

CSD 4203 – Database Programming

Term : 2023S

Student ID : C0930321

Student Name : Shreejana Shrestha

Assignment # 10

Inventory Management Functions

Task 1: Table creation and data insertion

The screenshot displays the Oracle SQL Developer interface. On the left, the 'Connections' pane shows a connection to 'ShreejanaDB'. The main workspace is divided into a 'Worksheet' and a 'Query Builder'. The 'Worksheet' contains the following SQL code:

```
CREATE TABLE products (  
    product_id NUMBER PRIMARY KEY,  
    name VARCHAR2(100),  
    stock_quantity NUMBER,  
    category VARCHAR2(50),  
    price NUMBER(10,2)  
);  
  
DESC products;
```

Below the SQL editor, the 'Script Output' pane shows the execution results. A message indicates 'Task completed in 0.084 seconds'. Below this, a table displays the structure of the 'products' table:

Name	Null?	Type
PRODUCT_ID	NOT NULL	NUMBER
NAME		VARCHAR2(100)
STOCK_QUANTITY		NUMBER
CATEGORY		VARCHAR2(50)
PRICE		NUMBER(10,2)

Connections

SQL Worksheet History

Worksheet Query Builder

```
select * from products;
```

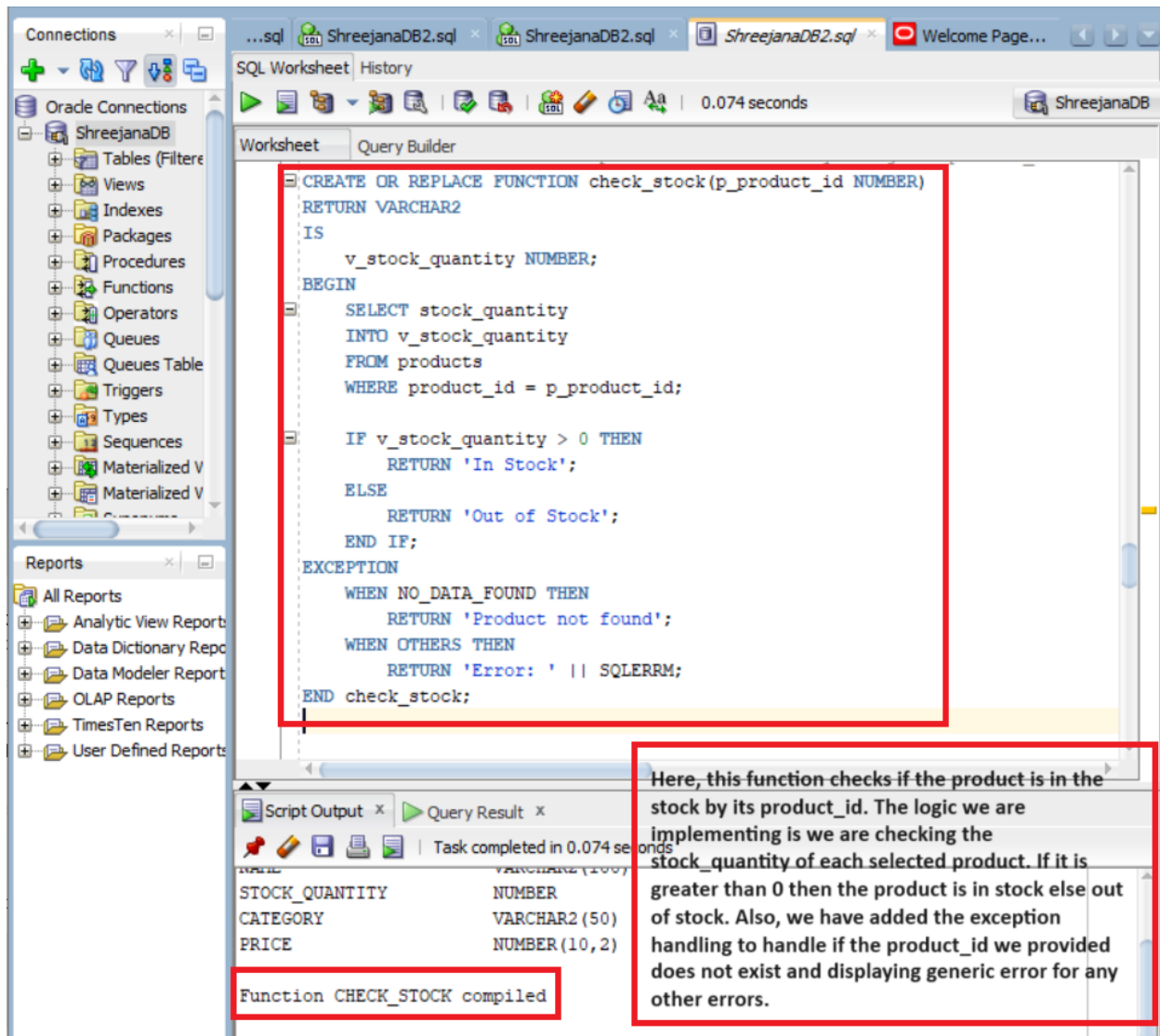
Script Output x Query Result x

SQL | All Rows Fetched: 5 in 0.003 seconds

PRODUCT_ID	NAME	STOCK_QUANTITY	CATEGORY	PRICE
1	1 Laptop	50	Electronics	999.99
2	2 Smartphone	150	Electronics	699.99
3	3 Tablet	80	Electronics	399.99
4	4 Headphones	200	Accessories	49.99
5	5 Smartwatch	120	Wearables	199.99

Task 2: Functions creation

check_stock



The screenshot displays the Oracle SQL Developer interface. On the left, the 'Connections' pane shows 'ShreejanaDB' selected. The main window is titled 'SQL Worksheet' and contains the following PL/SQL code for creating a function:

```
CREATE OR REPLACE FUNCTION check_stock(p_product_id NUMBER)
RETURN VARCHAR2
IS
    v_stock_quantity NUMBER;
BEGIN
    SELECT stock_quantity
    INTO v_stock_quantity
    FROM products
    WHERE product_id = p_product_id;

    IF v_stock_quantity > 0 THEN
        RETURN 'In Stock';
    ELSE
        RETURN 'Out of Stock';
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 'Product not found';
    WHEN OTHERS THEN
        RETURN 'Error: ' || SQLERRM;
END check_stock;
```

Below the code editor, the 'Script Output' pane shows the message: 'Function CHECK_STOCK compiled'. To the right of the script output, a text box explains the function's logic:

Here, this function checks if the product is in the stock by its product_id. The logic we are implementing is we are checking the stock_quantity of each selected product. If it is greater than 0 then the product is in stock else out of stock. Also, we have added the exception handling to handle if the product_id we provided does not exist and displaying generic error for any other errors.

reorder_level

The screenshot displays the SQL Developer interface with the following components:

- Connections:** Shows the 'ShreejanaDB' connection.
- SQL Worksheet:** Contains the PL/SQL code for the `reorder_level` function.
- Script Output:** Displays the message 'Function REORDER_LEVEL compiled'.

PL/SQL Code:

```
-- function that calculates the reorder level for a product based on
-- sales rate and lead time
CREATE OR REPLACE FUNCTION reorder_level(p_product_id NUMBER)
RETURN NUMBER
IS
-- Assumption:
--     a constant sales rate
--     a constant lead time
    v_sales_rate NUMBER := 10;
    v_lead_time NUMBER := 5;
    v_stock_quantity NUMBER;
BEGIN
    SELECT stock_quantity
    INTO v_stock_quantity
    FROM products
    WHERE product_id = p_product_id;

    RETURN v_sales_rate * v_lead_time;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN -1; -- the product was not found
    WHEN OTHERS THEN
        RETURN -1; -- error
END reorder_level;
```

Annotations:

- Function Definition:** Here, we have created a function that calculates the reorder level for a product based on sales rate and lead time.
- Query Execution:** We get the products' quantity from the table for the given product_id then calculated the reorder level.
- Exception Handling:** We have also added the exception handling that will handle the error when the given product with product_id we provided does not exist and handle any other errors as well.

Line 230 Column 19 | Insert | Modified | Windows: CI

format_product_info

The screenshot displays the SQL Developer interface with the 'ShreejanaDB2.sql' file open. The main window shows the SQL Worksheet with the following code:

```
-- function that formats product details as a readable string
CREATE OR REPLACE FUNCTION format_product_info(p_product_id NUMBER)
RETURN VARCHAR2
IS
    v_product_info VARCHAR2(500);
    v_name VARCHAR2(100);
    v_category VARCHAR2(50);
    v_stock_quantity NUMBER;
    v_price NUMBER(10, 2);
BEGIN
    SELECT name, category, stock_quantity, price
    INTO v_name, v_category, v_stock_quantity, v_price
    FROM products
    WHERE product_id = p_product_id;

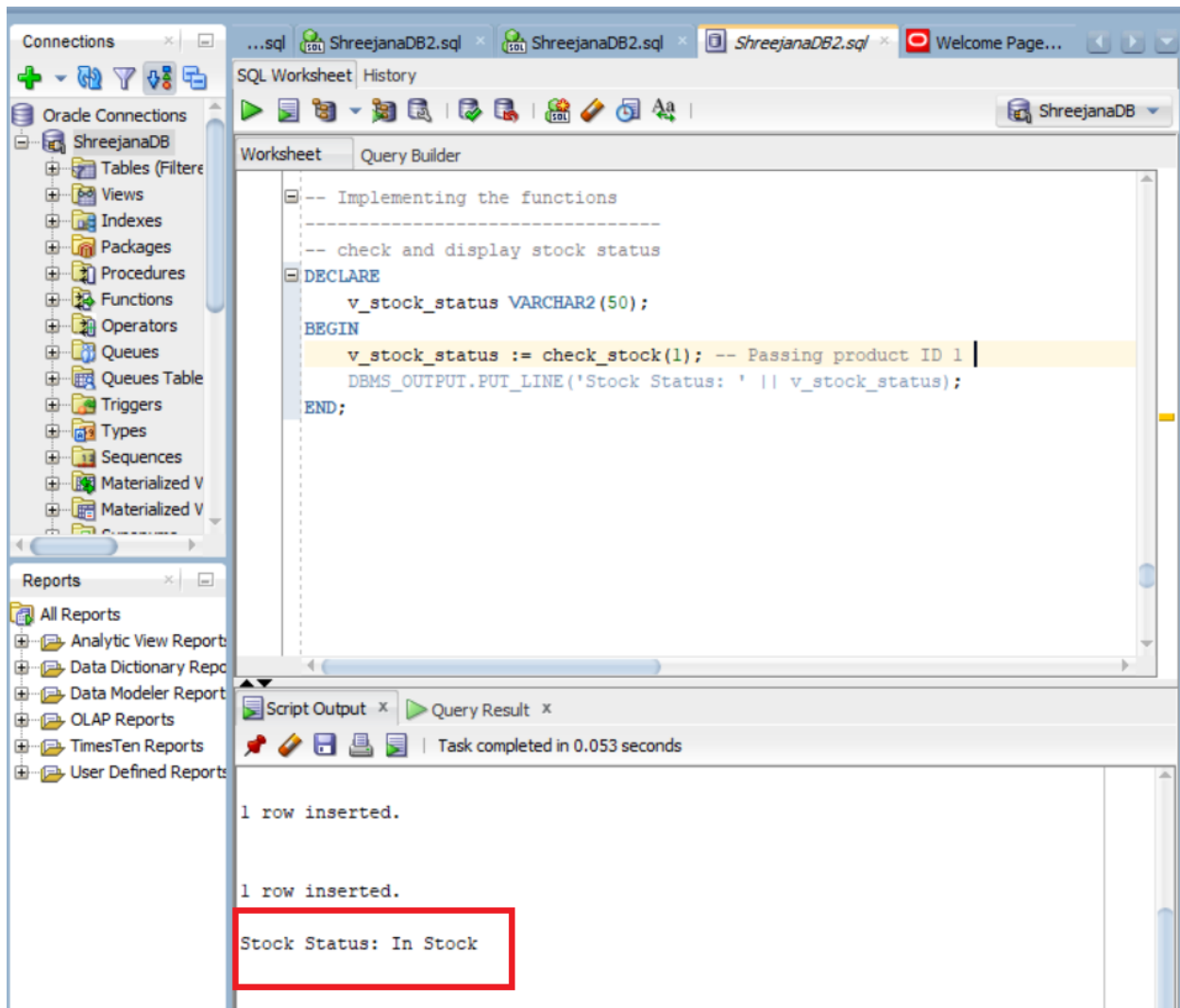
    v_product_info := 'Product ID: ' || p_product_id || ', Name: ' || v_name;
    RETURN v_product_info;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 'Product not found';
    WHEN OTHERS THEN
        RETURN 'Error: ' || SQLERRM;
END format_product_info;
```

The 'Script Output' pane at the bottom shows the message: 'Function FORMAT_PRODUCT_INFO compiled'.

Here, this function formats the product details as a readable string. At first, we are selecting all the details of the product from the products table for the given product_id using the select query. After that we have formatted the product information into a readable string. If the product is not found, we are showing product not found message through exception handling.

Task 3: Implementation

implementation of check_stock function



The screenshot displays the Oracle SQL Developer interface. On the left, the 'Connections' pane shows 'ShreejanaDB' selected. The main window is titled 'SQL Worksheet' and contains the following PL/SQL code:

```
-- Implementing the functions
-- check and display stock status
DECLARE
    v_stock_status VARCHAR2(50);
BEGIN
    v_stock_status := check_stock(1); -- Passing product ID 1
    DBMS_OUTPUT.PUT_LINE('Stock Status: ' || v_stock_status);
END;
```

Below the code editor, the 'Script Output' pane shows the execution results:

```
1 row inserted.

1 row inserted.

Stock Status: In Stock
```

The text 'Stock Status: In Stock' is highlighted with a red rectangular box.

Implementation of reorder_level function

The screenshot displays the SQL Developer interface with the following components:

- Connections Panel:** Shows 'ShreejanaDB' selected under 'Oracle Connections'.
- SQL Worksheet:** Contains a PL/SQL procedure named 'reorder_level'.
- Script Output:** Shows the execution result 'Reorder Level: 50' highlighted with a red box.
- Status Bar:** Indicates 'Task completed in 0.068 seconds' and 'PL/SQL procedure successfully completed.'

PL/SQL Code:

```
-- calculate and display reorder level
DECLARE
    v_reorder_level NUMBER;
BEGIN
    v_reorder_level := reorder_level(1); -- Passing product ID 1
    IF v_reorder_level != -1 THEN
        DBMS_OUTPUT.PUT_LINE('Reorder Level: ' || v_reorder_level);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Error: Could not calculate reorder level');
    END IF;
END;
```

Execution Output:

```
Reorder Level: 50
```

Status: PL/SQL procedure successfully completed.

Implementation of format_product_info function

The screenshot displays the SQL Developer interface with the following components:

- Connections:** Shows a tree view of the database schema for 'ShreejanaDB', including Tables, Views, Indexes, Packages, Procedures, Functions, Operators, Queues, Queues Table, Triggers, Types, Sequences, and Materialized Views.
- SQL Worksheet:** Contains the following PL/SQL code:

```
-- format and display product details
DECLARE
    v_product_info VARCHAR2(500);
BEGIN
    v_product_info := format_product_info(1); -- Passing product ID 1
    DBMS_OUTPUT.PUT_LINE('Product Info: ' || v_product_info);
END;
```
- Script Output:** Shows the execution results:

```
Reorder Level: 50
PL/SQL procedure successfully completed.
Product Info: Product ID: 1, Name: Laptop, Category: Electronics, Stock Quantity: 50, Price: $999.9
PL/SQL procedure successfully completed.
```

The output line 'Product Info: Product ID: 1, Name: Laptop, Category: Electronics, Stock Quantity: 50, Price: \$999.9' is highlighted with a red border.