

Snowflake – IDE Code Assistant

The screenshot shows a code editor interface with several tabs at the top:

- test-code.py 2, M X
- 1-create-script.sql
- app_mcp.py 1, U
- ai_tools_mcp.py 1, U
- mcp_server_5.py 4, U
- app.py

The main area displays Python code for a Streamlit application. The code includes imports for os, streamlit, snowflake.snowpark.Session, openai.OpenAI, and re. It defines two functions: getSession() and getChatResponse(). The getSession() function retrieves credentials from secrets and creates a Session object. The getChatResponse() function uses an OpenAI client to generate a completion based on a prompt.

```
01-vscode-assistant > test-code.py > ...
1 # This code is designed to create a Streamlit application that allows users to analyze and optimize
2 # Snowflake SQL queries. It includes the following features:
3 import os
4 import streamlit as st
5 from snowflake.snowpark import Session
6 from openai import OpenAI
7 import re
8
9 @st.cache_resource(show_spinner="Connecting...")
10 def getSession():
11     account = st.secrets["connections"]["snowflake"]["account"]
12     user = st.secrets["connections"]["snowflake"].get("user")
13     password = os.environ.get("SNOWFLAKE_PASSWORD")
14
15     if not all([account, user, password]):
16         raise RuntimeError("Snowflake credentials not fully configured")
17
18     return Session.builder.configs({
19         "account": account,
20         "user": user,
21         "password": password
22     }).create()
23
24
25 def getChatResponse(prompt):
26     client = OpenAI(api_key=os.environ["OPENAI_API_KEY"])
27     response = client.chat.completions.create(
28         model="gpt-4o-mini",
29         messages=[{"role": "user", "content": prompt}]
30     )
31     return response.choices[0].message.content
32
33
```

Query Analyzer and Optimizer

Analyze and optimize Snowflake SQL queries for correctness and performance.

Query Plan Description Comments Optimization Encapsulation

```
select count(*) from snowflake_sample_data.tpch_sf1.lineitem
```

	COUNT(*)	
0		6001215

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:

176. Second Highest Salary

Table: Employee

```
+-----+-----+
| Column Name | Type |
+-----+-----+
| id          | int   |
| salary      | int   |
+-----+-----+
```

id is the primary key (column with unique values) for this table.
Each row of this table contains information about the salary of an employee.

Write a solution to find the second highest **distinct** salary from the `Employee` table. If there is no second highest salary, return `null` (return `None` in Pandas).

The result format is in the following example.

Example 1:

Input:
Employee table:

id	salary
1	100
2	200
3	300

Output:

SecondHighestSalary
null

4.1K 379 102 Online

Copy paste the problem to the application:

Snowflake Connection

Keep your credential options here. You can use secrets/env or type them below.

Account: jt 9

User: abhijitsnowdemo

Password: *****

Role (optional): ACCOUNTADMIN

Warehouse (optional): R_DB

Database (optional): PUBLIC

Schema (optional): Connect

Snowflake LeetCode SQL Assistant

Paste a LeetCode SQL problem + sample input, auto-create TEMP tables, run query, and get AI explain/comment/optimize.

1) Paste Problem Statement

LeetCode problem statement (include the Table schema blocks)

```
Employee table:
+-----+-----+
| id | salary |
+-----+-----+
| 1  | 100  |
+-----+-----+
Output:
+-----+
| null |
+-----+
```

2) Paste Sample Input (Optional but recommended)

LeetCode 'Input' tables (ASCII tables).
Paste the Employee table / Bonus table input blocks here...

Prepare Tables Generate Solution SQL Run Query Reset Problem Session

Query

SQL to run (logical table names will be rewritten to the TEMP tables automatically)
Click 'Generate Solution SQL' or write your own SQL here...

Deploy

```
+-----+  
| SecondHighestSalary |  
+-----+  
| null |  
+-----+
```

```
+-----+  
| Output:  
+-----+  
| SecondHighestSalary |
```

Prepare Tables
Generate Solution SQL
Run Query
Reset Problem Session

Detected schema / temp table mapping

Database/Schema in use: RETAILDB.PUBLIC

Employee → TEMP_EMPLOYEE_2560396D

```
id INT, salary INT  
+-----+  
+-----+→ TEMP_+-----+_2560396D  
id VARCHAR, l 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 < > ↻

Description | Accepted | Editorial | Solutions | Submissions
Submit

All Submissions
MySQL | Auto

Accepted 10 / 10 testcases passed

abhijit193 submitted at Dec 31, 2025 10:17

2025 Rewind Remember the Journey, Carry It Forward!

Runtime 239 ms | Beats 92.61% Analyze Complexity

Code : MySQL

```
1 SELECT
2   MAX(salary) AS SecondHighestSalary
3 FROM
4   Employee
5 WHERE
6   salary < (SELECT MAX(salary) FROM Employee);
```

Code

```
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 your code to see the results.

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_2560360

```
id INT, salary INT
+----+-----+ → TEMP_+----+-----+_2560360
id VARCHAR, l1 VARCHAR
```

Query Plan Description Comments Optimization Encapsulation

Encapsulation (AI)

```
CREATE OR REPLACE PROCEDURE GetSecondHighestSalary()
RETURNS FLOAT
LANGUAGE SQL
AS
$$
SELECT
    MAX(salary) AS SecondHighestSalary
FROM
    Employee
WHERE
    salary < (SELECT MAX(salary) FROM Employee);
$$;
```