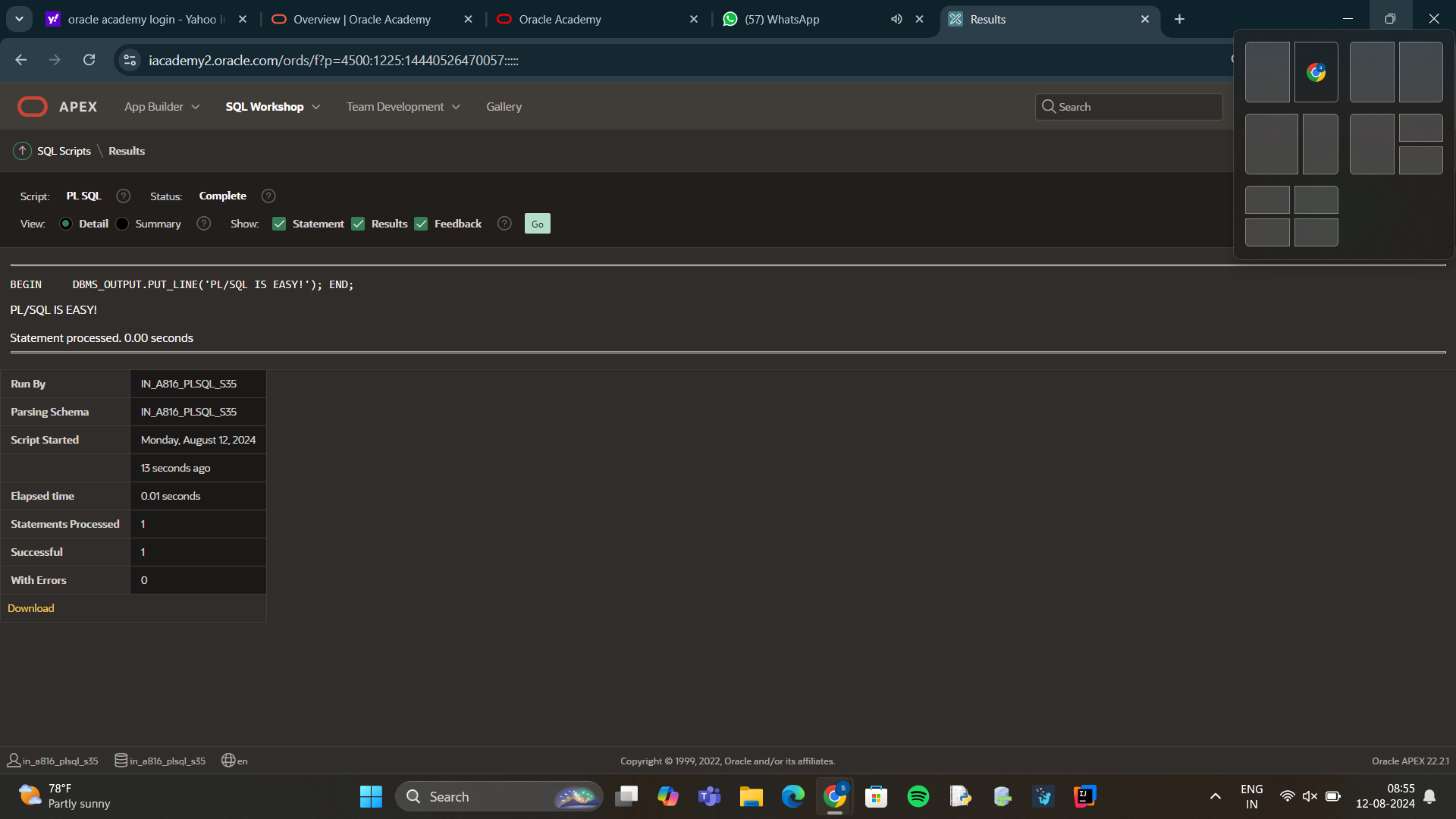
BEGIN

DBMS\_OUTPUT.PUT\_LINE('PL/SQL IS EASY!');

END;

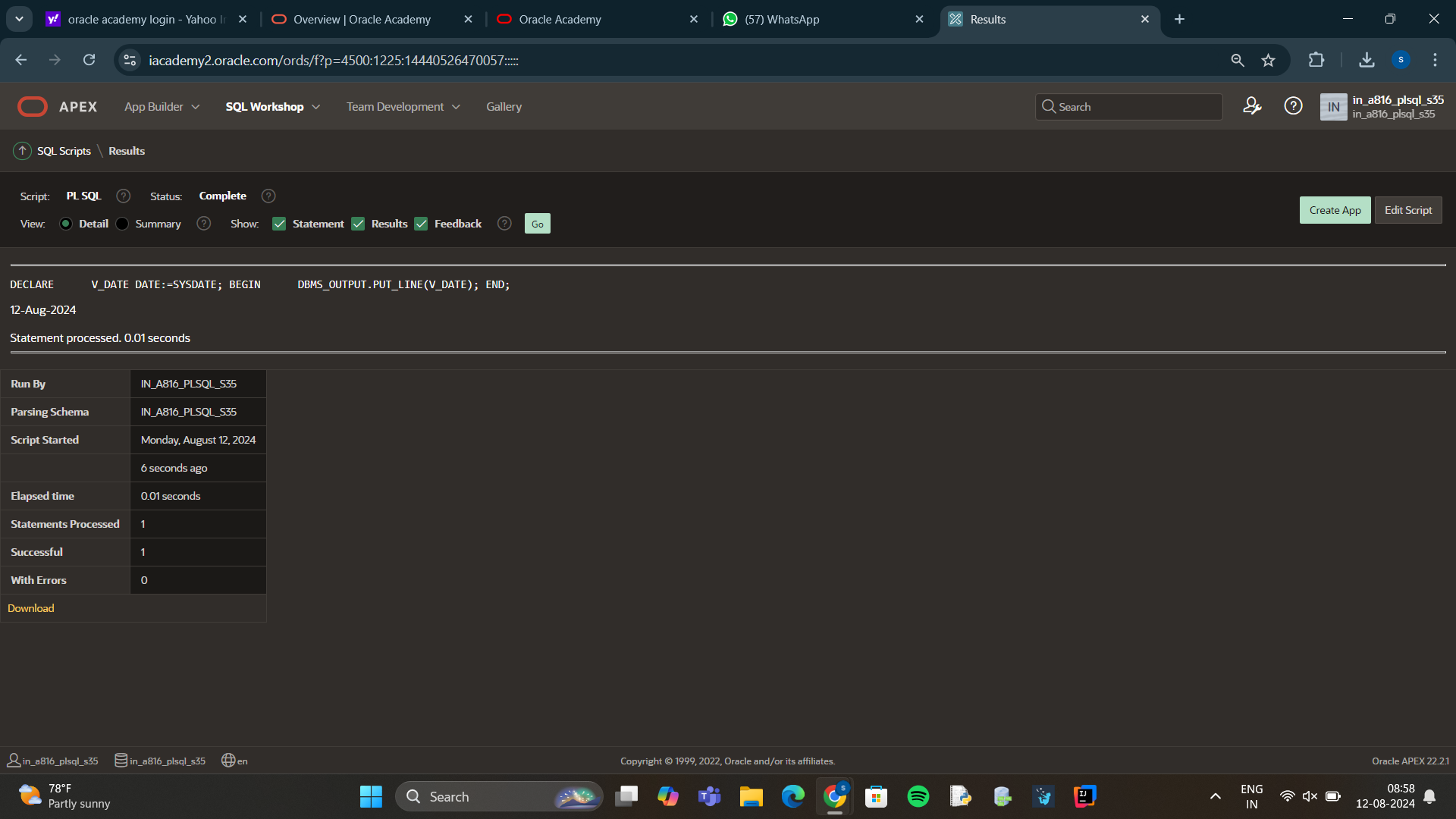
DECLARE

V\_DATE DATE:=SYSDATE;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(V\_DATE);

END;



DECLARE

FIRST\_NAME VARCHAR2(25);

LAST\_NAME VARCHAR2(25);

BEGIN

SELECT FIRST\_NAME,LAST\_NAME

INTO FIRST\_NAME, LAST\_NAME

FROM ROMA

WHERE last\_name = 'JAYASHANKAR';

DBMS\_OUTPUT.PUT\_LINE ('The employee of the month is: '

|| FIRST\_NAME || ' ' || LAST\_NAME || '.');

EXCEPTION

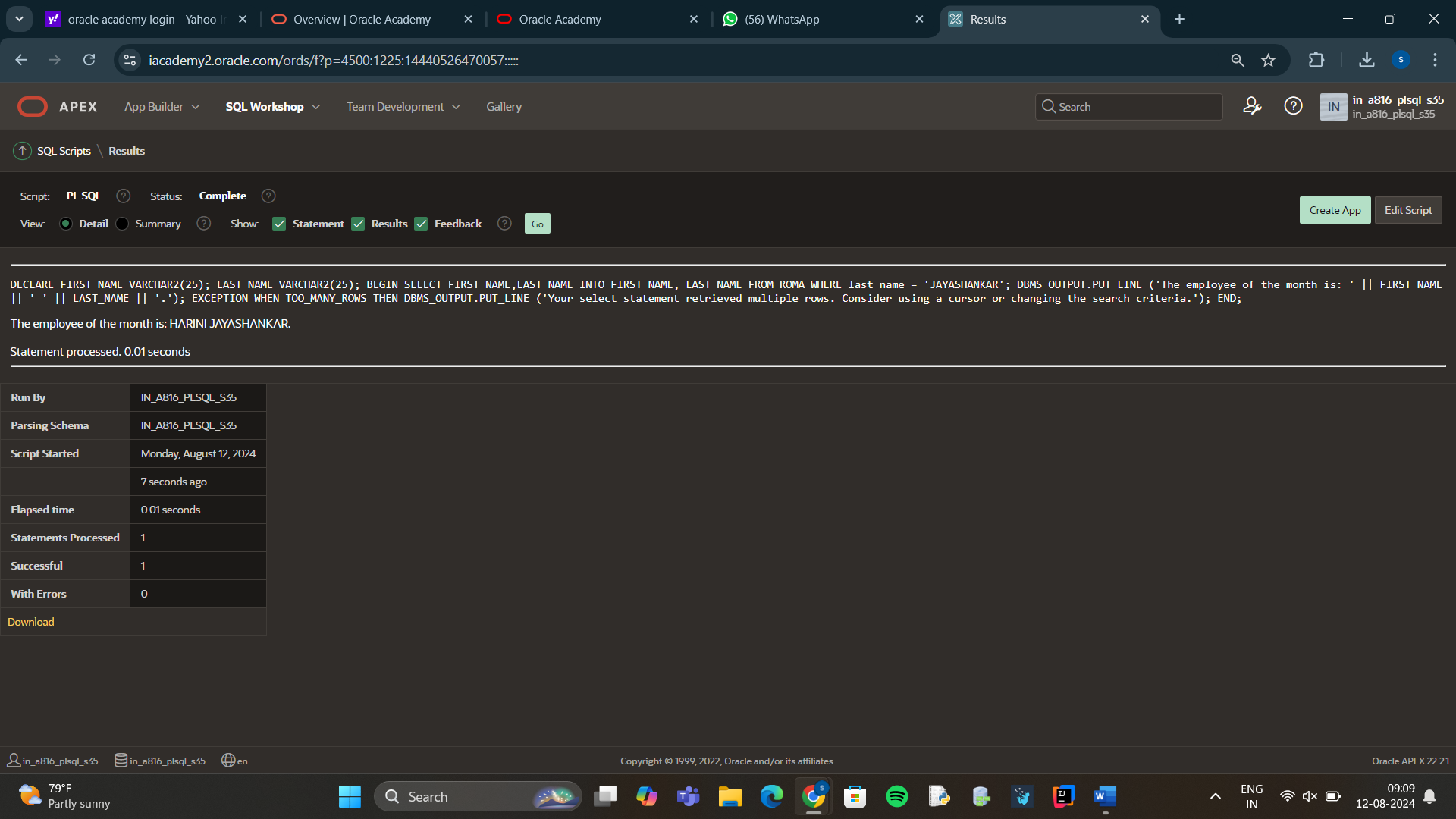
WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE ('Your select statement retrieved

multiple rows. Consider using a cursor or changing

the search criteria.');

END;



**SUBPROGRAM:**

CREATE OR REPLACE PROCEDURE print\_date IS

v\_date VARCHAR2(30);

BEGIN

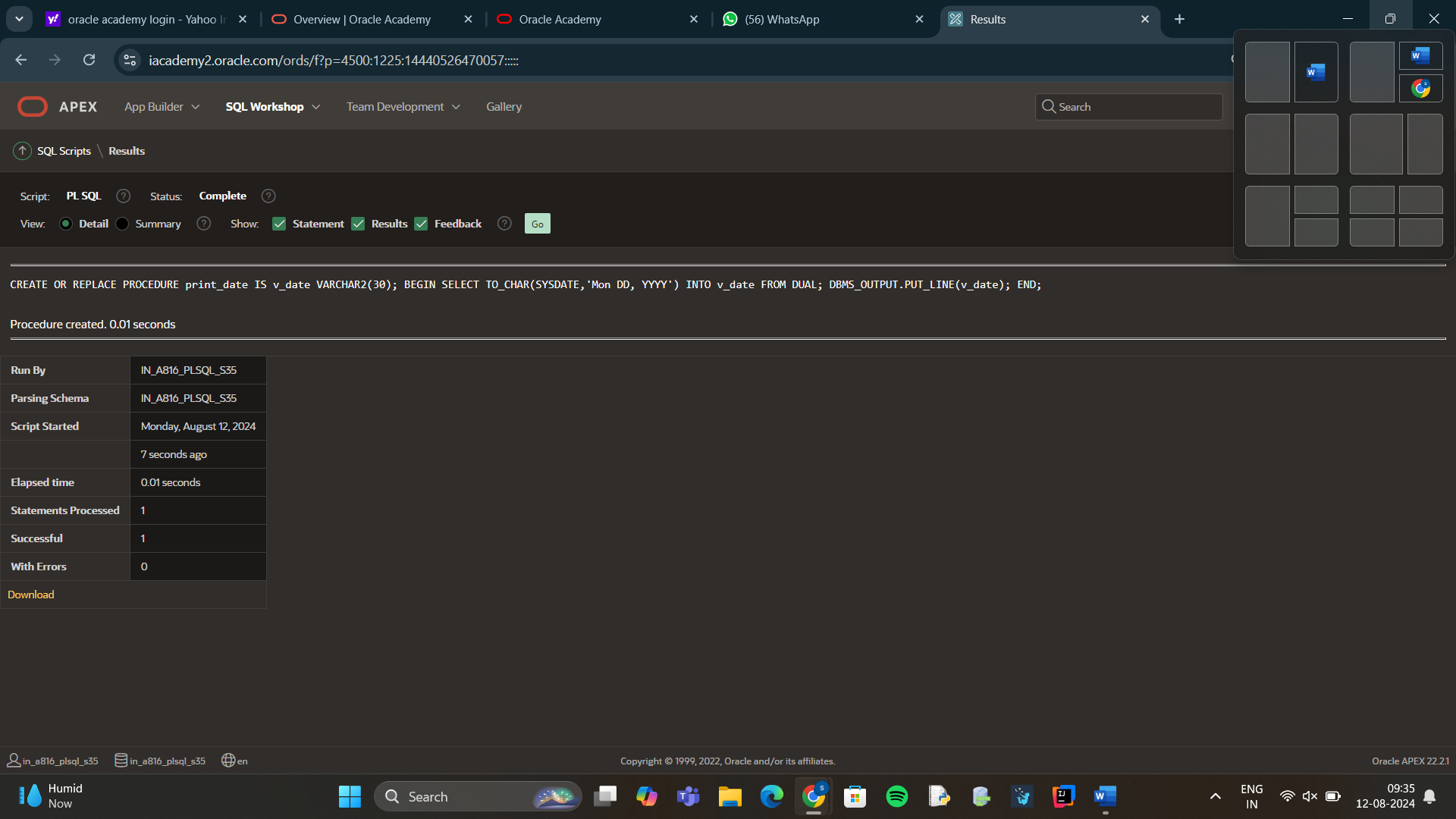
SELECT TO\_CHAR(SYSDATE,'Mon DD, YYYY')

INTO v\_date

FROM DUAL;

DBMS\_OUTPUT.PUT\_LINE(v\_date);

END;



DECLARE

a1 integer := 10;

b1 integer := 20;

c1 integer;

f1 real;

BEGIN

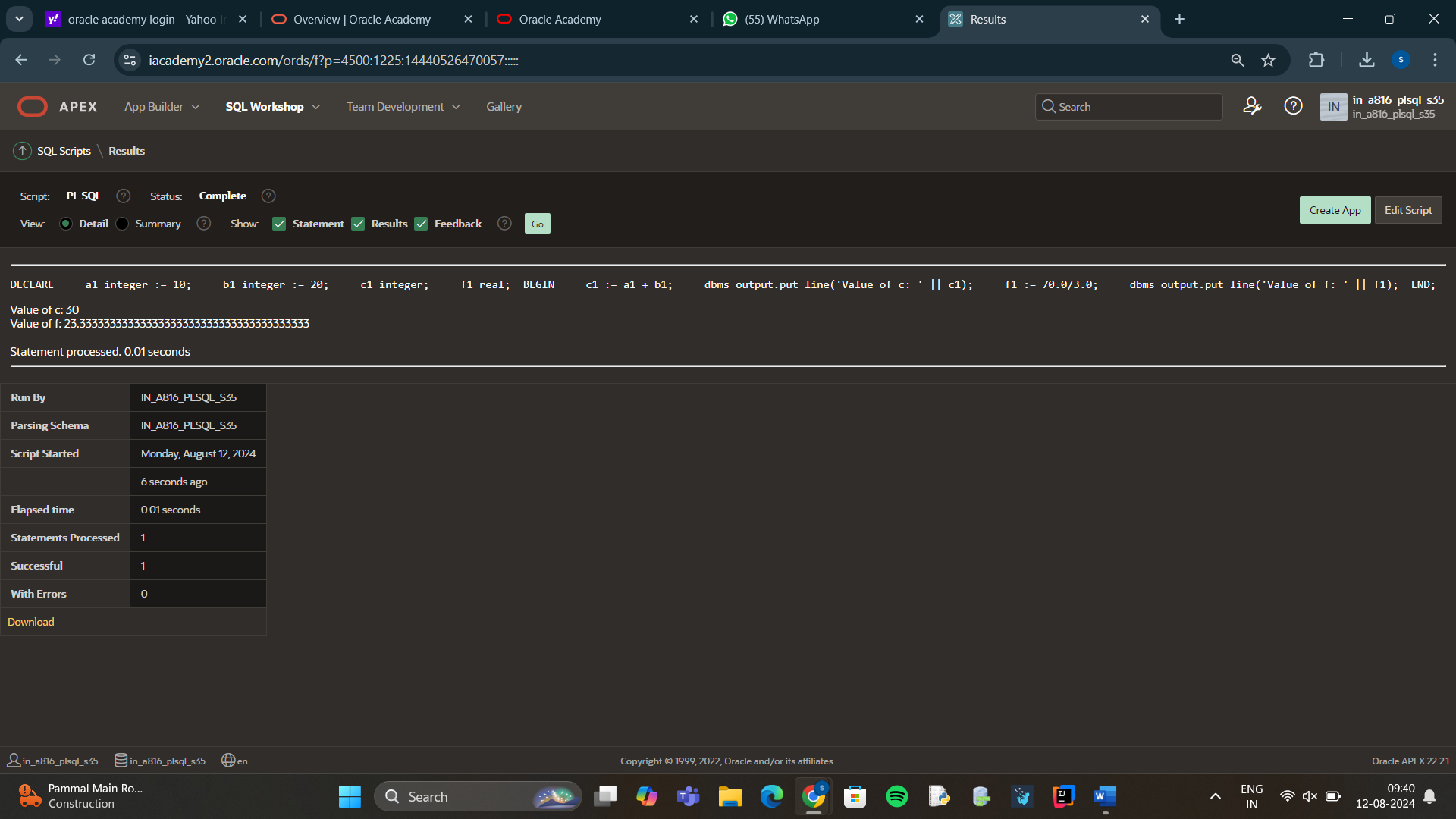
c1 := a1 + b1;

dbms\_output.put\_line('Value of c: ' || c1);

f1 := 70.0/3.0;

dbms\_output.put\_line('Value of f: ' || f1);

END;



DECLARE

-- constant declaration

pi constant number := 3.141592654;

-- other declarations

radius number(5,2);

dia number(5,2);

circumference number(7, 2);

area number (10, 2);

BEGIN

-- processing

radius := 9.5;

dia := radius \* 2;

circumference := 2.0 \* pi \* radius;

area := pi \* radius \* radius;

-- output

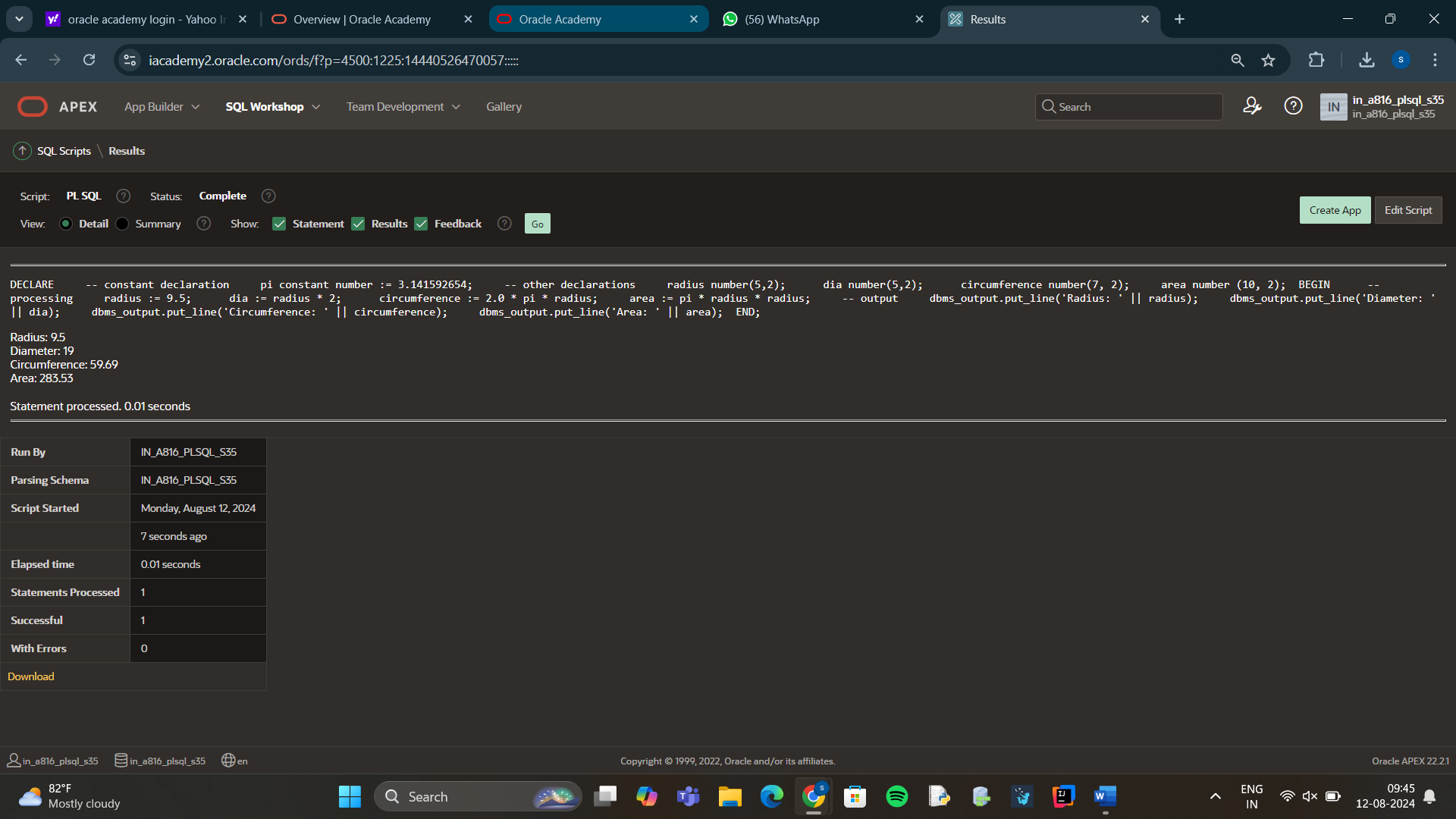
dbms\_output.put\_line('Radius: ' || radius);

dbms\_output.put\_line('Diameter: ' || dia);

dbms\_output.put\_line('Circumference: ' || circumference);

dbms\_output.put\_line('Area: ' || area);

END;



DECLARE

A2 integer := 10;

B2 integer := 20;

BEGIN

IF a2 > b2 THEN

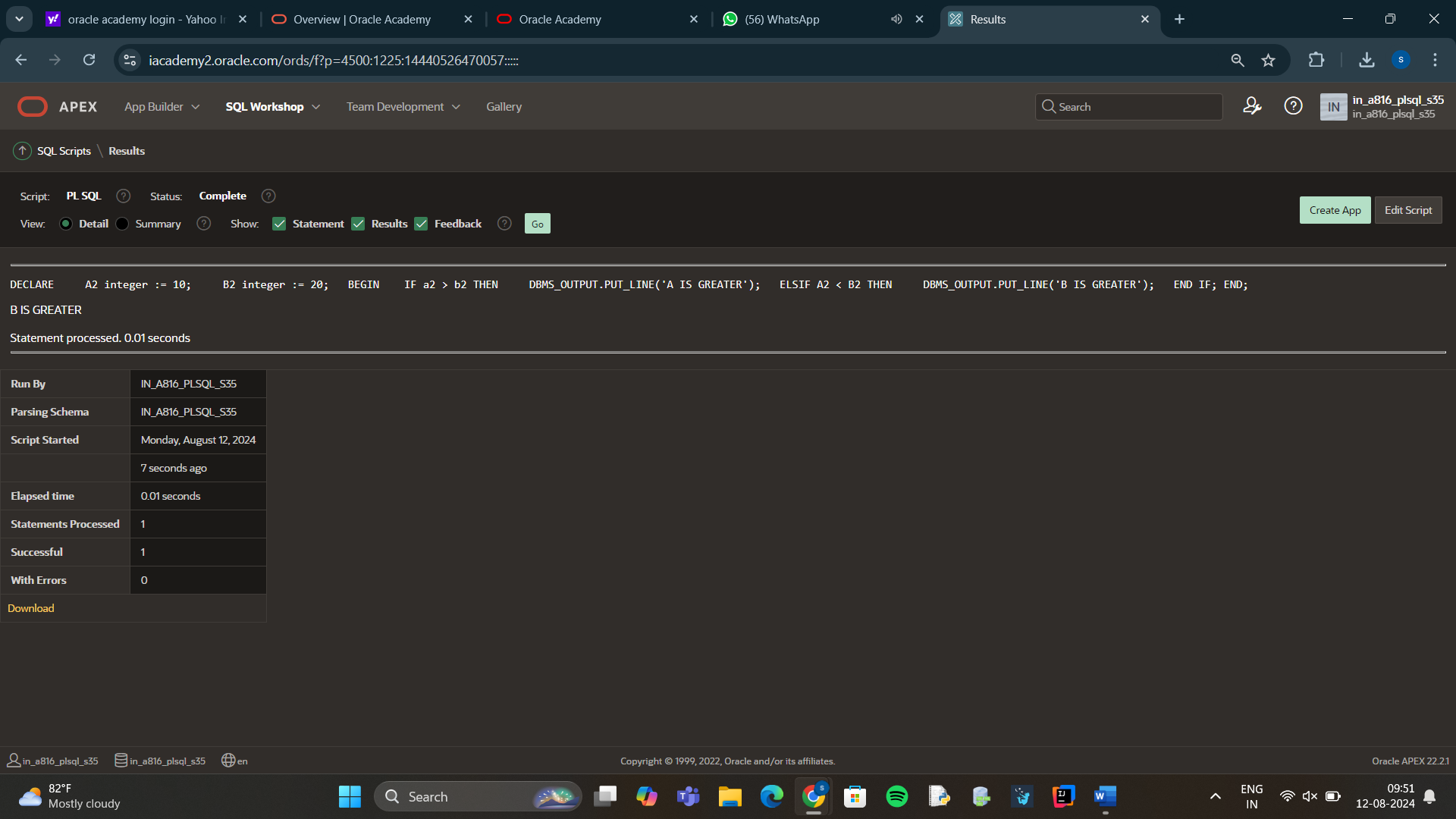
DBMS\_OUTPUT.PUT\_LINE('A IS GREATER');

ELSIF A2 < B2 THEN

DBMS\_OUTPUT.PUT\_LINE('B IS GREATER');

END IF;

END;



DECLARE

str VARCHAR2(40) := 'Tutorials Point';

nchars NUMBER(4) := 0;

nwords NUMBER(4) := 1;

s CHAR;

BEGIN

FOR i IN 1..Length(str) LOOP

s := Substr(str, i, 1);

nchars:= nchars+ 1;

IF s = ' ' THEN

nwords := nwords + 1;

END IF;

END LOOP;

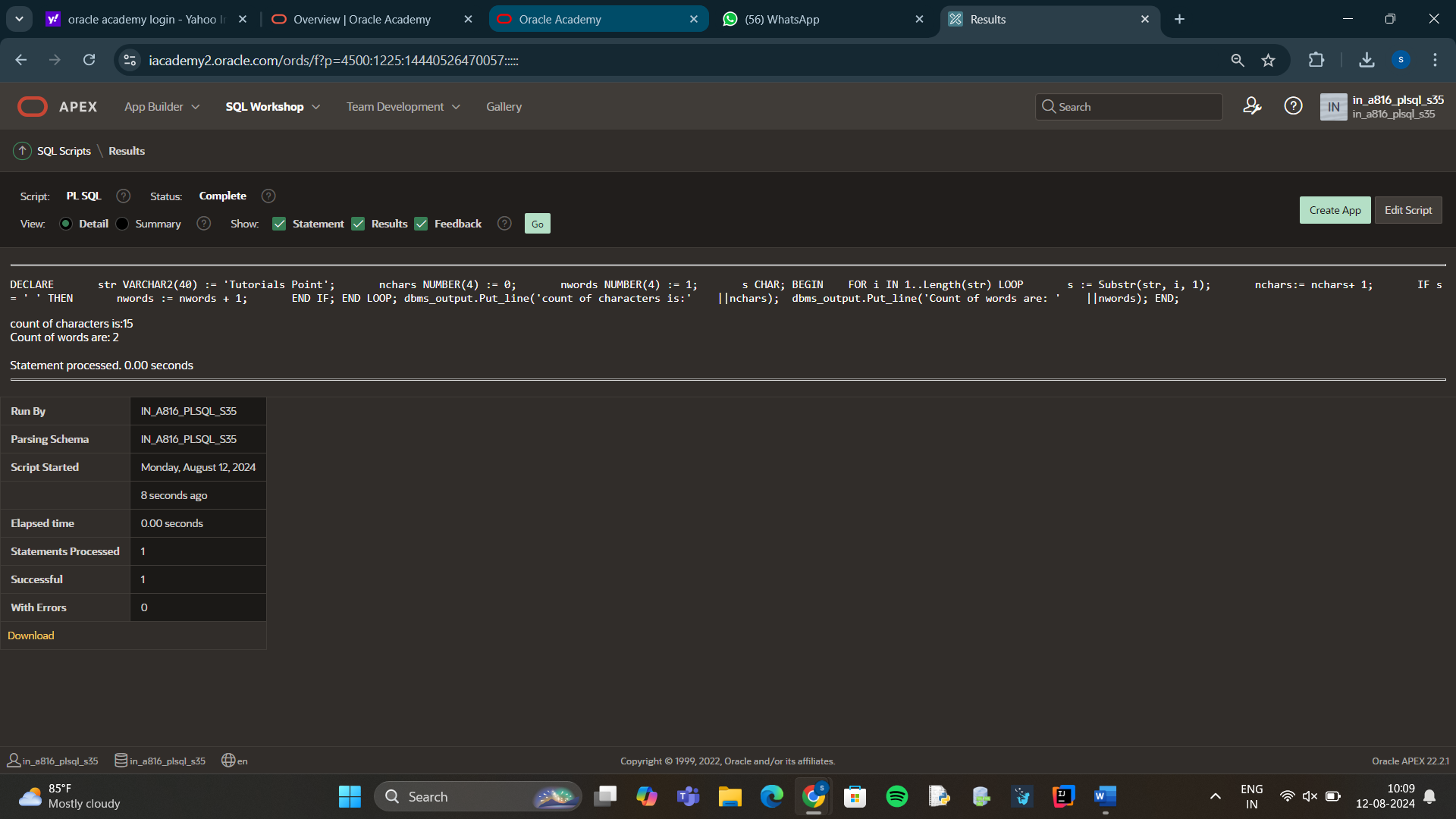
dbms\_output.Put\_line('count of characters is:'

||nchars);

dbms\_output.Put\_line('Count of words are: '

||nwords);

END;



DECLARE

x NUMBER;

n NUMBER;

i NUMBER;

FUNCTION Findmax(n IN NUMBER)

RETURN NUMBER

IS

sums NUMBER := 0;

BEGIN

FOR i IN 1..n

LOOP

sums := sums + i\*(i+1)/2;

END LOOP;

RETURN sums;

END;

BEGIN

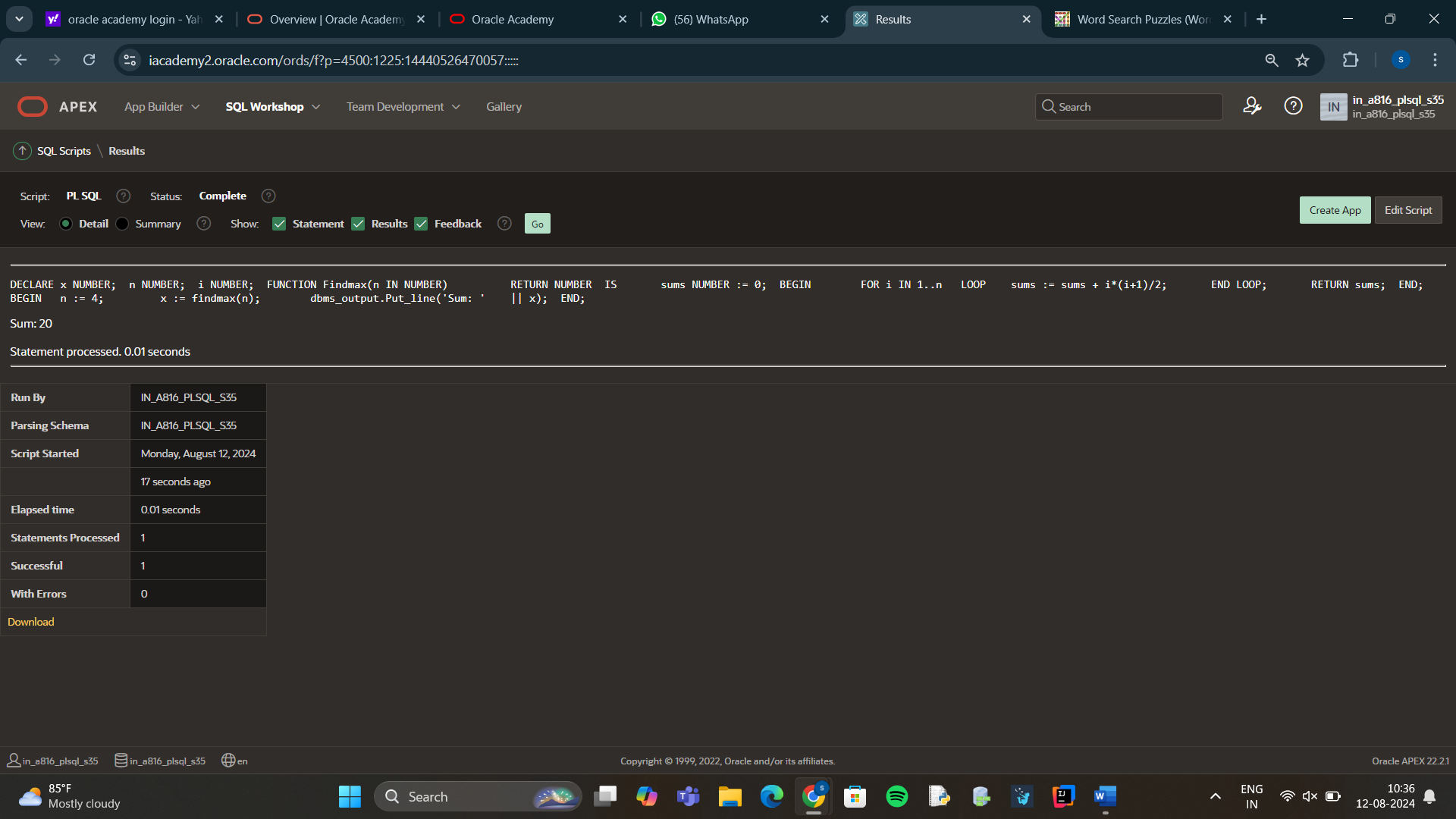
n := 4;

x := findmax(n);

dbms\_output.Put\_line('Sum: '

|| x);

END;



DECLARE

start\_num NUMBER := 1;

end\_num NUMBER :=10 ;

BEGIN

FOR i IN start\_num..end\_num LOOP

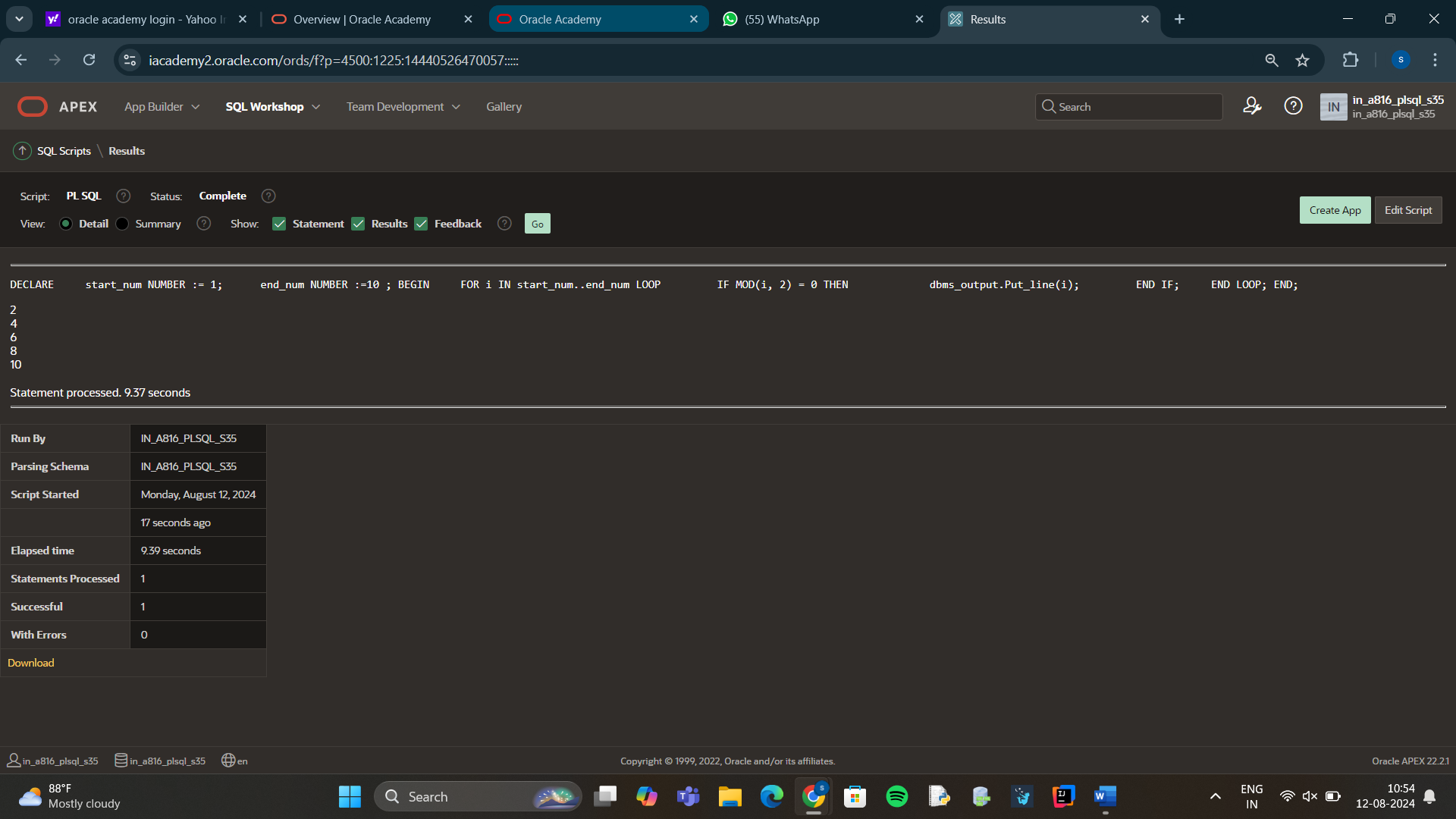
IF MOD(i, 2) = 0 THEN

dbms\_output.Put\_line(i);

END IF;

END LOOP;

END;



DECLARE

type namesarray IS VARRAY(5) OF VARCHAR2(10);

type grades IS VARRAY(5) OF INTEGER;

names namesarray;

marks grades;

total integer;

BEGIN

names := namesarray('Kavita', 'Pritam', 'Ayan', 'Rishav', 'Aziz');

marks:= grades(98, 97, 78, 87, 92);

total := names.count;

dbms\_output.put\_line('Total '|| total || ' Students');

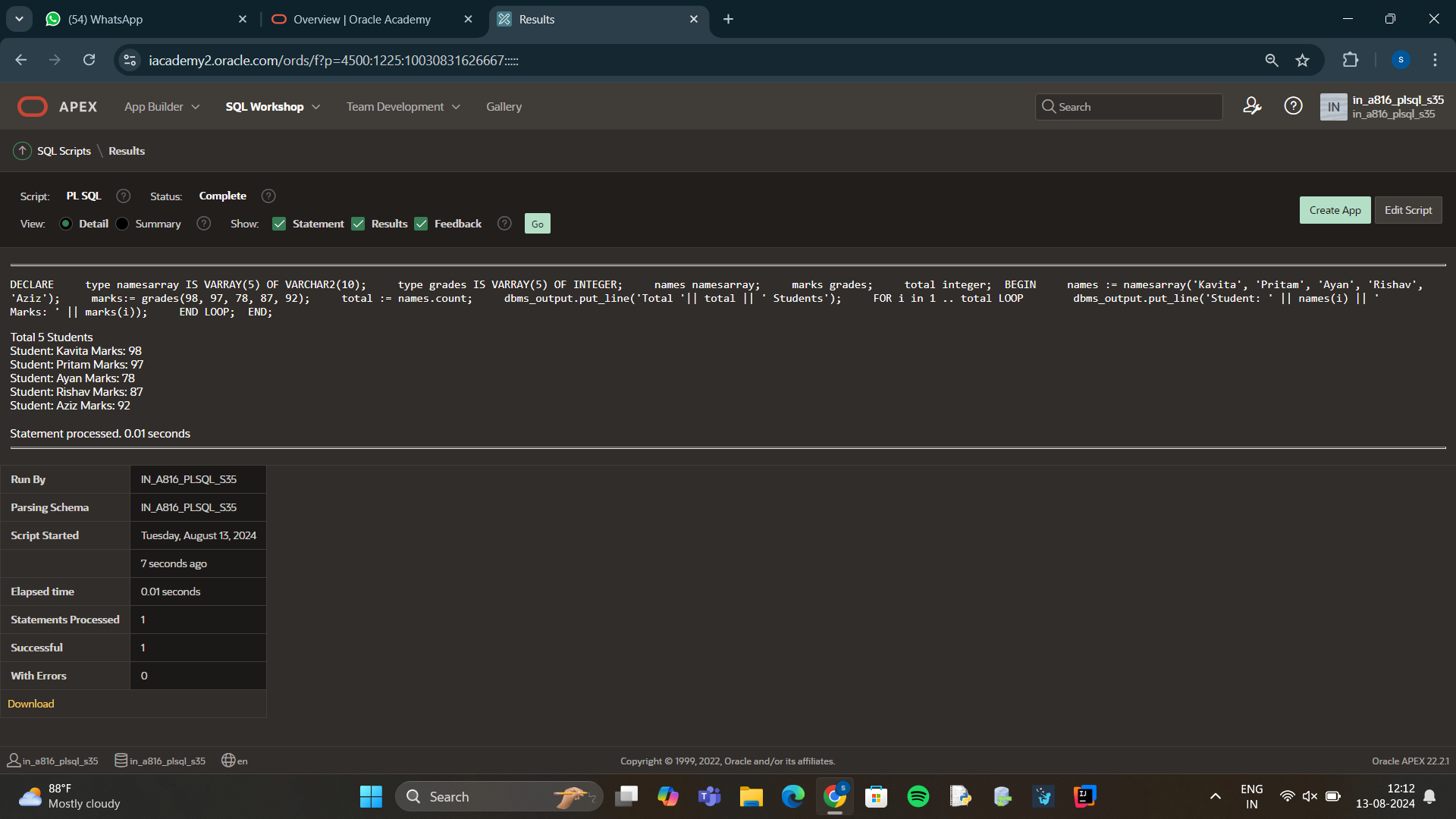
FOR i in 1 .. total LOOP

dbms\_output.put\_line('Student: ' || names(i) || '

Marks: ' || marks(i));

END LOOP;

END;



DECLARE

type namesarray IS VARRAY(5) OF VARCHAR2(10);

type grades IS VARRAY(5) OF INTEGER;

type grade\_labels IS VARRAY(5) OF VARCHAR2(2);

names namesarray;

marks grades;

total integer;

grade\_label varchar(2);

BEGIN

names := namesarray('Kavita', 'Pritam', 'Ayan', 'Rishav', 'Aziz');

marks:= grades(98, 97, 78, 87, 92);

total := names.count;

dbms\_output.put\_line('Total '|| total || ' Students');

FOR i in 1 .. total LOOP

IF marks(i) >= 90 THEN

grade\_label := 'A';

ELSIF marks(i) >= 80 THEN

grade\_label := 'B';

ELSIF marks(i) >= 70 THEN

grade\_label := 'C';

ELSIF marks(i) >= 60 THEN

grade\_label := 'D';

ELSE

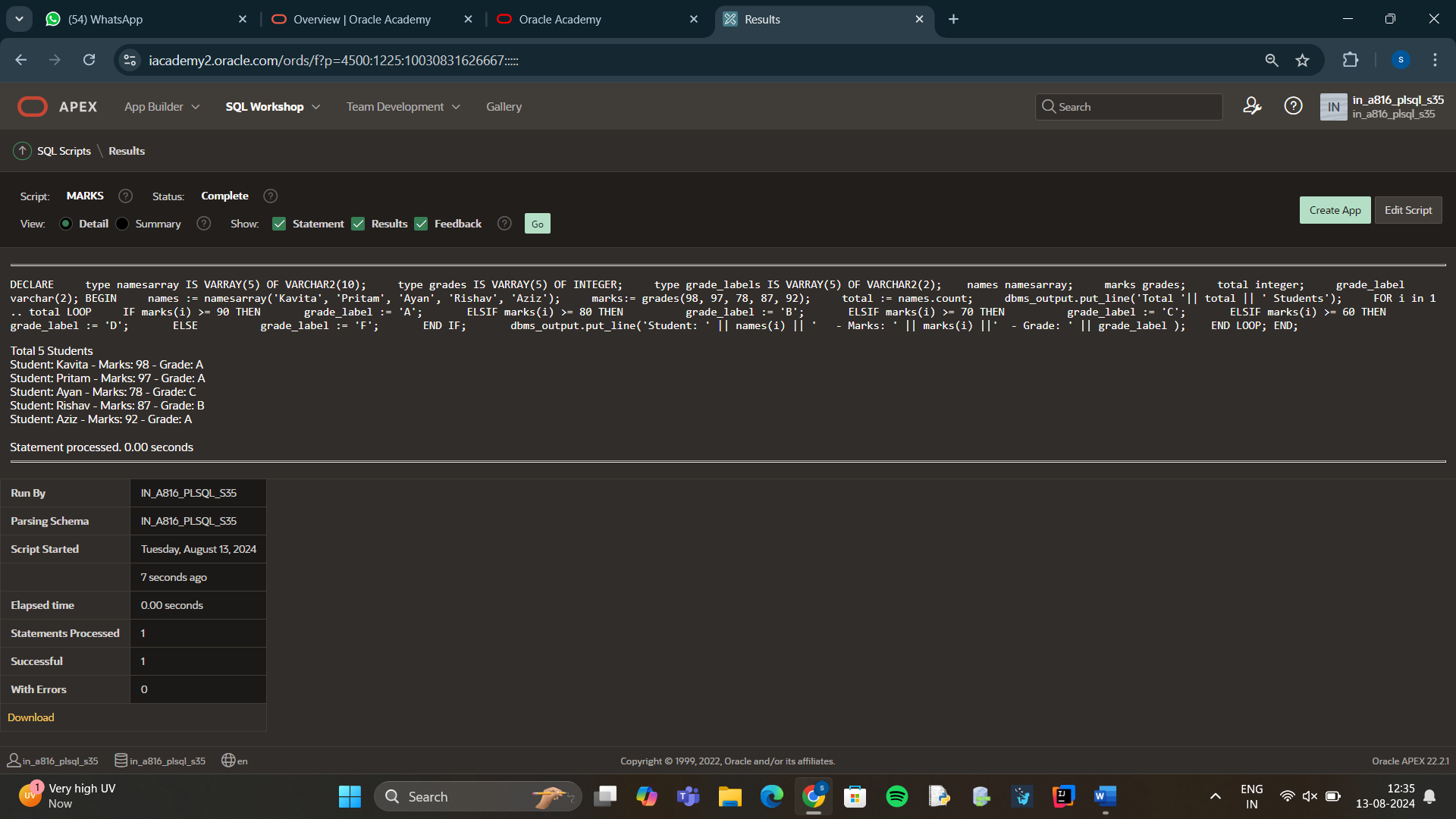
grade\_label := 'F';

END IF;

dbms\_output.put\_line('Student: ' || names(i) || ' - Marks: ' || marks(i) ||' - Grade: ' || grade\_label );

END LOOP;

END;



DECLARE

NUMBER\_TO\_CHECK NUMBER := 23146579;

ODD\_COUNT NUMBER := 0;

EVEN\_COUNT NUMBER := 0;

CURRENT\_DIGIT NUMBER;

BEGIN

WHILE NUMBER\_TO\_CHECK > 0 LOOP

CURRENT\_DIGIT := MOD(NUMBER\_TO\_CHECK, 10);

IF MOD(CURRENT\_DIGIT, 2) = 0 THEN

EVEN\_COUNT := EVEN\_COUNT + 1;

ELSE

ODD\_COUNT := ODD\_COUNT + 1;

END IF;

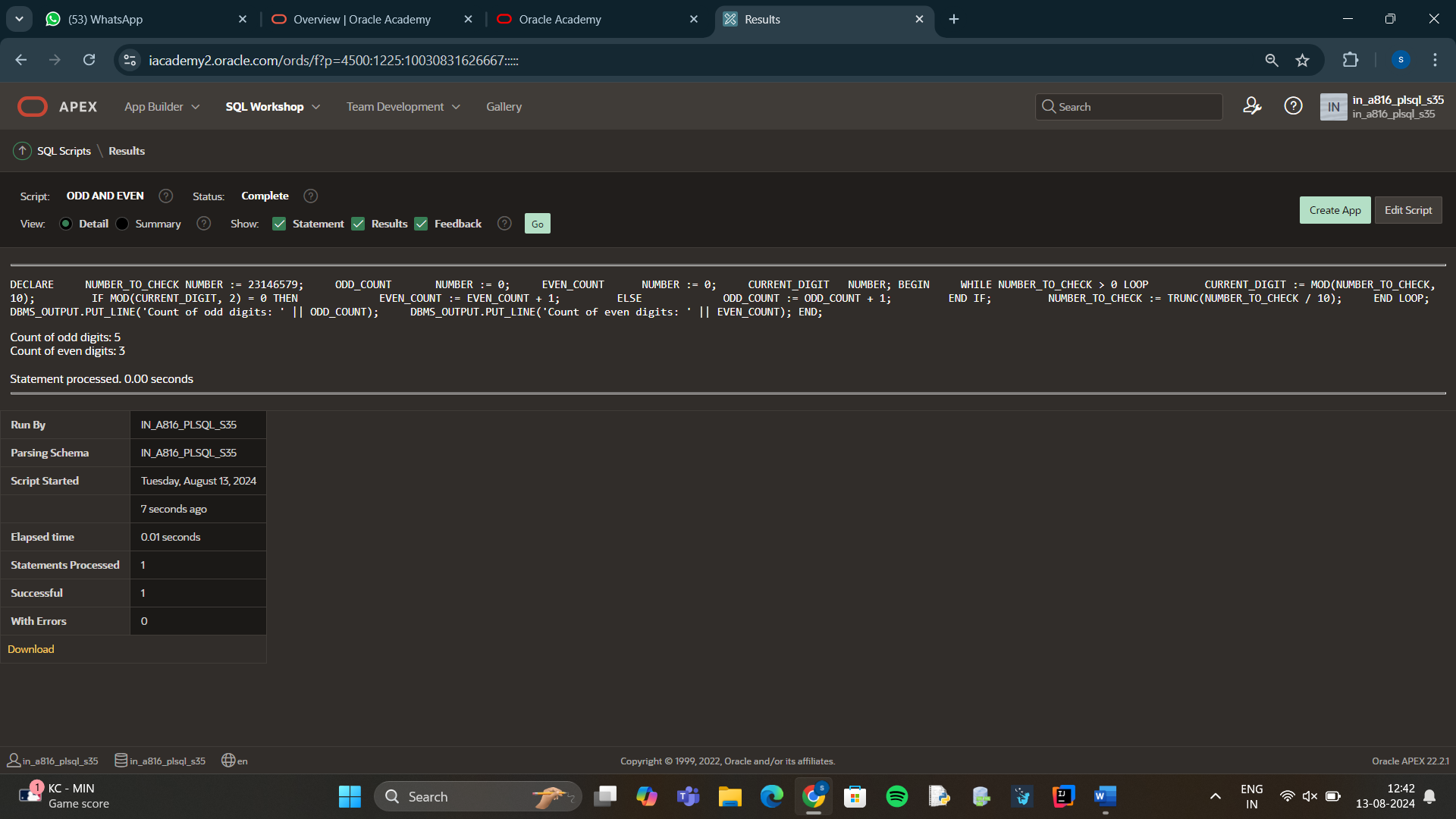
NUMBER\_TO\_CHECK := TRUNC(NUMBER\_TO\_CHECK / 10);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Count of odd digits: ' || ODD\_COUNT);

DBMS\_OUTPUT.PUT\_LINE('Count of even digits: ' || EVEN\_COUNT);

END;



DECLARE

v\_emp\_count NUMBER;

BEGIN

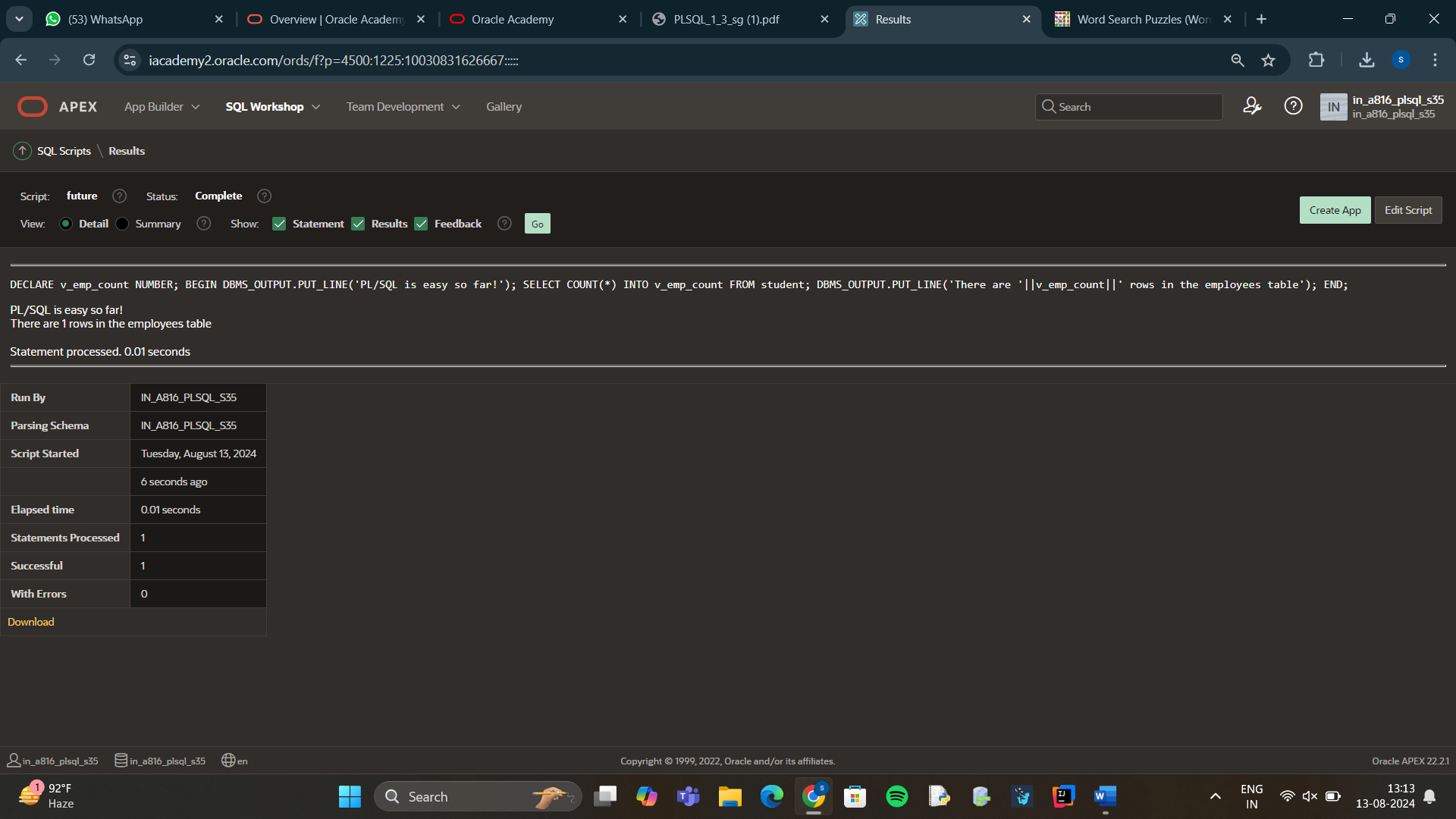
DBMS\_OUTPUT.PUT\_LINE('PL/SQL is easy so far!');

SELECT COUNT(\*) INTO v\_emp\_count FROM student;

DBMS\_OUTPUT.PUT\_LINE('There are '||v\_emp\_count||'

rows in the employees table');

END;



DECLARE

a number;

b number;

c number;

PROCEDURE findMin(x IN number, y IN number, z OUT number) IS

BEGIN

IF x < y THEN

z:= x;

ELSE

z:= y;

END IF;

END;

BEGIN

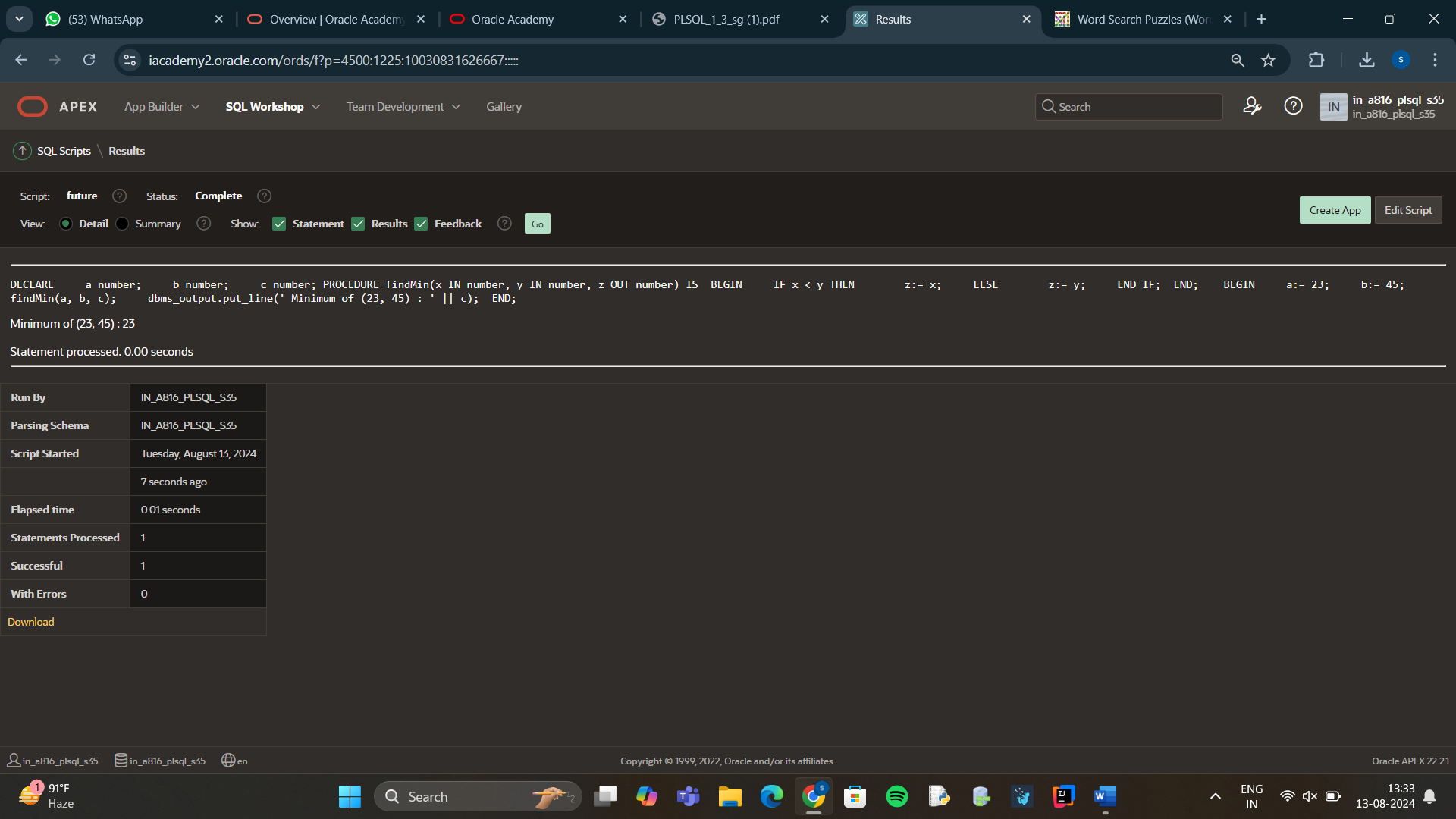
a:= 23;

b:= 45;

findMin(a, b, c);

dbms\_output.put\_line(' Minimum of (23, 45) : ' || c);

END;



DECLARE

num1 NUMBER := 10;

num2 NUMBER := 5;

ops VARCHAR2(10);

sum\_result NUMBER;

subtract\_result NUMBER;

PROCEDURE findCALCULATOR(num1 IN NUMBER, num2 IN NUMBER, ops IN VARCHAR2, sum\_result OUT NUMBER, subtract\_result OUT NUMBER) IS

BEGIN

IF ops = 'ADD' THEN

sum\_result := num1 + num2;

ELSIF ops = 'SUBTRACT' THEN

subtract\_result := num1 - num2;

END IF;

END findCALCULATOR;

BEGIN

ops := 'ADD';

findCALCULATOR(num1, num2, ops, sum\_result, subtract\_result);

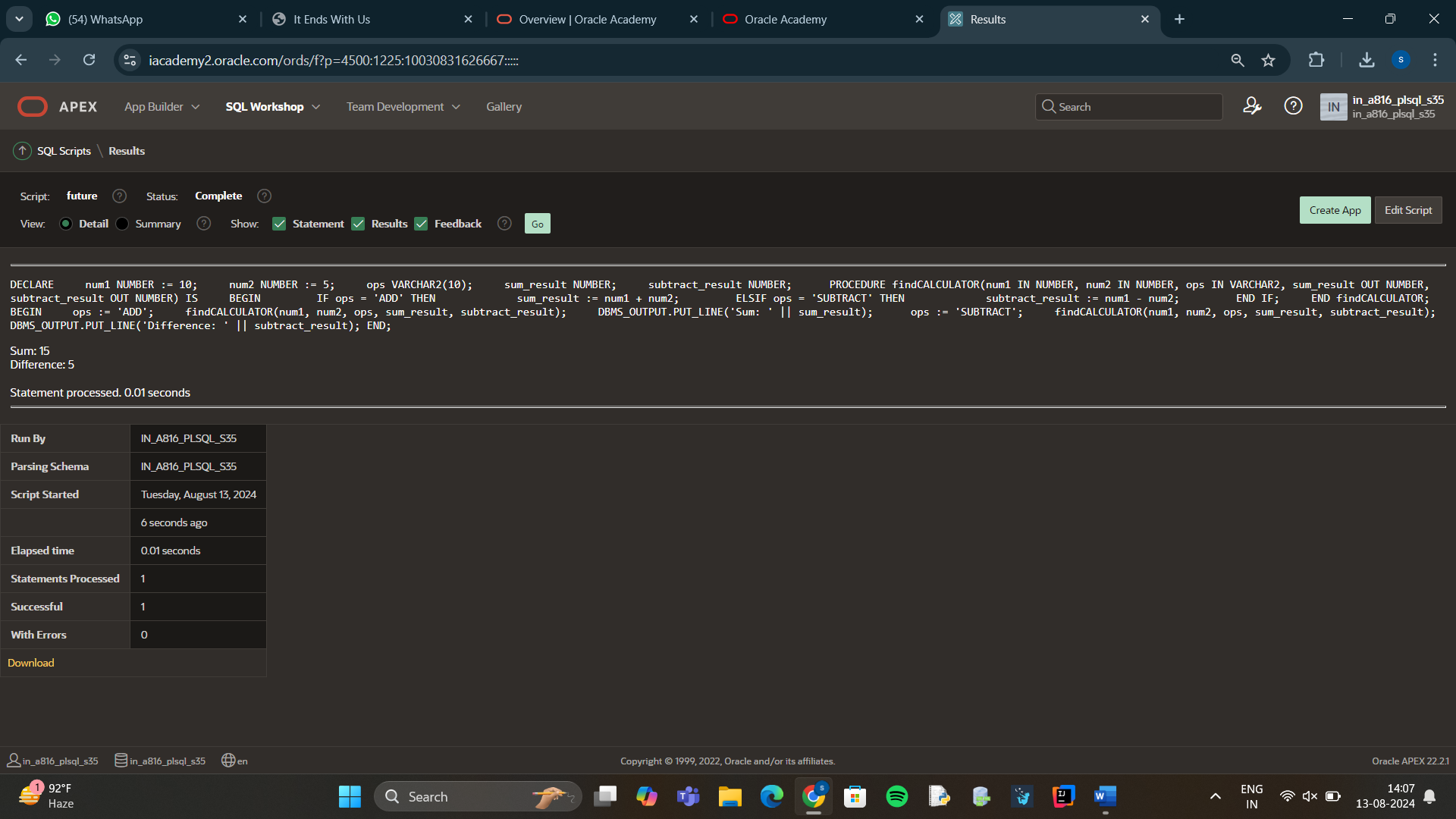
DBMS\_OUTPUT.PUT\_LINE('Sum: ' || sum\_result);

ops := 'SUBTRACT';

findCALCULATOR(num1, num2, ops, sum\_result, subtract\_result);

DBMS\_OUTPUT.PUT\_LINE('Difference: ' || subtract\_result);

END;



DECLARE

num number;

factorial number;

FUNCTION fact(x number)

RETURN number

IS

f number;

BEGIN

IF x=0 THEN

f := 1;

ELSE

f := x \* fact(x-1);

END IF;

RETURN f;

END;

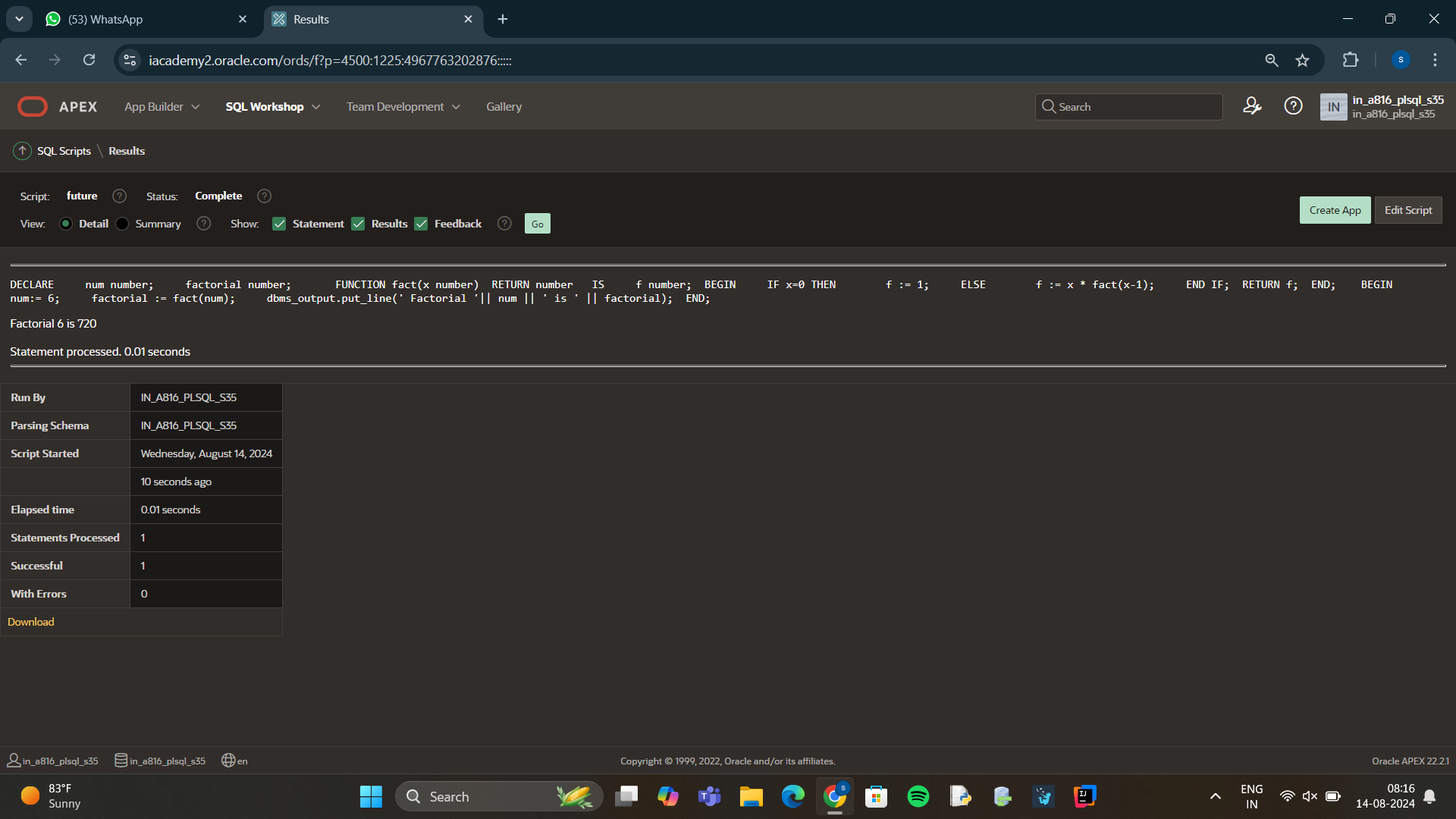
BEGIN

num:= 6;

factorial := fact(num);

dbms\_output.put\_line(' Factorial '|| num || ' is ' || factorial);

END;



CREATE OR REPLACE FUNCTION fibonacci(n IN NUMBER) RETURN NUMBER IS

result NUMBER;

BEGIN

IF n <= 0 THEN

result := 0;

ELSIF n = 1 THEN

result := 1;

ELSE

result := fibonacci(n - 1) + fibonacci(n - 2);

END IF;

RETURN result;

END;

/

DECLARE

num\_terms NUMBER := 10;

i NUMBER;

fib\_num NUMBER;

BEGIN

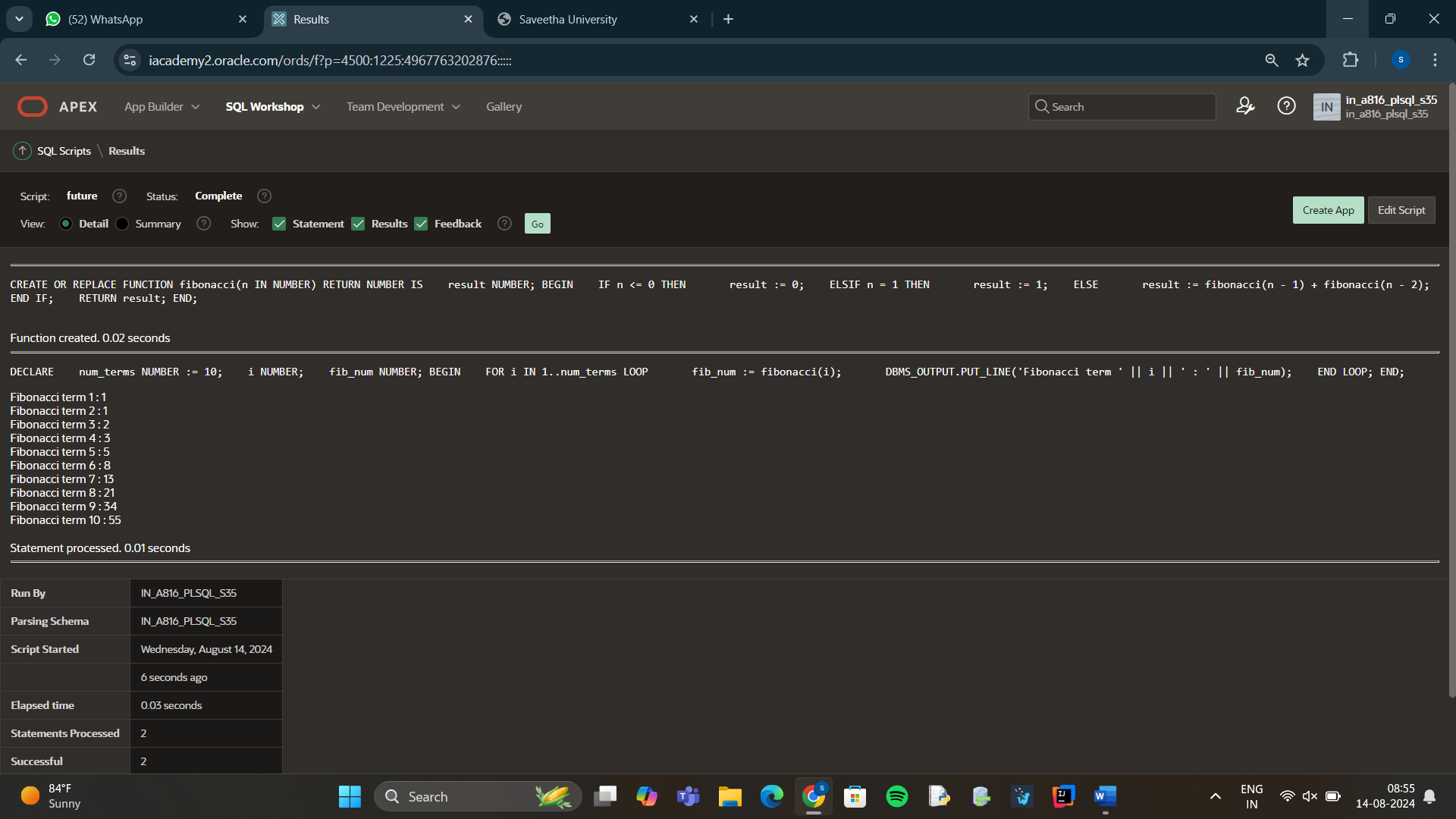
FOR i IN 1..num\_terms LOOP

fib\_num := fibonacci(i);

DBMS\_OUTPUT.PUT\_LINE('Fibonacci term ' || i || ' : ' || fib\_num);

END LOOP;

END;



DECLARE

total\_rows number(2);

BEGIN

UPDATE PERSON1

SET MARKS = MARKS +50;

IF sql%notfound THEN

dbms\_output.put\_line('no customers selected');

ELSIF sql%found THEN

total\_rows := sql%rowcount;

dbms\_output.put\_line( total\_rows || ' customers selected ');

END IF;

END;

**OUTPUT:**

6 customers selected

**MERGE:**

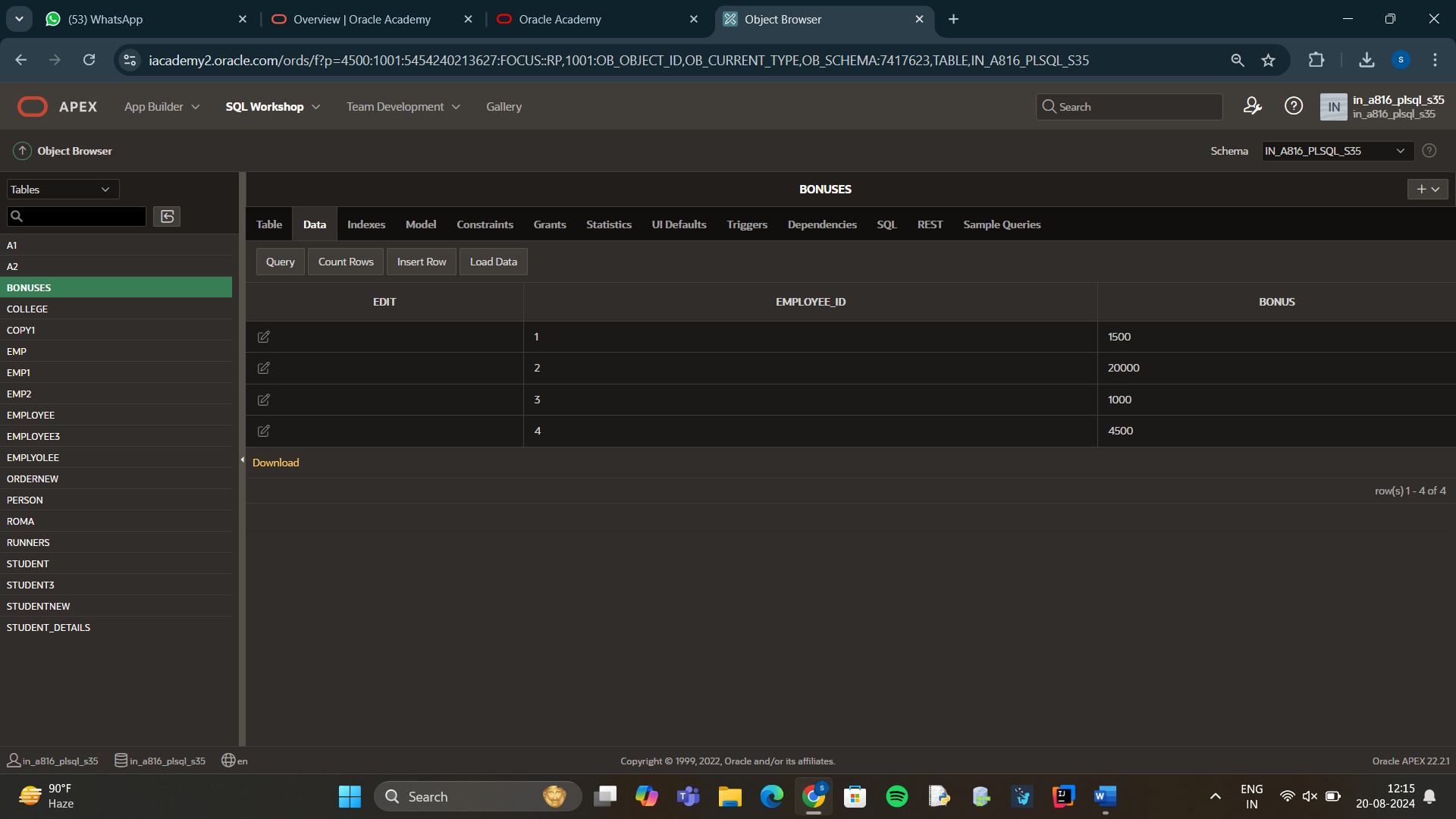
MERGE INTO BONUSES b

USING EMPLOYEES1 e

ON (b.EMPLOYEE\_ID = e.EMPLOYEE\_ID)

WHEN MATCHED THEN

UPDATE SET b.BONUS = e.SALARY \*.05;



DECLARE

V\_EMPLOYEE\_NAME EMPLOYEES1.EMPLOYEE\_NAME% TYPE;

BEGIN

SELECT EMPLOYEE\_NAME

INTO V\_EMPLOYEE\_NAME

FROM EMPLOYEES1

WHERE EMPLOYEE\_ID=1;

DBMS\_OUTPUT.PUT\_LINE('NAME:'||V\_EMPLOYEE\_NAME) ;

END

