TEXT

```
PROCEDURE accept_order(p_order_id IN NUMBER) IS
BEGIN
    -- Update order status to "Accepted"
    UPDATE orders
    SET status = 'Your order has been accepted'
    WHERE order_id = p_order_id;

    -- Insert payment record (payment_id will be generated using the sequence)
    INSERT INTO payments (payment_id, order_id, amount, payment_date)
    VALUES (payment_sequence.nextval, p_order_id,
            (SELECT total_amount FROM orders WHERE order_id = p_order_id), SYSDATE);

    COMMIT;
END accept_order;
PROCEDURE add_product (
    p_product_name IN VARCHAR2,
    p_category IN VARCHAR2,
    p_price IN NUMBER,
    p_stock IN NUMBER,
    p_supplier_id IN NUMBER,
    p_image IN BLOB
) IS
BEGIN
    INSERT INTO products (product_name, category, price, stock_quantity, supplier_id, product_image)
    VALUES (p_product_name, p_category, p_price, p_stock, p_supplier_id, p_image);
    COMMIT;
END add_product;
PROCEDURE add_supplier(
    p_supplier_name IN VARCHAR2,
    p_contact_number IN VARCHAR2,
    p_email IN VARCHAR2,
    p_address IN VARCHAR2
) AS
BEGIN
    -- Check if a supplier with the same contact number or email already exists
    DECLARE
        v_count INTEGER;
    BEGIN
        -- Check if contact number already exists
        SELECT COUNT(*) INTO v_count
        FROM suppliers
        WHERE contact_number = p_contact_number OR email = p_email;

        IF v_count > 0 THEN
            RAISE_APPLICATION_ERROR(-20001, 'A supplier with this contact number or email already exists.');
        ELSE
            -- If no existing record found, insert the new supplier
```

TEXT

```
            INSERT INTO suppliers (supplier_name, contact_number, email, address)
            VALUES (p_supplier_name, p_contact_number, p_email, p_address);
        END IF;
    END;
END add_supplier;
PROCEDURE cancel_order(p_order_id IN NUMBER) IS
BEGIN
    -- Update order status to "Cancelled"
    UPDATE orders
    SET status = 'Your order has been cancelled'
    WHERE order_id = p_order_id;

    -- Delete associated payment record
    DELETE FROM payments WHERE order_id = p_order_id;

    -- Restore stock for the products associated with the order
    FOR rec IN (SELECT product_id, quantity FROM order_items WHERE order_id = p_order_id) LOOP
        -- Update stock in the products table
        UPDATE products
        SET stock_quantity = stock_quantity + rec.quantity
        WHERE products.id = rec.product_id;
    END LOOP;

    COMMIT;
END cancel_order;
PROCEDURE check_contact_uniqueness (
    p_contact IN VARCHAR2,
    p_customer_id IN NUMBER,
    p_is_unique OUT VARCHAR2
) IS
    v_contact_exists NUMBER;
BEGIN
    -- Check if the contact number already exists for another customer
    SELECT COUNT(*) INTO v_contact_exists
    FROM customers
    WHERE contact = p_contact AND id != p_customer_id;

    IF v_contact_exists > 0 THEN
        p_is_unique := 'contact_exists';
    ELSE
        p_is_unique := 'unique';
    END IF;
END;
PROCEDURE check_customer_uniqueness (
    p_email IN VARCHAR2,
    p_contact IN VARCHAR2,
    p_customer_id IN NUMBER,
```

TEXT

```
    p_is_unique OUT VARCHAR2
) IS
    v_email_exists NUMBER;
    v_contact_exists NUMBER;
BEGIN
    -- Check if the email already exists for another customer
    SELECT COUNT(*) INTO v_email_exists
    FROM customers
    WHERE email = p_email AND id != p_customer_id;

    -- Check if the contact number already exists for another customer
    SELECT COUNT(*) INTO v_contact_exists
    FROM customers
    WHERE contact = p_contact AND id != p_customer_id;

    IF v_email_exists > 0 THEN
        p_is_unique := 'email_exists';
    ELSIF v_contact_exists > 0 THEN
        p_is_unique := 'contact_exists';
    ELSE
        p_is_unique := 'unique';
    END IF;
END;
PROCEDURE check_email_uniqueness (
    p_email IN VARCHAR2,
    p_customer_id IN NUMBER,
    p_is_unique OUT VARCHAR2
) IS
    v_email_exists NUMBER;
BEGIN
    -- Check if the email already exists for another customer
    SELECT COUNT(*) INTO v_email_exists
    FROM customers
    WHERE email = p_email AND id != p_customer_id;

    IF v_email_exists > 0 THEN
        p_is_unique := 'email_exists';
    ELSE
        p_is_unique := 'unique';
    END IF;
END;
PROCEDURE check_supplier_uniqueness(
    p_supplier_id IN NUMBER,
    p_contact_number IN VARCHAR2,
    p_email IN VARCHAR2,
    p_is_contact_unique OUT NUMBER,
    p_is_email_unique OUT NUMBER
```

TEXT

```
) AS
BEGIN
    -- Check if the contact number is unique (excluding the current supplier)
    SELECT COUNT(*)
    INTO p_is_contact_unique
    FROM suppliers
    WHERE contact_number = p_contact_number
    AND id != p_supplier_id;

    -- Check if the email is unique (excluding the current supplier)
    SELECT COUNT(*)
    INTO p_is_email_unique
    FROM suppliers
    WHERE email = p_email
    AND id != p_supplier_id;

    -- Determine if the contact number is unique (0 = not unique, 1 = unique)
    IF p_is_contact_unique > 0 THEN
        p_is_contact_unique := 0;  -- Not unique
    ELSE
        p_is_contact_unique := 1;  -- Unique
    END IF;

    -- Determine if the email is unique (0 = not unique, 1 = unique)
    IF p_is_email_unique > 0 THEN
        p_is_email_unique := 0;  -- Not unique
    ELSE
        p_is_email_unique := 1;  -- Unique
    END IF;
END check_supplier_uniqueness;
PROCEDURE delete_product (p_product_id IN NUMBER) IS
BEGIN
    -- Delete product from the products table
    DELETE FROM products
    WHERE id = p_product_id;

    -- Commit the transaction
    COMMIT;

    -- Output message to indicate successful deletion
    DBMS_OUTPUT.PUT_LINE('Product with ID ' || p_product_id || ' has been deleted.');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        -- If no product is found with the provided ID, display a message
        DBMS_OUTPUT.PUT_LINE('No product found with ID ' || p_product_id);
    WHEN OTHERS THEN
        -- Handle other exceptions
```

TEXT

```
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
        ROLLBACK; -- Rollback the transaction in case of error
END delete_product;
PROCEDURE delete_supplier(p_supplier_id IN NUMBER) IS
BEGIN
    -- Begin transaction
    SAVEPOINT delete_supplier_point;

    -- Check if supplier exists
    DECLARE
        v_count NUMBER;
    BEGIN
        SELECT COUNT(*) INTO v_count
        FROM suppliers
        WHERE id = p_supplier_id;

        IF v_count = 0 THEN
            RAISE_APPLICATION_ERROR(-20001, 'Supplier not found!');
        END IF;
    END;

    -- Delete the supplier
    DELETE FROM suppliers WHERE id = p_supplier_id;

    -- Commit the transaction
    COMMIT;

    -- Output success message
    DBMS_OUTPUT.PUT_LINE('Supplier deleted successfully');
EXCEPTION
    WHEN OTHERS THEN
        -- Rollback if an error occurs
        ROLLBACK TO delete_supplier_point;
        RAISE;
END delete_supplier;
PROCEDURE edit_supplier(
    p_supplier_id IN NUMBER,
    p_supplier_name IN VARCHAR2,
    p_contact_number IN VARCHAR2,
    p_email IN VARCHAR2,
    p_address IN VARCHAR2
) IS
BEGIN
    UPDATE suppliers
    SET supplier_name = p_supplier_name,
        contact_number = p_contact_number,
        email = p_email,
```

TEXT

```
        address = p_address
    WHERE id = p_supplier_id;
    COMMIT;
END edit_supplier;
PROCEDURE fetch_suppliers (
    p_supplier_name IN VARCHAR2,
    p_suppliers OUT SYS_REFCURSOR
)
IS
BEGIN
    OPEN p_suppliers FOR
        SELECT id, supplier_name, contact_number, email, address
        FROM suppliers
        WHERE supplier_name LIKE '%' || p_supplier_name || '%';

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        -- Handle no data found case
        NULL;  -- No action needed for this example
    WHEN OTHERS THEN
        -- Log the error
        DBMS_OUTPUT.PUT_LINE('Error occurred: ' || SQLERRM);
END fetch_suppliers;
```