Q1. Tenure analysis:

Table: employee

Column Name	Data Type	Description
employee_id	Integer	Unique identifier for each employee
department	Varchar	The department of the employee
job_level	Varchar	The level of the job (e.g., Junior, Mid-level, Senior)
employee_status	Varchar	The status of the employee (Active, Termed)
hire_date	Date	The date the employee was hired
term_date	Date	The date the employee left, NULL if currently employed

Task:

1. Write a SQL query to calculate the average tenure of active employees in each job level within each department. Assume today's date as the end date you need to calculate tenure for active employees.

```
SELECT
   department,
   job_level,
   AVG(CURRENT_DATE - hire_date) AS tenure
FROM employee
WHERE employee_status = 'Active'
GROUP BY 1,2;
```

2. Due to data error, there may be issues with term date and hire date. Write the issues that might occur in the date fields and how will you modify Task 1 to account for those conditions.

```
Some issues with the date fields and potential solutions

String instead of Date: Sometimes dates in a database are stored as a string instead
of a DATE or TIMESTAMP field. In these cases I would likely CAST the STRING to DATE:
CAST(hire_date AS DATE) AS hire_date


NULL hire_date: There could be a case where the hire date is NULL. In these cases I
would likely inform someone of the issue and likely use a COALESCE function:
COALESCE(hire_date, CURRENT_DATE) AS hire_date


Incorrect date format: There could be a case where the date is stored incorrectly,
such as MM-DD-YYYY (e.g. 12-25-2023). In these cases I would use a LIKE, CASE WHEN and
SUBSTR function to reformat the dates. This is something that I would also recommend
be done upstream in the data pipeline.


CASE
```

```
   WHEN hire_date LIKE '____--__--__' THEN hire_date
   WHEN '__-__-____' THEN SUBSTR(hire_date, 7, 4) || '-' || SUBSTR(hire_date, 1, 2) ||
'-' || SUBSTR(4, 2)
```

3. Write a SQL query to find the employee(s) in each department with the longest tenure, along with their tenure. Also, identify any employees whose tenure is more than two standard deviations above the average tenure for their department and job level.

```
WITH employee_tenure AS (
   SELECT
       department,
       job_level,
       employee_id,
       CURRENT_DATE - hire_date AS tenure
   FROM
       employee
   WHERE
       employee_status = 'Active'
)
, department_max_tenure AS (
   SELECT
       department,
       MAX(tenure) AS max_tenure
   FROM
       employee_tenure
   GROUP BY
       1
)
, employees_with_longest_tenure AS (
   SELECT
       e.department,
       e.job_level,
       e.employee_id,
       e.tenure
   FROM
       employee_tenure e
   JOIN
       department_max_tenure dmt ON e.department = dmt.department AND e.tenure =
dmt.max_tenure
)
, department_avgstdev AS (
```

```
   SELECT
       department,
       job_level,
       AVG(tenure) AS avg_tenure,
       STDDEV(tenure) AS stddev_tenure
   FROM
       employee_tenure
   GROUP BY
       1,2
)
, lt AS (
SELECT
   ewt.department,
   ewt.job_level,
   ewt.employee_id,
   ewt.tenure
FROM
   employees_with_longest_tenure ewt
ORDER BY
   1,2
)
, stdv AS (
SELECT
   e.department,
   e.job_level,
   e.employee_id,
   e.tenure
   -- DAS.avg_tenure,
   -- DAS.stddev_tenure,
   -- (DAS.avg_tenure + 2 * DAS.stddev_tenure)
FROM
   employee_tenure e
JOIN
   department_avgstdev das ON e.department = das.department AND e.job_level =
das.job_level
WHERE
   e.tenure > (das.avg_tenure + 2 * das.stddev_tenure))

SELECT *
FROM lt
UNION ALL
SELECT * FROM stdv;
```

Q2. Analysis of workforce data, performance, and engagement.
Datasets:

Employees Table

Column Name	Data Type	Description
employee_id	Integer	Unique identifier for each employee
department_id	Integer	Identifier for the employee's department
job_level	String	The level of the job (e.g., Junior, Mid-level, Senior)
tenure_in_level	Integer	Years in job level
tenure	Integer	Years with the company

Performance Table

Column Name	Data Type	Description
employee_id	Integer	Unique identifier for each employee
year	Integer	Performance review year
performance_rating	Integer	Rating on a 1-5 scale

Engagement Survey Table

Column Name	Data Type	Description
employee_id	Integer	Unique identifier for each employee
year	Integer	Survey year
engagement_score	Integer	Score on a 1-10 scale

All employees might not have an engagement score

Task:

1. Write a SQL query to show average score across the company

```
-- This shows the average for all years
SELECT
   AVG(engagement_score) AS average_engagement_score
FROM engagement_survey;


-- This shows the average across all years
SELECT
   year,
   AVG(engagement_score) AS average_engagement_score
FROM engagement_survey
GROUP BY 1;
```

2. Construct a SQL query to identify the top 5 departments with the highest average engagement score among employees with a performance rating of 4 or above in 2023.

```
SELECT
   e.department_id,
```

```
   AVG(es.engagement_score) AS average_engagement_score
FROM employees e
JOIN performance p ON e.employee_id = p.employee_id
JOIN engagement_survey es ON e.employee_id = es.employee_id
WHERE p.year = 2023 AND p.performance_rating >= 4
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5;
```

3. Only employees who responded to the survey will have a score. Write a SQL query to show the top 5 departments with least participation rates in year 2022 and year 2023

```
WITH rate AS (
SELECT
   e.department_id,
   es.year,
   COUNT(DISTINCT e.employee_id) AS employee_count,
   COUNT(DISTINCT CASE WHEN es.engagement_score IS NOT NULL THEN es.employee_id END)
AS participating_employees,
   ROUND(COUNT(DISTINCT CASE WHEN es.engagement_score IS NOT NULL THEN es.employee_id
END)*1.0 / COUNT(DISTINCT e.employee_id),2) AS participation_rate
FROM employees e
LEFT JOIN engagement_survey es ON e.employee_id = es.employee_id
WHERE es.year IN (2022, 2023)
GROUP BY 1, 2
ORDER BY 2 ASC, 5 ASC)

SELECT
   department_id,
   year,
   participation_rate
FROM (
SELECT
*,
ROW_NUMBER() OVER (PARTITION BY year ORDER BY participation_rate ASC) AS drnk
FROM rate) z
WHERE drnk <= 5;
```

4. Create a SQL query to rank each department based on how much above or below they score compared to the average company score (Hint: Rank based on Company Avg Score – Department Avg Score)

```sql
-- overall (all years)
WITH department_scores AS (
   SELECT
       e.department_id,
       AVG(es.engagement_score) AS department_avg_score
   FROM
       employees e
   JOIN
       engagement_survey es ON e.employee_id = es.employee_id
   WHERE
       es.year IN (2022, 2023)
   GROUP BY
       e.department_id
)
, avg_score AS (
   SELECT
       AVG(es.engagement_score) AS company_avg_score
   FROM
       engagement_survey es
   WHERE
       es.year IN (2022, 2023)
)
SELECT
   d.department_id,
   d.department_avg_score,
   (a.company_avg_score - d.department_avg_score) AS score_difference,
   DENSE_RANK() OVER (ORDER BY a.company_avg_score - d.department_avg_score DESC) AS
drnk
FROM
   department_scores d
CROSS JOIN
   avg_score a
ORDER BY
   4;


-- split by year
WITH department_scores AS (
   SELECT
       e.department_id,
       es.year,
       AVG(es.engagement_score) AS department_avg_score
   FROM
```

```
        employees e
    JOIN
        engagement_survey es ON e.employee_id = es.employee_id
    WHERE
        es.year IN (2022, 2023)
    GROUP BY
        e.department_id, es.year
)
, avg_scores AS (
    SELECT
        year,
        AVG(es.engagement_score) AS company_avg_score
    FROM
        engagement_survey es
    WHERE
        es.year IN (2022, 2023)
    GROUP BY
        es.year
)
SELECT
    d.department_id,
    d.year,
    d.department_avg_score,
    a.company_avg_score,
    a.company_avg_score - d.department_avg_score AS score_difference,
    RANK() OVER (PARTITION BY d.year ORDER BY a.company_avg_score -
d.department_avg_score) AS department_rank
FROM
    department_scores d
JOIN
    avg_scores a ON d.year = a.year
ORDER BY 2 ASC, 6 DESC;
```