queries.sql 42ug4sujy

```sql
-- Create the Students table
CREATE TABLE Students (
  student_id      NUMBER PRIMARY KEY,
  first_name      VARCHAR2(50),
  last_name       VARCHAR2(50),
  department_id   NUMBER,
  enrollment_date DATE
);

-- Create or replace the procedure
CREATE OR REPLACE PROCEDURE GetStudentCountByDepartment(
  dept_id IN NUMBER,
  student_count OUT NUMBER
) AS
BEGIN
  SELECT COUNT(*)
  INTO student_count

  FROM Students
  WHERE department_id = dept_id;

END;

```

queries.sql 42ug4sujy NEW PLSQL RUN

```plsql
-- Create the Courses table
CREATE TABLE Courses (
  course_id      NUMBER PRIMARY KEY,
  course_name    VARCHAR2(100),
  department_id  NUMBER,
  created_date   DATE,
  last_modified  DATE
);

-- Create or replace the trigger
CREATE OR REPLACE TRIGGER UpdateLastModified
BEFORE UPDATE ON Courses
FOR EACH ROW
BEGIN
  :NEW.last_modified := SYSDATE;
END;
```

STDIN

Input for the program (Optional)

Output:

Program did not output anything!

queries.sql 42ug53cqx NEW PLSQL RUN

```sql
-- Create the Students table
CREATE TABLE Students (
  student_id    NUMBER PRIMARY KEY,
  student_name  VARCHAR2(100),
  department_id NUMBER,
  GPA           NUMBER(3, 2) -- Assuming GPA is a number with up to 2 decimal places
);

-- PL/SQL block with cursor
DECLARE
  CURSOR student_cursor IS
    SELECT student_name FROM Students WHERE GPA > 3.0;
  student_name_var Students.student_name%TYPE;
BEGIN
  OPEN student_cursor;
  LOOP
    FETCH student_cursor INTO student_name_var;
    EXIT WHEN student_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(student_name_var);
  END LOOP;
  CLOSE student_cursor;
END;
```

STDIN

Input for the program (Optional)

Output:

```
Program did not output anything!
```

```sql
-- Create the Students table
CREATE TABLE Students (
  student_id      NUMBER PRIMARY KEY,
  student_name    VARCHAR2(100),
  department_id   NUMBER,
  GPA             NUMBER(3, 2),
  graduation_year NUMBER
);

-- PL/SQL block to create ArchivedStudents table dynamically
DECLARE
  sql_stmt VARCHAR2(200);
BEGIN
  sql_stmt := 'CREATE TABLE ArchivedStudents AS SELECT * FROM Students WHERE graduation_year < 2020';
  EXECUTE IMMEDIATE sql_stmt;
END;
```

STDIN

Input for the program (Optional)

Output:

```
Program did not output anything!
```

queries.sql 42ug53cqx NEW PLSQL RUN

```sql
-- Create the Employees table
CREATE TABLE employees (
  employee_id   NUMBER PRIMARY KEY,
  first_name    VARCHAR2(50),
  last_name     VARCHAR2(50),
  department_id NUMBER,
  salary        NUMBER(8, 2)
);

-- Insert data into the Employees table
INSERT INTO employees (employee_id, first_name, last_name, department_id, salary) VALUES (110, 'John', 'Doe', 1
INSERT INTO employees (employee_id, first_name, last_name, department_id, salary) VALUES (111, 'Jane', 'Smith',
INSERT INTO employees (employee_id, first_name, last_name, department_id, salary) VALUES (112, 'Alice', 'Johnso
INSERT INTO employees (employee_id, first_name, last_name, department_id, salary) VALUES (113, 'Bob', 'Brown',

-- Commit the inserted data
COMMIT;

-- PL/SQL block to calculate the incentive
DECLARE
  incentive   NUMBER(8, 2);
BEGIN
  SELECT salary * 0.12 INTO incentive
  FROM employees
  WHERE employee_id = 110;

  DBMS_OUTPUT.PUT_LINE('Incentive = ' || TO_CHAR(incentive));
END;
/
```

STDIN

Input for the program (Optional)

Output:

```
Incentive = 600
```

queries.sql 42ug53cqx NEW PLSQL RUN

```sql
-- Create the Employees table
CREATE TABLE employees (
  employee_id   NUMBER PRIMARY KEY,
  first_name    VARCHAR2(50),
  last_name     VARCHAR2(50),
  department_id NUMBER,
  salary        NUMBER(8, 2)
);

-- Insert data into the Employees table
INSERT INTO employees (employee_id, first_name, last_name, department_id, salary) VALUES (110, 'John', 'Doe', 1
INSERT INTO employees (employee_id, first_name, last_name, department_id, salary) VALUES (111, 'Jane', 'Smith',
INSERT INTO employees (employee_id, first_name, last_name, department_id, salary) VALUES (112, 'Alice', 'Johnso
INSERT INTO employees (employee_id, first_name, last_name, department_id, salary) VALUES (113, 'Bob', 'Brown',

-- Commit the inserted data
COMMIT;

-- PL/SQL block to demonstrate variable reference
DECLARE
    MyVariable NUMBER := 100;  -- Quoted identifier (case-sensitive)
    myvariable2 NUMBER := 200;    -- Unquoted identifier
BEGIN
    -- Valid reference to the quoted identifier (case-sensitive, ust match exactly)
    DBMS_OUTPUT.PUT_LINE('Value of "MyVariable": ' || MyVariable);
    -- Invalid reference to unquoted identifier (this will cause an error)
    -- DBMS_OUTPUT.PUT_LINE('Invalid reference to MyVariable: ' || MyVariable);
    DBMS_OUTPUT.PUT_LINE('Value of myvariable2: ' || myvariable2);
    -- Invalid reference to MYVARIABLE2 (case-sensitive)
    -- DBMS_OUTPUT.PUT_LINE('Invalid reference to MYVARIABLE2: ' || MYVARIABLE2);
END;
/
```

STDIN

Input for the program (Optional)

Output:

```
Value of "MyVariable": 100
Value of myvariable2: 200
```