SQL> -- 1.Write a stored procedure to display the total number of pizza's ordered by the given order number.

SQL> CREATE OR REPLACE PROCEDURE TotalPizzas(order\_input IN ORDERS. order\_no%TYPE, totalnum OUT NUMBER) IS

2 BEGIN

3 SELECT SUM(ol.qty) INTO totalnum

4 FROM ORDER\_LIST ol

5 WHERE ol.order\_no = order\_input;

6 END;

7 /

Procedure created.

SQL>

SQL> -- Valid Input

SQL> DECLARE

2 o\_no varchar2(5) := '&order\_no';

3 qty number;

4 BEGIN

5 TotalPizzas(o\_no, qty);

6 IF qty>0 THEN

7 DBMS\_OUTPUT.PUT\_LINE('Total number of pizzas ordered for Order ' || o\_no || ': ' || qty);

8 ELSE

9 DBMS\_OUTPUT.PUT\_LINE('Invalid Order Number.');

10 END IF;

11 END;

12 /

Enter value for order\_no: OP500

old 2: o\_no varchar2(5) := '&order\_no';

new 2: o\_no varchar2(5) := 'OP500';

Total number of pizzas ordered for Order OP500: 11

PL/SQL procedure successfully completed.

SQL>

SQL> -- Invalid Input

SQL> DECLARE

2 o\_no varchar2(5) := '&order\_no';

3 qty number;

4 BEGIN

5 TotalPizzas(o\_no, qty);

6 IF qty>0 THEN

7 DBMS\_OUTPUT.PUT\_LINE('Total number of pizzas ordered for Order ' || o\_no || ': ' || qty);

8 ELSE

9 DBMS\_OUTPUT.PUT\_LINE('Invalid Order Number.');

10 END IF;

11 END;

12 /

Enter value for order\_no: OP900

old 2: o\_no varchar2(5) := '&order\_no';

new 2: o\_no varchar2(5) := 'OP900';

Invalid Order Number.

PL/SQL procedure successfully completed.

SQL>

SQL> -- 2. For the given order number, calculate the Discount as follows:

SQL> -- For total amount > 2000 and total amount < 5000: Discount=5%

SQL> -- For total amount > 5000 and total amount < 10000: Discount=10%

SQL> -- For total amount > 10000: Discount=20%

SQL> -- Calculate the total amount (after the discount) and update the same in

SQL> -- orders table.

SQL>

SQL> SAVEPOINT A;

Savepoint created.

SQL>

SQL> ALTER TABLE orders

2 ADD discount\_price FLOAT;

Table altered.

SQL>

SQL> CREATE OR REPLACE PROCEDURE CalculateAndUpdateDiscount(order\_no\_input IN ORDERS.order\_no%TYPE) IS

2 order\_total NUMBER := 0;

3 disc\_rate NUMBER := 0;

4 disc\_amt NUMBER := 0;

5 final\_amt NUMBER := 0;

6 op\_qty ORDER\_LIST.qty%TYPE;

7 amt NUMBER := 0;

8 cust\_name CUSTOMER.cust\_name%TYPE;

9 order\_date ORDERS.order\_date%TYPE;

10 phone CUSTOMER.phone%TYPE;

11 BEGIN

12 -- Query to retrieve customer details for the given order number

13 SELECT c.cust\_name, o.order\_date, c.phone

14 INTO cust\_name, order\_date, phone

15 FROM ORDERS o

16 JOIN CUSTOMER c ON o.cust\_id = c.cust\_id

17 WHERE o.order\_no = order\_no\_input;

18

19 -- Output customer details

20 DBMS\_OUTPUT.PUT\_LINE('\*');

21 DBMS\_OUTPUT.PUT\_LINE('Order Number: ' || order\_no\_input || ' Customer Name: ' || cust\_name);

22 DBMS\_OUTPUT.PUT\_LINE('Order Date: ' || TO\_CHAR(order\_date, 'DD-Mon-YYYY') || ' Phone: ' || phone);

23 DBMS\_OUTPUT.PUT\_LINE('\*');

24

25 -- Loop through order details to calculate total amount

26 FOR order\_detail IN (

27 SELECT p.pizza\_type, ol.qty, p.unit\_price

28 FROM ORDER\_LIST ol

29 JOIN PIZZA p ON ol.pizza\_id = p.pizza\_id

30 WHERE ol.order\_no = order\_no\_input

31 ) LOOP

32 op\_qty := order\_detail.qty;

33 amt := op\_qty \* order\_detail.unit\_price;

34 order\_total := order\_total + amt;

35

36 -- Output order details

37 DBMS\_OUTPUT.PUT\_LINE(order\_detail.pizza\_type || ' ' || op\_qty || ' ' || order\_detail.unit\_price || ' ' || amt);

38 END LOOP;

39

40 -- Calculate discount rate based on order total

41 IF order\_total > 2000 AND order\_total < 5000 THEN

42 disc\_rate := 0.05;

43 ELSIF order\_total > 5000 AND order\_total < 10000 THEN

44 disc\_rate := 0.1;

45 ELSIF order\_total > 10000 THEN

46 disc\_rate := 0.2;

47 ELSE

48 disc\_rate := 0;

49 END IF;

50

51 -- Calculate discount amount and final amount

52 disc\_amt := order\_total \* disc\_rate;

53 final\_amt := order\_total - disc\_amt;

54

55 -- Output total amount, discount, and final amount

56 DBMS\_OUTPUT.PUT\_LINE('Total Amount : Rs.' || order\_total);

57 DBMS\_OUTPUT.PUT\_LINE('Discount (' || TO\_CHAR(disc\_rate \* 100, 'FM9990.00') || '%) : Rs.' || disc\_amt);

58 DBMS\_OUTPUT.PUT\_LINE('\*');

59 DBMS\_OUTPUT.PUT\_LINE('Amount to be paid : Rs.' || final\_amt);

60 DBMS\_OUTPUT.PUT\_LINE('\*');

61 DBMS\_OUTPUT.PUT\_LINE('Great Offers! Discount up to 25% on DIWALI Festival Day...');

62 DBMS\_OUTPUT.PUT\_LINE('\*');

63

64 -- Update order with discount price

65 UPDATE ORDERS

66 SET discount\_price = final\_amt

67 WHERE order\_no = order\_no\_input;

68

69 EXCEPTION

70 WHEN NO\_DATA\_FOUND THEN

71 DBMS\_OUTPUT.PUT\_LINE('No order found for the given order number.');

72 END;

73 /

Procedure created.

SQL>

SQL> -- Valid Input

SQL> DECLARE

2 o\_no ORDERS.order\_no%TYPE := '&order\_number';

3 BEGIN

4 CalculateAndUpdateDiscount(o\_no);

5 END;

6 /

Enter value for order\_number: OP500

old 2: o\_no ORDERS.order\_no%TYPE := '&order\_number';

new 2: o\_no ORDERS.order\_no%TYPE := 'OP500';

\*

Order Number: OP500 Customer Name: Hari

Order Date: 29-Jun-2015 Phone: 9001200031

\*

italian 6 200 1200

spanish 5 260 1300

Total Amount : Rs.2500

Discount (5.00%) : Rs.125

\*

Amount to be paid : Rs.2375

\*

Great Offers! Discount up to 25% on DIWALI Festival Day...

\*

PL/SQL procedure successfully completed.

SQL>

SQL> -- Invalid Input

SQL> DECLARE

2 o\_no ORDERS.order\_no%TYPE := '&order\_number';

3 BEGIN

4 CalculateAndUpdateDiscount(o\_no);

5 END;

6 /

Enter value for order\_number: OP900

old 2: o\_no ORDERS.order\_no%TYPE := '&order\_number';

new 2: o\_no ORDERS.order\_no%TYPE := 'OP900';

No order found for the given order number.

PL/SQL procedure successfully completed.

SQL>

SQL> -- 3. Write a stored function to display the customer name who ordered highest among the total number of pizzas for a given pizza type.

SQL> CREATE OR REPLACE FUNCTION MostOrderedPizza(pizza\_type\_input IN VARCHAR2)

2 RETURN VARCHAR2 IS cust\_name\_output CUSTOMER.cust\_name%TYPE;

3 BEGIN

4 SELECT c.cust\_name INTO cust\_name\_output

5 FROM CUSTOMER c, ORDERS o ,ORDER\_LIST ol,PIZZA p

6 WHERE ol. pizza\_id = p.pizza\_id AND o. order\_no = ol.order\_no AND c.cust\_id = o.cust\_id AND p.pizza\_type =pizza\_type\_input

7 GROUP BY c.cust\_name

8 ORDER BY SUM(ol.qty) DESC

9 FETCH FIRST 1 ROW ONLY;

10 RETURN cust\_name\_output;

11 EXCEPTION

12 WHEN NO\_DATA\_FOUND THEN RETURN NULL;

13 END;

14 /

Function created.

SQL>

SQL> -- Valid Input

SQL> DECLARE

2 most\_ordered\_customer CUSTOMER.cust\_name%TYPE;

3 pizza\_type\_input varchar2(10) := '&pizza\_type';

4 BEGIN

5 most\_ordered\_customer := MostOrderedPizza(pizza\_type\_input);

6 IF most\_ordered\_customer IS NULL THEN

7 DBMS\_OUTPUT.PUT\_LINE('Invalid Pizza Type.');

8 ELSE

9 DBMS\_OUTPUT.PUT\_LINE(pizza\_type\_input || ' pizza type was most ordered by customer : ' || most\_ordered\_customer);

10 END IF;

11 END;

12 /

Enter value for pizza\_type: italian

old 3: pizza\_type\_input varchar2(10) := '&pizza\_type';

new 3: pizza\_type\_input varchar2(10) := 'italian';

italian pizza type was most ordered by customer : Hari

PL/SQL procedure successfully completed.

SQL>

SQL> -- Invalid Input

SQL> DECLARE

2 most\_ordered\_customer CUSTOMER.cust\_name%TYPE;

3 pizza\_type\_input varchar2(10) := '&pizza\_type';

4 BEGIN

5 most\_ordered\_customer := MostOrderedPizza(pizza\_type\_input);

6 IF most\_ordered\_customer IS NULL THEN

7 DBMS\_OUTPUT.PUT\_LINE('Invalid Pizza Type.');

8 ELSE

9 DBMS\_OUTPUT.PUT\_LINE(pizza\_type\_input || ' pizza type was most ordered by customer : ' || most\_ordered\_customer);

10 END IF;

11 END;

12 /

Enter value for pizza\_type: pal

old 3: pizza\_type\_input varchar2(10) := '&pizza\_type';

new 3: pizza\_type\_input varchar2(10) := 'pal';

Invalid Pizza Type.

PL/SQL procedure successfully completed.

SQL>

SQL> -- 4. Implement Question (2) using a stored function to return the amount to be paid and update the same, for the given order number.

SQL> CREATE OR REPLACE FUNCTION CalculateAndUpdateDiscountFunc(order\_no\_input IN ORDERS.order\_no%TYPE)

2 RETURN NUMBER

3 IS

4 order\_total NUMBER := 0;

5 disc\_rate NUMBER := 0;

6 disc\_amt NUMBER := 0;

7 final\_amt NUMBER := 0;

8 op\_qty ORDER\_LIST.qty%TYPE;

9 amt NUMBER := 0;

10 CURSOR order\_details\_cur IS

11 SELECT p.unit\_price, ol.qty

12 FROM ORDER\_LIST ol

13 JOIN PIZZA p ON ol.pizza\_id = p.pizza\_id

14 WHERE ol.order\_no = order\_no\_input;

15 BEGIN

16 FOR order\_detail IN order\_details\_cur LOOP

17 op\_qty := order\_detail.qty;

18 amt := op\_qty \* order\_detail.unit\_price;

19 order\_total := order\_total + amt;

20 END LOOP;

21 IF order\_total > 2000 AND order\_total < 5000 THEN

22 disc\_rate := 0.05;

23 ELSIF order\_total > 5000 AND order\_total < 10000 THEN

24 disc\_rate := 0.1;

25 ELSIF order\_total > 10000 THEN

26 disc\_rate := 0.2;

27 ELSE

28 disc\_rate := 0;

29 END IF;

30 disc\_amt := order\_total \* disc\_rate;

31 final\_amt := order\_total - disc\_amt;

32 UPDATE ORDERS

33 SET discount\_price = final\_amt

34 WHERE order\_no = order\_no\_input;

35 RETURN final\_amt;

36 EXCEPTION

37 WHEN NO\_DATA\_FOUND THEN

38 DBMS\_OUTPUT.PUT\_LINE('No order found for the given order number.');

39 RETURN NULL;

40 END;

41 /

Function created.

SQL>

SQL> -- Valid Input

SQL> DECLARE

2 order\_no\_input ORDERS.order\_no%TYPE := '&order\_no';

3 final\_amt NUMBER;

4 BEGIN

5 final\_amt := CalculateAndUpdateDiscountFunc(order\_no\_input);

6 IF final\_amt IS NOT NULL THEN

7 DBMS\_OUTPUT.PUT\_LINE('The amount to be paid for Order ' || order\_no\_input || ' after discount is: ' || final\_amt);

8 END IF;

9 END;

10 /

Enter value for order\_no: OP500

old 2: order\_no\_input ORDERS.order\_no%TYPE := '&order\_no';

new 2: order\_no\_input ORDERS.order\_no%TYPE := 'OP500';

The amount to be paid for Order OP500 after discount is: 2375

PL/SQL procedure successfully completed.

SQL>

SQL> -- Invalid Input

SQL> DECLARE

2 order\_no\_input ORDERS.order\_no%TYPE := '&order\_no';

3 final\_amt NUMBER;

4 BEGIN

5 final\_amt := CalculateAndUpdateDiscountFunc(order\_no\_input);

6 IF final\_amt IS NOT NULL THEN

7 DBMS\_OUTPUT.PUT\_LINE('The amount to be paid for Order ' || order\_no\_input || ' after discount is: ' || final\_amt);

8 END IF;

9 END;

10 /

Enter value for order\_no: OP900

old 2: order\_no\_input ORDERS.order\_no%TYPE := '&order\_no';

new 2: order\_no\_input ORDERS.order\_no%TYPE := 'OP900';

The amount to be paid for Order OP900 after discount is: 0

PL/SQL procedure successfully completed.

