PL/SQL Assignment

Krishna Kumar P

07/07/2023

Tables:

1)CREATE TABLE customers

```
 (
   customer_id INT PRIMARY KEY,
   name        VARCHAR( 255 ) NOT NULL,
   address     VARCHAR( 255 )        ,
   website     VARCHAR( 255 )        ,
   credit_limit DECIMAL( 8, 2 )
 );
```

2)CREATE TABLE employees

```
 (
   employee_id INT PRIMARY KEY,
   first_name VARCHAR( 255 ) NOT NULL,
   last_name  VARCHAR( 255 ) NOT NULL,
   email      VARCHAR( 255 ) NOT NULL,
   phone      VARCHAR( 50 ) NOT NULL ,
   hire_date  DATE NOT NULL         ,
   manager_id INT       , -- fk
   job_title  VARCHAR( 255 ) NOT NULL,
   salary INT(50)
 );
```

3)CREATE TABLE orders

```
 (
```

```
  order_id INT PRIMARY KEY,
  customer_id INT NOT NULL, -- fk
  status      VARCHAR( 20 ) NOT NULL ,
  salesman_id INT         , -- fk
  order_date  DATE NOT NULL         ,
  CONSTRAINT fk_orders_customers
    FOREIGN KEY( customer_id )
    REFERENCES customers( customer_id )
    ON DELETE CASCADE,
  CONSTRAINT fk_orders_employees
    FOREIGN KEY( salesman_id )
    REFERENCES employees( employee_id )
    ON DELETE SET NULL
);
```

4)

```
CREATE TABLE regions
(
  region_id INT PRIMARY KEY,
  region_name VARCHAR( 50 ) NOT NULL
);
```

5)

```
CREATE TABLE countries
(
  country_id   CHAR( 2 ) PRIMARY KEY,
  country_name VARCHAR( 40 ) NOT NULL,
  region_id    INT,
  CONSTRAINT fk_countries_regions FOREIGN KEY( region_id )
    REFERENCES regions( region_id )
    ON DELETE CASCADE
);
```

6)CREATE TABLE locations

```
(
  location_id INT PRIMARY KEY       ,
  address     VARCHAR( 255 ) NOT NULL,
  postal_code VARCHAR( 20 )         ,
  city        VARCHAR( 50 )         ,
  state       VARCHAR( 50 )         ,
  country_id  CHAR( 2 )             , -- fk
  CONSTRAINT fk_locations_countries
    FOREIGN KEY( country_id )
    REFERENCES countries( country_id )
    ON DELETE CASCADE
);
```

7)CREATE TABLE warehouses

```
(
  warehouse_id INT PRIMARY KEY,
  warehouse_name VARCHAR( 255 ) ,
  location_id    INT, -- fk
  CONSTRAINT fk_warehouses_locations
    FOREIGN KEY( location_id )
    REFERENCES locations( location_id )
    ON DELETE CASCADE
);
```

8) CREATE TABLE product_categories

```
(
  category_id INT PRIMARY KEY,
  category_name VARCHAR( 255 ) NOT NULL
);
```

9)CREATE TABLE products

```
(
  product_id INT PRIMARY KEY,
  product_name  VARCHAR( 255 ) NOT NULL,
  description   VARCHAR( 2000 )       ,
  standard_cost DECIMAL( 9, 2 )         ,
  list_price    DECIMAL( 9, 2 )         ,
  category_id   INT NOT NULL        ,
  CONSTRAINT fk_products_categories
    FOREIGN KEY( category_id )
    REFERENCES product_categories( category_id )
    ON DELETE CASCADE
);
```

10)CREATE TABLE contacts

```
(
  contact_id INT PRIMARY KEY,
  first_name  VARCHAR( 255 ) NOT NULL,
  last_name   VARCHAR( 255 ) NOT NULL,
  email       VARCHAR( 255 ) NOT NULL,
  phone       VARCHAR( 20 )         ,
  customer_id INT                   ,
  CONSTRAINT fk_contacts_customers
    FOREIGN KEY( customer_id )
    REFERENCES customers( customer_id )
    ON DELETE CASCADE
);
```

11)CREATE TABLE order_items

```
(
  order_id   INT,
```

```
  item_id    INT,
  product_id INT NOT NULL,
  quantity   DECIMAL( 8, 2 ) NOT NULL ,
  unit_price DECIMAL( 8, 2 ) NOT NULL ,
  CONSTRAINT pk_order_items
    PRIMARY KEY( order_id, item_id ),
  CONSTRAINT fk_order_items_products
    FOREIGN KEY( product_id )
    REFERENCES products( product_id )
    ON DELETE CASCADE,
  CONSTRAINT fk_order_items_orders
    FOREIGN KEY( order_id )
    REFERENCES orders( order_id )
    ON DELETE CASCADE
);
```

12)

```
CREATE TABLE inventories
(
  product_id   INT,
  warehouse_id INT,
  quantity INT NOT NULL,
  CONSTRAINT pk_inventories
    PRIMARY KEY( product_id, warehouse_id ),
  CONSTRAINT fk_inventories_products
    FOREIGN KEY( product_id )
    REFERENCES products( product_id )
    ON DELETE CASCADE,
  CONSTRAINT fk_inventories_warehouses
    FOREIGN KEY( warehouse_id )
    REFERENCES warehouses( warehouse_id )
    ON DELETE CASCADE
```

```
);
```

Assignment Questions and Answers:

1)Create a PL/SQL block to adjust the salary from 7900 to 9000 of the employee whose ID 122.

```
delimiter @@
create procedure sp_salary_update (in x int, in new_salary  int  )
begin
update employees
set salary = new_salary
where employee_id = x ;
end @@
delimiter ;

call sp_salary_update(122,9000);

select * from employees where employee_id = 122;
```

Result:

Result Grid | Filter Rows: | Edit:

employee_id	first_name	last_name	hire_date	salary
122	Summer	Payne	2016-06-07	9000
NULL	NULL	NULL	NULL	NULL

3. Create a PL/SQL procedure to calculate the incentive achieved according to the specific sale limit

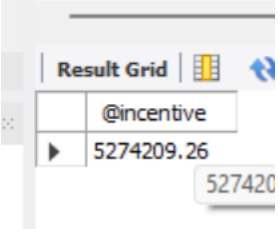
```
DELIMITER //
CREATE PROCEDURE calculate_incentive(IN sales_limit DECIMAL(10, 2), OUT incentive DECIMAL(10, 2))
BEGIN
  -- Declare variables
  DECLARE total_sales DECIMAL(10, 2);
  -- Calculate the total sales
```

```
select SUM(quantity * unit_price) into total_sales from order_items;

  -- Calculate the incentive
  IF total_sales >= sales_limit THEN
    SET incentive = total_sales * 0.1; -- 10% incentive rate
  ELSE
    SET incentive = 0;
  END IF;
END //
DELIMITER ;

CALL calculate_incentive(10000000, @incentive);
SELECT @incentive;
```

Result:



@incentive
5274209.26

4. Print a list of managers using PL/SQL explicit cursors:

```
DELIMITER //
CREATE PROCEDURE list_managers()
BEGIN
  -- Declare variables
  DECLARE done INT DEFAULT FALSE;
  DECLARE manager_first_name VARCHAR(50);
  DECLARE manager_last_name VARCHAR(50);
-- Declare cursor-like result set
  DECLARE cur CURSOR FOR
```

```
    SELECT e.first_name, e.last_name
    FROM employees e
    JOIN employees m ON e.manager_id = m.employee_id;
  -- Declare handler for NOT FOUND condition
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
  -- Open the result set
  OPEN cur;
  -- Fetch the rows and process them
  read_loop: LOOP
    -- Fetch the next row into variables
    FETCH cur INTO manager_first_name, manager_last_name;
    -- Exit the loop if no more rows
    IF done THEN
      LEAVE read_loop;
    END IF;
  -- Perform operations with the fetched data
    -- Example: Print the manager names
    SELECT CONCAT(manager_first_name, ' ', manager_last_name) AS manager_name;
  END LOOP;
-- Close the result set
  CLOSE cur;
END //
DELIMITER ;
CALL list_managers();
```

Result:

5. Create a PL/SQL cursor to calculate total salary from employee table without using sum() function.

```
DELIMITER //

CREATE PROCEDURE calculate_total_salary()

BEGIN

 -- Declare variables

 DECLARE done INT DEFAULT FALSE;

 DECLARE employee_salary DECIMAL(10, 2);

 DECLARE total_salary DECIMAL(10, 2) DEFAULT 0;

 -- Declare cursor

 DECLARE curr CURSOR FOR

  SELECT salary FROM employees where employee_id = 2;

 -- Declare handler for NOT FOUND condition

 DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

 -- Open the cursor

 OPEN curr;

 -- Fetch the rows and calculate total salary

 read_loop: LOOP

  -- Fetch the next row into the variable

  FETCH curr INTO employee_salary ;

  -- Exit the loop if no more rows

  IF done THEN

    LEAVE read_loop;

  END IF;

  -- Accumulate the salary

  SET total_salary = total_salary + employee_salary;

 END LOOP;

 -- Close the cursor

 CLOSE curr;
```

```
 -- Print the total salary
 SELECT total_salary AS TotalSalary;
END //
DELIMITER ;

CALL calculate_total_salary();
```

Result:

Result Grid | Filter

	TotalSalary
▶	2000000.00

6.Display the name and salary of each employee in the EMPLOYEES table whose salary is less than that specified by a passed-in parameter value:

```
delimiter //
CREATE PROCEDURE display_employees_below_salary(IN max_salary DECIMAL(10, 2))
BEGIN
 -- Display employee name and salary
 SELECT concat( first_name ,' ',last_name)  AS Employee,  salary
 AS Salary
 FROM employees
 WHERE salary < max_salary;
END //
delimiter ;
CALL display_employees_below_salary(20000);
```

Result:

Result Grid | Filter Rows:

	Employee	Salary
▶	Carter Gonzales	12312
	Caleb Diaz	1
	Harper Spencer	12324
	Summer Payne	9000

Result 63 ×

7.Display the name of the employee and increment percentage of salary according to their working experiences.

```
create view employees_experience as

 select concat(first_name,' ', last_name) as Employees_name, 2031 -year(hire_date) as experience
from employees;


 select * from employees_experience;


 DELIMITER //

CREATE PROCEDURE display_employee_increment()

BEGIN

  -- Display employee name and increment percentage

  SELECT

    employees_name AS Employee,

    CASE

              when experience >= 10 then '15%'

      WHEN experience >= 5 THEN '10%' -- 10% increment for experience >= 5 years

      WHEN experience >= 3 THEN '5%' -- 5% increment for experience >= 3 years

      ELSE 'No increment'

    END AS IncrementPercentage

  FROM employees_experience;

END //

DELIMITER ;


CALL display_employee_increment();
```

Result:

Result Grid | Filter Rows:

Employee	IncrementPercentage
Ellis Washington	No increment
Olivia Ford	No increment
Poppy Jordan	No increment
Lily Fisher	No increment
Kai Long	5%
Holly Shaw	5%
Tommy Bailey	15%
Jude Rivera	15%
Blake Cooper	15%
Louie Richardson	15%

Result 64

8. Display the number of employees by month using PL/SQL block:

```
DELIMITER //
CREATE PROCEDURE display_employees_by_month()
BEGIN
  -- Display the number of employees by month
 SELECT MONTH(hire_date) AS Month, COUNT(*) AS NumEmployees
FROM employees
GROUP BY MONTH(hire_date) order by month;
END //
DELIMITER ;
CALL display_employees_by_month();
```

Result:

Result Grid | Filter Rows:

Month	NumEmployees
1	14
2	13
3	17
4	7
5	6
6	11
7	7
8	9
9	5
10	6
11	5
12	7

Result 65

9. Create a PL/SQL block to insert records from employee table to another table:

```
CREATE TABLE employees_info
  (
    employee_id INT PRIMARY KEY,
    first_name VARCHAR( 255 ) NOT NULL,
    last_name  VARCHAR( 255 ) NOT NULL,
    email      VARCHAR( 255 ) NOT NULL,
    phone      VARCHAR( 50 ) NOT NULL
  );

 DELIMITER //
CREATE PROCEDURE insert_employee_records()
BEGIN
  -- Insert records from employee table into another table
  INSERT INTO employees_info (EMPLOYEE_ID,FIRST_NAME,LAST_NAME,EMAIL,PHONE)
  SELECT EMPLOYEE_ID,FIRST_NAME,LAST_NAME,EMAIL,PHONE
  FROM employees;
END //
DELIMITER ;

CALL  insert_employee_records();

select * from employees_info;
```

Result:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell

employee_id	first_name	last_name	email	phone
1	Tommy	Bailey	tommy.bailey@example.com	515.123.4567
2	Jude	Rivera	jude.rivera@example.com	515.123.4568
3	Blake	Cooper	blake.cooper@example.com	515.123.4569
4	Louie	Richardson	louie.richardson@example.com	590.423.4567
5	Nathan	Cox	nathan.cox@example.com	590.423.4568
6	Gabriel	Howard	gabriel.howard@example.com	590.423.4569
7	Charles	Ward	charles.ward@example.com	590.423.4560
8	Bobby	Torres	bobby.torres@example.com	590.423.5567
9	Mohammad	Peterson	mohammad.peterson@example.com	515.124.4569
10	Ryan	Gray	ryan.gray@example.com	515.124.4169
11	Tyler	Ramirez	tyler.ramirez@example.com	515.124.4269
12	Elliott	James	elliott.james@example.com	515.124.4369

employees_info 66

10. Create the PL/SQL package to calculate net value of the ordered items done by a particular customer in a specific year:

Create a stored function to calculate the net value of ordered items for a customer in a specific year

```
DELIMITER //

CREATE FUNCTION calculate_net_value(customer_id INT, order_year INT)

RETURNS DECIMAL(10,2)

deterministic

BEGIN

  DECLARE total_amount DECIMAL(10,2);

  -- Calculate the net value by summing the amounts of ordered items

  SELECT SUM(quantity*unit_price) INTO total_amount

  FROM (select o.order_id,customer_id ,quantity,unit_price,order_date from  orders o join
order_items i on o.order_id = i.order_id ) t

  WHERE customer_id = customer_id AND YEAR(order_date) = order_year;

  RETURN total_amount;

END //

DELIMITER ;

select calculate_net_value(2,2027) as purchase_value;
```

```
-- Create a stored procedure to display the net value of ordered items for a customer in a specific year

DELIMITER //

CREATE PROCEDURE display_net_value(IN customer_id INT, IN order_year INT)

BEGIN

  DECLARE net_value DECIMAL(10,2);

  -- Call the calculate_net_value function to get the net value

  SET net_value = calculate_net_value(customer_id, order_year);

  -- Display the net value

  SELECT CONCAT('Net Value for Customer ID ', customer_id, ' in Year ', order_year, ': ', net_value) AS NetValue;

END //

DELIMITER ;


CALL display_net_value(1, 2027) ;
```

Result:

Result Grid | Filter Rows: | Export:

NetValue
Net Value for Customer ID 1 in Year 2027: 3752809.94

11. Display the first 10 customers using nested table:

```
SELECT *

FROM (

  SELECT c.customer_id,concat(name,',',address,',', website,',',credit_limit) as Customer_info,

  co.phone

  FROM customers c join contacts co on c.customer_id = co.customer_id

  ORDER BY customer_id

  LIMIT 10

) AS r;
```

Result:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	Customer_info	phone
1	Raytheon,514 W Superior St, Kokomo, IN,http://www.raytheon.com,100.00	+1 317 123 4104
2	Plains GP Holdings,2515 Bloyd Ave, Indianapolis, IN,http://www.plainsallamerican.com,100.00	+1 317 123 4111
3	US Foods Holding,8768 N State Rd 37, Bloomington, IN,http://www.usfoods.com,100.00	+1 812 123 4115
4	AbbVie,6445 Bay Harbor Ln, Indianapolis, IN,http://www.abbvie.com,100.00	+1 317 123 4126
5	Centene,4019 W 3Rd St, Bloomington, IN,http://www.centene.com,100.00	+1 812 123 4129
6	Community Health Systems,1608 Portage Ave, South Bend, IN,http://www.chs.net,100.00	+1 219 123 4136
7	Alcoa,23943 Us Highway 33, Elkhart, IN,http://www.alcoa.com,100.00	+1 219 123 4138
8	International Paper,136 E Market St # 800, Indianapolis, IN,http://www.internationalpaper.com,10...	+1 317 123 4141
9	Emerson Electric,1905 College St, South Bend, IN,http://www.emerson.com,100.00	+1 219 123 4142
10	Union Pacific,3512 Rockville Rd # 137C, Indianapolis, IN,http://www.up.com,200.00	+1 317 123 4146

12. Fetch the data from employees table for employee_id '101' using native dynamic SQL (Execute Immediate)

```
DELIMITER //

CREATE PROCEDURE get_employee_details(IN employee_id INT)

BEGIN

  -- Declare variables

  DECLARE dynamic_sql_statement VARCHAR(1000);

  -- Construct dynamic SQL statement

  SET dynamic_sql_statement = CONCAT('SELECT * FROM employees WHERE employee_id = ',
employee_id);

  -- Execute dynamic SQL statement

  SET @sql = dynamic_sql_statement;

  PREPARE stmt FROM @sql;

  EXECUTE stmt;

  DEALLOCATE PREPARE stmt;

END //

DELIMITER ;

drop procedure get_employee_details;


CALL get_employee_details(20);
```

Result:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

employee_id	first_name	last_name	hire_date	salary
20	Dexter	Barnes	2010-08-16	562324

13. Create a statement-level trigger, when CRUD operation is performed on employees table:

```
CREATE  TABLE crud_info (
  employees_id INT,
  action VARCHAR(50)
);

drop table crud_info ;

DELIMITER //
CREATE  TRIGGER employees_insert_trigger
AFTER insert on  employees
FOR EACH ROW
BEGIN
  INSERT INTO crud_info (employees_id, action)
  values(new.employee_id,'insert_operation')  ;
END //
delimiter ;

delimiter //
CREATE TRIGGER employees_update_trigger
AFTER UPDATE ON employees
FOR EACH ROW
BEGIN
  INSERT INTO crud_info (employees_id, action)
  values(new.employee_id,'update_operation')  ;
```

```
END //
delimiter ;

delimiter //
CREATE TRIGGER employees_delete_trigger
AFTER DELETE ON employees
FOR EACH ROW
BEGIN
  INSERT INTO crud_info (employees_id, action)
  values(old.employee_id,'delete_operation') ;
END //
DELIMITER ;
Insert into EMPLOYEES
(EMPLOYEE_ID,FIRST_NAME,LAST_NAME,EMAIL,PHONE,HIRE_DATE,MANAGER_ID,JOB_TITLE,SALAR
Y) values (122,'Summer','Payne','summer.payne@example.com','515.123.8181','2016-06-
07',106,'Public Accountant',54236);

delete from employees where employee_id = 122;

update employees
set salary = 20000
where employee_id = 2;
select * from crud_info;
```

Result:

Result Grid | Filter Rows:

employees_id	action
122	delete_operation
122	insert_operation
2	update_operation
122	update_operation