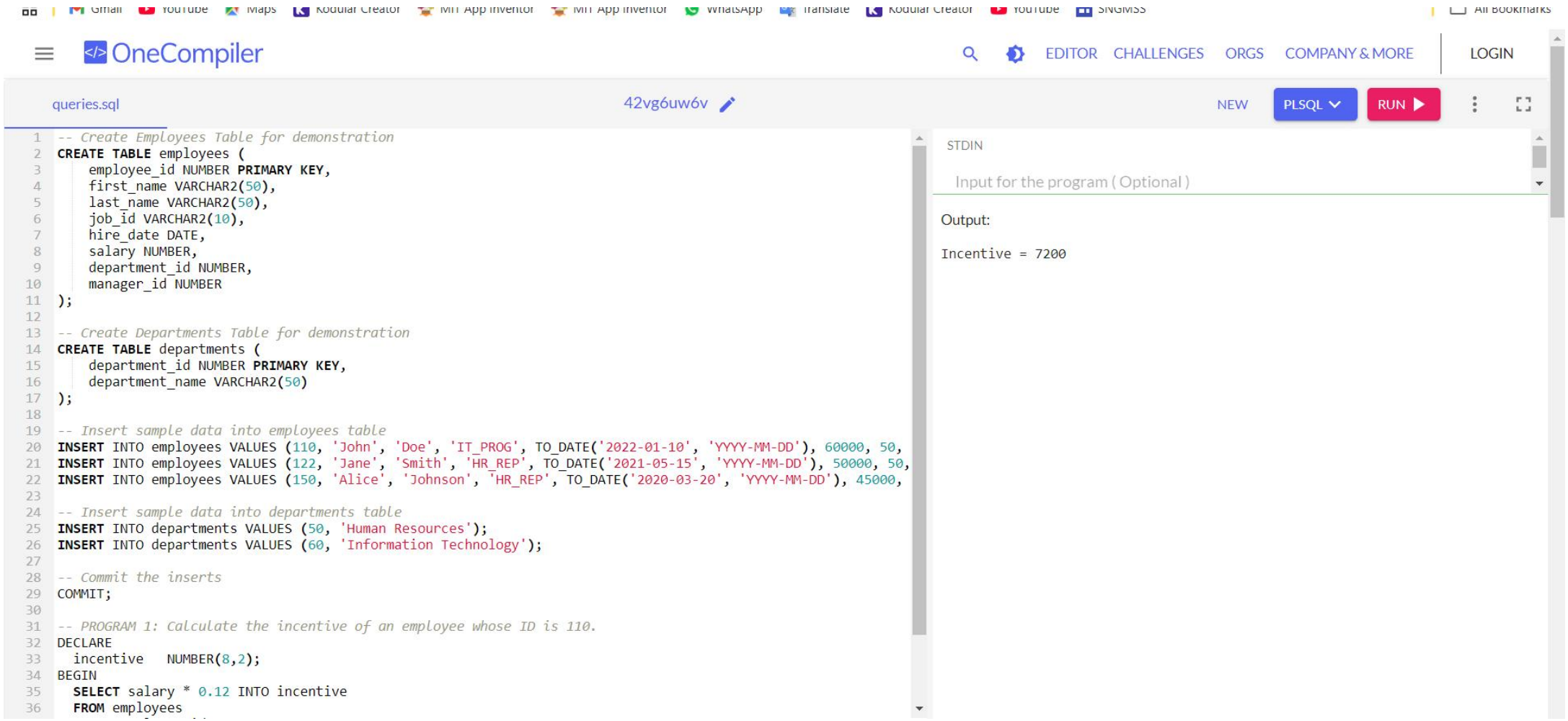
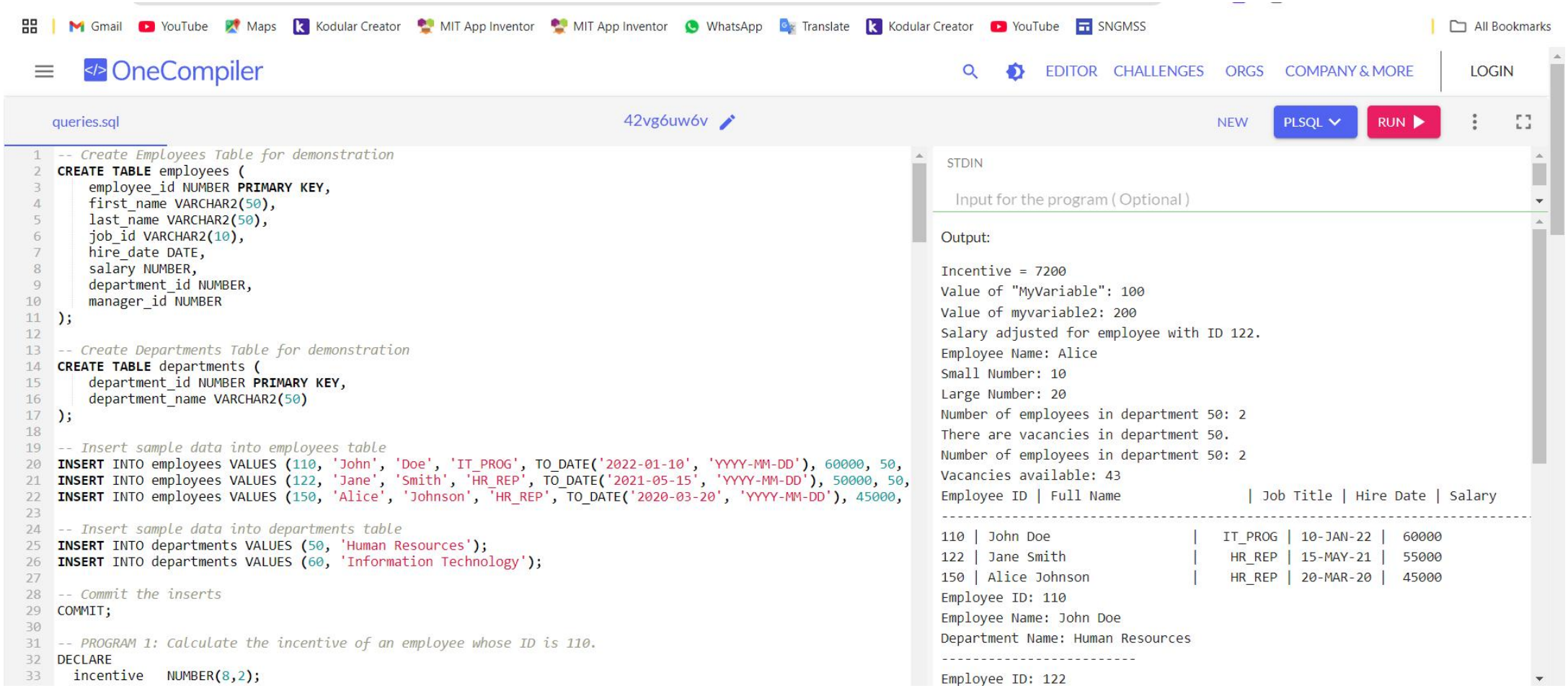
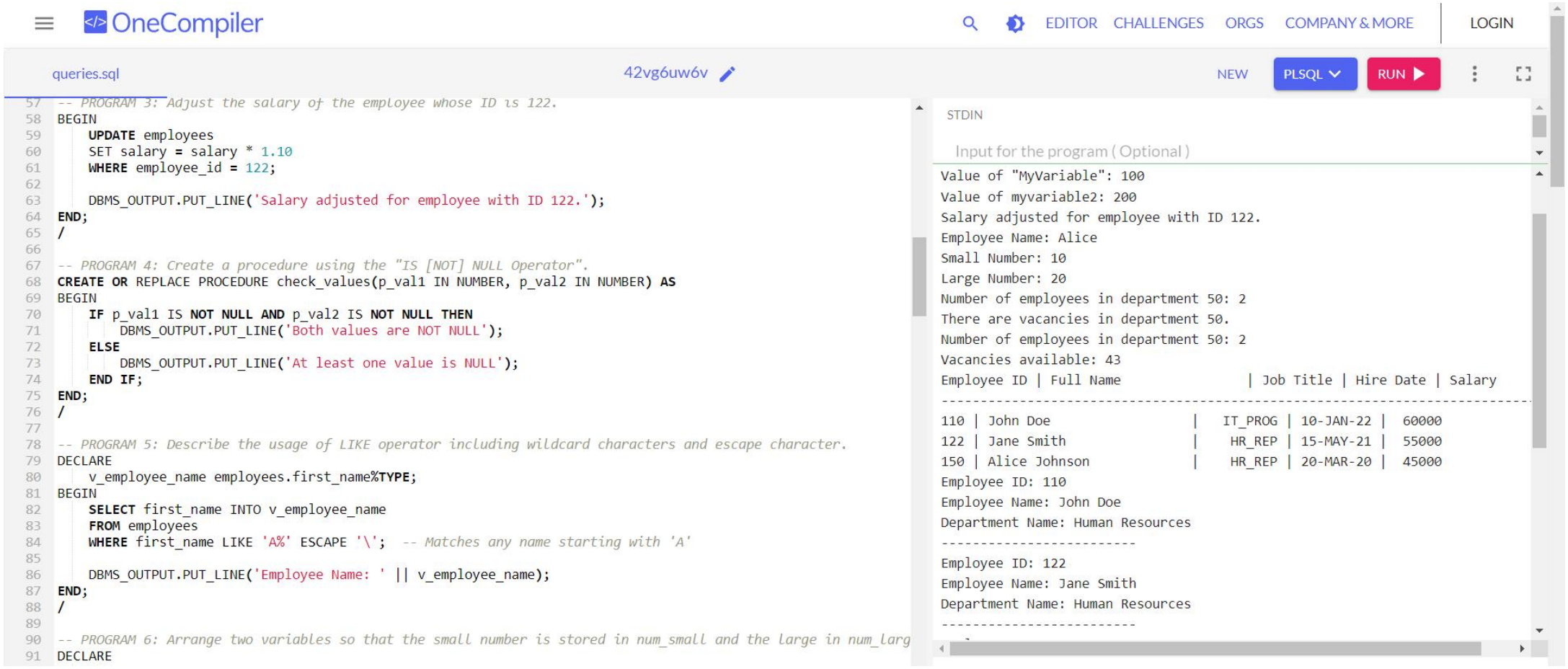
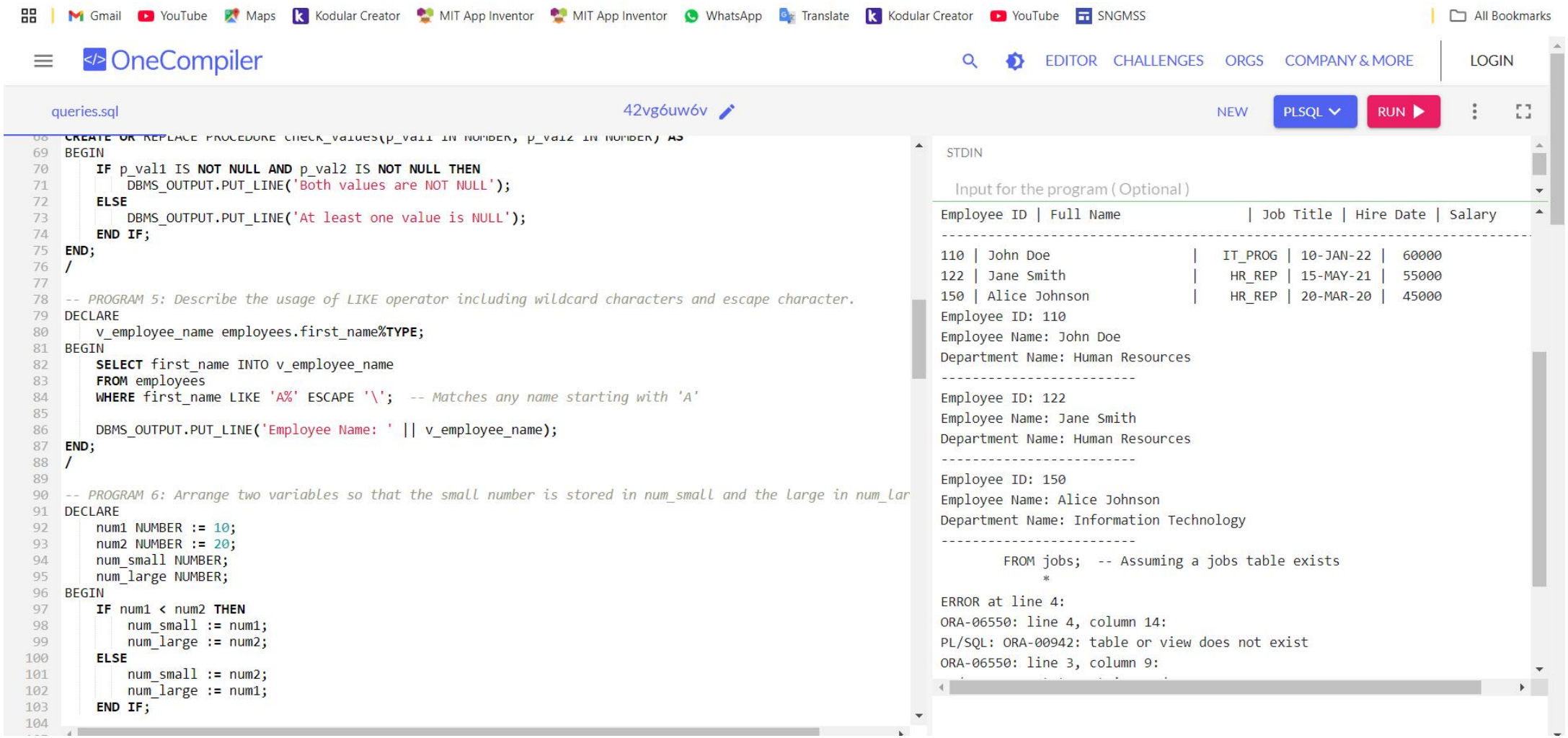



```sql
-- Create Employees Table for demonstration
CREATE TABLE employees (
    employee_id NUMBER PRIMARY KEY,
    first_name VARCHAR2(50),
    last_name VARCHAR2(50),
    job_id VARCHAR2(10),
    hire_date DATE,
    salary NUMBER,
    department_id NUMBER,
    manager_id NUMBER
);

-- Create Departments Table for demonstration
CREATE TABLE departments (
    department_id NUMBER PRIMARY KEY,
    department_name VARCHAR2(50)
);

-- Insert sample data into employees table
INSERT INTO employees VALUES (110, 'John', 'Doe', 'IT_PROG', TO_DATE('2022-01-10', 'YYYY-MM-DD'), 60000, 50,
INSERT INTO employees VALUES (122, 'Jane', 'Smith', 'HR_REP', TO_DATE('2021-05-15', 'YYYY-MM-DD'), 50000, 50,
INSERT INTO employees VALUES (150, 'Alice', 'Johnson', 'HR_REP', TO_DATE('2020-03-20', 'YYYY-MM-DD'), 45000,

-- Insert sample data into departments table
INSERT INTO departments VALUES (50, 'Human Resources');
INSERT INTO departments VALUES (60, 'Information Technology');

-- Commit the inserts
COMMIT;

-- PROGRAM 1: Calculate the incentive of an employee whose ID is 110.
DECLARE
  incentive   NUMBER(8,2);
BEGIN
  SELECT salary * 0.12 INTO incentive
  FROM employees
```


Gmail YouTube Maps Kodular Creator MIT App Inventor MIT App Inventor WhatsApp Translate Kodular Creator YouTube SNGMSS All Bookmarks
OneCompiler
EDITOR CHALLENGES ORGS COMPANY & MORE LOGIN
queries.sql
42vg6uw6v
NEW
PLSQL
RUN
-- Create Employees Table for demonstration
CREATE TABLE employees (
employee_id NUMBER PRIMARY KEY,
first_name VARCHAR2(50),
last_name VARCHAR2(50),
job_id VARCHAR2(10),
hire_date DATE,
salary NUMBER,
department_id NUMBER,
manager_id NUMBER
);
-- Create Departments Table for demonstration
CREATE TABLE departments (
department_id NUMBER PRIMARY KEY,
department_name VARCHAR2(50)
);
-- Insert sample data into employees table
INSERT INTO employees VALUES (110, 'John', 'Doe', 'IT_PROG', TO_DATE('2022-01-10', 'YYYY-MM-DD'), 60000, 50,
INSERT INTO employees VALUES (122, 'Jane', 'Smith', 'HR_REP', TO_DATE('2021-05-15', 'YYYY-MM-DD'), 50000, 50,
INSERT INTO employees VALUES (150, 'Alice', 'Johnson', 'HR_REP', TO_DATE('2020-03-20', 'YYYY-MM-DD'), 45000,
-- Insert sample data into departments table
INSERT INTO departments VALUES (50, 'Human Resources');
INSERT INTO departments VALUES (60, 'Information Technology');
-- Commit the inserts
COMMIT;
-- PROGRAM 1: Calculate the incentive of an employee whose ID is 110.
DECLARE
incentive NUMBER(8,2);
STDIN
Input for the program (Optional)
Output:
Incentive = 7200
Value of "MyVariable": 100
Value of myvariable2: 200
Salary adjusted for employee with ID 122.
Employee Name: Alice
Small Number: 10
Large Number: 20
Number of employees in department 50: 2
There are vacancies in department 50.
Number of employees in department 50: 2
Vacancies available: 43
Employee ID | Full Name | Job Title | Hire Date | Salary
110 | John Doe | IT_PROG | 10-JAN-22 | 60000
122 | Jane Smith | HR_REP | 15-MAY-21 | 55000
150 | Alice Johnson | HR_REP | 20-MAR-20 | 45000
Employee ID: 110
Employee Name: John Doe
Department Name: Human Resources
Employee ID: 122


OneCompiler
EDITOR CHALLENGES ORGS COMPANY & MORE
LOGIN
queries.sql
42vg6uw6v
NEW
PLSQL
RUN
-- PROGRAM 3: Adjust the salary of the employee whose ID is 122.
BEGIN
UPDATE employees
SET salary = salary * 1.10
WHERE employee_id = 122;
DBMS_OUTPUT.PUT_LINE('Salary adjusted for employee with ID 122.');
END;
/
-- PROGRAM 4: Create a procedure using the "IS [NOT] NULL Operator".
CREATE OR REPLACE PROCEDURE check_values(p_val1 IN NUMBER, p_val2 IN NUMBER) AS
BEGIN
IF p_val1 IS NOT NULL AND p_val2 IS NOT NULL THEN
DBMS_OUTPUT.PUT_LINE('Both values are NOT NULL');
ELSE
DBMS_OUTPUT.PUT_LINE('At least one value is NULL');
END IF;
END;
/
-- PROGRAM 5: Describe the usage of LIKE operator including wildcard characters and escape character.
DECLARE
v_employee_name employees.first_name%TYPE;
BEGIN
SELECT first_name INTO v_employee_name
FROM employees
WHERE first_name LIKE 'A%' ESCAPE '\'; -- Matches any name starting with 'A'
DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_employee_name);
END;
/
-- PROGRAM 6: Arrange two variables so that the small number is stored in num_small and the large in num_larg
DECLARE
STDIN
Input for the program (Optional)
Value of "MyVariable": 100
Value of myvariable2: 200
Salary adjusted for employee with ID 122.
Employee Name: Alice
Small Number: 10
Large Number: 20
Number of employees in department 50: 2
There are vacancies in department 50.
Number of employees in department 50: 2
Vacancies available: 43
Employee ID | Full Name | Job Title | Hire Date | Salary
110 | John Doe | IT_PROG | 10-JAN-22 | 60000
122 | Jane Smith | HR_REP | 15-MAY-21 | 55000
150 | Alice Johnson | HR_REP | 20-MAR-20 | 45000
Employee ID: 110
Employee Name: John Doe
Department Name: Human Resources
Employee ID: 122
Employee Name: Jane Smith
Department Name: Human Resources


Gmail YouTube Maps Kodular Creator MIT App Inventor MIT App Inventor WhatsApp Translate Kodular Creator YouTube SNGMSS All Bookmarks
OneCompiler
EDITOR CHALLENGES ORGS COMPANY & MORE LOGIN
queries.sql
42vg6uw6v
NEW PLSQL RUN
CREATE OR REPLACE PROCEDURE check_values(p_val1 IN NUMBER, p_val2 IN NUMBER) AS
BEGIN
IF p_val1 IS NOT NULL AND p_val2 IS NOT NULL THEN
DBMS_OUTPUT.PUT_LINE('Both values are NOT NULL');
ELSE
DBMS_OUTPUT.PUT_LINE('At least one value is NULL');
END IF;
END;
/
-- PROGRAM 5: Describe the usage of LIKE operator including wildcard characters and escape character.
DECLARE
v_employee_name employees.first_name%TYPE;
BEGIN
SELECT first_name INTO v_employee_name
FROM employees
WHERE first_name LIKE 'A%' ESCAPE '\'; -- Matches any name starting with 'A'
DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_employee_name);
END;
/
-- PROGRAM 6: Arrange two variables so that the small number is stored in num_small and the large in num_lar
DECLARE
num1 NUMBER := 10;
num2 NUMBER := 20;
num_small NUMBER;
num_large NUMBER;
BEGIN
IF num1 < num2 THEN
num_small := num1;
num_large := num2;
ELSE
num_small := num2;
num_large := num1;
END IF;
STDIN
Input for the program (Optional)
Employee ID | Full Name | Job Title | Hire Date | Salary
110 | John Doe | IT_PROG | 10-JAN-22 | 60000
122 | Jane Smith | HR_REP | 15-MAY-21 | 55000
150 | Alice Johnson | HR_REP | 20-MAR-20 | 45000
Employee ID: 110
Employee Name: John Doe
Department Name: Human Resources
Employee ID: 122
Employee Name: Jane Smith
Department Name: Human Resources
Employee ID: 150
Employee Name: Alice Johnson
Department Name: Information Technology
FROM jobs; -- Assuming a jobs table exists
*
ERROR at line 4:
ORA-06550: line 4, column 14:
PL/SQL: ORA-00942: table or view does not exist
ORA-06550: line 3, column 9:

queries.sql 42vg6uw6v NEW PLSQL RUN

```
            DBMS_OUTPUT.PUT_LINE('Record updated with new incentive.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Target not achieved. No record updated.');
    END IF;
END;
/

-- PROGRAM 8: Calculate incentive achieved according to a specific sale limit.
CREATE OR REPLACE PROCEDURE calculate_specific_incentive(p_sales_amount IN NUMBER) AS
    v_incentive NUMBER;
BEGIN
    IF p_sales_amount > 5000 THEN
        v_incentive := p_sales_amount * 0.15;  -- 15% incentive for sales above 5000
        DBMS_OUTPUT.PUT_LINE('Incentive calculated: ' || v_incentive);
    ELSE
        DBMS_OUTPUT.PUT_LINE('No incentive, sales below limit.');
    END IF;
END;
/

-- PROGRAM 9: Count the number of employees in department 50 and check for vacancies.
DECLARE
    v_emp_count NUMBER;
    v_vacancies NUMBER := 45;  -- Total vacancies in department 50
BEGIN
    SELECT COUNT(*) INTO v_emp_count FROM employees WHERE department_id = 50;

    DBMS_OUTPUT.PUT_LINE('Number of employees in department 50: ' || v_emp_count);

    IF v_emp_count < v_vacancies THEN
        DBMS_OUTPUT.PUT_LINE('There are vacancies in department 50.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('No vacancies in department 50.');
    END IF;
END;
/
```

STDIN

Input for the program (Optional)

```
Small Number: 10
Large Number: 20
Number of employees in department 50: 2
There are vacancies in department 50.
Number of employees in department 50: 2
Vacancies available: 43
Employee ID | Full Name                | Job Title | Hire Date | Salary
--------------------------------------------------------------------------
110 | John Doe                  |   IT_PROG | 10-JAN-22 |  60000
122 | Jane Smith                |    HR_REP | 15-MAY-21 |  55000
150 | Alice Johnson             |    HR_REP | 20-MAR-20 |  45000
Employee ID: 110
Employee Name: John Doe
Department Name: Human Resources
-------------------------
Employee ID: 122
Employee Name: Jane Smith
Department Name: Human Resources
-------------------------
Employee ID: 150
Employee Name: Alice Johnson
Department Name: Information Technology
-------------------------
```

queries.sql 42vg6uw6v NEW PLSQL RUN

```
DECLARE
    v_department_id NUMBER := 50;  -- Example department ID
    v_emp_count NUMBER;
    v_vacancies NUMBER := 45;  -- Total vacancies in the department
BEGIN
    SELECT COUNT(*) INTO v_emp_count FROM employees WHERE department_id = v_department_id;

    DBMS_OUTPUT.PUT_LINE('Number of employees in department ' || v_department_id || ': ' || v_emp_count);

    IF v_emp_count < v_vacancies THEN
        DBMS_OUTPUT.PUT_LINE('Vacancies available: ' || (v_vacancies - v_emp_count));
    ELSE
        DBMS_OUTPUT.PUT_LINE('No vacancies in department ' || v_department_id);
    END IF;
END;
/

-- PROGRAM 11: Display employee IDs, names, job titles, hire dates, and salaries of all employees.
DECLARE
    v_employee_id     employees.employee_id%TYPE;
    v_full_name       VARCHAR2(50);
    v_job_id          employees.job_id%TYPE;
    v_hire_date       employees.hire_date%TYPE;
    v_salary          employees.salary%TYPE;

    CURSOR c_employees IS
        SELECT employee_id, first_name || ' ' || last_name AS full_name, job_id, hire_date, salary
        FROM employees;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Employee ID | Full Name               | Job Title | Hire Date | Salary');
    DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------------------');

    OPEN c_employees;
    FETCH c_employees INTO v_employee_id, v_full_name, v_job_id, v_hire_date, v_salary;

    WHILE c_employees%FOUND LOOP
```

STDIN

Input for the program (Optional)

```
Small Number: 10
Large Number: 20
Number of employees in department 50: 2
There are vacancies in department 50.
Number of employees in department 50: 2
Vacancies available: 43
Employee ID | Full Name               | Job Title | Hire Date | Salary
-------------------------------------------------------------------------------------------
110 | John Doe                  |   IT_PROG | 10-JAN-22 |  60000
122 | Jane Smith                |    HR_REP | 15-MAY-21 |  55000
150 | Alice Johnson             |    HR_REP | 20-MAR-20 |  45000
Employee ID: 110
Employee Name: John Doe
Department Name: Human Resources
-------------------------
Employee ID: 122
Employee Name: Jane Smith
Department Name: Human Resources
-------------------------
Employee ID: 150
Employee Name: Alice Johnson
Department Name: Information Technology
-------------------------
```

```
        DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp_record.employee_id);
        DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_record.employee_name);
        DBMS_OUTPUT.PUT_LINE('Department Name: ' || NVL(emp_record.department_name, 'No Department'));
        DBMS_OUTPUT.PUT_LINE('-------------------------');

        FETCH emp_cursor INTO emp_record;
    END LOOP;

    CLOSE emp_cursor;
END;
/

-- PROGRAM 13: Display job IDs, titles, and minimum salaries of all jobs.
DECLARE
    CURSOR job_cursor IS
        SELECT job_id, job_title, min_salary
        FROM jobs;  -- Assuming a jobs table exists

    job_record job_cursor%ROWTYPE;
BEGIN
    OPEN job_cursor;
    FETCH job_cursor INTO job_record;

    WHILE job_cursor%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE('Job ID: ' || job_record.job_id);
        DBMS_OUTPUT.PUT_LINE('Job Title: ' || job_record.job_title);
        DBMS_OUTPUT.PUT_LINE('Minimum Salary: ' || job_record.min_salary);
        DBMS_OUTPUT.PUT_LINE('-------------------------');

        FETCH job_cursor INTO job_record;
    END LOOP;

    CLOSE job_cursor;
END;
/
```

STDIN

Input for the program (Optional)

```
Value of myvariable2: 200
Salary adjusted for employee with ID 122.
Employee Name: Alice
Small Number: 10
Large Number: 20
Number of employees in department 50: 2
There are vacancies in department 50.
Number of employees in department 50: 2
Vacancies available: 43
Employee ID | Full Name                | Job Title | Hire Date | Salary
-------------------------------------------------------------------------
110 | John Doe                  |   IT_PROG | 10-JAN-22 |  60000
122 | Jane Smith                |    HR_REP | 15-MAY-21 |  55000
150 | Alice Johnson             |    HR_REP | 20-MAR-20 |  45000
Employee ID: 110
Employee Name: John Doe
Department Name: Human Resources
-------------------------
Employee ID: 122
Employee Name: Jane Smith
Department Name: Human Resources
-------------------------
Employee ID: 150
```