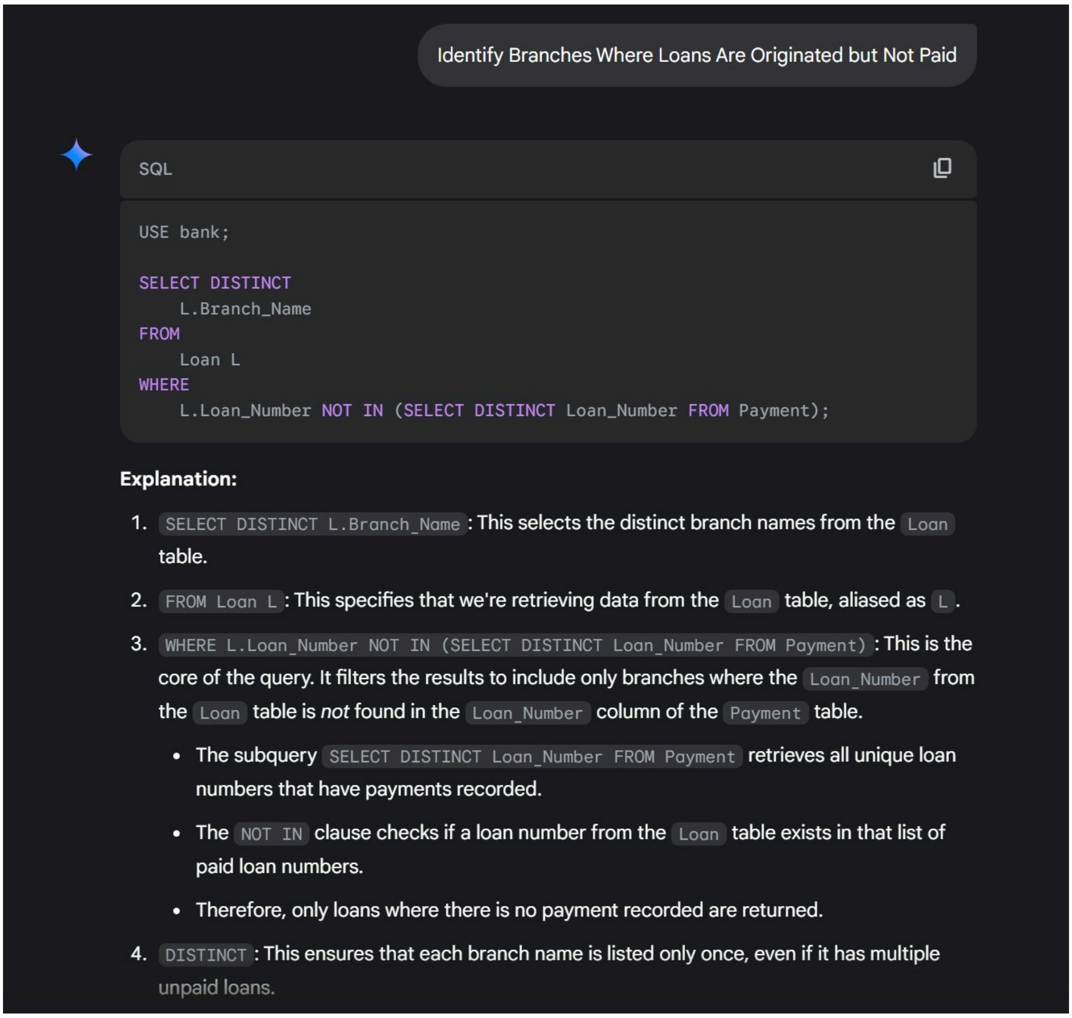
INCORRECT QUERIES

We used Gemini to generate some queries and have highlighted the ones we found incorrect.

QUERY-1

Identify Branches Where Loans Are Originated but Not Paid



Identify Branches Where Loans Are Originated but Not Paid

```sql
USE bank;

SELECT DISTINCT
    L.Branch_Name
FROM
    Loan L
WHERE
    L.Loan_Number NOT IN (SELECT DISTINCT Loan_Number FROM Payment);
```

Explanation:

1. `SELECT DISTINCT L.Branch_Name`: This selects the distinct branch names from the `Loan` table.
2. `FROM Loan L`: This specifies that we're retrieving data from the `Loan` table, aliased as `L`.
3. `WHERE L.Loan_Number NOT IN (SELECT DISTINCT Loan_Number FROM Payment)`: This is the core of the query. It filters the results to include only branches where the `Loan_Number` from the `Loan` table is *not* found in the `Loan_Number` column of the `Payment` table.
    - The subquery `SELECT DISTINCT Loan_Number FROM Payment` retrieves all unique loan numbers that have payments recorded.
    - The `NOT IN` clause checks if a loan number from the `Loan` table exists in that list of paid loan numbers.
    - Therefore, only loans where there is no payment recorded are returned.
4. `DISTINCT`: This ensures that each branch name is listed only once, even if it has multiple unpaid loans.

🔴 Issues

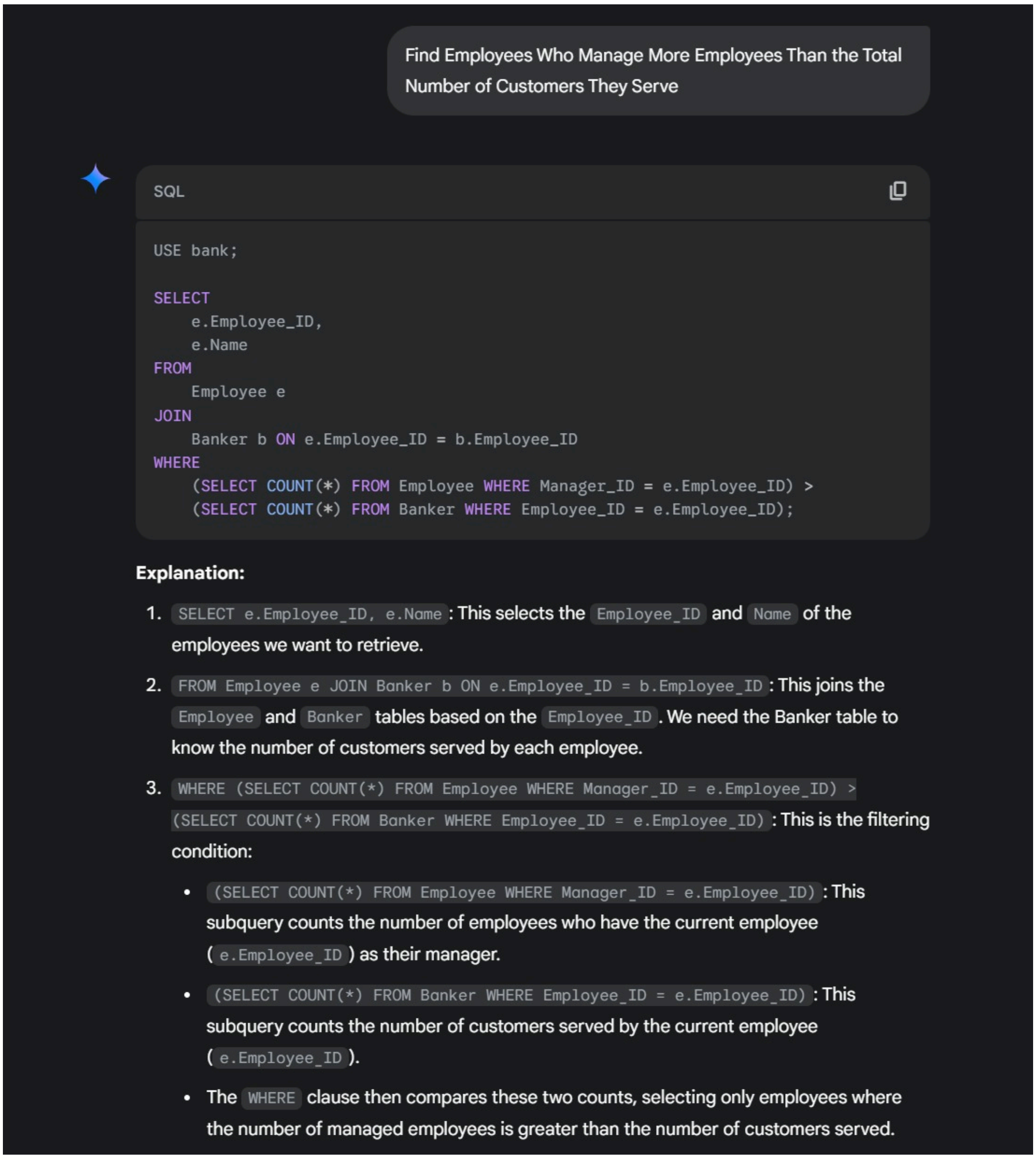
Issue	Why is it incorrect?	Fix
1️⃣ **`NOT IN` Fails with NULL Values**	If `Payment.Loan_Number` contains `NULL`, `NOT IN` fails because `NULL NOT IN (...)` evaluates to `FALSE`, potentially returning **incorrect or empty results.**	Use `NOT EXISTS` instead of `NOT IN`.
2️⃣ **`Loan.Branch_Name` Doesn't Exist**	The schema does **not** store `Branch_Name` in the `Loan` table; it is in the **`Originated_By` table**, which links loans to branches.	Use `Originated_By` to correctly link branches.
3️⃣ **`DISTINCT` is Unnecessary**	`DISTINCT` is redundant when filtering by `Branch_Name` in `WHERE NOT EXISTS`.	Remove `DISTINCT` to improve performance.

✅ Final Corrected Query

```
sql
CopyEdit
SELECT O.Branch_Name
FROM Originated_By O
JOIN Loan L ON O.Loan_Number = L.Loan_Number
WHERE NOT EXISTS (
   SELECT 1 FROM Payment P WHERE P.Loan_Number = L.Loan_Number
);
```

QUERY-2

Find Employees Who Manage More Employees Than the Total Number of Customers They Serve



Find Employees Who Manage More Employees Than the Total Number of Customers They Serve

```sql
USE bank;

SELECT
    e.Employee_ID,
    e.Name
FROM
    Employee e
JOIN
    Banker b ON e.Employee_ID = b.Employee_ID
WHERE
    (SELECT COUNT(*) FROM Employee WHERE Manager_ID = e.Employee_ID) >
    (SELECT COUNT(*) FROM Banker WHERE Employee_ID = e.Employee_ID);
```

Explanation:

1. `SELECT e.Employee_ID, e.Name`: This selects the `Employee_ID` and `Name` of the employees we want to retrieve.
2. `FROM Employee e JOIN Banker b ON e.Employee_ID = b.Employee_ID`: This joins the `Employee` and `Banker` tables based on the `Employee_ID`. We need the Banker table to know the number of customers served by each employee.
3. `WHERE (SELECT COUNT(*) FROM Employee WHERE Manager_ID = e.Employee_ID) > (SELECT COUNT(*) FROM Banker WHERE Employee_ID = e.Employee_ID)`: This is the filtering condition:
    - `(SELECT COUNT(*) FROM Employee WHERE Manager_ID = e.Employee_ID)`: This subquery counts the number of employees who have the current employee (`e.Employee_ID`) as their manager.
    - `(SELECT COUNT(*) FROM Banker WHERE Employee_ID = e.Employee_ID)`: This subquery counts the number of customers served by the current employee (`e.Employee_ID`).
    - The `WHERE` clause then compares these two counts, selecting only employees where the number of managed employees is greater than the number of customers served.

🔴 Issues

Issue	Why is it Incorrect?	Fix

1️⃣ **Uses `JOIN` Incorrectly**	The `JOIN Banker b ON e.Employee_ID = b.Employee_ID` turns the query into an **INNER JOIN**, which **excludes employees who don't serve any customers**.	Use `LEFT JOIN` instead of `JOIN`.
2️⃣ **Employees Without Customers Are Excluded**	If an employee **has no customers**, they should still be counted as long as they manage more employees.	Use `LEFT JOIN` to ensure employees with zero customers are included.
3️⃣ **Misplaced `WHERE` Clause**	The `WHERE` clause filters out employees **before checking counts**, **excluding potential valid managers**.	Move the filtering logic to `HAVING`.
4️⃣ **`COUNT(*)` Counts All Rows**	`COUNT(*)` on `Banker` **includes duplicate customers**, since an employee can serve multiple customers.	Use `COUNT(DISTINCT Banker.Customer_ID)`.

✅ Corrected Query

```
SELECT
    e.Employee_ID,
    e.Name
FROM
    Employee e
LEFT JOIN Banker b ON e.Employee_ID = b.Employee_ID
GROUP BY e.Employee_ID, e.Name
HAVING
    (SELECT COUNT(*) FROM Employee WHERE Manager_ID = e.Employee_ID) >
    COUNT(DISTINCT b.Customer_ID);
```

CHAT LINK

📎 https://g.co/gemini/share/bad9b748e77a