Documentation: Finding Customers Who Never Placed an Order

Problem Overview:

- The task is to identify customers from the Customers table who have never placed any orders, which are recorded in the Orders table. The goal is to return the names of such customers in the format of a single column titled "Customers."

Input:

- **The solution works with two input tables:** Customers and Orders.

Table 1: *Customers*

- **id (int):** Unique identifier for each customer (Primary key).
- **name (varchar):** Name of the customer.

This table holds the information of all registered customers.

Table 2: *Orders*

- **id (int):** Unique identifier for each order (Primary key).
- **customerId (int):** Foreign key that references the customer id from the Customers table.

This table stores the orders placed by customers, associating each order with the customer who placed it.

Output:

- The solution will return a single table that contains the names of customers who have never placed an order. This table will have the following structure:

Customers
customer1
customer2

Approach:

To find customers who have never placed an order, the following steps are performed:

1. Left Join the Customers Table with the Orders Table:

- A left join between the Customers table and the Orders table is performed based on the id from Customers and customerId from Orders.
- In this process, all records from the Customers table are retained, even if there is no matching entry in the Orders table (i.e., the customer never placed any order).
- If a customer has not placed an order, the customerId in the merged table will be NaN (missing).

2. Filter Customers with No Orders:

- After the join, we filter the rows where the customerId is NaN, which indicates that these customers never placed any orders.

3. Extract the Customer Names:

- The result consists of the names of customers who have not placed any orders. The output is a single-column DataFrame renamed to "Customers."

Example Input:

Consider the following two tables:

Customers Table:

id	name
1	Joe
2	Henry
3	Sam
4	Max

Orders Table:

id	customerId
1	3
2	1

- The Customers table contains four customers: Joe, Henry, Sam, and Max.
- The Orders table shows that customer with id = 3 (Sam) and customer with id = 1 (Joe) have placed orders.
- Therefore, customers Henry and Max have not placed any orders.

Example Output:

Customers
Henry
Max

Steps Breakdown:

1. Perform a Left Join:

- Left join the Customers table with the Orders table on Customers.id = Orders.customerId.
- *After joining, the merged table will look like this:*

id	name	id (order)	customerId
1	Joe	2	3
2	Henry	NaN	NaN
3	Sam	1	3
4	Max	NaN	NaN

2. Filter Out Customers with Orders:

- From the merged result, filter rows where customerId is NaN (customers who have not placed any orders).
- ***Filtered result:***

id	name	id (order)	customerId
2	Henry	NaN	NaN
4	Max	NaN	NaN

3. Return the Customer Names:

- Select only the name column from the filtered result and rename it to Customers for the final output.
- ***Final result:***

Customers
Henry
Max

Performance Consideration:

- **Time Complexity:** The solution involves a left join operation, which in most cases operates in linear time relative to the number of rows in the Customers and Orders tables, i.e., O(n + m) where n is the number of customers and m is the number of orders.

- **Space Complexity:** Space used is proportional to the size of the merged DataFrame, which depends on the number of customers and orders.

Edge Cases:

1. **No Orders Exist:**
   - If the Orders table is empty, the solution will return all customers from the Customers table.

2. **All Customers Have Orders:**
   - If all customers have placed at least one order, the solution will return an empty result.

3. **Empty Customers Table:**
   - If the Customers table is empty, the result will also be empty, as there are no customers to check against.

Conclusion:

- This approach efficiently identifies customers who have never placed any orders by leveraging a left join between the Customers and Orders tables, followed by filtering and formatting the result.