Case Study 1 (Job Data):-

Project Description: - This project will analyze a job data table to get insights into job review performance. The goal is to examine the number of tasks examined over time, throughput, the percentage share of each language, and the amount of duplicate rows in the data.

Approach: - The study began with an overview of the dataset and the identification of the important indicators to be examined. The SQL queries were created to compute the necessary metrics, and the results were evaluated to acquire insights. To display the data, the analysis used a combination of SQL and Excel.

Tech-Stack Used: - The project was executed using MySQL Workbench and Microsoft Excel.

Insight: -

1.The number of jobs reviewed per hour per day for November 2020 shows a decreasing trend over the month, with a few spikes in between.
2.The 7-day rolling average of throughput shows a similar trend with spikes in between. It is preferred to use the 7-day rolling average to smoothen out the spikes and get a better understanding of the trend.
3.The percentage share of each language in the last 30 days shows that English is the most reviewed language, followed by Persian, Arabic, Italian, Hindi, and French.
4.There were no duplicate rows found in the dataset.

Result: - We were able to obtain insight into numerous areas of job evaluating utilising SQL as a result of this project. We were able to answer queries regarding the number of tasks examined over time, throughput, the percentage share of each language, and how to identify and show duplicate rows by evaluating the job data. This project has aided in the development of abilities in data analysis and SQL programming, which will be useful in future projects.

Q1-Number of jobs reviewed: Amount of jobs reviewed over time.

task: Calculate the number of jobs reviewed per hour per day for November 2020?

```
SELECT

  DATE_FORMAT(ds, '%Y-%m-%d %H:00:00') as hour,
```

```
  COUNT(*) as num_jobs_reviewed
FROM
  job_data
WHERE
  ds >= '2020-11-01' AND ds < '2020-12-01'
GROUP BY
  hour;
```

	hour	num_jobs_reviewed
▶	2020-11-30 00:00:00	2
	2020-11-29 00:00:00	1
	2020-11-28 00:00:00	2
	2020-11-27 00:00:00	1
	2020-11-26 00:00:00	1
	2020-11-25 00:00:00	1

Q2-Throughput: It is the no. of events happening per second.

Your task: Let's say the above metric is called throughput. Calculate 7 day rolling average of throughput? For throughput, do you prefer daily metric or 7-day rolling and why?

```
SELECT ds,
    COUNT(*)/(24*3600) AS throughput_per_sec,
    AVG(COUNT(*)) OVER (ORDER BY ds ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS seven_day_rolling_avg
FROM job_data
WHERE ds >= '2020-11-01' AND ds < '2020-12-01'
GROUP BY ds
ORDER BY ds;
```

	ds	throughput_per_sec	seven_day_rolling_avg
▶	2020-11-25 00:00:00	0.0000	1.0000
	2020-11-26 00:00:00	0.0000	1.0000
	2020-11-27 00:00:00	0.0000	1.0000
	2020-11-28 00:00:00	0.0000	1.2500
	2020-11-29 00:00:00	0.0000	1.2000
	2020-11-30 00:00:00	0.0000	1.3333

1.25

Q3-Percentage share of each language: Share of each language for different contents.

Your task: Calculate the percentage share of each language in the last 30 days?

```
WITH CTE AS (
SELECT
Language,
COUNT(job_id) AS num_jobs
FROM
job_data
WHERE
event IN('transfer','decision')
AND ds BETWEEN '2020-11-01' AND '2020-11-30'
GROUP BY
language
),
total AS (
SELECT
COUNT(job_id) AS total_jobs
FROM
job_data
WHERE
event IN('transfer','decision')
AND ds BETWEEN '2020-11-01' AND '2020-11-30'
)
SELECT
language,
ROUND(100.0*num_jobs/total_jobs,2) AS perc_jobs
FROM
CTE
CROSS JOIN
total
```

```
ORDER BY
perc_jobs DESC
```

	language	perc_jobs
▶	Persian	33.33
	Arabic	16.67
	Hindi	16.67
	French	16.67
	Italian	16.67

Q4-Duplicate rows: Rows that have the same value present in them.
Your task: Let's say you see some duplicate rows in the data. How will you display duplicates from the table?

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

	ds	job_id	actor_id	event	language	time_spent	org	num_duplicates

Case Study 2 (Investigating metric spike): -

Project Description: -

The goal of this project is to look at a metric spike in a product by measuring user engagement, user growth, weekly retention, weekly engagement, and email engagement. The project entails constructing a database and tables, executing SQL analysis to answer queries about metrics, and reporting the results.

Approach: -

To begin the project, I examined the table architecture as well as the significance of each event in the events and email events tables. Finally, in MySQL Workbench, I built a new database and tables based on the table structure supplied. After that, I built SQL queries to extract the information needed to calculate the metrics specified in the project requirements. Eventually, I presented the findings in the form of a report.

Tech-Stack Used: -

For this project, I used MySQL Workbench version 8.0.26 as the software to create the database and tables. I used SQL queries to perform data analysis and extract the required information to calculate the metrics

.

Insight: -

I received expertise and insights into how to examine numerous metrics relating to user engagement, growth, retention, and email engagement as a result of this project. I learnt how to perform SQL queries to get the information I needed from the database and how to construct metrics based on the data I got. I also learned how to deliver the findings to the leadership team in a clear and straightforward manner.

Result: -

During this project, I gained a better grasp of how to utilise SQL queries to assess user interaction, user growth, weekly retention, weekly engagement, and email engagement data. I was able to extract the essential information from the database and generate the metrics required to evaluate the product's metric increase. The project provided me with

practical experience in SQL and data analysis, which I feel will be useful in future projects.

==========

A.User Engagement: To measure the activeness of a user. Measuring if the user finds quality in a product/service.
Your task: Calculate the weekly user engagement?

```
SELECT events.user_id, DATE_FORMAT(events.occurred_at, '%x-%v') AS week,
COUNT(DISTINCT events.event_type) AS engagement
FROM events
LEFT JOIN email_events ON events.user_id = email_events.user_id AND
DATE(events.occurred_at) = DATE(email_events.occurred_at)
GROUP BY events.user_id, week;
```

	user_id	week	engagement
▶	10522.0	2014-18	1

==========

B.User Growth: Amount of users growing over time for a product.
Your task: Calculate the user growth for product?

```
/*month basis*/
SELECT DATE_FORMAT(activated_at, '%Y-%m') AS month, COUNT(DISTINCT user_id)
AS new_users
FROM users
GROUP BY month;
```

	month	new_users
▶	NULL	9685
	2013-01	160
	2013-02	160
	2013-03	150
	2013-04	181
	2013-05	214
	2013-06	213
	2013-07	284
	2013-08	316
	2013-09	330
	2013-10	390
	2013-11	399
	2013-12	486
	2014-01	552
	2014-02	525
	2014-03	615

```
/*week basis*/
SELECT DATE_FORMAT(activated_at, '%Y-%v') AS week, COUNT(DISTINCT user_id) AS
new_users
FROM users
GROUP BY week;
```

week	new_users
NULL	9685
2013-01	67
2013-02	29
2013-03	47
2013-04	36
2013-05	30
2013-06	48
2013-07	41
2013-08	39
2013-09	33
2013-10	43
2013-11	33
2013-12	32
2013-13	33
2013-14	40
2013-15	35

==========

C.Weekly Retention: Users getting retained weekly after signing-up for a product.
Your task: Calculate the weekly retention of users-sign up cohort?

```
SELECT DATE_FORMAT(occurred_at, '%Y-%m-%d') AS week,
AVG(age_at_event) AS "Average age during week",
COUNT(DISTINCT CASE WHEN user_age > 70 THEN z.user_id ELSE NULL END) AS
"10+ weeks",
COUNT(DISTINCT CASE WHEN user_age < 70 AND user_age >=63 THEN z.user_id ELSE
NULL END) AS "9 weeks",
COUNT(DISTINCT CASE WHEN user_age < 63 AND user_age >=56 THEN z.user_id ELSE
NULL END) AS "8 weeks",
COUNT(DISTINCT CASE WHEN user_age < 56 AND user_age >=49 THEN z.user_id ELSE
NULL END) AS "7 weeks",
COUNT(DISTINCT CASE WHEN user_age < 49 AND user_age >=42 THEN z.user_id ELSE
NULL END) AS "6 weeks",
COUNT(DISTINCT CASE WHEN user_age < 42 AND user_age >=35 THEN z.user_id ELSE
NULL END) AS "5 weeks",
COUNT(DISTINCT CASE WHEN user_age < 35 AND user_age >=28 THEN z.user_id ELSE
NULL END) AS "4 weeks",
COUNT(DISTINCT CASE WHEN user_age < 28 AND user_age >=21 THEN z.user_id ELSE
NULL END) AS "3 weeks",
COUNT(DISTINCT CASE WHEN user_age < 21 AND user_age >=14 THEN z.user_id ELSE
NULL END) AS "2 weeks",
COUNT(DISTINCT CASE WHEN user_age < 14 AND user_age >=7 THEN z.user_id ELSE
NULL END) AS "1 week",
COUNT(DISTINCT CASE WHEN user_age < 7 AND user_age >=0 THEN z.user_id ELSE
NULL END) AS "Less than a week"
FROM (
SELECT occurred_at, u.user_id, DATE_FORMAT(u.activated_at, '%Y-%m-%d') AS
activation_week,
DATEDIFF(occurred_at, u.activated_at) AS age_at_event,
DATEDIFF('2014-09-01', u.activated_at) AS user_age
FROM new_schema.users u
JOIN new_schema.events e
```

```
ON u.user_id = e.user_id
AND e.event_type = 'engagement'
AND e.event_name= 'login'
AND e.occurred_at >= '2014-05-01'
AND e.occurred_at < '2014-09-01'
WHERE u.activated_at IS NOT NULL
) z
GROUP BY 1
ORDER BY 1
LIMIT 100;
```

week	Average age during week	10+ weeks	9 weeks	8 weeks	7 weeks	6 weeks	5 weeks	4 weeks	3 weeks	2 weeks	1 week	Less than a week
2014-05-02	28.0000	1	0	0	0	0	0	0	0	0	0	0

==========

D.Weekly Engagement: To measure the activeness of a user. Measuring if the user finds quality in a product/service weekly.

```
SELECT
  date_format('week', occurred_at) AS week,
  COUNT(DISTINCT e.user_id) AS weekly_active_users,
SELECT
  date_format('week', occurred_at) AS week,
  COUNT(DISTINCT e.user_id) AS weekly_active_users,
  COUNT(DISTINCT CASE WHEN e.device IN ('macbook pro', 'lenovo thinkpad', 'macbook air', 'dell inspiron notebook', 'asus chromebook', 'dell inspiron desktop', 'acer aspire notebook', 'hp pavilion desktop', 'acer aspire desktop', 'mac mini') THEN e.user_id ELSE NULL END) AS computer,
  COUNT(DISTINCT CASE WHEN e.device IN ('iphone 5', 'samsung galaxy s4', 'nexus 5', 'iphone 5s', 'iphone 4s', 'nokia lumia 635', 'htc one', 'samsung galaxy note', 'amazon fire phone') THEN e.user_id ELSE NULL END) AS phone,
  COUNT(DISTINCT CASE WHEN e.device IN ('ipad air', 'nexus 7', 'ipad mini', 'nexus 10', 'kindle fire', 'windows surface', 'samsung galaxy tablet') THEN e.user_id ELSE NULL END) AS tablet
FROM events e
WHERE e.event_type = 'engagement'
AND e.event_name = 'login'
GROUP BY 1
ORDER BY 1
LIMIT 100
```

week	weekly_active_users	computer	phone	tablet
NULL	1	1	0	0

==========

E.Email Engagement: Users engaging with the email service.
Your task: Calculate the email engagement metrics?

```
SELECT DATE_FORMAT(occurred_at, '%Y-%u') AS week,
COUNT(CASE WHEN e.action = 'sent weekly digest' THEN e.user_id
ELSE NULL END) AS weekly_emails,
COUNT(CASE WHEN e.action = 'sent reengagement email' THEN e.user_id
ELSE NULL END) AS reengagement_emails,
COUNT(CASE WHEN e.action = 'email open' THEN e.user_id
ELSE NULL END) AS email_opens,
COUNT(CASE WHEN e.action = 'email clickthrough' THEN e.user_id
ELSE NULL END) AS email_clickthroughs
FROM email_events e
GROUP BY 1;
```

week	weekly_emails	reengagement_emails	email_opens	email_clickthroughs
2014-19	0	0	0	0

0

==========