```
from unittest.mock import patch

from swarms_cloud.auth_with_swarms_cloud import fetch_api_key_info

import asyncio



@patch("swarms_cloud.auth_with_swarms_cloud.supabase_client_init")

def test_fetch_api_key_info_valid_token(mock_supabase):

    # Arrange

    mock_supabase.table().select().filter().execute.return_value = {

        "data": [{"id": "1", "user_id": "2", "key": "valid_token"}]

    }

    token = "valid_token"


    # Act

    result = asyncio.run(fetch_api_key_info(token, mock_supabase))


    # Assert

    assert result == {"id": "1", "user_id": "2", "key": "valid_token"}



@patch("swarms_cloud.auth_with_swarms_cloud.supabase_client_init")

def test_fetch_api_key_info_invalid_token(mock_supabase):

    # Arrange

    mock_supabase.table().select().filter().execute.return_value = {"data": None}

    token = "invalid_token"
    # Act
    result = asyncio.run(fetch_api_key_info(token, mock_supabase))

    # Assert
    assert result is None


@patch("swarms_cloud.auth_with_swarms_cloud.supabase_client_init")
def test_fetch_api_key_info_exception(mock_supabase):
    # Arrange
    mock_supabase.table().select().filter().execute.side_effect = Exception(
        "Database error"
    )
    token = "valid_token"

    # Act
    result = asyncio.run(fetch_api_key_info(token, mock_supabase))

    # Assert
    assert result is None
```