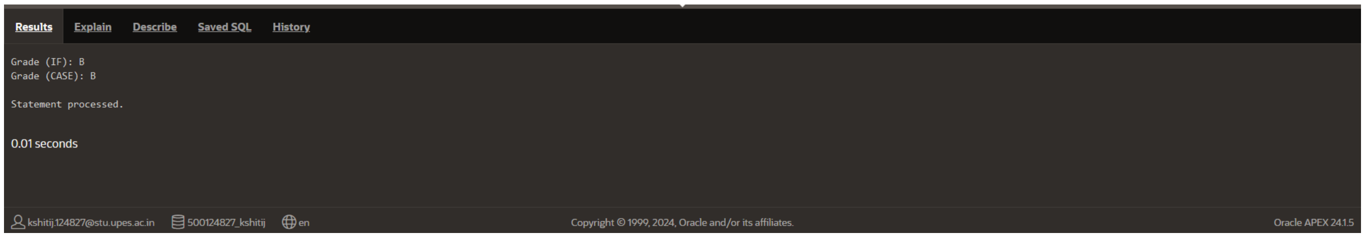
Name: Kshitij Chandrakar SAP: 500124827 Batch: B49 Exp: 11

Q1.

```
DECLARE
score NUMBER := 85; -- Input exam score
grade CHAR(1);
BEGIN
-- Using IF statements
IF score BETWEEN 90 AND 100 THEN
grade := 'A';
ELSIF score BETWEEN 80 AND 89 THEN
grade := 'B';
ELSIF score BETWEEN 70 AND 79 THEN
grade := 'C';
ELSE
grade := 'F';
END IF;
DBMS_OUTPUT.PUT_LINE('Grade (IF): ' || grade);
-- Using CASE statement
grade := CASE
WHEN score BETWEEN 90 AND 100 THEN 'A'
WHEN score BETWEEN 80 AND 89 THEN 'B'
WHEN score BETWEEN 70 AND 79 THEN 'C'
ELSE 'F'
END;
DBMS_OUTPUT.PUT_LINE('Grade (CASE): ' || grade);
END;
OUTPUT
```



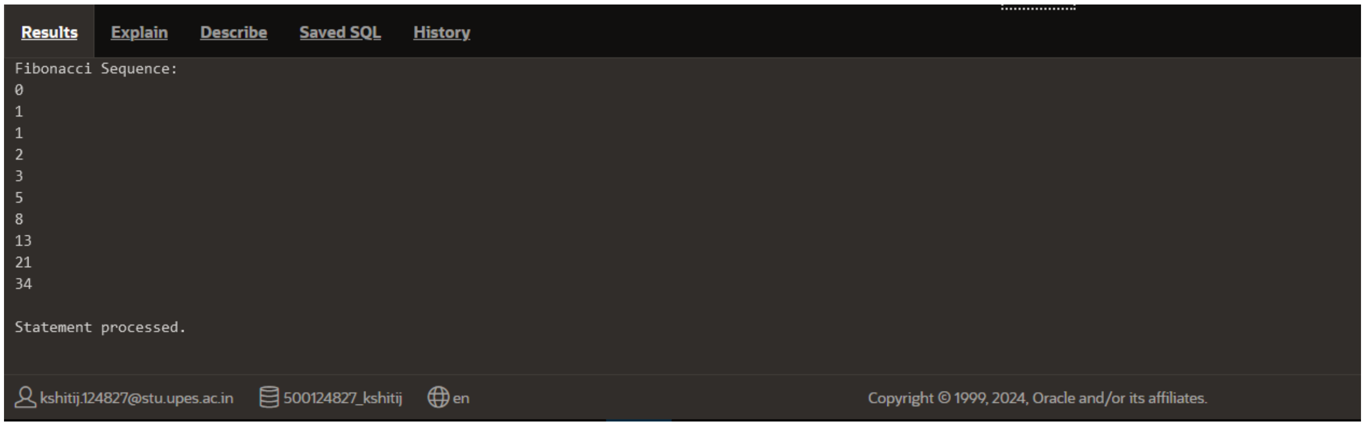
```
Grade (IF): B
Grade (CASE): B

Statement processed.

0.01 seconds
```

kshitij.124827@stu.upes.ac.in 500124827_kshitij en Copyright © 1999, 2024, Oracle and/or its affiliates. Oracle APEX 24.1.5

Q2.

```
DECLARE
a NUMBER := 0;
b NUMBER := 1;
temp NUMBER;
```

```
counter NUMBER := 1;
BEGIN
DBMS_OUTPUT.PUT_LINE('Fibonacci Sequence:');
WHILE counter <= 10 LOOP
DBMS_OUTPUT.PUT_LINE(a);
temp := a + b;
a := b;
b := temp;
counter := counter + 1;
END LOOP;
END;
OUTPUT
```



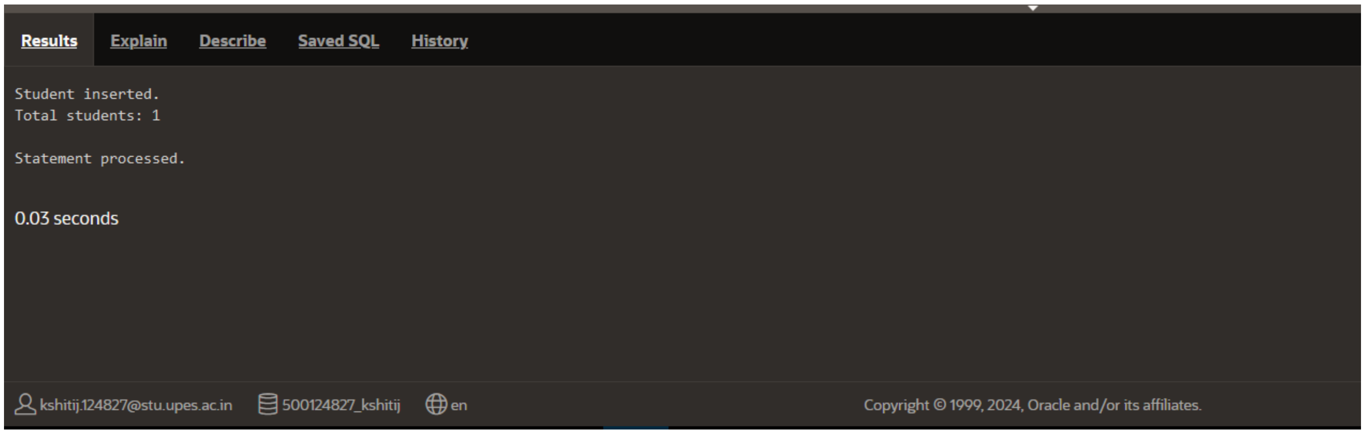
```
Fibonacci Sequence:
0
1
1
2
3
5
8
13
21
34

Statement processed.
```

kshitij.124827@stu.upes.ac.in 500124827_kshitij en Copyright © 1999, 2024, Oracle and/or its affiliates.

3.

```
-- Create the table
CREATE TABLE STUDENTS (
student_name VARCHAR2(50),
student_id NUMBER PRIMARY KEY,
score NUMBER
);
-- Procedure to insert new records
CREATE OR REPLACE PROCEDURE insert_student(
p_name VARCHAR2,
p_id NUMBER,
p_score NUMBER
) AS
BEGIN
INSERT INTO STUDENTS (student_name, student_id, score)
VALUES (p_name, p_id, p_score);
DBMS_OUTPUT.PUT_LINE('Student inserted.');
END;
-- Function to get the total number of students
CREATE OR REPLACE FUNCTION total_students RETURN NUMBER IS
```

```
student_count NUMBER;
BEGIN
SELECT COUNT(*) INTO student_count FROM STUDENTS;
RETURN student_count;
END;
-- Test the procedure and function
BEGIN
insert_student('Alice', 1, 85);
DBMS_OUTPUT.PUT_LINE('Total students: ' || total_students);
END;
OUTPUT
```



```
Student inserted.
Total students: 1

Statement processed.
```

0.03 seconds

kshitij.124827@stu.upes.ac.in 500124827_kshitij en

Copyright © 1999, 2024, Oracle and/or its affiliates.

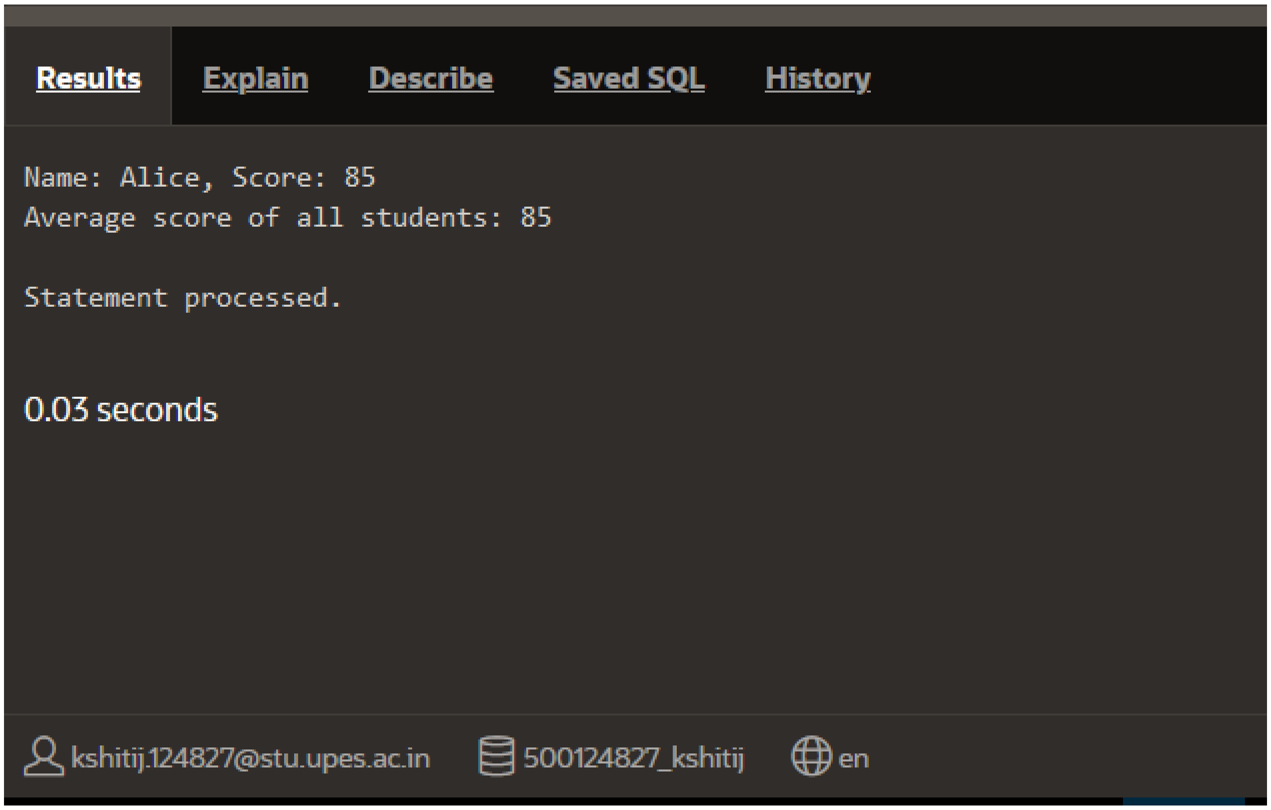
Q4.

```
CREATE OR REPLACE PACKAGE student_pkg AS
PROCEDURE display_student(p_id NUMBER);
FUNCTION student_count RETURN NUMBER;
END student_pkg;
CREATE OR REPLACE PACKAGE BODY student_pkg AS
PROCEDURE display_student(p_id NUMBER) IS
v_name STUDENTS.student_name%TYPE;
v_score STUDENTS.score%TYPE;
BEGIN
SELECT student_name, score INTO v_name, v_score
FROM STUDENTS WHERE student_id = p_id;
DBMS_OUTPUT.PUT_LINE('Name: ' || v_name || ', Score: ' || v_score);
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Student not found');
END;
FUNCTION student_count RETURN NUMBER IS
v_count NUMBER;
BEGIN
SELECT COUNT(*) INTO v_count FROM STUDENTS;
```

```
RETURN v_count;
END;
END student_pkg;
BEGIN
student_pkg.display_student(1);
DBMS_OUTPUT.PUT_LINE('Total students: ' || student_pkg.student_count);
END;
```

Q5.

```
DECLARE
CURSOR high_score_cur IS
SELECT student_name, score FROM STUDENTS WHERE score > 75;
v_name STUDENTS.student_name%TYPE;
v_score STUDENTS.score%TYPE;
CURSOR avg_score_cur IS
SELECT AVG(score) AS avg_score FROM STUDENTS;
v_avg_score NUMBER;
BEGIN
-- Display students with scores > 75
OPEN high_score_cur;
LOOP
FETCH high_score_cur INTO v_name, v_score;
EXIT WHEN high_score_cur%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('Name: ' || v_name || ', Score: ' || v_score);
END LOOP;
CLOSE high_score_cur;
-- Calculate average score
OPEN avg_score_cur;
FETCH avg_score_cur INTO v_avg_score;
DBMS_OUTPUT.PUT_LINE('Average score of all students: ' || v_avg_score);
CLOSE avg_score_cur;
END;
```



```
Name: Alice, Score: 85
Average score of all students: 85

Statement processed.
```

0.03 seconds

kshitij.124827@stu.upes.ac.in 500124827_kshitij en

Q6

```
CREATE TABLE STUDENTS (
student_name VARCHAR2(50),
student_id NUMBER PRIMARY KEY,
sem NUMBER,
dept VARCHAR2(50),
enrollment_date DATE
);
CREATE TABLE STUDENT_LOG (
log_id NUMBER GENERATED ALWAYS AS IDENTITY,
student_id NUMBER,
updated_date DATE
);
CREATE OR REPLACE TRIGGER set_enrollment_date
BEFORE INSERT ON STUDENTS
FOR EACH ROW
BEGIN
:NEW.enrollment_date := SYSDATE;
END;
CREATE OR REPLACE TRIGGER log_student_update
AFTER UPDATE ON STUDENTS
FOR EACH ROW
BEGIN
INSERT INTO STUDENT_LOG (student_id, updated_date)
```

```sql
VALUES (:OLD.student_id, SYSDATE);
END;
```