ANS 2

```
CREATE PROCEDURE UpdateOrderDetails
    @OrderID INT,
    @ProductID INT,
    @UnitPrice DECIMAL(18, 2) = NULL,
    @Quantity INT = NULL,
    @Discount DECIMAL(4, 2) = NULL
AS
BEGIN
    UPDATE OrderDetails
    SET
        UnitPrice = ISNULL(@UnitPrice, UnitPrice),
        Quantity = ISNULL(@Quantity, Quantity),
        Discount = ISNULL(@Discount, Discount)
    WHERE
        OrderID = @OrderID AND
        ProductID = @ProductID;

    DECLARE @OldQuantity INT;
    SELECT @OldQuantity = Quantity
    FROM OrderDetails
    WHERE OrderID = @OrderID AND ProductID = @ProductID;

    IF @Quantity IS NOT NULL
    BEGIN
        UPDATE Products
        SET UnitsInStock = UnitsInStock - (@Quantity - @OldQuantity)
        WHERE ProductID = @ProductID;
    END
END;
```

ANS 4

```
CREATE PROCEDURE DeleteOrderDetail(IN p_OrderID INT, IN p_ProductID INT)
BEGIN
    DECLARE num_rows INT;
    SELECT COUNT(*)
    INTO num_rows
    FROM OrderDetails
    WHERE OrderID = p_OrderID;
    IF num_rows = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: OrderID does not exist.', MYSQL_ERRNO = -1;
        LEAVE;
    END IF;

    SELECT COUNT(*)
    INTO num_rows
    FROM OrderDetails
    WHERE OrderID = p_OrderID AND ProductID = p_ProductID;
    IF num_rows = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: ProductID does not exist for the given OrderID.', MYSQL_ERRNO = -
1;
        LEAVE;
    END IF;

    -- Delete the row from OrderDetails table
    DELETE FROM OrderDetails
    WHERE OrderID = p_OrderID AND ProductID = p_ProductID;

END
```

Ans 5

```
from datetime import datetime
def format_date(dt):
    return dt.strftime('%m/%d/%Y')
# Example usage
example_dt = datetime.strptime('2006-11-21 23:34:05.920', '%Y-%m-%d %H:%M:%S.%f')
formatted_date = format_date(example_dt)
print(formatted_date)
# Output: 11/21/2006
```

Ans 6

```
CREATE FUNCTION FormatDateYYYYMMDD (@inputDate DATE)
RETURNS VARCHAR(8)
AS
BEGIN
    RETURN CONVERT(VARCHAR(8), @inputDate, 112);
END
GO
```

Ans 7

```
CREATE VIEW vwCustomerOrders AS
SELECT
    c.CompanyName,
    o.OrderID,
    o.OrderDate,
    od.ProductID,
    p.ProductName,
    od.Quantity,
    od.UnitPrice,
    (od.Quantity * od.UnitPrice) AS TotalPrice
FROM
```

```
    Customers c
    JOIN Orders o ON c.CustomerID = o.CustomerID
    JOIN OrderDetails od ON o.OrderID = od.OrderID
    JOIN Products p ON od.ProductID = p.ProductID;
```

ANS 8

```
CREATE VIEW vwCustomerOrdersYesterday AS
SELECT
    c.CompanyName,
    o.OrderId,
    o.OrderDate,
    p.ProductId,
    p.ProductName,
    od.Quantity,
    od.UnitPrice,
    od.Quantity * od.UnitPrice AS TotalPrice
FROM
    Customers c
JOIN
    Orders o ON c.CustomerId = o.CustomerId
JOIN
    OrderDetails od ON o.OrderId = od.OrderId
JOIN
    Products p ON od.ProductId = p.ProductId
WHERE
    o.OrderDate = CAST(CONVERT(VARCHAR, DATEADD(DAY, -1, GETDATE()), 101) AS DATE);
```

ANS 9

```
CREATE VIEW MyProducts AS
SELECT
    p.ProductID,
    p.ProductName,
```

```
    p.QuantityPerUnit,
    p.UnitPrice,
    s.CompanyName AS SupplierName,
    c.CategoryName
FROM
    Products p
JOIN
    Suppliers s ON p.SupplierID = s.SupplierID
JOIN
    Categories c ON p.CategoryID = c.CategoryID
WHERE
    p.Discontinued = 0;
```

ANS 10

```
IF OBJECT_ID('trg_InsteadOfDeleteOrder', 'TR') IS NOT NULL
DROP TRIGGER trg_InsteadOfDeleteOrder;
GO
CREATE TRIGGER trg_InsteadOfDeleteOrder
ON Orders
INSTEAD OF DELETE
AS
BEGIN
    DELETE FROM [Order Details]
    WHERE OrderID IN (SELECT OrderID FROM DELETED);
    DELETE FROM Orders
    WHERE OrderID IN (SELECT OrderID FROM DELETED);
END;
```

ANS 11

```
IF OBJECT_ID('trg_InsteadOfInsertOrderDetails', 'TR') IS NOT NULL
DROP TRIGGER trg_InsteadOfInsertOrderDetails;
GO
```

```
CREATE TRIGGER trg_InsteadOfInsertOrderDetails
ON [Order Details]
INSTEAD OF INSERT
AS
BEGIN
   DECLARE @OrderID INT, @ProductID INT, @Quantity INT;
   DECLARE cur CURSOR FOR
   SELECT OrderID, ProductID, Quantity
   FROM INSERTED;

   OPEN cur;
   FETCH NEXT FROM cur INTO @OrderID, @ProductID, @Quantity;

   WHILE @@FETCH_STATUS = 0
   BEGIN
      IF EXISTS (SELECT 1 FROM Products WHERE ProductID = @ProductID AND UnitsInStock >=
@Quantity)
      BEGIN
        UPDATE Products
          SET UnitsInStock = UnitsInStock - @Quantity
          WHERE ProductID = @ProductID;
          INSERT INTO [Order Details] (OrderID, ProductID, UnitPrice, Quantity, Discount)
          SELECT OrderID, ProductID, UnitPrice, Quantity, Discount
          FROM INSERTED
          WHERE OrderID = @OrderID AND ProductID = @ProductID;
      END
      ELSE
      BEGIN
         RAISERROR('Insufficient stock for ProductID %d. OrderID %d could not be filled.', 16, 1,
@ProductID, @OrderID);
      END
```

```
        FETCH NEXT FROM cur INTO @OrderID, @ProductID, @Quantity;
    END;

    CLOSE cur;
    DEALLOCATE cur;
END;
GO
```

ANS 3

```
CREATE PROCEDURE GetOrderDetails @OrderID INT
AS
BEGIN
    IF EXISTS (SELECT * FROM OrderDetails WHERE OrderID = @OrderID)
    BEGIN
        SELECT * FROM OrderDetails WHERE OrderID = @OrderID;
    END
    ELSE
    BEGIN
        PRINT 'The OrderID ' + CAST(@OrderID AS VARCHAR) + ' does not exist';
        RETURN 1;
    END
END
```

ANS 1

```
CREATE PROCEDURE InsertOrderDetails
    @OrderID INT,
    @ProductID INT,
    @Quantity INT,
    @UnitPrice DECIMAL(10, 2) = NULL,
    @Discount DECIMAL(4, 2) = NULL
AS
```

```
BEGIN
    DECLARE @ProductUnitPrice DECIMAL(10, 2);
    DECLARE @ProductDiscount DECIMAL(4, 2);
    DECLARE @StockToUpdate INT;
    DECLARE @UnitsInStock INT;
    DECLARE @ReorderLevel INT;
    SELECT @ProductUnitPrice = ISNULL(@UnitPrice, UnitPrice),
        @ProductDiscount = ISNULL(@Discount, 0),
        @UnitsInStock = UnitsInStock,
        @ReorderLevel = ReorderLevel
    FROM Products
    WHERE ProductID = @ProductID;
    IF @Quantity > @UnitsInStock
    BEGIN
        PRINT 'Failed to place the order. Not enough stock available.';
        RETURN;
    END
    INSERT INTO OrderDetails (OrderID, ProductID, UnitPrice, Quantity, Discount)
    VALUES (@OrderID, @ProductID, @ProductUnitPrice, @Quantity, @ProductDiscount);
    IF @@ROWCOUNT = 1
    BEGIN
        PRINT 'Order placed successfully.';
        SET @StockToUpdate = @UnitsInStock - @Quantity;
        IF @StockToUpdate < @ReorderLevel
        BEGIN
            PRINT 'Warning: Quantity in stock for ProductID ' + CAST(@ProductID AS VARCHAR) + ' drops
below reorder level.';
        END
        UPDATE Products
        SET UnitsInStock = @StockToUpdate
        WHERE ProductID = @ProductID;
```

```
    END
    ELSE
    BEGIN
        PRINT 'Failed to place the order. Please try again.';
    END
END
```