**PL/SQL Assignment Questions:**

CREATE TABLE EMPLOYEES (

EMP\_ID NUMBER PRIMARY KEY,

EMP\_NAME VARCHAR2(100),

DEPARTMENT VARCHAR2(50),

SALARY NUMBER

);

# 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)

*ANS:*

CREATE PROCEDURE insert\_employee (

p\_emp\_id IN NUMBER,

p\_emp\_name IN VARCHAR2,

p\_department IN VARCHAR2,

p\_salary IN NUMBER

) AS

BEGIN

INSERT INTO EMPLOYEES (EMP\_ID, EMP\_NAME, DEPARTMENT, SALARY)

VALUES (p\_emp\_id, p\_emp\_name, p\_department, p\_salary);

END;

//

Insert Values:

BEGIN

insert\_employee(1, 'Kowsalya', 'HR', 50000);

insert\_employee(2, 'Santhiya', 'Sales', 60000);

insert\_employee(3, 'Sowmiya', 'IT', 4000);

insert\_employee(3, 'Swathi', 'Manager', 55000);

END;

/

# 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 PROCEDURE update\_salary (

IN p\_emp\_id INT

)

BEGIN

DECLARE current\_salary INT;

SELECT SALARY INTO current\_salary

FROM EMPLOYEES

WHERE EMP\_ID = p\_emp\_id

LIMIT 1;

IF current\_salary < 5000 THEN

UPDATE EMPLOYEES

SET SALARY = current\_salary \* 1.10

WHERE EMP\_ID = p\_emp\_id;

ELSEIF current\_salary BETWEEN 5000 AND 10000 THEN

UPDATE EMPLOYEES

SET SALARY = current\_salary \* 1.075

WHERE EMP\_ID = p\_emp\_id;

ELSE

UPDATE EMPLOYEES

SET SALARY = current\_salary \* 1.05

WHERE EMP\_ID =p\_emp\_id;

END IF;

END //

SET SQL\_SAFE\_UPDATES = 1;

CALL update\_salary(1);

CALL update\_salary(2);

SELECT \* FROM EMPLOYEES;

# 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.

CREATE PROCEDURE display\_employee\_names()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE emp\_name VARCHAR(100);

-- Declare cursor

DECLARE emp\_cursor CURSOR FOR SELECT EMP\_NAME FROM EMPLOYEES;

-- Declare continue handler

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

-- Open cursor

OPEN emp\_cursor;

-- Fetch each row and display employee name read\_loop:

LOOP

FETCH emp\_cursor INTO emp\_name;

IF done THEN

LEAVE read\_loop;

END IF;

-- Display employee name using a SELECT statement to mimic printing

SELECT emp\_name AS Employee\_Name;

END LOOP;

-- Close cursor CLOSE emp\_cursor;

END

//

CALL display\_employee\_names();

# 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 VIEW high\_salary\_employees AS

SELECT EMP\_ID, EMP\_NAME, DEPARTMENT, SALARY FROM EMPLOYEES

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 FUNCTION Calculate\_bonus (p\_salary DECIMAL(10, 2))

RETURNS DECIMAL(10, 2)

DETERMINISTIC BEGIN

DECLARE bonus DECIMAL(10, 2);

IF p\_salary < 5000 THEN

SET bonus = p\_salary \* 0.10;

ELSEIF p\_salary BETWEEN 5000 AND 10000 THEN

SET bonus = p\_salary \* 0.075;

ELSE

SET bonus = p\_salary \* 0.05; END IF;

RETURN bonus;

END;

//

SELECT EMP\_ID, EMP\_NAME, SALARY, Calculate\_bonus(SALARY) AS BONUS

FROM EMPLOYEES;

# 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 (

Log\_Id INT AUTO\_INCREMENT PRIMARY KEY,

EMP\_ID INT,

EMP\_NAME VARCHAR(100),

DEPARTMENT VARCHAR(50),

SALARY DECIMAL(10, 2),

INSERTION\_TIME TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TRIGGER log\_employee\_insert

AFTER INSERT ON EMPLOYEES

FOR EACH ROW

BEGIN

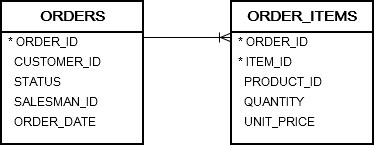
INSERT INTO Employee\_Log (EMP\_ID, EMP\_NAME, DEPARTMENT, SALARY)

VALUES (NEW.EMP\_ID, NEW.EMP\_NAME, NEW.DEPARTMENT, NEW.SALARY);

END ;

SELECT \* FROM Employee\_Log;

**Question 7:**Consider the orders and order\_items tables from the sample database.



1. Create a view that returns the sales revenues by customers. The values of the credit column are 5% of the total sales revenues.
2. Write the PL/SQL query to develop an anonymous block which:
3. Reset the credit limits of all customers to zero.
4. Fetch customers sorted by sales in descending order and give them new credit limits from a budget of 1 million.

CREATE TABLE customers (

Customer\_id INT PRIMARY KEY,

Customer\_name VARCHAR(100)

);

ALTER TABLE customers

ADD credit\_limit DECIMAL(10, 2);

CREATE TABLE orders (

Order\_id INT PRIMARY KEY,

Customer\_id INT,

status VARCHAR(50),

salesman\_id INT,

order\_date DATE,

FOREIGN KEY (Customer\_id) REFERENCES customers(Customer\_id)

);

SET SQL\_SAFE\_UPDATES = 0;

UPDATE customers SET credit\_limit = 0;

CREATE TABLE Order\_Items (

Orde\_rid INT,

Item\_id INT,

Product\_id INT,

quantity INT,

unit\_price DECIMAL(10, 2),

PRIMARY KEY (Order\_id, Item\_id),

FOREIGN KEY (Order\_id) REFERENCES orders(Order\_id)

);

INSERT INTO customers (Customer\_id, Customer\_name)

VALUES (1, 'Sowmiya'),

(2, 'Santhiya'),

(3, 'Kowsi');

INSERT INTO orders (Order\_id, Customer\_id, status, salesman\_id, order\_date) VALUES

(1, 1, 'Shipped', 101, '2023-01-15'),

(2, 2, 'Pending', 102, '2023-02-21'),

(3, 1, 'Pending', 103, '2023-03-05'),

(4, 3, 'Shipped', 101, '2023-05-16');

INSERT INTO Order\_Items (Order\_id, Item\_id, Product\_id, quantity, unit\_price) VALUES

(1, 1, 1001, 2, 500.00),

(1, 2, 1002, 1, 1500.00),

(2, 1, 1001, 3, 500.00),

(3, 1, 1003, 4, 250.00),

(4, 1, 1002, 2, 1500.00);

CREATE VIEW sales\_revenues\_by\_customers AS

SELECT

order.Customer\_id,

SUM(oi.quantity \* oi.unit\_price) AS total\_sales\_revenue,

SUM(oi.quantity \* oi.unit\_price) \* 0.05 AS credit

FROM orders order

JOIN

Order\_Items oi ON order.Order\_id = oi.Order\_id

GROUP BY order.Customer\_id;

DELIMITER //

CREATE PROCEDURE update\_credit\_limits()

BEGIN

DECLARE v\_budget DECIMAL(10, 2) DEFAULT 1000000;

DECLARE v\_remaining\_budget DECIMAL(10, 2) DEFAULT 1000000;

DECLARE v\_credit\_limit DECIMAL(10, 2);

DECLARE v\_customerid INT;

DECLARE v\_total\_sales\_revenue DECIMAL(10, 2);

DECLARE done INT DEFAULT 0; -- Declare done as a local variable

DECLARE customer\_cursor CURSOR FOR

SELECT Customer\_id, total\_sales\_revenue

FROM sales\_revenues\_by\_customers

ORDER BY total\_sales\_revenue DESC;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

UPDATE customers SET credit\_limit = 0;

OPEN customer\_cursor;

read\_loop: LOOP

FETCH customer\_cursor INTO v\_customerid, v\_total\_sales\_revenue;

IF done THEN LEAVE read\_loop;

END IF;

SET v\_credit\_limit = v\_total\_sales\_revenue \* 0.05;

IF v\_credit\_limit > v\_remaining\_budget THEN

SET v\_credit\_limit = v\_remaining\_budget;

END IF;

UPDATE customers

SET credit\_limit = v\_credit\_limit

WHERE customerid = v\_customerid;

SET v\_remaining\_budget = v\_remaining\_budget - v\_credit\_limit;

IF v\_remaining\_budget <= 0 THEN LEAVE read\_loop;

END IF; END LOOP;

-- Close the cursor CLOSE customer\_cursor;

END //

CALL update\_credit\_limits();

**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 | |  | archar(25) |
| phone\_number | |  | varchar(15) |
| hire\_date | |  | date |
| job\_id | |  | varchar(25) |
| salary | |  | integer |
| commission\_pct | |  | decimal(5,2) |
| manager\_id | |  | integer |
| department\_id | |  | integer |
|  | |  |  |
|  |  |  |  |

CREATE TABLE employee (

employee\_id INT PRIMARY KEY,

first\_name VARCHAR(25),

last\_name VARCHAR(25),

email VARCHAR(25),

phone\_number VARCHAR(15),

hire\_date DATE,

job\_id VARCHAR(25),

salary INT,

commission\_pct DECIMAL(5,2),

manager\_id INT,

department\_id INT

);

INSERT INTO employee (employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id)

VALUES

(1, 'sathiya', 'banu', 'sathiya@gmail..com', '123-456-7890', '2020-07-19', 'IT\_PROG', 50000, NULL, 101, 10),

(2, 'siva', 'kumar', 'siva@gmail.com', '987-654-3210', '2019-03-23', 'HR\_REP', 35000, NULL, 102, 20),

(3, 'abi', 'nithi', 'abi@gmail.com', '456-789-0123', '2021-06-30', 'FIN\_ANALYST', 60000, 0.10, 103, 30);

//

DROP PROCEDURE IF EXISTS display\_employee\_info;

//

CREATE PROCEDURE display\_employee\_info ()

BEGIN

DECLARE v\_employee\_id INT;

DECLARE v\_first\_name VARCHAR(25);

DECLARE v\_last\_name VARCHAR(25);

DECLARE v\_email VARCHAR(25);

DECLARE v\_phone\_number VARCHAR(15);

DECLARE v\_hire\_date DATE;

DECLARE v\_job\_id VARCHAR(25);

DECLARE v\_salary INT;

DECLARE v\_commission\_pct DECIMAL(5,2);

DECLARE v\_manager\_id INT;

DECLARE v\_department\_id INT;

SELECT employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id

INTO v\_employee\_id, v\_first\_name, v\_last\_name, v\_email, v\_phone\_number, v\_hire\_date, v\_job\_id, v\_salary, v\_commission\_pct, v\_manager\_id, v\_department\_id

FROM employees

WHERE employee\_id = 1;

SELECT 'Employee ID: ', v\_employee\_id;

SELECT 'First Name: ', v\_first\_name;

SELECT 'Last Name: ', v\_last\_name;

SELECT 'Email: ', v\_email;

SELECT 'Phone Number: ', v\_phone\_number;

SELECT 'Hire Date: ', v\_hire\_date;

SELECT 'Job ID: ', v\_job\_id;

SELECT 'Salary: ', v\_salary;

SELECT 'Commission Pct: ', v\_commission\_pct;

SELECT 'Manager ID: ', v\_manager\_id;

SELECT 'Department ID: ', v\_department\_id;

END //

CALL display\_employee\_info();

**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 passedin parameter value. Table: employees

|  |  |  |
| --- | --- | --- |
| employee\_id |  | integer |
| first\_name |  | varchar(25) |
| last\_name |  | varchar(25) |
| email |  | archar(25) |
| phone\_number | | varchar(15) |
| hire\_date | | date |
| job\_id | | varchar(25) |
| salary | | integer |
| commission\_pct | | decimal(5,2) |
| manager\_id | | integer |
| department\_id | | integer |

DROP PROCEDURE IF EXISTS display\_low\_salary\_employees;

//

CREATE PROCEDURE display\_low\_salary\_employees(IN max\_salary INT)

BEGIN

DECLARE v\_first\_name VARCHAR(25);

DECLARE v\_last\_name VARCHAR(25);

DECLARE v\_salary INT;

DECLARE done INT DEFAULT 0;

DECLARE cur\_employee CURSOR FOR

SELECT first\_name, last\_name, salary

FROM employees

WHERE salary < max\_salary;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

FETCH cur\_employee INTO v\_first\_name, v\_last\_name, v\_salary;

IF done THEN

LEAVE read\_loop;

END IF;

SELECT CONCAT('Name: ', v\_first\_name, ' ', v\_last\_name, ' - Salary: ', v\_salary) AS Employee\_Info;

END LOOP;

CLOSE cur\_employee;

END //

SET SQL\_SAFE\_UPDATES = 0;

CALL display\_low\_salary\_employees(50000);

**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 TRIGGER check\_duplicate\_email

BEFORE INSERT ON employees

FOR EACH ROW

BEGIN

DECLARE email\_count INT;

SELECT COUNT(\*)

INTO email\_count

FROM employees

WHERE email = NEW.email AND employee\_id != NEW.employee\_id;

IF email\_count > 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Duplicate email address found. Each employee must have a unique email address.';

END IF;

END //

INSERT INTO employees (employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id)

VALUES (4, 'Kowsy', 's', 'kowsalya.s@example.com', '555-1234', '2024-07-19', 'IT\_PROG', 60000, NULL, NULL, 60);

INSERT INTO employees (employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id)

VALUES (5, ‘santhiya', 'p', 'santhiya.p@example.com', '555-5678', '2024-07-19', 'IT\_PROG', 70000, NULL, NULL, 60)

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

CREATE PROCEDURE get\_employees\_by\_salary(

IN min\_salary DECIMAL(10, 2),

IN max\_salary DECIMAL(10, 2)

);

BEGIN

SELECT employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id

FROM employees

WHERE salary BETWEEN min\_salary AND max\_salary;

END //

CALL get\_employees\_by\_salary(30000, 70000);

 Consider that we are fetching details of employees from ib\_employee table where salary is a parameter for filter.

**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.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **EMPLOYE**  **E\_ID** | | 100 | | 101 | | 102 | | 103 | | |  | | --- | | **FIRST\_NA**  **ME** | | ABC | | GHI | | MNO | | STU | | |  | | --- | | **LAST\_NA**  **ME** | | DEF | | JKL | | PQR | | VWX | | |  | | --- | | **EMAIL**  **\_ID** | | abef | | ghkl | | mnqr | | stwx | | |  | | --- | | **PHONE\_NU**  **MBER** | | 9876543210 | | 9876543211 | | 9876543212 | | 9876543213 | | |  | | --- | | **JOIN\_D**  **ATE** | | 2020-06-  06 | | |  | | --- | | **JOB\_I**  **D** | | AD\_PR  ES | | |  | | --- | | **SALA**  **RY** | | 24000. | |
| 2021-02-    08  AD\_VP  17000.  00    2016-05-  14    AD\_VP  17000.  00    2019-06-  24    IT\_PR  OG  9000.0  0 | | |

CREATE TABLE employeesTable (

employee\_id INT PRIMARY KEY,

first\_name VARCHAR(25),

last\_name VARCHAR(25),

email\_id VARCHAR(50),

phone\_number VARCHAR(15),

join\_date DATE,

job\_id VARCHAR(25),

salary DECIMAL(10, 2)

);

INSERT INTO employeesTable (employee\_id, first\_name, last\_name, email\_id, INSERT INTO employeesTable (employee\_id, first\_name, last\_name, email\_id, phone\_number,

join\_date, job\_id, salary)

VALUES

(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 PROCEDURE increment\_salary(

employee\_id INT,

increment\_amount DECIMAL(10,2)

)

BEGIN

UPDATE employeesTable

SET salary = salary + increment\_amount

WHERE employee\_id = employee\_id;

END;

//

CALL increment\_salary(102, 1000);

SELECT \* FROM employeesTable

WHERE employee\_id = 102;