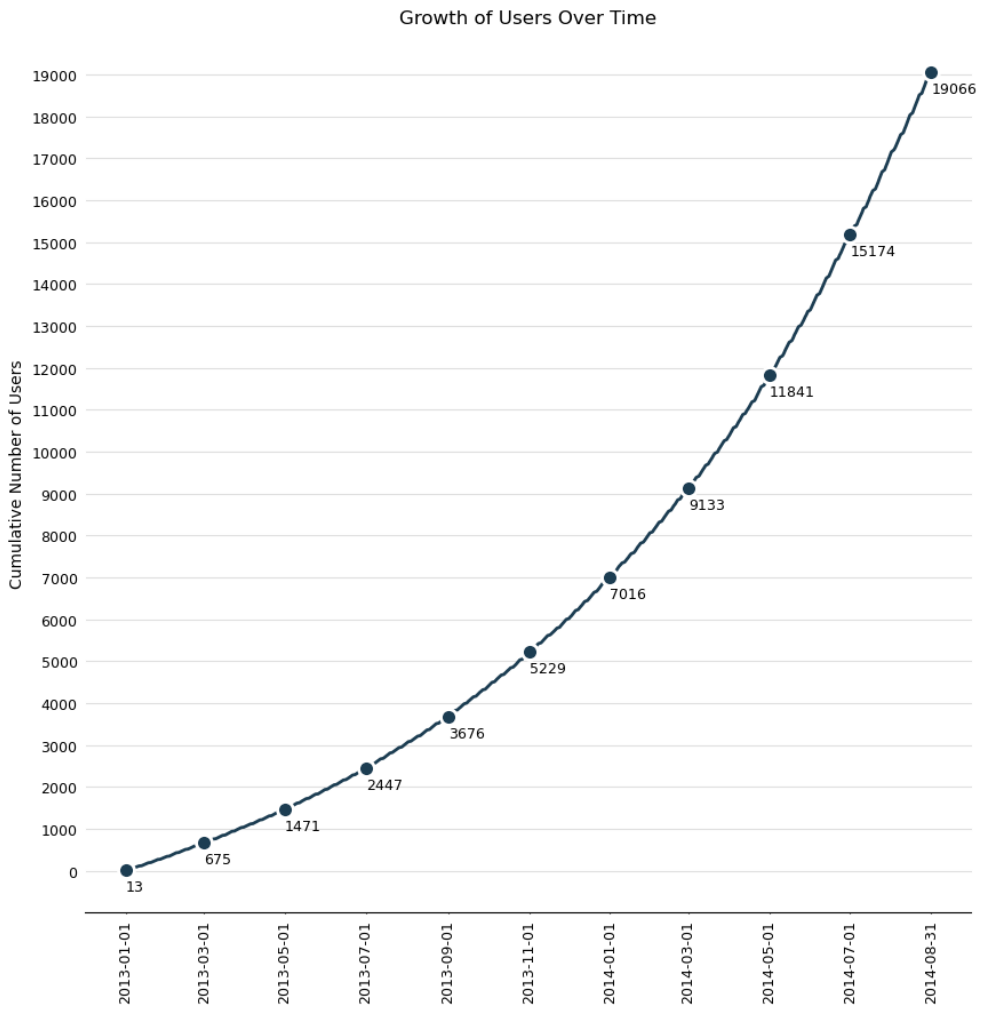
Query:-

```
query1='''SELECT DATE(STR_TO_DATE(created_at, \'%Y-%m-%d %H:%i:%s\')) AS dt, COUNT(user_id) AS cnt
            FROM cs_2_t_1
            GROUP BY dt'''

query2='''SELECT dt, cnt, SUM(cnt) OVER (ORDER BY dt) DIV 1 AS cml_cnt
            FROM ({}) AS q1'''.format(query1)

df2=pd.read_sql_query(query2, conn)
```

Insights

Case Study 2: Investigating Metric Spike

Result:



Insights:-

- From the plot, we can observe that user growth is experiencing almost **exponential** growth. Here the number of users increased from **13** on **1st January 2013** to **19066** on **31st August 2014**.

- **Exponential** growth indicates a **positive** sign for the organization.

Insights

Case Study 2: Investigating Metric Spike

C. Weekly Retention Analysis:

Objective: Analyze the retention of users on a weekly basis after signing up for a product.

Task: Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

Query:-

```
query1='''SELECT user_id, TIMESTAMPDIFF(WEEK, \'2013-01-01 04:40:10\', activated_at) as week_no
            FROM cs_2_t_1
            WHERE state = \'active\'''‘


query2='''SELECT user_id, DATE_FORMAT(STR_TO_DATE(occurred_at, \'%d-%m-%Y %H:%i\'), \'%Y-%m-%d %H:%i:%S\') AS date_time,
            TIMESTAMPDIFF(WEEK, \'2013-01-01 04:40:10\', DATE_FORMAT(STR_TO_DATE(occurred_at, \'%d-%m-%Y %H:%i\'), \'%Y-%m-
            %d %H:%i:%S\')) as week_no
            FROM cs_2_t_2 WHERE event_type = \'engagement\‘
            UNION ALL
            SELECT user_id, occurred_at AS date_time, TIMESTAMPDIFF(WEEK, \'2013-01-01 04:40:10\', occurred_at) as week_no
            FROM cs_2_t_3 ORDER BY user_id, week_no'''


query3='''SELECT q1.user_id AS q1id, q1.week_no AS q1wn, q2.user_id AS q2id, q2.week_no AS q2wn
            FROM ({0}) AS q1
            LEFT OUTER JOIN ({1}) AS q2 ON q1.user_id=q2.user_id
            UNION
            SELECT q1.user_id AS q1id, q1.week_no AS q1wn, q2.user_id AS q2id, q2.week_no AS q2wn
            FROM ({0}) AS q1
            RIGHT OUTER JOIN ({1}) AS q2  ON q1.user_id=q2.user_id'''.format(query1, query2)
```

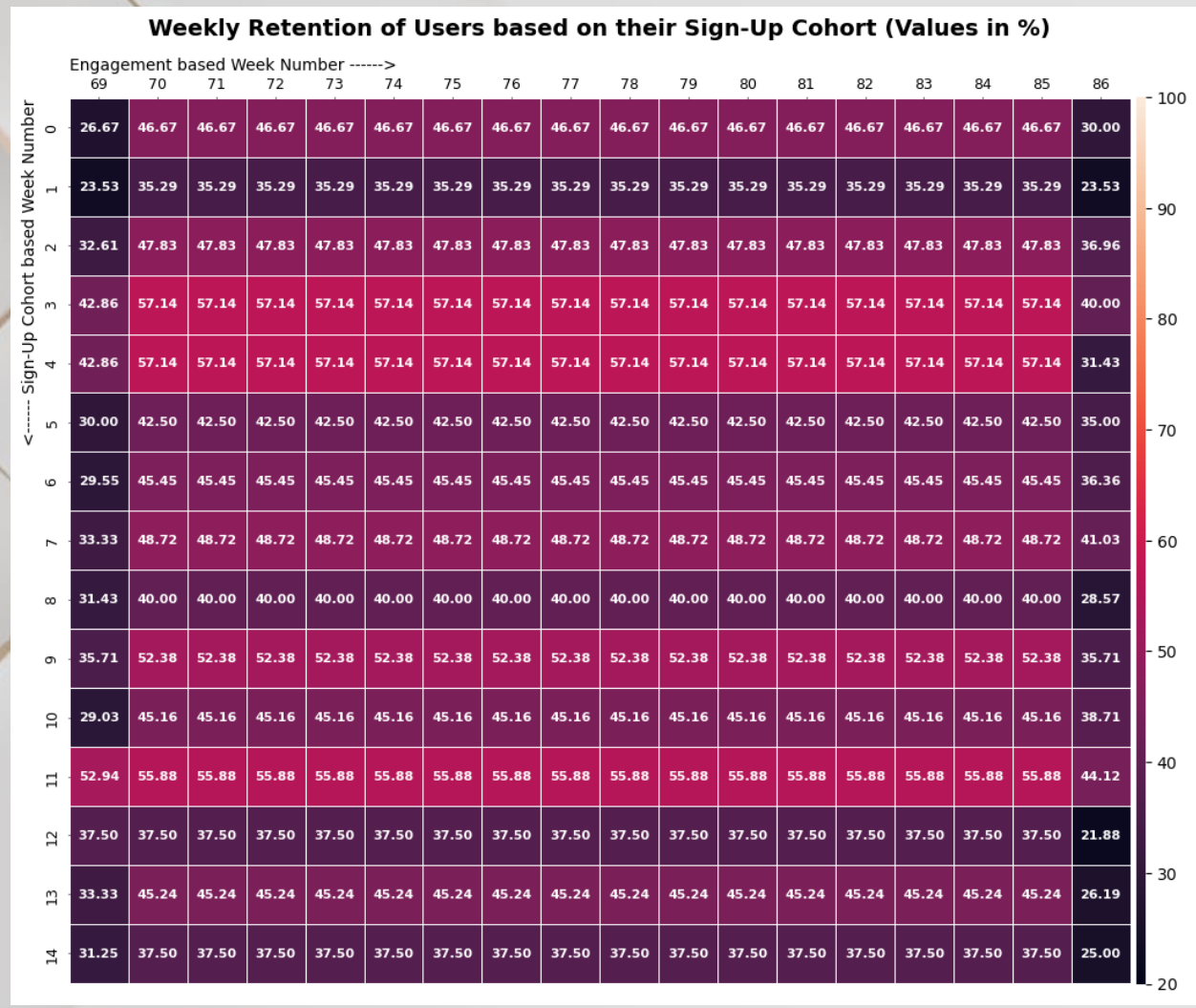
Query:-

```
query4='''SELECT q3.q1id AS q3id, q3.q1wn AS q3wn1, q3.q2wn AS q3wn2
            FROM ({0}) AS q3
            ORDER BY q3wn1, q3wn2'''.format(query3)

query5='''SELECT q1_.week_no AS q5wn, COUNT(q1_.user_id) AS ttl_ui_cnt
            FROM ({0}) AS q1_
            GROUP BY q5wn'''.format(query1)

query6='''SELECT q4.q3wn1 AS q4wn1, q4.q3wn2 AS q4wn2, COUNT(q4.q3id) AS cnt_q4id
            FROM ({0}) AS q4
            GROUP BY q3wn1, q3wn2'''.format(query4)

query7='''SELECT q6.q5wn AS q6wn, q7.q4wn2 AS q7wn2, ROUND((100*q7.cnt_q4id/q6.ttl_ui_cnt),2) AS perc_ret
            FROM ({0}) AS q6
            JOIN ({1}) AS q7
            ON (q6.q5wn=q7.q4wn1 AND q6.q5wn<q7.q4wn2)
            ORDER BY q6wn, q7wn2'''.format(query5, query6)

df3=pd.read_sql_query(query7, conn)
```

Insights

Case Study 2: Investigating Metric Spike

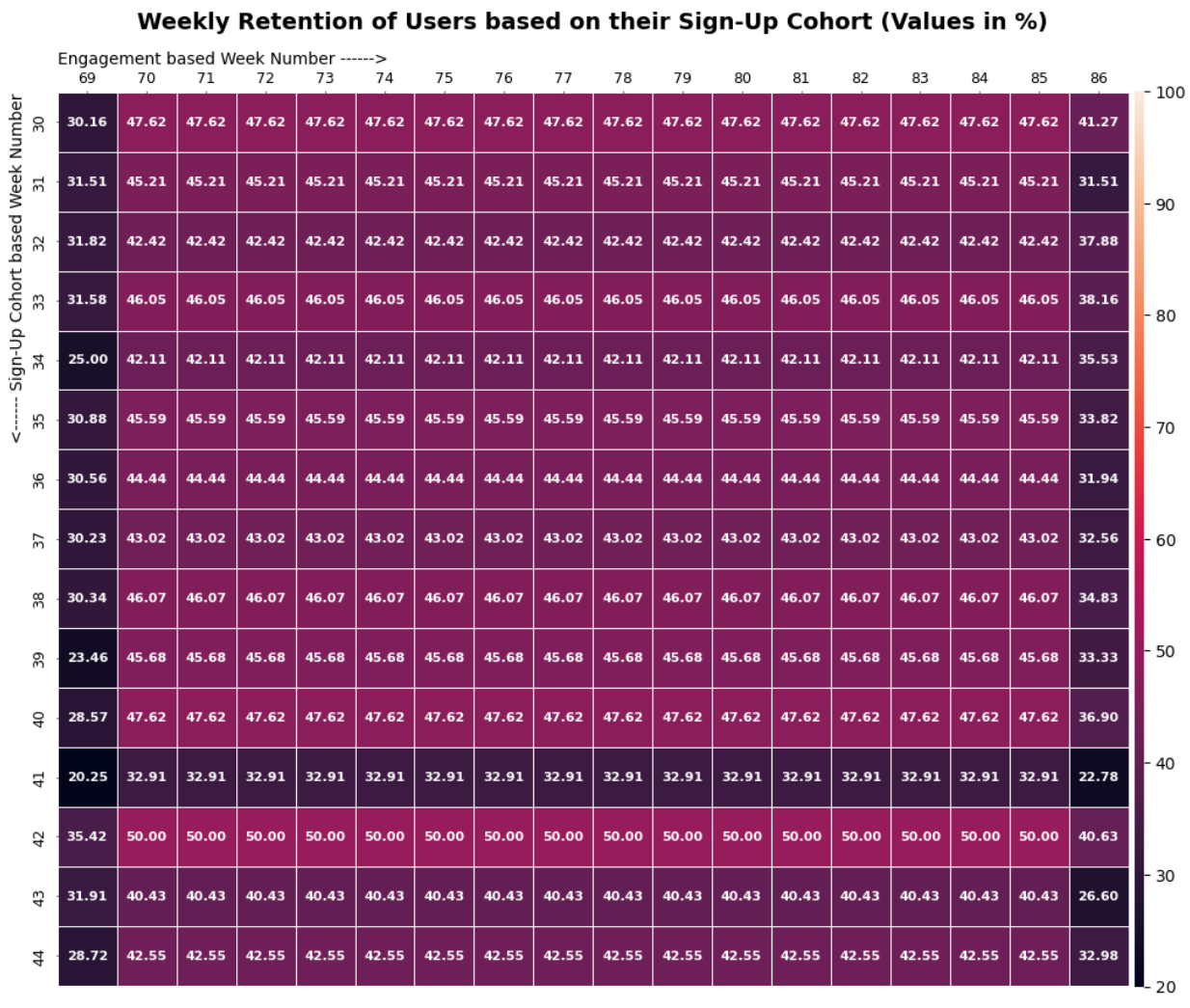
Result:-

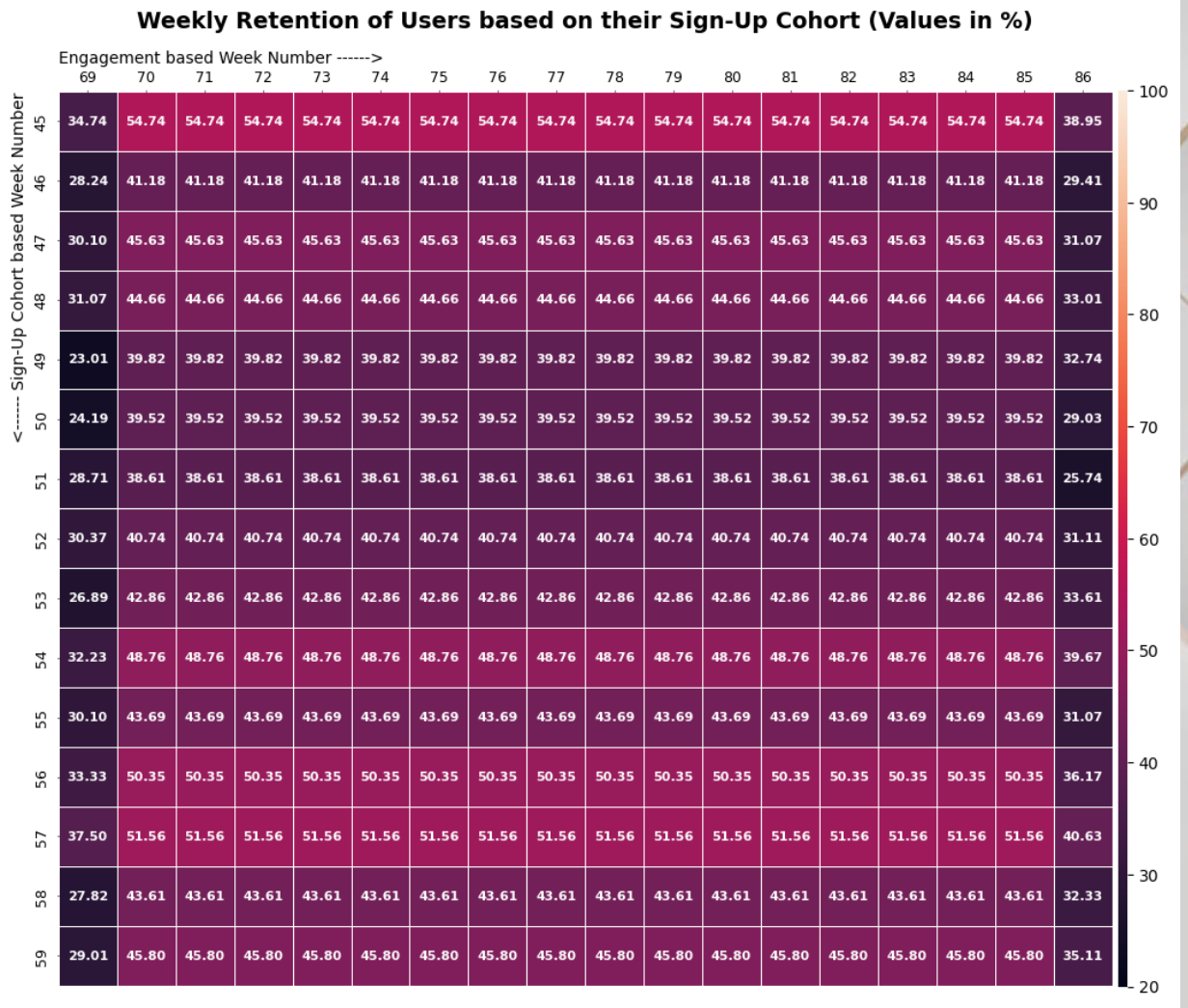


Insights

Case Study 2: Investigating Metric Spike

Result:-

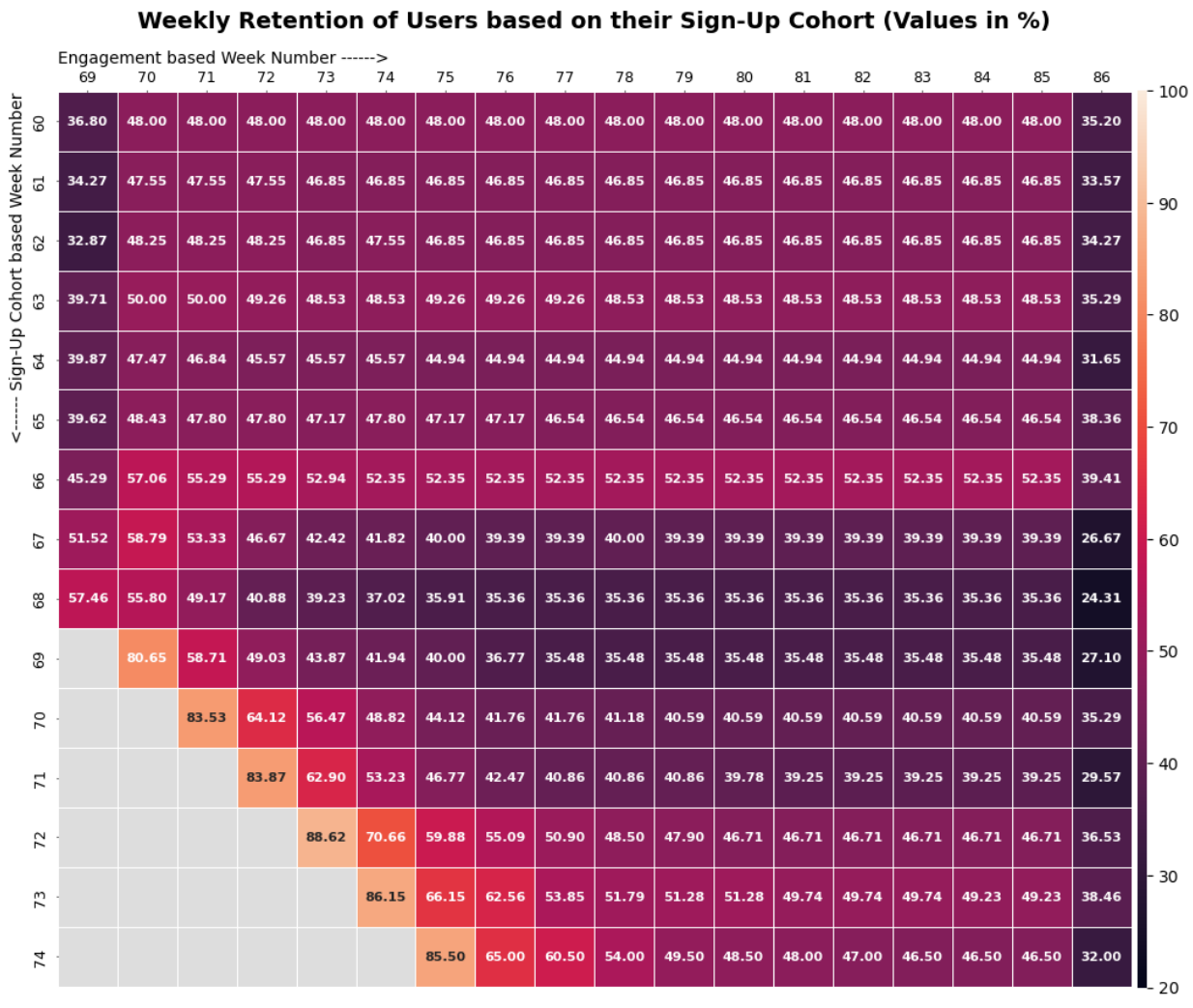


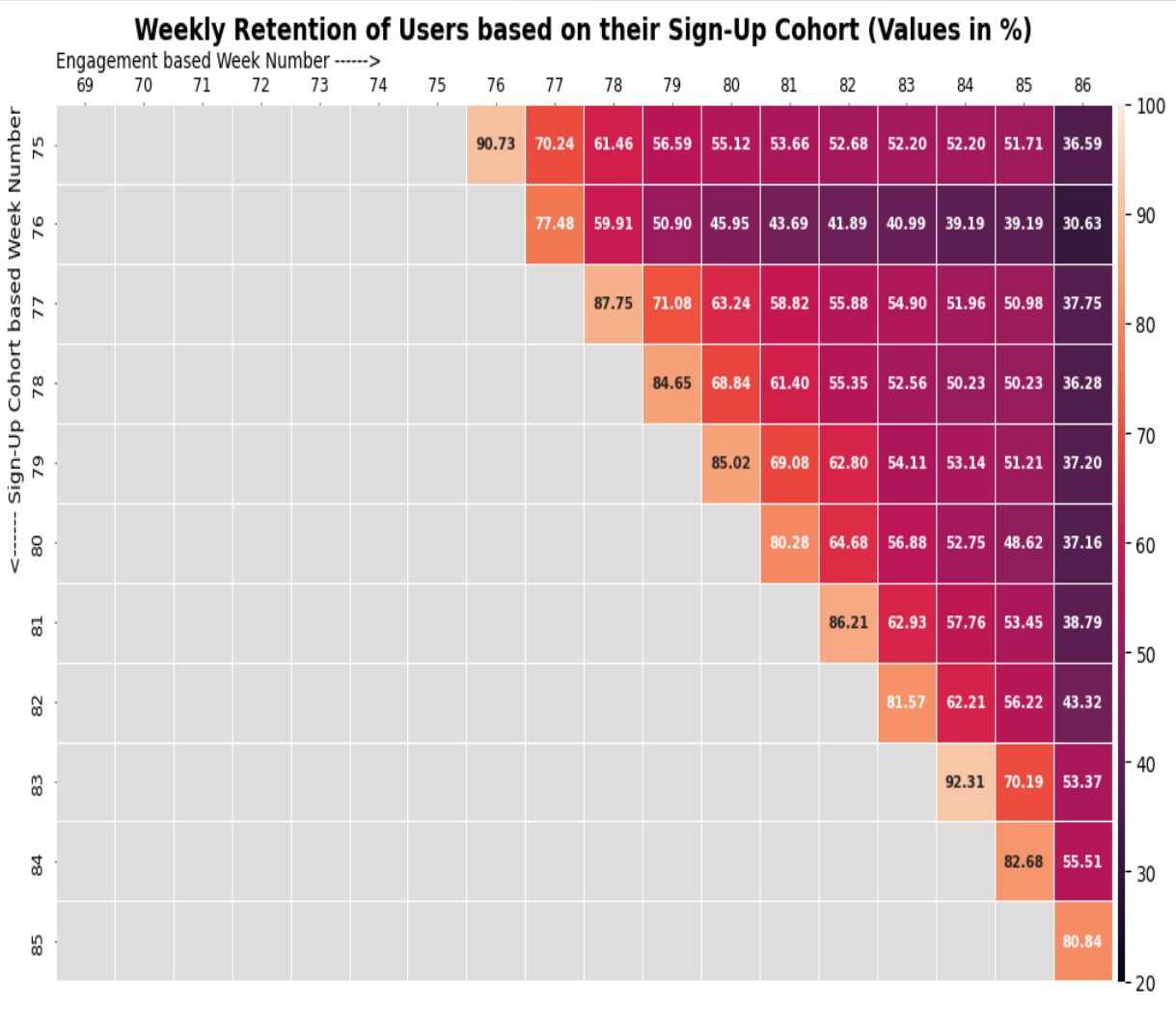


Insights

Case Study 2: Investigating Metric Spike

Result:-





Insights

Case Study 2: Investigating Metric Spike

Insights:-

- From the plots, we can observe that for sign-up cohort weeks **before 69**, the retention rate for **69th event week** is very **low** i.e. between 20% and 45% for **most** sign-up cohort weeks but it **increases slightly** after that to remain between 40% and 50% for **most** of the event weeks. One reason that may be contributed to above trend is that for sign-up cohort weeks before 69th week, the **users were dormant for a number of days** before starting to perform various events on the 69th week and increasing the number of events after that week.

- From 69th sign-up week onwards, the retention rate of the very next week is very **high** i.e. between 77% and 93% but it **decreases** after that as the weeks pass. One reason that may be contributed to above trend is that for sign-up cohort weeks on and after 69th week, the event weeks are starting right away i.e. the users are **not dormant** after signing up. The reason for decrease after the first event week may be contributed to the **excitement of users** after signing up which decreases as weeks pass.

Insights

Case Study 2: Investigating Metric Spike

Insights:-

- We can observe that the retention rate is **remaining same** for **most** of the **middle event weeks**. On further investigation, we found that event "**sent_weekly_digest**" forms majority of the events for **most** of the event weeks from 69 to 85 and the number of occurrence of this event is **remaining constant** for **most** of the middle overs for a given sign-up cohort week.

- The last event week i.e. **86th event week** have the **lowest** retention rate except for the 85th sign-up cohort week. The retention rate for **most** of the sign-up cohort weeks are between 20% and 35% for 86th event week. One reason that may be contributed to this decrease is that as the week calculation started on Tuesday since 2013-01-01 was Tuesday, so the next Monday will be counted as one whole week. Now the last event week **ends on Sunday**, i.e. one day less which means **less number of events** which contributes to comparatively low retention rate.

Insights

Case Study 2: Investigating Metric Spike

D. Weekly Engagement Per Device:

Objective: Measure the activeness of users on a weekly basis per device.

Task: Write an SQL query to calculate the weekly engagement per device.

Query:-

```
query1='''SELECT device, TIMESTAMPDIFF(WEEK, \'2013-01-01 04:40:10\', DATE_FORMAT(STR_TO_DATE(occurred_at, \'%d-%m-%Y
        %H:%i\'), \'%Y-%m-%d %H:%i:%S\')) AS wk, COUNT(user_id) as Cnt
        FROM cs_2_t_2 WHERE event_type = \'engagement\‘ GROUP BY device, wk ORDER BY device'''

query2='''SELECT device AS Device, ROUND(AVG(q1.Cnt), 2) AS Avg_Week_Eng_P_Dev
        FROM ({}) AS q1 GROUP BY device ORDER BY Avg_Week_Eng_P_Dev DESC'''.format(query1)

df5=pd.read_sql_query(query2, conn)
```

Insights

Case Study 2: Investigating Metric Spike

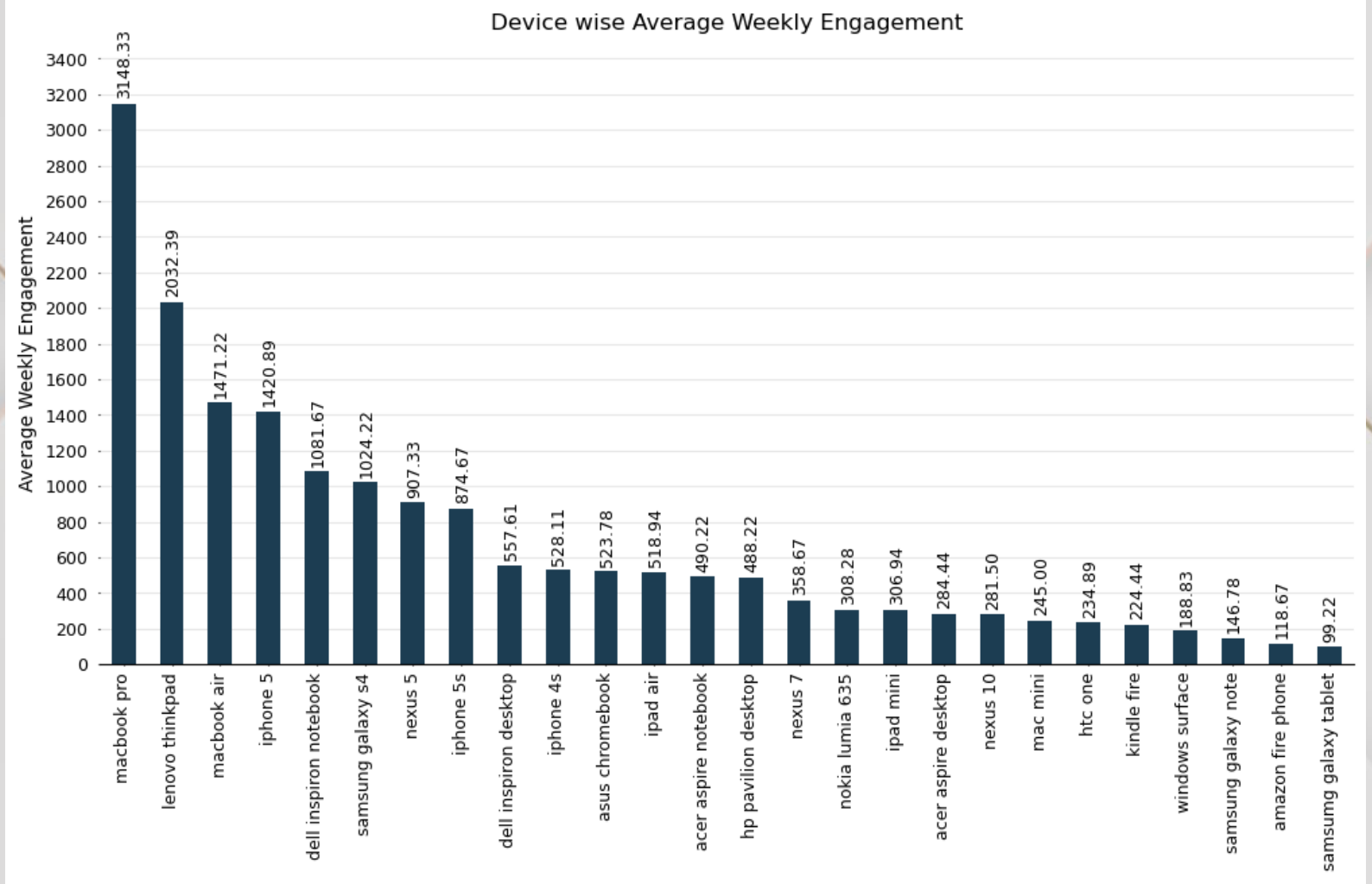
Result:-

	Device	Avg_Week_Eng_P_Dev		Device	Avg_Week_Eng_P_Dev
0	macbook pro	3148.33	13	hp pavilion desktop	488.22
1	lenovo thinkpad	2032.39	14	nexus 7	358.67
2	macbook air	1471.22	15	nokia lumia 635	308.28
3	iphone 5	1420.89	16	ipad mini	306.94
4	dell inspiron notebook	1081.67	17	acer aspire desktop	284.44
5	samsung galaxy s4	1024.22	18	nexus 10	281.50
6	nexus 5	907.33	19	mac mini	245.00
7	iphone 5s	874.67	20	htc one	234.89
8	dell inspiron desktop	557.61	21	kindle fire	224.44
9	iphone 4s	528.11	22	windows surface	188.83
10	asus chromebook	523.78	23	samsung galaxy note	146.78
11	ipad air	518.94	24	amazon fire phone	118.67
12	acer aspire notebook	490.22	25	samsumg galaxy tablet	99.22

Insights

Case Study 2: Investigating Metric Spike

Result:



Insights

Case Study 2: Investigating Metric Spike

Query: (Here we group the devices based on their companies and check the weekly engagement per company)

```
query1='''SELECT CASE
            WHEN INSTR(device, \'mac\') OR INSTR(device, \'macbook\') OR INSTR(device, \'iphone\') OR INSTR(device, '\ipad\') THEN \'APPLE\'
            WHEN INSTR(device, \'dell\') THEN \'DELL\' WHEN INSTR(device, \'windows\') THEN \'MICROSOFT\'
            WHEN INSTR(device, \'kindle\') OR LOCATE(\'amazon\', device) THEN \'AMAZON\'
            WHEN INSTR(device, \'samsung\') OR INSTR(device, \'samsumg\') THEN \'SAMSUNG\'
            WHEN INSTR(device, \'lenovo\') THEN \'LENOVO\' WHEN INSTR(device, \'nexus\') THEN \'LG\'
            WHEN INSTR(device, \'acer\') THEN \'ACER\' WHEN INSTR(device, \'asus\') THEN \'ASUS\'
            WHEN INSTR(device, \'htc\') THEN \'HTC\' WHEN INSTR(device, \'nokia\') THEN \'NOKIA\'
            WHEN INSTR(device, \'hp\') THEN \'HP\‘
             ELSE \'NO COMPANY\'
      END AS Company,
      TIMESTAMPDIFF(WEEK, \'2013-01-01 04:40:10\', DATE_FORMAT(STR_TO_DATE(occurred_at, \'%d-%m-%Y %H:%i\'), \'%Y-%m-%d %H:%i:%S\')) AS wk,
          COUNT(user_id) AS cnt FROM cs_2_t_2 WHERE event_type = \'engagement\‘ GROUP BY Company, wk ORDER BY Company'''
query2='''SELECT Company, ROUND(AVG(q1.Cnt), 2) AS Avg_Week_Eng_P_Com FROM ({}) AS q1 GROUP BY Company ORDER BY Avg_Week_Eng_P_Com
          DESC'''.format(query1)
df6=pd.read_sql_query(query2, conn)
```

Insights

Case Study 2: Investigating Metric Spike

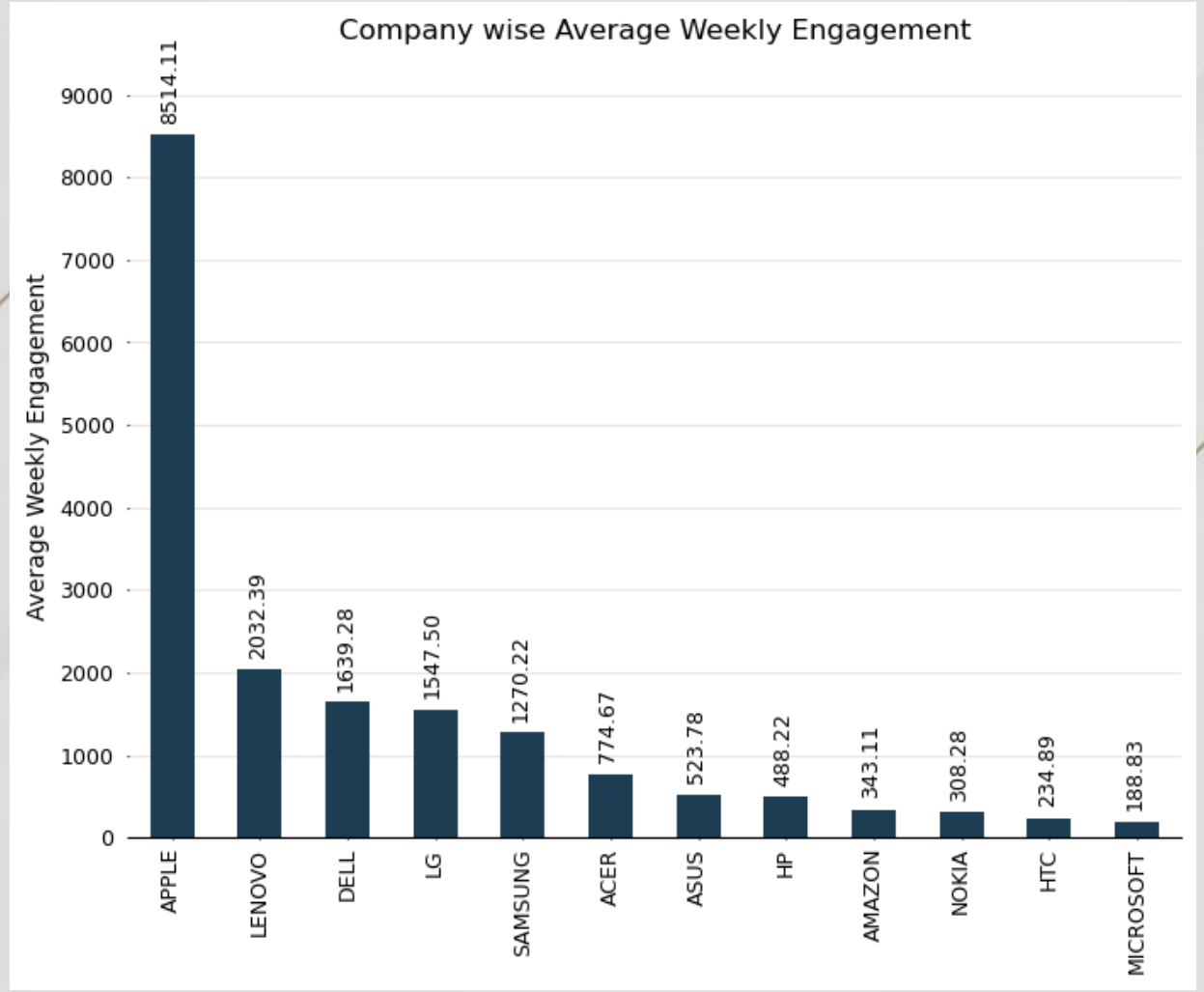
Result:-

	Company	Avg_Week_Eng_P_Com
0	APPLE	8514.11
1	LENOVO	2032.39
2	DELL	1639.28
3	LG	1547.50
4	SAMSUNG	1270.22
5	ACER	774.67
6	ASUS	523.78
7	HP	488.22
8	AMAZON	343.11
9	NOKIA	308.28
10	HTC	234.89
11	MICROSOFT	188.83

Insights

Case Study 2: Investigating Metric Spike

Result:



Insights

Case Study 2: Investigating Metric Spike

Insights:-

- From the first plot, we can observe that the most widely used device for engagement on weekly basis is **Macbook Pro** followed by **Lenovo thinkpad** and **Macbook Air** at distant second and third position respectively. All three devices are **laptops**. It is understandable as these are **formal events** mostly used in **corporate environment**.

- From the second plot, we can observe that devices from **Apple** are most widely used for engagement on weekly basis followed by **Lenovo** and **Dell** at distant second and third position respectively. This trend may again be contributed to the fact that the events are **formal** in nature which are mostly used in **corporate environment**. And for efficient corporate works, Apple's Macbooks are **famous** and **reliable**. Also all three are primarily laptop brands with Apple producing phones as well.

Insights

Case Study 2: Investigating Metric Spike

E. Email Engagement Analysis:

Objective: Analyze how users are engaging with the email service.

Task: Write an SQL query to calculate the email engagement metrics.

Query:-

```
query1='''SELECT action, TIMESTAMPDIFF(WEEK, \'2013-01-01 04:40:10\', occurred_at) AS wk, COUNT(user_id) as Cnt
            FROM cs_2_t_3 GROUP BY action, wk ORDER BY action'''

query2='''SELECT action, ROUND(AVG(q1.Cnt), 2) AS Avg_Week_Email_Eng
            FROM ({}) AS q1 GROUP BY action ORDER BY Avg_Week_Email_Eng  DESC'''.format(query1)

df7=pd.read_sql_query(query2, conn))
```

Insights

Case Study 2: Investigating Metric Spike

Result:-

	action	Avg_Week_Email_Eng
0	sent_weekly_digest	3181.50
1	email_open	1136.61
2	email_clickthrough	500.56
3	sent_reengagement_email	202.94

Insights

Case Study 2: Investigating Metric Spike

Insights:-

- From the above table, we can observe that most email activity is related to **sent_weekly_digest** followed by **email_open**. It is understandable since **sent_weekly_digest** from the name suggests some **weekly email** to be sent to every user and **email_open** from the name suggests opening of email.

Result

- Through this project, I was able to understand how important **Operational Analytics** is for an organizations as it helps in identifying and understanding areas where **improvement** is required.

- In this project I was able to get insights about various questions like rate of job reviews, share of languages, patterns of user engagement on weekly basis, growth of users etc.I can **communicate** these insights to the management team as per the requirements using which they can make proper **data-driven decisions**.

- This Project has also helped me in understanding the **advance functions of SQL** and its working. Also, since I used python environment to run the SQL queries, I was also able to understand how to link the two platforms.

Image Source: Google Images

Thank You