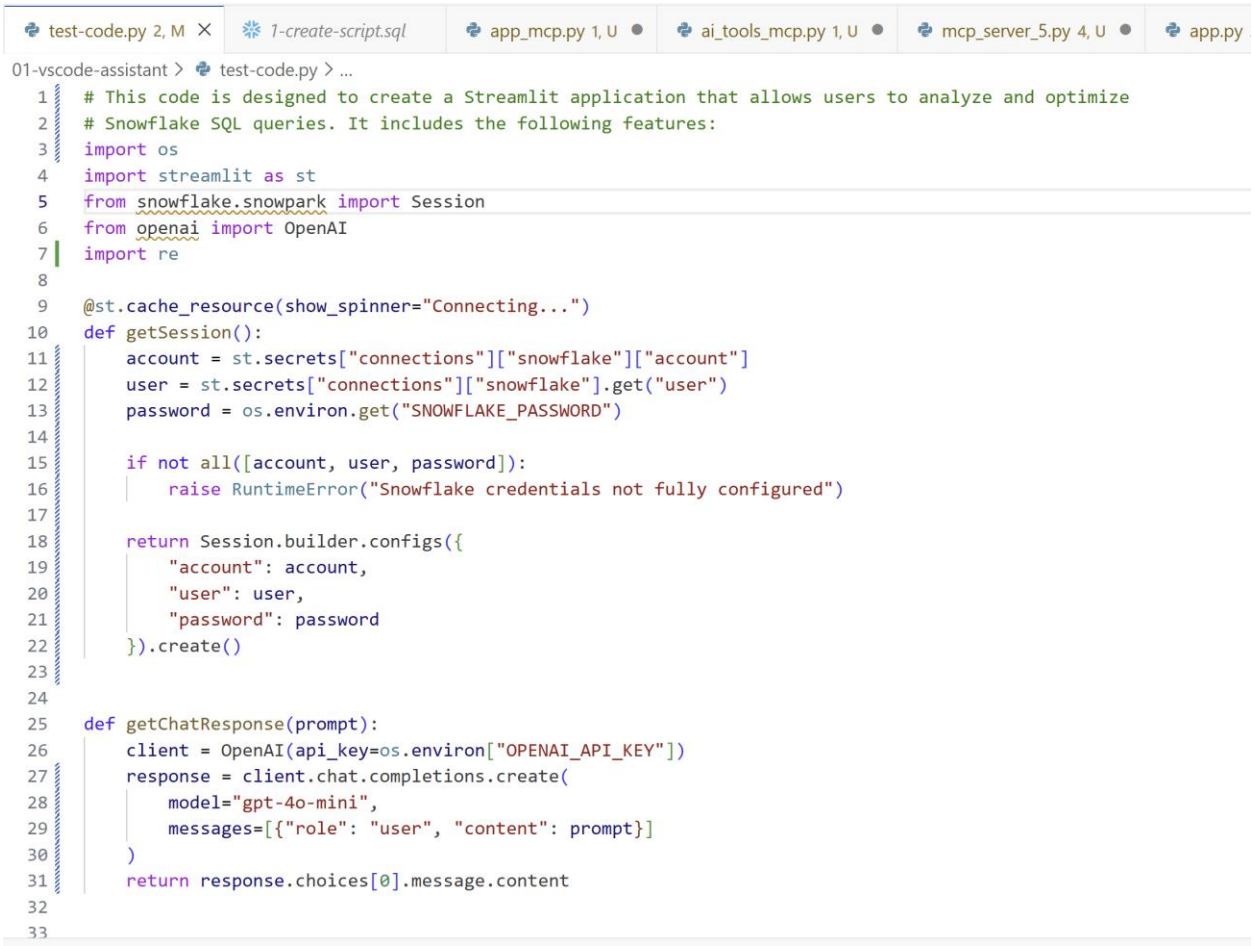
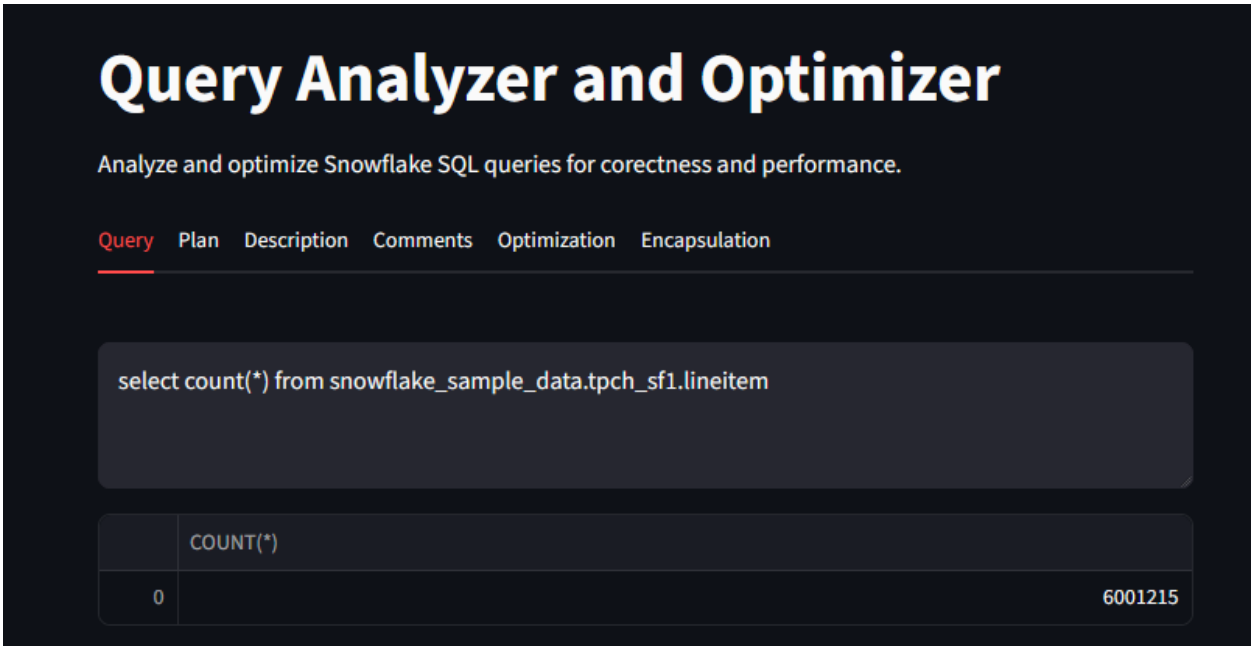
Snowflake – IDE Code Assistant

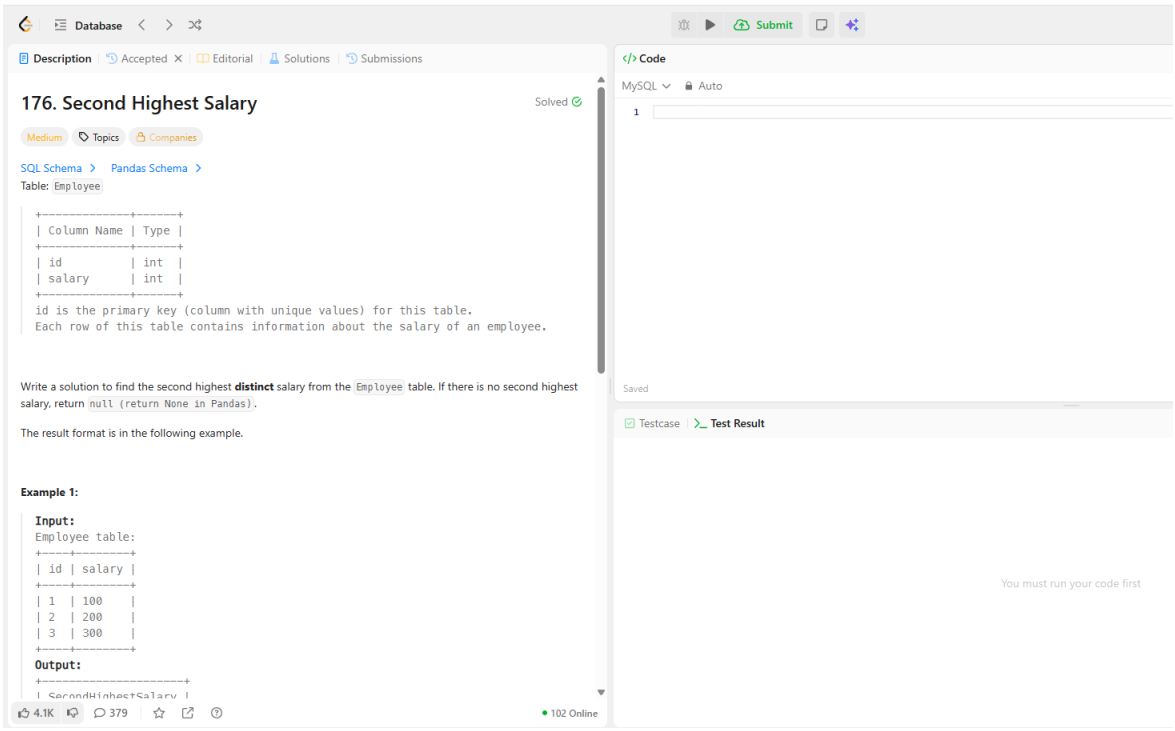




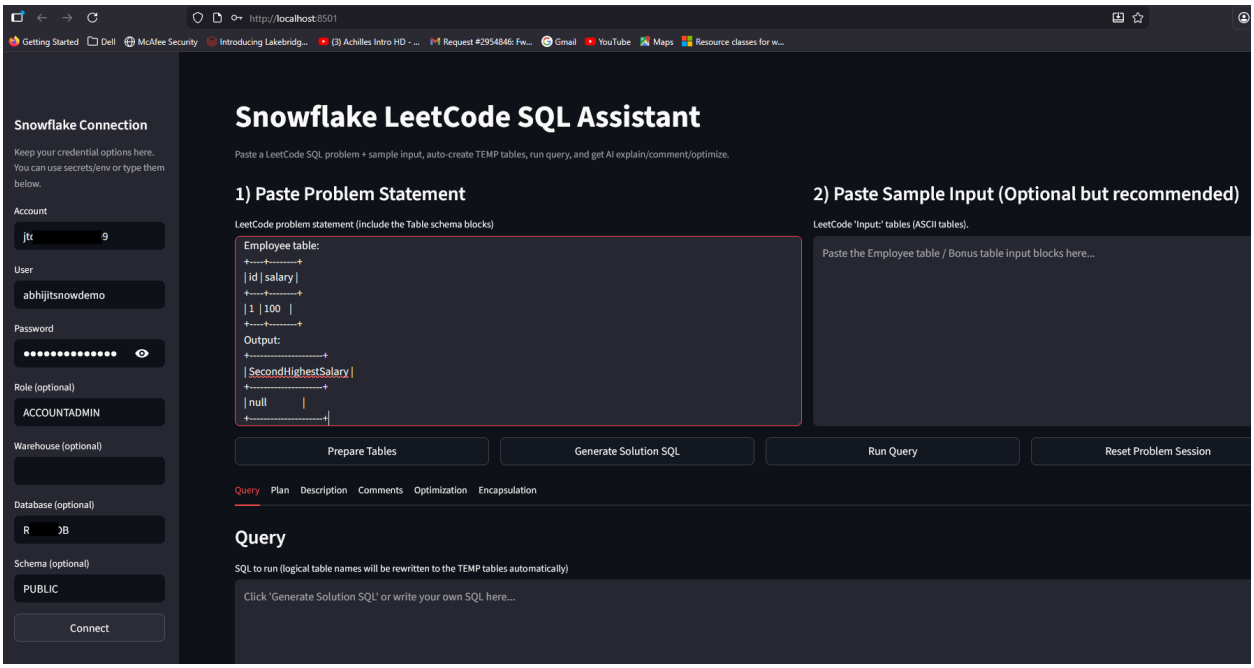
Additionally, I have built an application to everyone who are interest to learn SQL. This code is basically a LeetCode-style SQL practice "lab" inside Snowflake.

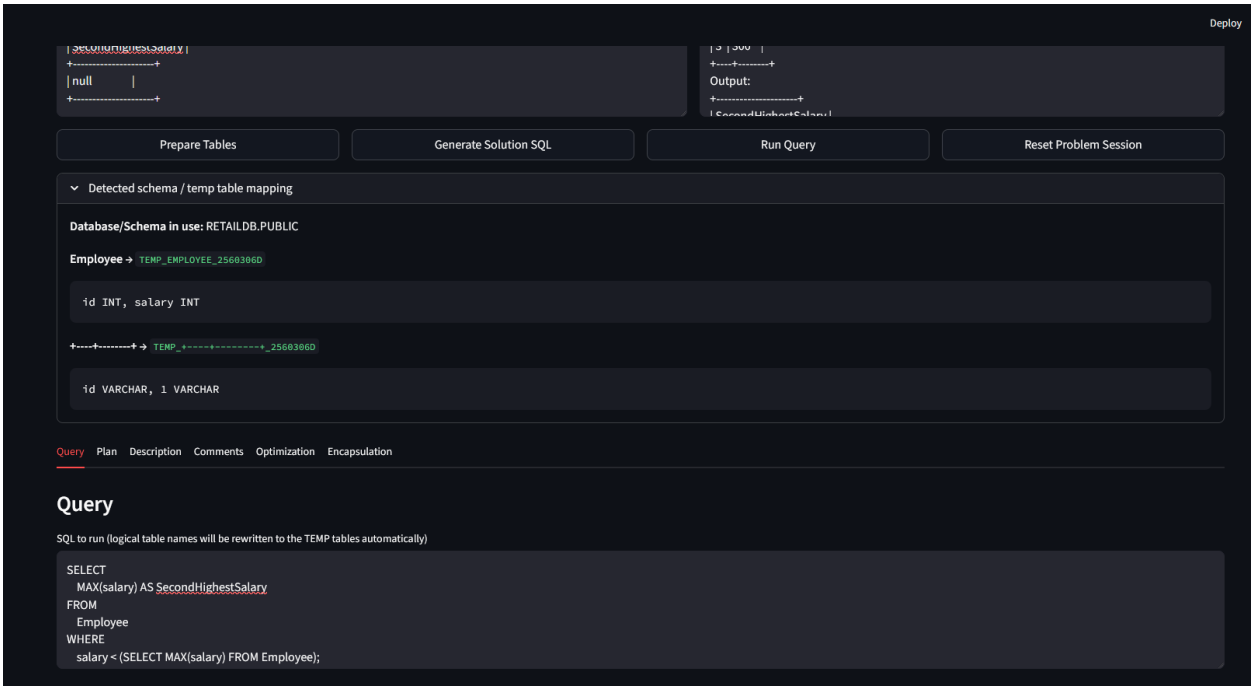
1. You paste a problem + sample input → it auto-detects table schemas, creates matching tables, and loads the sample rows so you can run queries instantly.
2. It can also generate a correct Snowflake SQL solution with AI, then explain / comment / optimize it—so you learn patterns (joins, windows, grouping, edge cases).
3. Best use case: SQL upskilling with real execution—practice problems end-to-end (setup → query → results → explain/optimize) without manually creating tables or copying data every time.

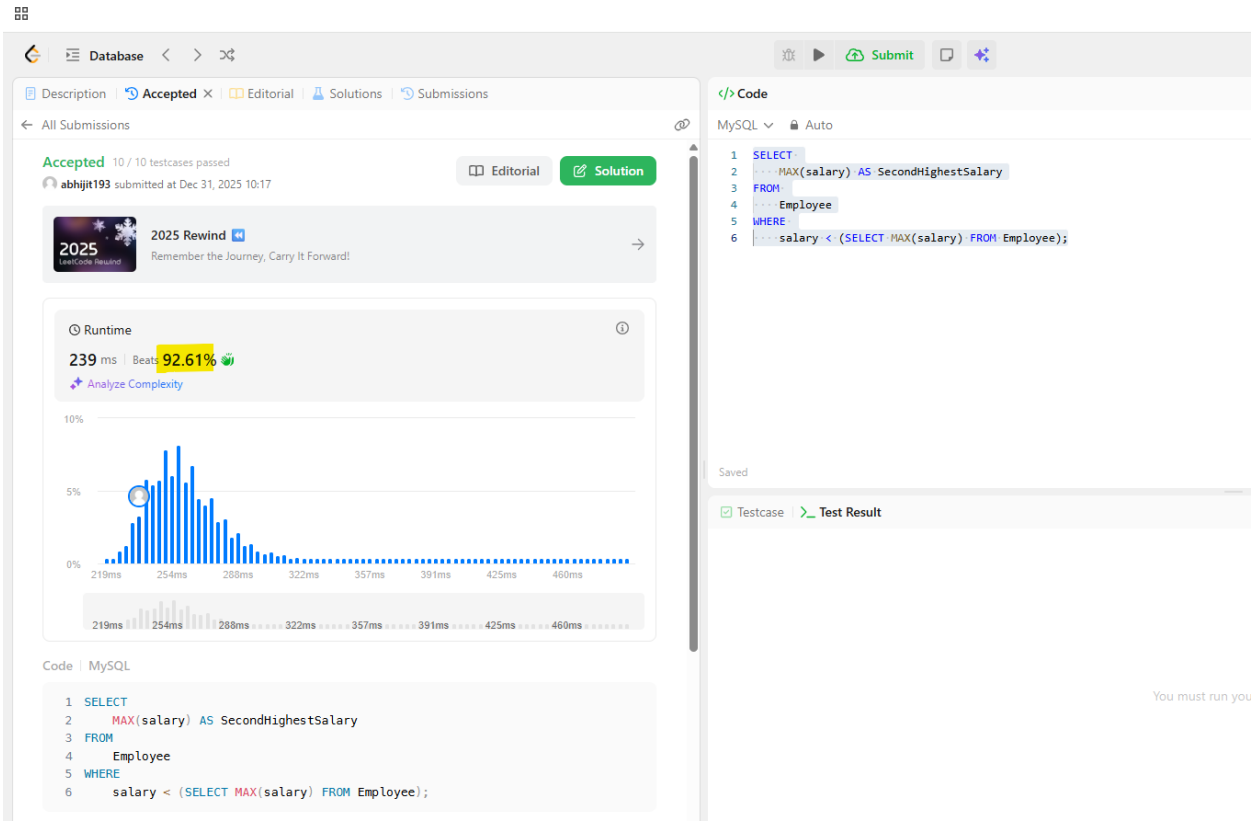
Here is a leetcode problem:

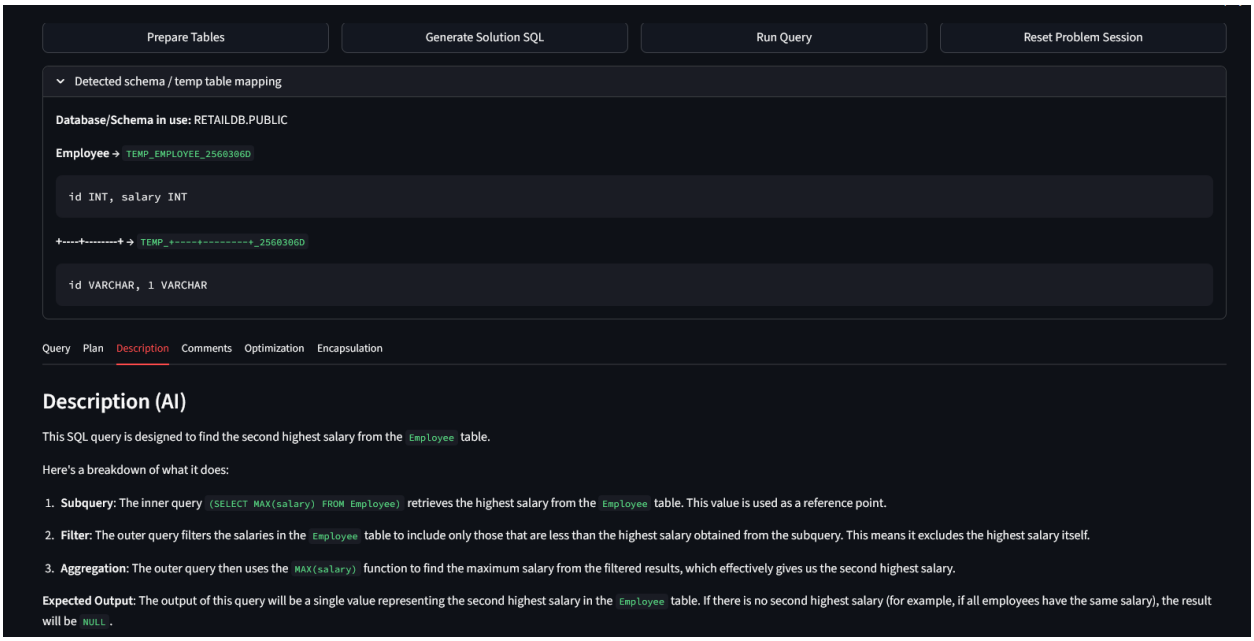


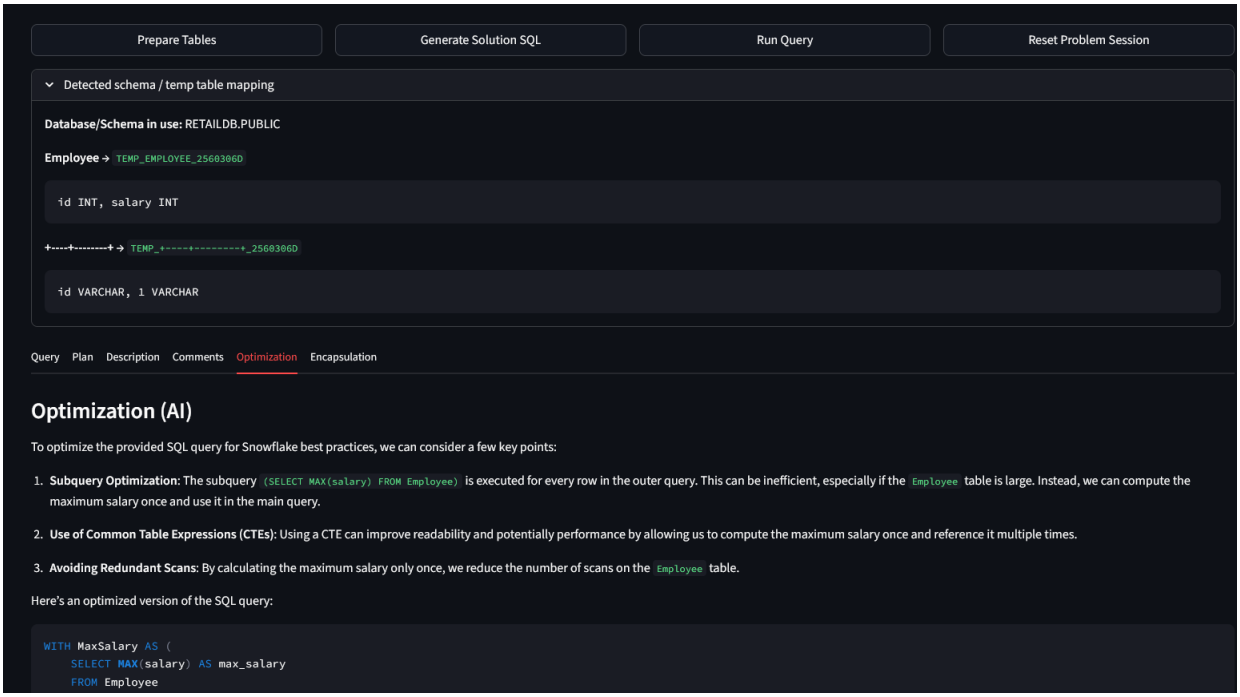
Copy paste the problem to the application:

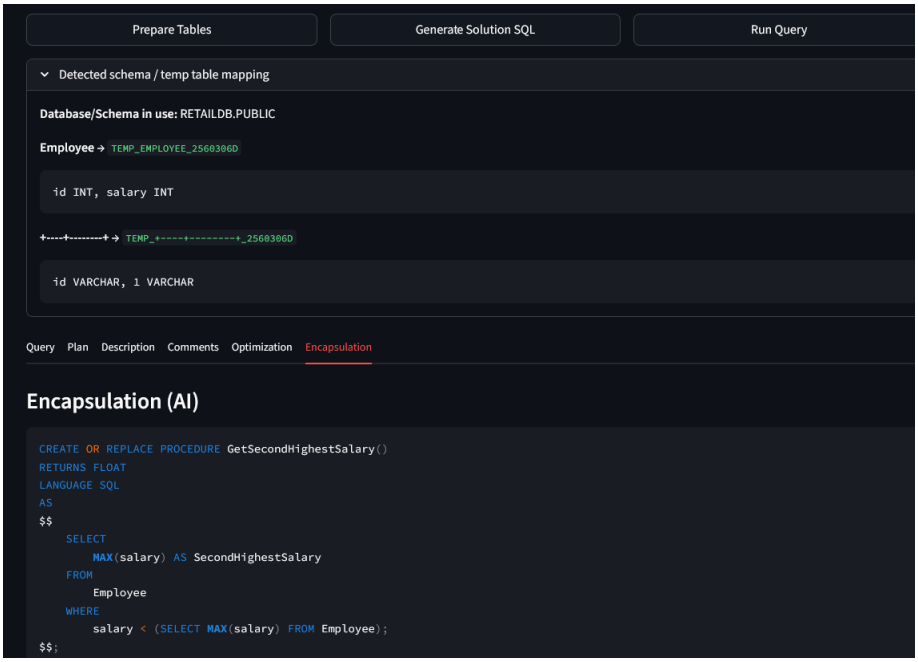



Deploy
| null |
Output:
Prepare Tables
Generate Solution SQL
Run Query
Reset Problem Session
Detected schema / temp table mapping
Database/Schema in use: RETAILDB.PUBLIC
Employee → TEMP_EMPLOYEE_2560306D
id INT, salary INT
id VARCHAR, 1 VARCHAR
Query Plan Description Comments Optimization Encapsulation
Query
SQL to run (logical table names will be rewritten to the TEMP tables automatically)
SELECT
MAX(salary) AS SecondHighestSalary
FROM
Employee
WHERE
salary < (SELECT MAX(salary) FROM Employee);


Database
Submit
Description Accepted Editorial Solutions Submissions
All Submissions
Accepted 10 / 10 testcases passed
abhijit193 submitted at Dec 31, 2025 10:17
Editorial
Solution
2025 Rewind
Remember the Journey, Carry It Forward!
Runtime
239 ms | Beats 92.61%
Analyze Complexity
10%
5%
0%
219ms 254ms 288ms 322ms 357ms 391ms 425ms 460ms
Code | MySQL
1 SELECT
2 MAX(salary) AS SecondHighestSalary
3 FROM
4 Employee
5 WHERE
6 salary < (SELECT MAX(salary) FROM Employee);
Code
MySQL
Auto
1 SELECT
2 MAX(salary) AS SecondHighestSalary
3 FROM
4 Employee
5 WHERE
6 salary < (SELECT MAX(salary) FROM Employee);
Saved
Testcase
Test Result
You must run you


Prepare Tables
Generate Solution SQL
Run Query
Reset Problem Session
Detected schema / temp table mapping
Database/Schema in use: RETAILDB.PUBLIC
Employee → TEMP_EMPLOYEE_2560306D
id INT, salary INT
+----+--------+ → TEMP_+----+--------+_2560306D
id VARCHAR, 1 VARCHAR
Query Plan Description Comments Optimization Encapsulation
Description (AI)
This SQL query is designed to find the second highest salary from the Employee table.
Here's a breakdown of what it does:
1. Subquery: The inner query (SELECT MAX(salary) FROM Employee) retrieves the highest salary from the Employee table. This value is used as a reference point.
2. Filter: The outer query filters the salaries in the Employee table to include only those that are less than the highest salary obtained from the subquery. This means it excludes the highest salary itself.
3. Aggregation: The outer query then uses the MAX(salary) function to find the maximum salary from the filtered results, which effectively gives us the second highest salary.
Expected Output: The output of this query will be a single value representing the second highest salary in the Employee table. If there is no second highest salary (for example, if all employees have the same salary), the result will be NULL .


Prepare Tables
Generate Solution SQL
Run Query
Reset Problem Session
Detected schema / temp table mapping
Database/Schema in use: RETAILDB.PUBLIC
Employee → TEMP_EMPLOYEE_2560306D
id INT, salary INT
+----+--------+ → TEMP_+----+--------+_2560306D
id VARCHAR, 1 VARCHAR
Query Plan Description Comments Optimization Encapsulation
Optimization (AI)
To optimize the provided SQL query for Snowflake best practices, we can consider a few key points:
1. Subquery Optimization: The subquery (SELECT MAX(salary) FROM Employee) is executed for every row in the outer query. This can be inefficient, especially if the Employee table is large. Instead, we can compute the maximum salary once and use it in the main query.
2. Use of Common Table Expressions (CTEs): Using a CTE can improve readability and potentially performance by allowing us to compute the maximum salary once and reference it multiple times.
3. Avoiding Redundant Scans: By calculating the maximum salary only once, we reduce the number of scans on the Employee table.
Here's an optimized version of the SQL query:
WITH MaxSalary AS (
SELECT MAX(salary) AS max_salary
FROM Employee


Prepare Tables
Generate Solution SQL
Run Query
Detected schema / temp table mapping
Database/Schema in use: RETAILDB.PUBLIC
Employee → TEMP_EMPLOYEE_2560306D
id INT, salary INT
+----+--------+ → TEMP_+----+--------+_2560306D
id VARCHAR, 1 VARCHAR
Query Plan Description Comments Optimization Encapsulation
Encapsulation (AI)
CREATE OR REPLACE PROCEDURE GetSecondHighestSalary()
RETURNS FLOAT
LANGUAGE SQL
AS
$$
SELECT
MAX(salary) AS SecondHighestSalary
FROM
Employee
WHERE
salary < (SELECT MAX(salary) FROM Employee);
$$;