TEXT

```
PROCEDURE get_all_customers AS
    CURSOR c_customers IS
        SELECT id, full_name, username, email, contact, password FROM customers;
BEGIN
DBMS_OUTPUT.PUT_LINE('--------------------------------------------------------------------------------');
    DBMS_OUTPUT.PUT_LINE(' ID  | FULL NAME      | USERNAME   | EMAIL                | CONTACT     | PASSWORD ');
DBMS_OUTPUT.PUT_LINE('--------------------------------------------------------------------------------');
    FOR rec IN c_customers LOOP
        DBMS_OUTPUT.PUT_LINE(
            LPAD(rec.id, 4) || ' | ' ||
           RPAD(SUBSTR(rec.full_name, 1, 15), 15) || ' | ' ||
           RPAD(SUBSTR(rec.username, 1, 10), 10) || ' | ' ||
           RPAD(SUBSTR(rec.email, 1, 20), 20) || ' | ' ||
            RPAD(rec.contact, 12) || ' | ' ||
          RPAD(SUBSTR(rec.password, 1, 10), 10) -- Truncate long passwords for alignment
        );
    END LOOP;
DBMS_OUTPUT.PUT_LINE('--------------------------------------------------------------------------------');
END get_all_customers;
PROCEDURE get_all_products(p_cursor OUT SYS_REFCURSOR) AS
BEGIN
    OPEN p_cursor FOR
    SELECT * FROM products;
END get_all_products;
PROCEDURE get_customer_by_username (
    p_username   IN customers.username%TYPE,
    p_id         OUT customers.id%TYPE,
    p_full_name  OUT customers.full_name%TYPE,
    p_email      OUT customers.email%TYPE,
    p_contact    OUT customers.contact%TYPE,
    p_password   OUT customers.password%TYPE
) AS
BEGIN
    SELECT id, full_name, email, contact, password
    INTO p_id, p_full_name, p_email, p_contact, p_password
    FROM customers
    WHERE username = p_username;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        p_id := NULL;
        p_full_name := NULL;
        p_email := NULL;
        p_contact := NULL;
        p_password := NULL;
       DBMS_OUTPUT.PUT_LINE('Customer not found.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
```

TEXT

```
PROCEDURE get_order_id(cursor OUT SYS_REFCURSOR) AS
BEGIN
    OPEN cursor FOR
    SELECT order_id
    FROM orders
    WHERE status = 'Accepted';
END;
PROCEDUREget_products_by_category(
    p_category IN VARCHAR2,
    p_cursor OUT SYS_REFCURSOR
) IS
BEGIN
    OPEN p_cursor FOR
       SELECT p.id, p.product_name, p.category, p.price, p.stock_quantity, s.supplier_name, p.product_image
        FROM products p
        JOIN suppliers s ON p.supplier_id = s.id
        WHERE p.category = p_category;
ENDget_products_by_category;
PROCEDURE get_product_by_id (
    p_id IN NUMBER,
    p_name OUT VARCHAR2,
    p_category OUT VARCHAR2,
    p_price OUT NUMBER,
    p_stock OUT NUMBER
) AS
BEGIN
    SELECT PRODUCT_NAME, CATEGORY, PRICE, STOCK_QUANTITY
    INTO p_name, p_category, p_price, p_stock
    FROM products
    WHERE ID = p_id;
END;
PROCEDURE get_suppliers_json (p_json OUT VARCHAR2) IS
  -- Declare a cursor to fetch supplier data
 CURSOR suppliers_cursor IS
    SELECT id, supplier_name, contact_number, email, address
    FROM suppliers;

  -- Declare variables to hold the data
 supplier_record suppliers_cursor%ROWTYPE;

  -- Declare a variable to hold the JSON result
 json_result VARCHAR2(4000) := '[';
BEGIN
  -- Open the cursor and fetch each row
 FOR supplier_record IN suppliers_cursor LOOP
   json_result := json_result || '{"id": "' || supplier_record.id || '", ' ||
                            '"supplier_name": "' || supplier_record.supplier_name || '", ' ||
                            '"contact_number": "' || supplier_record.contact_number || '", ' ||
                             '"email": "' || supplier_record.email || '", ' ||
```

TEXT

```
                        '"address": "' || supplier_record.address || '"},';
  END LOOP;

  -- Remove the last comma and close the JSON array
  json_result := RTRIM(json_result, ',') || ']';

  -- Assign the JSON result to the OUT parameter
  p_json := json_result;
END;
PROCEDUREget_supplier_by_id(
    p_id IN NUMBER,
    p_supplier_name OUT VARCHAR2,
    p_contact_number OUT VARCHAR2,
    p_email OUT VARCHAR2,
    p_address OUT VARCHAR2
) IS
BEGIN
    SELECT supplier_name, contact_number, email, address
    INTO p_supplier_name, p_contact_number, p_email, p_address
    FROM suppliers
    WHERE id = p_id;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        p_supplier_name := NULL;
        p_contact_number := NULL;
        p_email := NULL;
        p_address := NULL;
    WHEN OTHERS THEN
        p_supplier_name := NULL;
        p_contact_number := NULL;
        p_email := NULL;
        p_address := NULL;
END;
PROCEDURE handle_order(p_order_id IN NUMBER, p_action IN VARCHAR2) IS
BEGIN
    IF p_action = 'accept' THEN
        -- Call accept_order procedure
        accept_order(p_order_id);
    ELSIF p_action = 'cancel' THEN
        -- Call cancel_order procedure
        cancel_order(p_order_id);
    ELSE
        -- Handle invalid action
        RAISE_APPLICATION_ERROR(-20001, 'Invalid action: ' || p_action);
    END IF;
END handle_order;
PROCEDURE insert_delivery (
    p_order_id IN NUMBER,
    p_delivery_date IN DATE,
```

TEXT

```
    p_shipping_date IN DATE
)
AS
BEGIN
    -- Insert the delivery data into the deliveries table
    INSERT INTO delivery (order_id, delivery_date, shipping_date)
    VALUES (p_order_id, p_delivery_date, p_shipping_date);

    -- Update the order status in the orders table
    UPDATE orders
    SET status = 'Your order will be shipped on ' || TO_CHAR(p_shipping_date, 'YYYY-MM-DD')
    WHERE order_id = p_order_id;

    COMMIT;
END insert_delivery;
PROCEDURE login_user (
    p_username IN VARCHAR2,
    p_password IN VARCHAR2,
    p_status OUT VARCHAR2
) AS
BEGIN
    -- Dummy authentication logic (Replace with real DB query)
    IF p_username = 'admin' AND p_password = 'password' THEN
        p_status := 'Login successful';
    ELSE
        p_status := 'Invalid username or password';
    END IF;
END;
PROCEDURE process_order (
    p_customer_id IN NUMBER,
    p_total_amount IN NUMBER,
    p_products IN product_array -- Using the VARRAY type for products
)
IS
    v_order_id NUMBER;
BEGIN
    -- Insert into the orders table
    INSERT INTO orders (customer_id, total_amount, order_date, status)
    VALUES (p_customer_id, p_total_amount, SYSDATE, 'Order Placed')
    RETURNING order_id INTO v_order_id;

    -- Loop through each product in the order_items array
    FOR i IN 1..p_products.COUNT LOOP
        -- Insert each item into the order_items table
        INSERT INTO order_items (order_id, product_id, quantity)
       VALUES (v_order_id, p_products(i).product_id, p_products(i).quantity);
    END LOOP;

    COMMIT; -- Commit the transaction
```

TEXT

```
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END process_order;
PROCEDURE process_order_action (
    p_order_id     IN NUMBER,
    p_status       IN VARCHAR2,
    p_total_amount IN NUMBER DEFAULT NULL
)
AS
BEGIN
    -- Update order status
    UPDATE ORDERS
    SET STATUS = p_status
    WHERE ORDER_ID = p_order_id;

    IF p_status = 'Accepted' THEN
        -- Insert into payments
        INSERT INTO PAYMENTS (PAYMENT_ID, ORDER_ID, AMOUNT, PAYMENT_DATE)
        VALUES (PAYMENTS_SEQ.NEXTVAL, p_order_id, p_total_amount, SYSDATE);

    ELSIF p_status = 'Cancelled' THEN
        -- Delete payment records for the cancelled order
        DELETE FROM PAYMENTS
        WHERE ORDER_ID = p_order_id;

        -- Restore stock quantities
        FOR prod IN (
            SELECT PRODUCT_ID, QUANTITY
            FROM ORDER_ITEMS
            WHERE ORDER_ID = p_order_id
        ) LOOP
            UPDATE PRODUCTS
            SET STOCK_QUANTITY = STOCK_QUANTITY + prod.QUANTITY
            WHERE ID = prod.PRODUCT_ID;
        END LOOP;
    END IF;

    COMMIT;
END;
PROCEDUREregister_customer(
    p_full_name IN customers.full_name%TYPE,
    p_username  IN customers.username%TYPE,
    p_email     IN customers.email%TYPE,
    p_contact   IN customers.contact%TYPE,
    p_password  IN customers.password%TYPE,
    p_status    OUT VARCHAR2
```

TEXT

```
) AS
    v_count NUMBER;
BEGIN
    -- Check if the username or email already exists
    SELECT COUNT(*) INTO v_count FROM customers WHERE username = p_username OR email = p_email;

    IF v_count > 0 THEN
        p_status := 'Username or email already exists';
    ELSE
        -- Insert new customer record
        INSERT INTO customers (full_name, username, email, contact, password)
        VALUES (p_full_name, p_username, p_email, p_contact, p_password);

        p_status := 'Registration successful';
        COMMIT;
    END IF;
ENDregister_customer;
PROCEDUREupdate_customer(
    p_customer_id IN NUMBER,
    p_full_name IN VARCHAR2,
    p_username IN VARCHAR2,
    p_email IN VARCHAR2,
    p_contact IN VARCHAR2
) IS
    -- Exception handling for validation
    e_invalid_data EXCEPTION;
BEGIN
    -- Input validation
    IF p_full_name IS NULL OR p_username IS NULL OR p_email IS NULL OR p_contact IS NULL THEN
        RAISE e_invalid_data;
    END IF;

    -- Update query
    UPDATE customers
    SET full_name = p_full_name,
        username = p_username,
        email = p_email,
        contact = p_contact
    WHERE id = p_customer_id;

    -- Check if any rows were updated
    IF SQL%ROWCOUNT = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Customer not found or update failed.');
    END IF;

    -- Commit changes
    COMMIT;

    -- Success message
```

TEXT

```
   DBMS_OUTPUT.PUT_LINE('Customer details updated successfully.');
EXCEPTION
    WHEN e_invalid_data THEN
       DBMS_OUTPUT.PUT_LINE('All fields are required.');
    WHEN OTHERS THEN
       DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
        ROLLBACK;
END update_customer;
PROCEDURE update_customer_by_id (
    p_id         IN customers.id%TYPE,
   p_full_name  IN customers.full_name%TYPE,
   p_username   IN customers.username%TYPE,
    p_email      IN customers.email%TYPE,
   p_contact    IN customers.contact%TYPE,
   p_password   IN customers.password%TYPE
) AS
BEGIN
    UPDATE customers
   SET full_name = p_full_name,
       username = p_username,
       email = p_email,
       contact = p_contact,
       password = p_password
    WHERE id = p_id;

    COMMIT;

EXCEPTION
    WHEN OTHERS THEN
       DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
        ROLLBACK;
END;
PROCEDUREupdate_order_statusES(
   p_order_id IN NUMBER,
   p_action IN VARCHAR2,
   p_items IN SYS.ODCIVARCHAR2LIST -- This is a collection type that will hold product details for "cancel"
) IS
   v_total_amount NUMBER;
BEGIN
   -- Check action and perform respective logic
   IF p_action = 'acceptOrder' THEN
       -- Step 1: Update the order status
       UPDATE ORDERS
       SET STATUS = 'Your order has been accepted'
       WHERE ORDER_ID = p_order_id;

       -- Step 2: Get the total amount of the order
       SELECT TOTAL_AMOUNT INTO v_total_amount
        FROM ORDERS
```

TEXT

```
       WHERE ORDER_ID = p_order_id;

       -- Step 3: Insert payment record
       INSERT INTO PAYMENTS (ORDER_ID, PAYMENT_DATE, AMOUNT)
       VALUES (p_order_id, SYSDATE, v_total_amount);

   ELSIF p_action = 'cancelOrder' THEN
       -- Step 1: Update the order status
       UPDATE ORDERS
       SET STATUS = 'Order has been cancelled'
       WHERE ORDER_ID = p_order_id;

       -- Step 2: Restore stock for each product in the order
       FOR i IN 1..p_items.COUNT LOOP
           DECLARE
               v_product_id NUMBER;
               v_quantity NUMBER;
           BEGIN
               -- Get product ID and quantity from the item list
              v_product_id := TO_NUMBER(SUBSTR(p_items(i), 1, INSTR(p_items(i), ',') - 1));  -- Assuming fo...
              v_quantity := TO_NUMBER(SUBSTR(p_items(i), INSTR(p_items(i), ',') + 1));

               -- Update the stock quantity for the product
               UPDATE PRODUCTS
              SET STOCK_QUANTITY = STOCK_QUANTITY + v_quantity
               WHERE ID = v_product_id;
           END;
       END LOOP;
   END IF;

   COMMIT;
EXCEPTION
   WHEN OTHERS THEN
       ROLLBACK;
       RAISE;
ENDupdate_order_statusES;
PROCEDUREupdate_product(
   p_product_id       IN NUMBER,
   p_product_name     IN VARCHAR2,
   p_category         IN VARCHAR2,
   p_price            IN NUMBER,
   p_stock_quantity   IN NUMBER,
   p_supplier_id      IN NUMBER,
   p_product_image    IN VARCHAR2  -- This parameter will be NULL if no image is provided
) IS
BEGIN
   -- Update the product record
   UPDATE products
   SET
```

TEXT

```
        PRODUCT_NAME = p_product_name,
        CATEGORY = p_category,
        PRICE = p_price,
        STOCK_QUANTITY = p_stock_quantity,
        SUPPLIER_ID = p_supplier_id,
        CREATED_AT = CURRENT_TIMESTAMP
    WHERE ID = p_product_id;

    -- Only update the image if a new image is provided
    IF p_product_image IS NOT NULL THEN
        UPDATE products
        SET
            PRODUCT_IMAGE = p_product_image
        WHERE ID = p_product_id;
    END IF;

    COMMIT;

   DBMS_OUTPUT.PUT_LINE('Product updated successfully');
EXCEPTION
    WHEN OTHERS THEN
        -- Handle any errorsA
        ROLLBACK;
       DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END update_product;
```