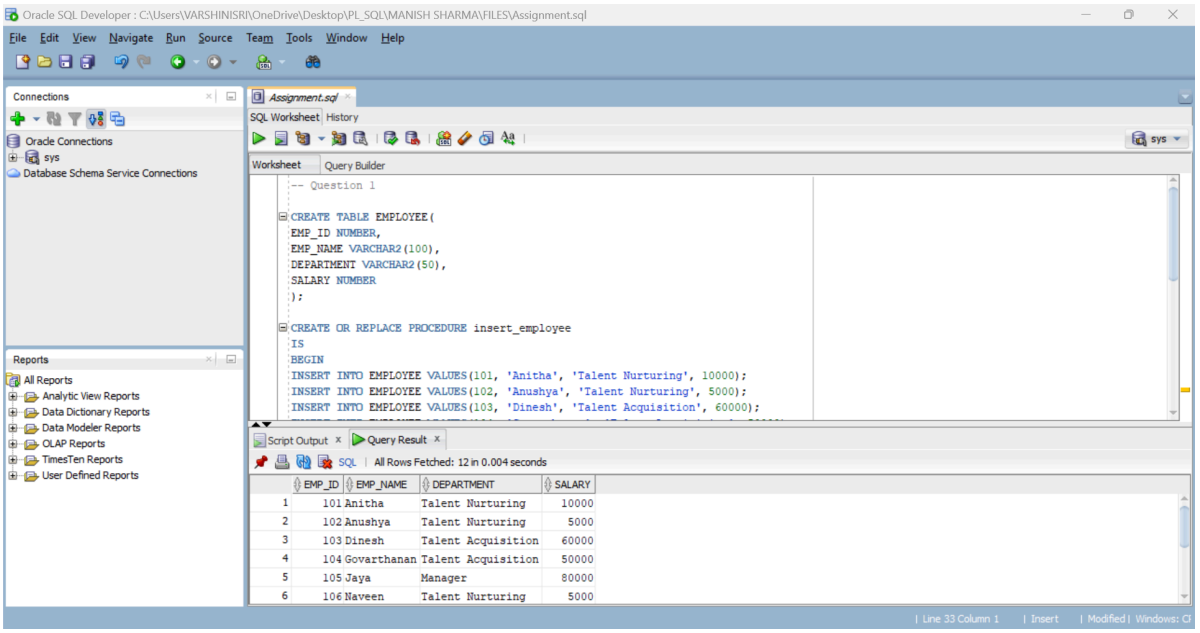
PL/SQL Assignment Questions:

Question 1: Create a Procedure to Insert Employee Data

Write a PL/SQL procedure named insert_employee to insert employee data into the EMPLOYEES table:

- Table structure: EMPLOYEES (EMP_ID NUMBER, EMP_NAME VARCHAR2(100), DEPARTMENT VARCHAR2(50), SALARY NUMBER)

```
CREATE TABLE EMPLOYEE(
EMP_ID NUMBER,
EMP_NAME VARCHAR2(100),
DEPARTMENT VARCHAR2(50),
SALARY NUMBER
);

CREATE OR REPLACE PROCEDURE insert_employee
IS
BEGIN
INSERT INTO EMPLOYEE VALUES(101, 'Anitha', 'Talent Nurturing', 10000);
INSERT INTO EMPLOYEE VALUES(102, 'Anushya', 'Talent Nurturing', 5000);
INSERT INTO EMPLOYEE VALUES(103, 'Dinesh', 'Talent Acquisition', 60000);
INSERT INTO EMPLOYEE VALUES(104, 'Govarthanan', 'Talent Acquisition', 50000);
INSERT INTO EMPLOYEE VALUES(105, 'Jaya', 'Manager', 80000);
INSERT INTO EMPLOYEE VALUES(106, 'Naveen', 'Talent Nurturing', 5000);
INSERT INTO EMPLOYEE VALUES(107, 'Sabapathi', 'Talent Nurturing', 5000);
INSERT INTO EMPLOYEE VALUES(108, 'Saraswathi', 'Talent Nurturing', 40000);
INSERT INTO EMPLOYEE VALUES(109, 'Savitha', 'Talent Nurturing', 10000);
INSERT INTO EMPLOYEE VALUES(110, 'Silpa', 'Talent Nurturing', 10000);
INSERT INTO EMPLOYEE VALUES(111, 'Snigdha', 'Talent Nurturing', 5000);
INSERT INTO EMPLOYEE VALUES(112, 'Subbu', 'Manager', 80000);
COMMIT;
END;

BEGIN
 insert_employee;
END;

SELECT * FROM EMPLOYEE;
```

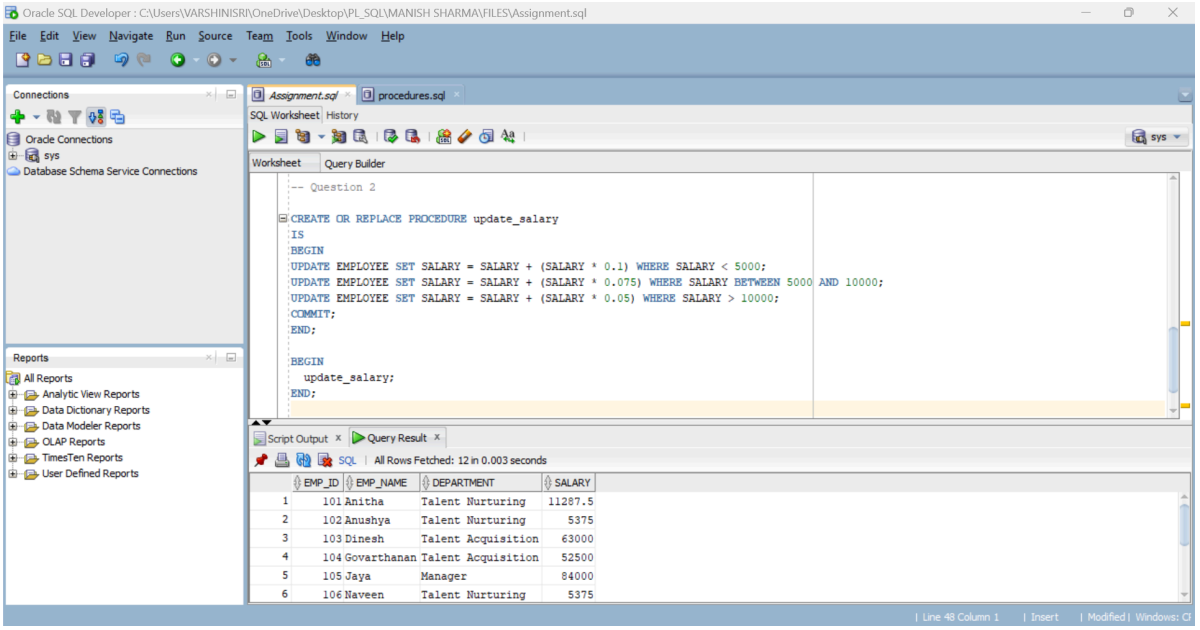


Question 2: Create a Procedure to Update Employee Salary

Write a PL/SQL procedure named update_salary to update an employee's salary based on their current salary:

- If the current salary is less than 5000, increase it by 10%.
- If the current salary is between 5000 and 10000, increase it by 7.5%.
- If the current salary is more than 10000, increase it by 5%.

```
CREATE OR REPLACE PROCEDURE update_salary
IS
BEGIN
UPDATE EMPLOYEE SET SALARY = SALARY + (SALARY * 0.1) WHERE SALARY < 5000;
UPDATE EMPLOYEE SET SALARY = SALARY + (SALARY * 0.075) WHERE SALARY BETWEEN
5000 AND 10000;
UPDATE EMPLOYEE SET SALARY = SALARY + (SALARY * 0.05) WHERE SALARY > 10000;
COMMIT;
END;

BEGIN
  update_salary;
END;

SELECT * FROM EMPLOYEE;
```

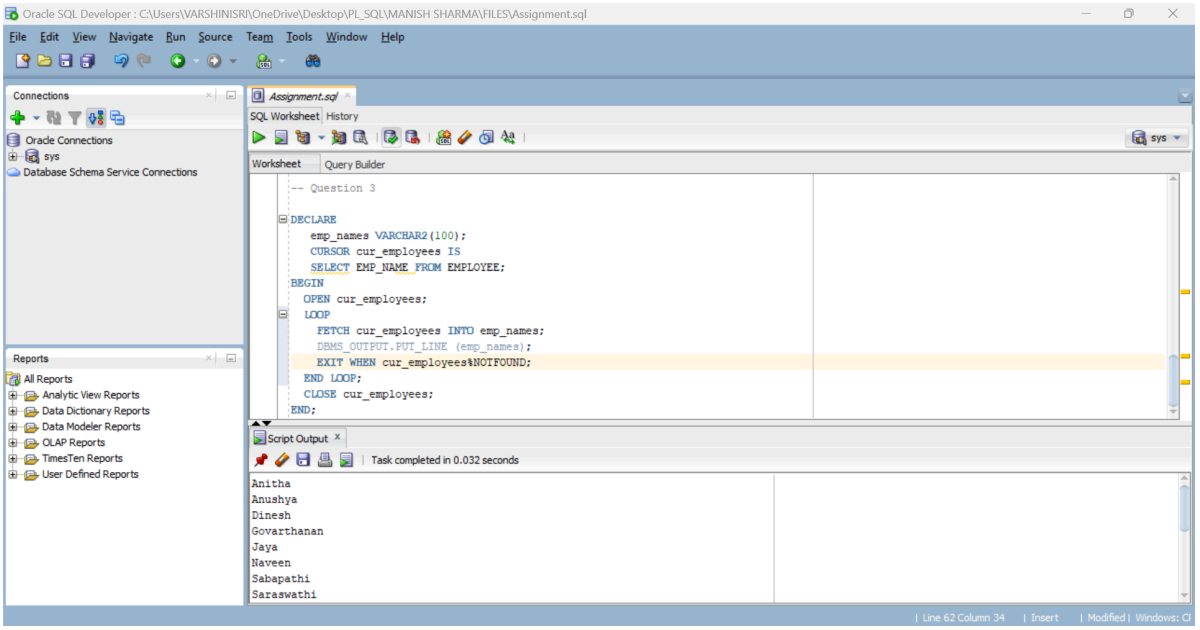


Cursors

Question 3: Use a Cursor to Display Employee Names

Write a PL/SQL block using a cursor to fetch and display all employee names from the EMPLOYEES table.

```
DECLARE
  emp_names VARCHAR2(100);
  CURSOR cur_employees IS
  SELECT EMP_NAME FROM EMPLOYEE;
BEGIN
 OPEN cur_employees;
 LOOP
   FETCH cur_employees INTO emp_names;
   DBMS_OUTPUT.PUT_LINE (emp_names);
   EXIT WHEN cur_employees%NOTFOUND;
 END LOOP;
 CLOSE cur_employees;
END;
```



Views

Question 4: Create a View for Employees with High Salary

Write a SQL statement to create a view named high_salary_employees that displays employees earning more than 10000.

```
CREATE OR REPLACE VIEW high_salary_employees AS
SELECT *
FROM EMPLOYEE
WHERE SALARY > 10000;

SELECT * FROM high_salary_employees;
```

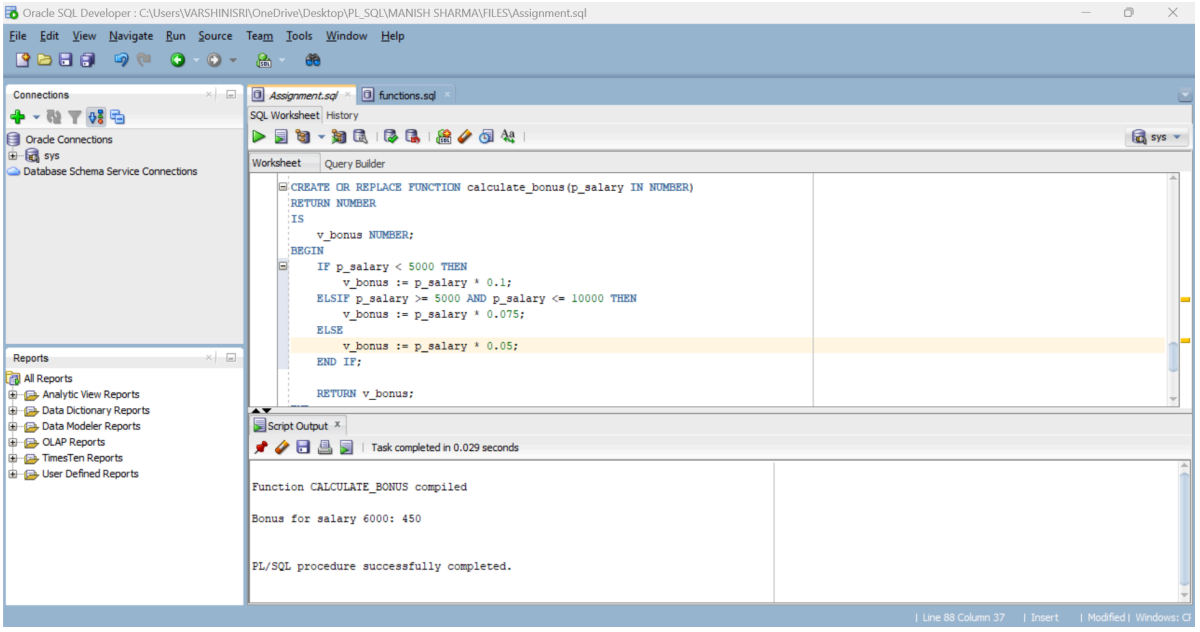
Functions

Question 5: Create a Function to Calculate Bonus

Write a PL/SQL function named calculate_bonus to calculate the bonus based on an employee's salary:

- Employees earning less than 5000 get a bonus of 10% of their salary.
- Employees earning between 5000 and 10000 get a bonus of 7.5% of their salary.
- Employees earning more than 10000 get a bonus of 5% of their salary.

```
CREATE OR REPLACE FUNCTION calculate_bonus(p_salary IN NUMBER)
RETURN NUMBER
IS
   v_bonus NUMBER;
BEGIN
   IF p_salary < 5000 THEN
      v_bonus := p_salary * 0.1;
   ELSIF p_salary >= 5000 AND p_salary <= 10000 THEN
      v_bonus := p_salary * 0.075;
   ELSE
      v_bonus := p_salary * 0.05;
   END IF;

   RETURN v_bonus;
END;

DECLARE
   v_salary NUMBER := 6000;
   v_bonus NUMBER;
BEGIN
   v_bonus := calculate_bonus(v_salary);
   DBMS_OUTPUT.PUT_LINE('Bonus for salary ' || v_salary || ': ' || v_bonus);
END;
```



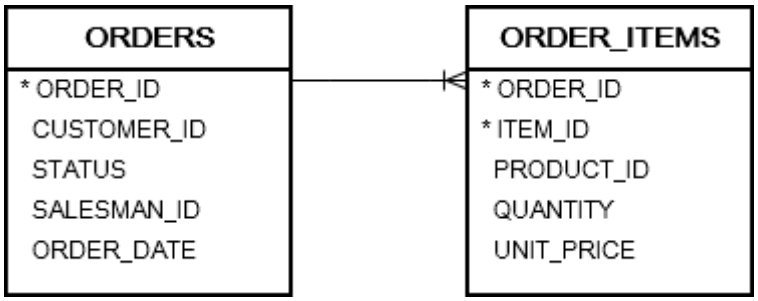
Triggers

Question 6: Create a Trigger to Log Employee Insertions

Write a PL/SQL trigger named log_employee_insert to log whenever an employee is inserted into the EMPLOYEES table.

```
CREATE TABLE EMPLOYEE_LOG(
 event_type VARCHAR2(20),
 LOG_DATE DATE,
 LOG_MESSAGE VARCHAR2(100)
);

CREATE OR REPLACE TRIGGER log_employee_insert
AFTER INSERT ON EMPLOYEE
FOR EACH ROW
DECLARE
   v_log_message VARCHAR2(100);
BEGIN
   v_log_message := 'Employee inserted - EMP_ID: ' || :NEW.EMP_ID || ', EMP_NAME: ' ||
:NEW.EMP_NAME || ', DEPARTMENT: ' || :NEW.DEPARTMENT || ', SALARY: ' || :NEW.SALARY;

   INSERT INTO EMPLOYEE_LOG (event_type, LOG_DATE, LOG_MESSAGE)
   VALUES (ora_sysevent, SYSDATE, v_log_message);

   DBMS_OUTPUT.PUT_LINE('Log message inserted: ' || v_log_message);
END;
```

Question 7: Consider the orders and order_items tables from the sample database.



A) Create a view that returns the sales revenues by customers. The values of the credit column are 5% of the total sales revenues.

B) Write the PL/SQL query to develop an anonymous block which:

1. Reset the credit limits of all customers to zero.
2. Fetch customers sorted by sales in descending order and give them new credit limits from a budget of 1 million.

```
CREATE TABLE orders
(
 order_id NUMBER
        GENERATED BY DEFAULT AS IDENTITY START WITH 201
        PRIMARY KEY,
 customer_id NUMBER(6, 0) NOT NULL,
 status      VARCHAR(20) NOT NULL,
 salesman_id NUMBER(6, 0),
 order_date  DATE NOT NULL
);

CREATE TABLE order_items
(
 order_id NUMBER(12, 0),
 item_id  NUMBER(12, 0),
 product_id NUMBER(12, 0) NOT NULL,
 quantity NUMBER(8, 2) NOT NULL,
 unit_price NUMBER(8, 2) NOT NULL
);

CREATE OR REPLACE VIEW Sales AS
SELECT customer_id,
      SUM(unit_price * quantity) total,
      ROUND(SUM(unit_price * quantity) * 0.05) credit,
FROM order_items
INNER JOIN orders USING (order_id)
WHERE status = "shipped"
GROUP BY customer_id;

DECLARE
 l_budget NUMBER := 1000000;
 CURSOR c_sales IS
 SELECT * FROM sales
 ORDER BY total DESC;
  r_sales c_sales%ROWTYPE;
BEGIN
 UPDATE customers SET credit_limit = 0;
 OPEN c_sales;
 LOOP
  FETCH c_sales INTO r_sales;
  EXIT WHEN c_sales%NOTFOUND;
  UPDATE
   customers
```

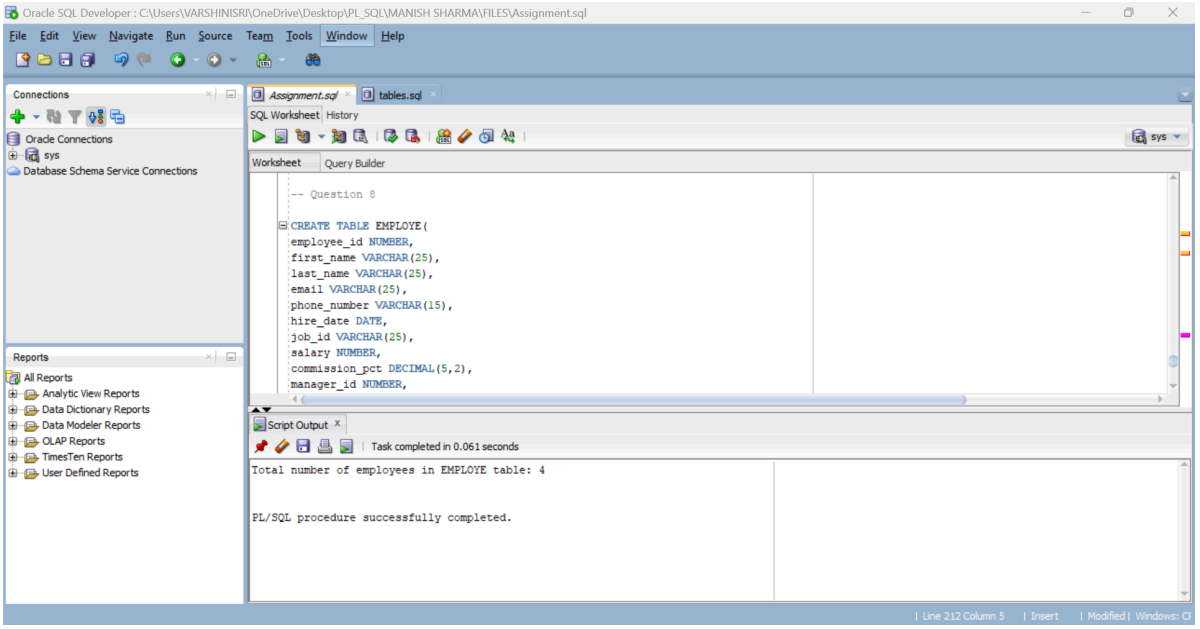
```
  SET
   credit_limit =
   CASE WHEN l_budget > r_sales.credit
    THEN r_sales.credit
     ELSE l_budget
   END
  WHERE
   customer_id = r_sales.customer_id;
  l_budget := l_budget - r_sales.credit;
  DBMS_OUTPUT.PUT_LINE("Customer id: " || r_sales.customer_id || "Credit: " || r_sales.credit ||
"Remaining Budget: " || l_budget);
  EXIT WHEN l_budget <= 0;
 END LOOP;
 CLOSE c_sales;
END;

SELECT customer_id, name, credit_limit FROM customers ORDER BY credit_limit DESC;
```

Question 8: Write a program in PL/SQL to show the uses of implicit cursor without using any attribute.

Table: employees

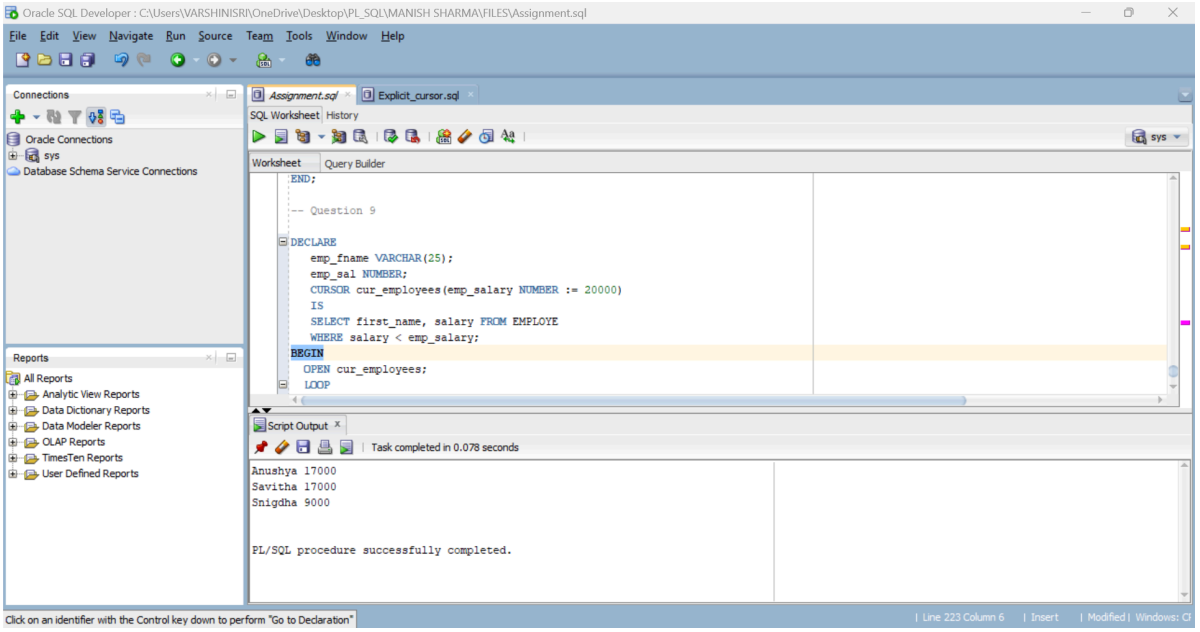
employee_id	integer
first_name	varchar(25)
last_name	varchar(25)
email	varchar(25)
phone_number	archar(25)
hire_date	varchar(15) date
job_id	varchar(25) integer
salary	decimal(5,2)
commission_pct	integer
manager_id	integer
department_id	integer

```
CREATE TABLE EMPLOYE(
employee_id NUMBER,
first_name VARCHAR(25),
last_name VARCHAR(25),
email VARCHAR(25),
phone_number VARCHAR(15),
hire_date DATE,
job_id VARCHAR(25),
salary NUMBER,
commission_pct DECIMAL(5,2),
manager_id NUMBER,
department_id NUMBER
);

INSERT INTO EMPLOYE VALUES(501, 'Anitha', 'Manogaran', 'ABC', '9876543210',
TO_DATE('2020-06-06', 'YYYY-MM-DD'), 'AD_PRES', 24000, 10.05, 101, 601);
INSERT INTO EMPLOYE VALUES(502, 'Anushya', 'Narayanan', 'DEF', '9876543211',
TO_DATE('2021-06-06', 'YYYY-MM-DD'), 'AD_VP', 17000, 12.05, 102, 602);
INSERT INTO EMPLOYE VALUES(503, 'Savitha', 'Ragunathan', 'GHI', '9876543212',
TO_DATE('2016-06-06', 'YYYY-MM-DD'), 'AD_VP', 17000, 14.05, 103, 603);
INSERT INTO EMPLOYE VALUES(504, 'Snigdha', 'Agarwal', 'JKL', '9876543213',
TO_DATE('2019-06-06', 'YYYY-MM-DD'), 'IT_PROG', 9000, 15.05, 104, 604);

DECLARE
  emp_count NUMBER;
BEGIN
  SELECT COUNT(*)INTO emp_count FROM EMPLOYE;
  DBMS_OUTPUT.PUT_LINE('Total number of employees in EMPLOYE table: ' || emp_count);
END;
```



Question 9: Write a program in PL/SQL to create a cursor displays the name and salary of each employee in the EMPLOYEES table whose salary is less than that specified by a passed-in parameter value.

```
DECLARE
  emp_fname VARCHAR(25);
  emp_sal NUMBER;
  CURSOR cur_employees(emp_salary NUMBER := 20000)
  IS
  SELECT first_name, salary FROM EMPLOYE
  WHERE salary < emp_salary;
BEGIN
 OPEN cur_employees;
 LOOP
   FETCH cur_employees INTO emp_fname, emp_sal;
   EXIT WHEN cur_employees%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE(emp_fname ||' '||emp_sal);
 END LOOP;
 CLOSE cur_employees;
END;
```



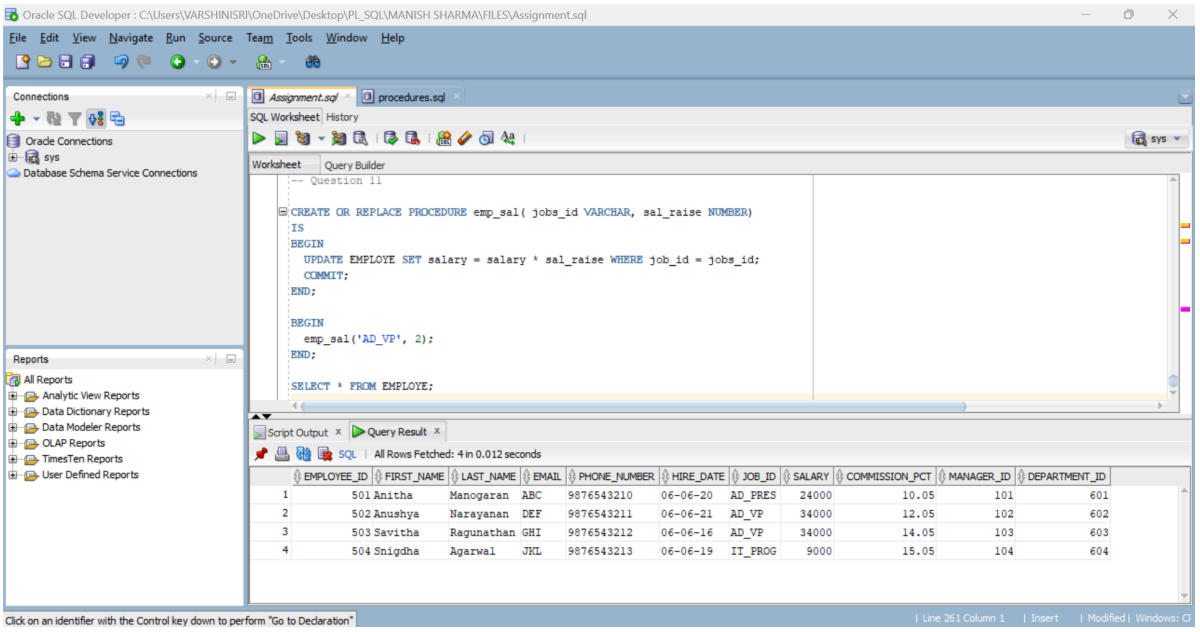
Question 10: Write a code in PL/SQL to create a trigger that checks for duplicate values in a specific column and raises an exception if found.

```
CREATE OR REPLACE TRIGGER check_duplicate_job_id
BEFORE INSERT OR UPDATE ON EMPLOYE
FOR EACH ROW
DECLARE
   v_count NUMBER;
BEGIN
   SELECT COUNT(*) INTO v_count FROM EMPLOYE WHERE job_id = :new.job_id;
   IF v_count > 0 THEN
      RAISE_APPLICATION_ERROR('Duplicate job_id found: ' || :new.job_id);
   END IF;
END;
```

Question 11: **Write a PL/SQL procedure for selecting some records from the database using some parameters as filters.**

Consider that we are fetching details of employees from ib_employee table where salary is a parameter for filter.

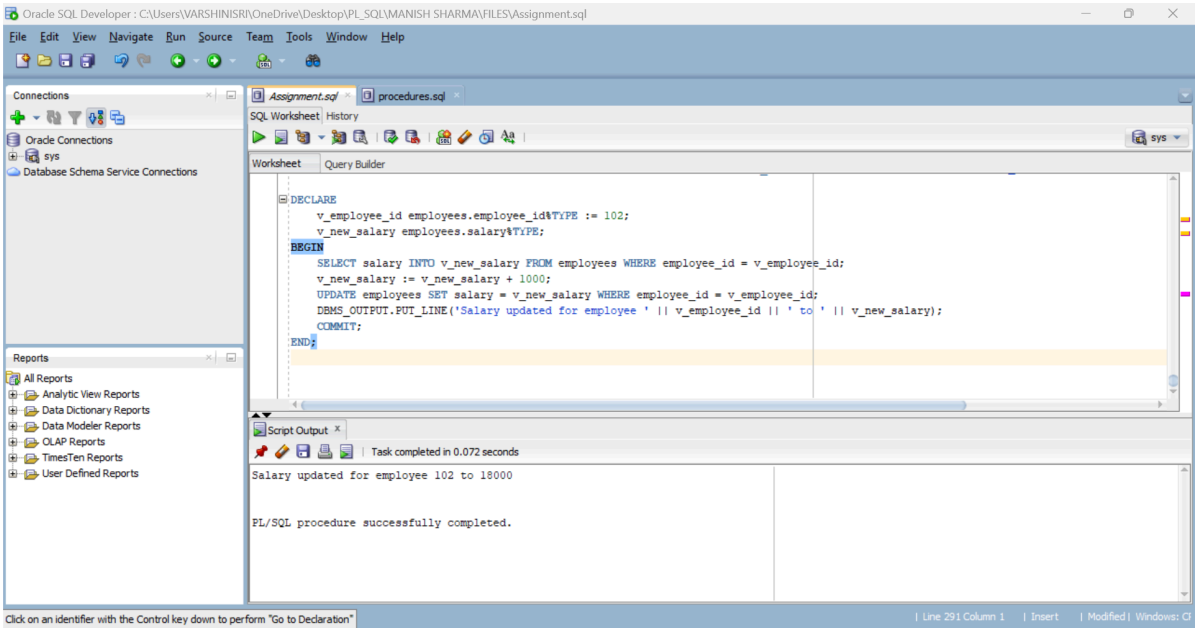
```
CREATE OR REPLACE PROCEDURE emp_sal( jobs_id VARCHAR, sal_raise NUMBER)
IS
BEGIN
  UPDATE EMPLOYE SET salary = salary * sal_raise WHERE job_id = jobs_id;
  COMMIT;
END;

BEGIN
  emp_sal('AD_VP', 2);
END;

SELECT * FROM EMPLOYE;
```



Question 12: **Write PL/SQL code block to increment the employee's salary by 1000 whose employee_id is 102 from the given table below.**

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL_ID	PHONE_NUMBER	JOIN_DATE	JOB_ID	SALARY
100	ABC	DEF	abef	9876543210	2020-06-06	AD_PRES	24000.00
101	GHI	JKL	ghkl	9876543211	2021-02-08	AD_VP	17000.00
102	MNO	PQR	mnqr	9876543212	2016-05-14	AD_VP	17000.00
103	STU	VWX	stwx	9876543213	2019-06-24	IT_PROG	9000.00

```
CREATE TABLE employees
(
 EMPLOYEE_ID NUMBER PRIMARY KEY,
 FIRST_NAME  VARCHAR2(255) NOT NULL,
 LAST_NAME   VARCHAR2(255) NOT NULL,
 EMAIL_ID    VARCHAR2(255) NOT NULL,
 PHONE_NUMBER VARCHAR2(50) NOT NULL,
 JOIN_DATE   DATE NOT NULL,
 JOB_ID      VARCHAR2(50) NOT NULL,
 SALARY      NUMBER NOT NULL
);

INSERT INTO employees VALUES(100, 'ABC', 'DEF', 'abef', '9876543210', TO_DATE('2020-06-06',
'YYYY-MM-DD'), 'AD_PRES', 24000.00);
INSERT INTO employees VALUES(101, 'GHI', 'JKL', 'ghkl', '9876543211', TO_DATE('2021-06-06',
'YYYY-MM-DD'), 'AD_VP', 17000.00);
INSERT INTO employees VALUES(102, 'MNO', 'PQR', 'mnqr', '9876543212', TO_DATE('2016-06-06',
'YYYY-MM-DD'), 'AD_VP', 17000.00);
INSERT INTO employees VALUES(103, 'STU', 'VWX', 'stwx', '9876543213', TO_DATE('2019-06-06',
'YYYY-MM-DD'), 'IT_PROG', 9000.00);

DECLARE
   v_employee_id employees.employee_id%TYPE := 102;
   v_new_salary employees.salary%TYPE;
BEGIN
   SELECT salary INTO v_new_salary FROM employees WHERE employee_id = v_employee_id;
   v_new_salary := v_new_salary + 1000;
   UPDATE employees SET salary = v_new_salary WHERE employee_id = v_employee_id;
   DBMS_OUTPUT.PUT_LINE('Salary updated for employee ' || v_employee_id || ' to ' ||
v_new_salary);
   COMMIT;
END;
```


Oracle SQL Developer : C:\Users\VARSHINISRI\OneDrive\Desktop\PL_SQL\MANISH SHARMA\FILES\Assignment.sql
File Edit View Navigate Run Source Team Tools Window Help
Connections
Oracle Connections
sys
Database Schema Service Connections
Reports
All Reports
Analytic View Reports
Data Dictionary Reports
Data Modeler Reports
OLAP Reports
TimesTen Reports
User Defined Reports
Assignment.sql
procedures.sql
SQL Worksheet History
Worksheet Query Builder
DECLARE
v_employee_id employees.employee_id%TYPE := 102;
v_new_salary employees.salary%TYPE;
BEGIN
SELECT salary INTO v_new_salary FROM employees WHERE employee_id = v_employee_id;
v_new_salary := v_new_salary + 1000;
UPDATE employees SET salary = v_new_salary WHERE employee_id = v_employee_id;
DBMS_OUTPUT.PUT_LINE('Salary updated for employee ' || v_employee_id || ' to ' || v_new_salary);
COMMIT;
END;
Script Output
Task completed in 0.072 seconds
Salary updated for employee 102 to 18000
PL/SQL procedure successfully completed.
Click on an identifier with the Control key down to perform "Go to Declaration"
Line 291 Column 1
Insert
Modified
Windows: CR