Name : Harshit Mishra

SQL Assignment Coding Quiz

Q1.

```
with cte_temp as (
select month(salary.pay_date) as pay_month, department_id,
avg(amount) over(partition by month(salary.pay_date), department_id) as dept_avg,
avg(amount) over(partition by month(salary.pay_date)) as company_avg
from salary inner join employee on salary.employee_id = employee.employee_id )
select distinct department_id, pay_month,
( case
when dept_avg > company_avg then 'Higher'
when dept_avg < company_avg then 'Lower'
else 'Same'
end
) as comparison from cte_temp ;
```

Q2.

```
WITH t1 AS(
SELECT student_id FROM
(
SELECT exam_id, student_id, student_name, score,
MIN(score) OVER(PARTITION BY exam_id) AS min_score,
MAX(score) OVER(PARTITION BY exam_id) AS max_score
FROM student INNER JOIN exam USING (student_id)
ORDER BY exam_id
) as a
WHERE min_score = score OR max_score = score
)
SELECT DISTINCT student_id, student_name
FROM exam JOIN student
USING (student_id)
WHERE student_id != ALL(SELECT student_id FROM t1)
ORDER BY 1;
```

Q3.

```
WITH StadiumCTE AS (
SELECT
stadium_id AS id,
visit_date,
attendance AS people,
stadium_id - ROW_NUMBER() OVER (ORDER BY visit_date) AS streak_num
FROM
YourStadiumTable -- Replace YourStadiumTable with the actual table name
```

```
WHERE
attendance >= 100
)
SELECT
id,
visit_date,
people
FROM
StadiumCTE
INNER JOIN
(
SELECT
streak_num,
COUNT(*) AS streak_size
FROM
StadiumCTE
GROUP BY
streak_num
HAVING
COUNT(*) >= 3
) AS StreakInfo ON StadiumCTE.streak_num = StreakInfo.streak_num;
```

Q4.

```
WITH RecursiveCTE AS (
SELECT
visit_date,
COALESCE(num_visits, 0) as num_visits,
COALESCE(num_transactions, 0) as num_transactions
FROM (
SELECT
visit_date,
user_id,
COUNT(*) as num_visits
FROM
YourVisitsTable -- Replace YourVisitsTable with the actual visits table name
GROUP BY
visit_date, user_id
) AS a
LEFT JOIN (
SELECT
transaction_date,
user_id,
COUNT(*) as num_transactions
FROM
GROUP BY
transaction_date, user_id
) AS b ON a.visit_date = b.transaction_date and a.user_id = b.user_id
), RecursiveCounter AS (
SELECT
```

```
MAX(num_transactions) as trans
FROM
RecursiveCTE
UNION ALL
SELECT
trans - 1
FROM
RecursiveCounter
WHERE
trans >= 1
)
SELECT
trans as transactions_count,
COALESCE(visits_count, 0) as visits_count
FROM
RecursiveCounter
LEFT JOIN (
SELECT
num_transactions as transactions_count,
COALESCE(COUNT(*), 0) as visits_count
FROM
RecursiveCTE
GROUP BY
num_transactions
) AS a ON a.transactions_count = RecursiveCounter.trans
ORDER BY
1;
```

Q5.

```
WITH SuccessCTE AS (
SELECT
MIN(successful_date) AS start_date,
MAX(successful_date) AS end_date,
success_state AS state
FROM (
SELECT
successful_date,
DATEADD(DAY, -ROW_NUMBER() OVER (ORDER BY successful_date),
successful_date) AS date_difference,
1 AS success_state
FROM
WHERE
successful_date BETWEEN '2019-01-01' AND '2019-12-31'
) AS a
GROUP BY
date_difference
), FailureCTE AS (
SELECT
MIN(failure_date) AS start_date,
MAX(failure_date) AS end_date,
failure_state AS state
```

```
FROM (
SELECT
failure_date,
DATEADD(DAY, -ROW_NUMBER() OVER (ORDER BY failure_date), failure_date) AS
date_difference,
0 AS failure_state
FROM
WHERE
failure_date BETWEEN '2019-01-01' AND '2019-12-31'
) AS b
GROUP BY
date_difference
)
SELECT
start_date,
end_date,
CASE
WHEN c.state = 1 THEN 'succeeded'
ELSE 'failed'
END AS period_state
FROM (
SELECT * FROM SuccessCTE
UNION ALL
SELECT * FROM FailureCTE
) AS c
ORDER BY
start_date;
```