1.Find employees whose salary is higher than their manager's salary:

```
SELECT e.id AS emp_id,
    e.name AS emp_name,
    e.salary AS emp_salary,
    m.id AS manager_id,
    m.name AS manager_name,
    m.salary AS manager_salary
FROM emp e
JOIN emp m
  ON e.manger_id = m.id
WHERE e.salary > m.salary;
```

Explanation:

- emp e and emp m → Self-join on the emp table. e is the employee, m is the manager.
- ON e.manger_id = m.id → Links each employee to their manager.
- WHERE e.salary > m.salary → Filters employees earning more than their manager.

2.You have a table called teams that contains a list of cricket teams. Write a SQL query to generate all possible matches between two different teams, ensuring that:
1)No team plays against itself.
2)Each matchup appears only once (i.e., (India, Australia) and (Australia, India) should not both appear).
3)Explain how your query avoids duplicates and self-matches.

```
SELECT t1.team_name AS team1,
    t2.team_name AS team2
FROM teams t1
JOIN teams t2
  ON t1.team_name < t2.team_name
ORDER BY t1.team_name, t2.team_name;
```

Explanation:

- teams t1 JOIN teams t2 → Self-join to create all team combinations.
- t1.team_name < t2.team_name → Ensures:
- A team cannot play itself.
- Each matchup appears only once (India, Australia appears, Australia, India does not).
- ORDER BY → Sorts matches alphabetically.

3.Customers who have never placed an order

```
SELECT c.customer_id, c.customer_name
FROM customers c
LEFT JOIN orders o
  ON c.customer_id = o.customer_id
WHERE o.order_id IS NULL;
```

Explanation:

- LEFT JOIN → Includes all customers, even if they have no orders.
- WHERE o.order_id IS NULL → Filters customers with no orders.
- Output → List of customers who never placed an order.

4.find missing dates in a sequence of dates in a table:

```
WITH calendar AS (
   SELECT generate_series(
      (SELECT MIN(sale_date) FROM sales),
      (SELECT MAX(sale_date) FROM sales),
      INTERVAL '1 day'
   )::DATE AS day
)
SELECT c.day AS missing_date
FROM calendar c
LEFT JOIN sales s
  ON c.day = s.sale_date
WHERE s.sale_date IS NULL
ORDER BY c.day;
```

Explanation:

- generate_series → Creates a complete list of dates from the minimum to maximum sale_date.
- LEFT JOIN sales → Matches existing dates with the calendar.
- WHERE s.sale_date IS NULL → Returns dates not present in sales.

5.find missing consecutive dates:

```
SELECT DATE_ADD(order_date, INTERVAL 1 DAY) AS missing_date
FROM (
   SELECT order_date,
        LEAD(order_date) OVER (ORDER BY order_date) AS next_date
   FROM orders
) t
```

WHERE DATEDIFF(next_date, order_date) > 1;

Explanation:

- LEAD(order_date) → Gets the next date in the ordered list.
- DATEDIFF(next_date, order_date) > 1 → Finds gaps in consecutive dates.
- DATE_ADD(order_date, INTERVAL 1 DAY) → Returns the first missing date in the gap.