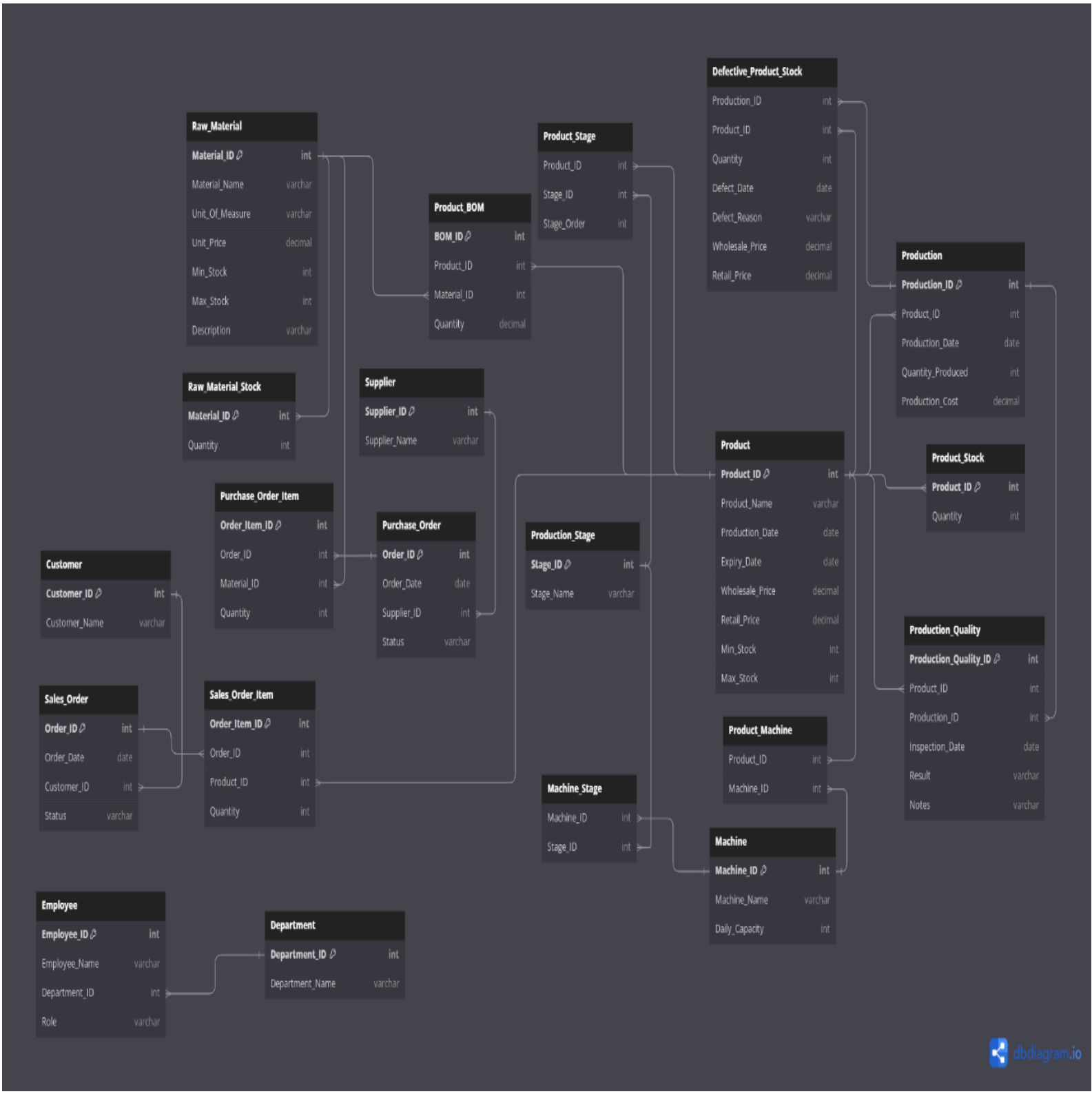
المادة: مخبر نظم قواعد البيانات BDBL501

الوظيفة F23

الطلاب المشاركون

اسم الطالب	الرقم الجامعي	الصف
يزن جوريه	Yazan_174681	C1

مخطط ERD :


Defective_Product_Stock
Production_ID int
Product_ID int
Quantity int
Defect_Date date
Defect_Reason varchar
Wholesale_Price decimal
Retail_Price decimal
Raw_Material
Material_ID int
Material_Name varchar
Unit_Of_Measure varchar
Unit_Price decimal
Min_Stock int
Max_Stock int
Description varchar
Product_Stage
Product_ID int
Stage_ID int
Stage_Order int
Product_BOM
BOM_ID int
Product_ID int
Material_ID int
Quantity decimal
Production
Production_ID int
Product_ID int
Production_Date date
Quantity_Produced int
Production_Cost decimal
Raw_Material_Stock
Material_ID int
Quantity int
Supplier
Supplier_ID int
Supplier_Name varchar
Product
Product_ID int
Product_Name varchar
Production_Date date
Expiry_Date date
Wholesale_Price decimal
Retail_Price decimal
Min_Stock int
Max_Stock int
Product_Stock
Product_ID int
Quantity int
Purchase_Order_Item
Order_Item_ID int
Order_ID int
Material_ID int
Quantity int
Purchase_Order
Order_ID int
Order_Date date
Supplier_ID int
Status varchar
Production_Stage
Stage_ID int
Stage_Name varchar
Customer
Customer_ID int
Customer_Name varchar
Production_Quality
Production_Quality_ID int
Product_ID int
Production_ID int
Inspection_Date date
Result varchar
Notes varchar
Sales_Order
Order_ID int
Order_Date date
Customer_ID int
Status varchar
Sales_Order_Item
Order_Item_ID int
Order_ID int
Product_ID int
Quantity int
Product_Machine
Product_ID int
Machine_ID int
Machine_Stage
Machine_ID int
Stage_ID int
Machine
Machine_ID int
Machine_Name varchar
Daily_Capacity int
Employee
Employee_ID int
Employee_Name varchar
Department_ID int
Role varchar
Department
Department_ID int
Department_Name varchar
dbdiagram.io

ملف دفعي لإنشاء جداول القاعدة مع قيودها :

```
-- Raw Material Table
CREATE TABLE Raw_Material (
  Material_ID INT PRIMARY KEY,
  Material_Name VARCHAR(255) NOT NULL,
  Unit_Of_Measure VARCHAR(50) NOT NULL,
  Unit_Price DECIMAL(10,2) NOT NULL,
  Min_Stock INT,
  Max_Stock INT,
  Description VARCHAR(255)
);

-- Raw Material Stock
CREATE TABLE Raw_Material_Stock (
  Material_ID INT PRIMARY KEY,
  Quantity INT NOT NULL,
  FOREIGN KEY (Material_ID) REFERENCES Raw_Material(Material_ID)
);

-- Product Table
CREATE TABLE Product (
  Product_ID INT PRIMARY KEY,
  Product_Name VARCHAR(255) NOT NULL,
  Production_Date DATE,
  Expiry_Date DATE,
  Wholesale_Price DECIMAL(10,2),
  Retail_Price DECIMAL(10,2),
  Min_Stock INT,
  Max_Stock INT
);

-- Product Stock
CREATE TABLE Product_Stock (
  Product_ID INT PRIMARY KEY,
  Quantity INT NOT NULL,
  FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID)
);

-- Machine Table
CREATE TABLE Machine (
  Machine_ID INT PRIMARY KEY,
  Machine_Name VARCHAR(255) NOT NULL,
  Daily_Capacity INT
);

-- Product_Machine
```

```sql
CREATE TABLE Product_Machine (
  Product_ID INT,
  Machine_ID INT,
  PRIMARY KEY (Product_ID, Machine_ID),
  FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID),
  FOREIGN KEY (Machine_ID) REFERENCES Machine(Machine_ID)
);

-- Production
CREATE TABLE Production (
  Production_ID INT PRIMARY KEY,
  Product_ID INT NOT NULL,
  Production_Date DATE NOT NULL,
  Quantity_Produced INT NOT NULL,
  Production_Cost DECIMAL(10,2),
  FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID)
);

   CREATE TABLE Production_Stage (
       Stage_ID INT PRIMARY KEY,
       Stage_Name VARCHAR(255) NOT NULL
   );

   CREATE TABLE Machine_Stage (
       Machine_ID INT,
       Stage_ID INT,
       PRIMARY KEY (Machine_ID, Stage_ID),
       FOREIGN KEY (Machine_ID) REFERENCES Machine(Machine_ID),
       FOREIGN KEY (Stage_ID) REFERENCES Production_Stage(Stage_ID)
   );

   CREATE TABLE Product_Stage (
       Product_ID INT,
       Stage_ID INT,
       Stage_Order INT, -- Order of the stage in the production process
       PRIMARY KEY (Product_ID, Stage_ID),
       FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID),
       FOREIGN KEY (Stage_ID) REFERENCES Production_Stage(Stage_ID)
   );


-- Product BOM (Bill of Materials)
CREATE TABLE Product_BOM (
    BOM_ID INT PRIMARY KEY,
    Product_ID INT NOT NULL,
    Material_ID INT NOT NULL,
    Quantity DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID),
```

```
    FOREIGN KEY (Material_ID) REFERENCES Raw_Material(Material_ID)
);



Create Table Production_Quality(
  Production_Quality_ID INT PRIMARY KEY,
  Product_ID INT NOT NULL,
  Production_ID INT NOT NULL,
  Inspection_Date DAte NOT NULL,
  Result varchar2(255) Not Null,
  Notes varchar2(255),
  FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID),
  FOREIGN KEY (Production_ID) REFERENCES Production(Production_ID)
);

-- Defective Products
CREATE TABLE Defective_Product_Stock (
  Production_ID INT NOT NULL,
  Product_ID INT NOT NULL ,
  Quantity INT NOT NULL,
  Defect_Date DATE NOT NULL,
  Defect_Reason VARCHAR(255),
  Wholesale_Price DECIMAL(10,2),
  Retail_Price DECIMAL(10,2),
  PRIMARY KEY (Production_ID,Product_ID),
  FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID),
  FOREIGN KEY (Production_ID) REFERENCES Production(Production_ID)
);

-- Supplier Table
CREATE TABLE Supplier (
  Supplier_ID INT PRIMARY KEY,
  Supplier_Name VARCHAR(255) NOT NULL
);

-- Purchase Order
CREATE TABLE Purchase_Order (
  Order_ID INT PRIMARY KEY,
  Order_Date DATE NOT NULL,
  Supplier_ID INT NOT NULL,
  Status VARCHAR(50) ,
  FOREIGN KEY (Supplier_ID) REFERENCES Supplier(Supplier_ID)
);

-- Purchase Order Item
CREATE TABLE Purchase_Order_Item (
  Order_Item_ID INT PRIMARY KEY,
  Order_ID INT NOT NULL,
```

```
  Material_ID INT NOT NULL,
  Quantity INT NOT NULL,
  FOREIGN KEY (Order_ID) REFERENCES Purchase_Order(Order_ID),
  FOREIGN KEY (Material_ID) REFERENCES Raw_Material(Material_ID)
);

-- Customer Table
CREATE TABLE Customer(
  Customer_ID INT PRIMARY KEY,
  Customer_Name VARCHAR(255) Not Null
);

-- Sales Order
CREATE TABLE Sales_Order (
  Order_ID INT PRIMARY KEY,
  Order_Date DATE NOT NULL,
  Customer_ID INT,
  Status VARCHAR(255) ,
  FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID)
);

-- Sales Order Item
CREATE TABLE Sales_Order_Item (
  Order_Item_ID INT PRIMARY KEY,
  Order_ID INT NOT NULL,
  Product_ID INT NOT NULL,
  Quantity INT NOT NULL,
  FOREIGN KEY (Order_ID) REFERENCES Sales_Order(Order_ID),
  FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID)
);

-- Department (purchasing, quality assurance, and sales)
CREATE TABLE Department (
  Department_ID INT PRIMARY KEY,
  Department_Name VARCHAR(255) NOT NULL
);

-- Employee
CREATE TABLE Employee (
  Employee_ID INT PRIMARY KEY,
  Employee_Name VARCHAR(255) NOT NULL,
  Department_ID INT,
  Role VARCHAR(255),
  FOREIGN KEY (Department_ID) REFERENCES Department(Department_ID)
);
```

الشرح:

المواد الخام (Raw Materials)

`Raw_Material*` تُخزّن معلومات المواد الخام الأساسية:

* Material_ID: مُعرّف فريد للمادة الخام (مفتاح أساسي).
* Material_Name: اسم المادة الخام.
* Unit_Of_Measure: وحدة قياس المادة الخام (مثال: كيلوجرام، لتر).
* Unit_Price: سعر وحدة قياس المادة الخام.
* Min_Stock: أقل كمية يجب توفرها من المادة الخام في المخزون.
* Max_Stock: أكبر كمية يمكن تخزينها من المادة الخام في المخزون.
* Description: وصف إضافي للمادة الخام.

`Raw_Material_Stock` *: تُخزّن كميات المواد الخام المتوفرة في المخزون:

* Material_ID: مُعرّف المادة الخام (مفتاح أساسي ومفتاح أجنبي مرتبط بجدول Raw_Material).
* Quantity: الكمية المتوفرة من المادة الخام في المخزون.

المنتجات (Products)

`Product*` تُخزّن معلومات المنتجات النهائية:

* Product_ID: مُعرّف فريد للمنتج (مفتاح أساسي).
* Product_Name: اسم المنتج.
* Production_Date: تاريخ إنتاج المنتج (اختياري).
* Expiry_Date: تاريخ انتهاء صلاحية المنتج (اختياري).
* Wholesale_Price: سعر بيع المنتج بسعر الجملة.
* Retail_Price: سعر بيع المنتج بالتجزئة.
* Min_Stock: أقل كمية يجب توفرها من المنتج في المخزون.
* Max_Stock: أكبر كمية يمكن تخزينها من المنتج في المخزون.

`Product_Stock` *: تُخزّن كميات المنتجات المتوفرة في المخزون:

* Product_ID: مُعرّف المنتج (مفتاح أساسي ومفتاح أجنبي مرتبط بجدول Product).
* Quantity: الكمية المتوفرة من المنتج في المخزون.

المَكائن ((Machines

` *Machine`: تُخزّن معلومات عن مَكائن الإنتاج:

* Machine_ID: مُعرّف فريد للمَكينة (مفتاح أساسي).
* Machine_Name: اسم المَكينة.
* Daily_Capacity: الطاقة الإنتاجية اليومية للمَكينة.

عملية الإنتاج ((Production

` *Production`: تُخزّن معلومات عمليات الإنتاج:

* Production_ID: مُعرّف فريد لعملية الإنتاج (مفتاح أساسي).
* Product_ID: مُعرّف المنتج الذي تم إنتاجه (مفتاح أجنبي مرتبط بجدول Product).
* Production_Date: تاريخ عملية الإنتاج.
* Quantity_Produced: كمية المنتج التي تم إنتاجها.
* Production_Cost: تكلفة عملية الإنتاج (اختياري).

` *Product_BOM`:

تُحدد قائمة مكونات كل منتج (Bill of Materials):

* BOM_ID: مُعرّف فريد لسجل قائمة مكونات الصنع (مفتاح أساسي).
* Product_ID: مُعرّف المنتج (مفتاح أجنبي مرتبط بجدول Product).
* Material_ID: مُعرّف المادة الخام (مفتاح أجنبي مرتبط بجدول Raw_Material).
* Quantity: كمية المادة الخام اللازمة لإنتاج وحدة واحدة من المنتج.

` *Production_Quality`:

تُخزّن نتائج فحص جودة الإنتاج:

* Production_Quality_ID: مُعرّف فريد لنتيجة فحص الجودة (مفتاح أساسي).
* Product_ID: مُعرّف المنتج (مفتاح أجنبي مرتبط بجدول Product).
* Production_ID: مُعرّف عملية الإنتاج (مفتاح أجنبي مرتبط بجدول Production).
* Inspection_Date: تاريخ فحص الجودة.
* Result: نتيجة فحص الجودة (مثال: ناجح، فاشل).
* Notes: ملاحظات إضافية حول فحص الجودة (اختياري).

المنتجات التالفة ((Defective Products

` *Defective_Product_Stock`: تُخزّن معلومات عن المنتجات التالفة:

* Production_ID: مُعرّف عملية الإنتاج (مفتاح أساسي ومفتاح أجنبي مرتبط بجدول Production).
* Product_ID: مُعرّف المنتج (مفتاح أساسي ومفتاح أجنبي مرتبط بجدول Product).
* Quantity: كمية المنتج التالف.
* Defect_Date: تاريخ اكتشاف العيب.
* Defect_Reason: سبب العيب (اختياري).
* Wholesale_Price: سعر بيع المنتج التالف بسعر الجملة (اختياري).
* Retail_Price: سعر بيع المنتج التالف بالتجزئة (اختياري).

المُورّدون ((Suppliers

` *Supplier`: تُخزّن معلومات المُورّدين:

* Supplier_ID: مُعرّف فريد للمُورّد (مفتاح أساسي).
* Supplier_Name: اسم المُورّد.

` *Purchase_Order`:

تُخزّن معلومات طلبات الشراء من المُورّدين:

* Order_ID: مُعرّف فريد لطلب الشراء (مفتاح أساسي).
* Order_Date: تاريخ طلب الشراء.
* Supplier_ID: مُعرّف المُورّد (مفتاح أجنبي مرتبط بجدول Supplier).
* Status: حالة طلب الشراء (مثال: جديد، قيد التنفيذ، مكتمل).

العملاء ((Customers

` *Customer`: تُخزّن معلومات العملاء:

* Customer_ID: مُعرّف فريد للعميل (مفتاح أساسي).
* Customer_Name: اسم العميل.

المبيعات ((Sales

` *Sales_Order`:

تُخزّن معلومات طلبات البيع:

` * Order_ID`: مُعرّف فريد لطلب البيع (مفتاح أساسي).

` * Order_Date`: تاريخ طلب البيع.

` * Customer_ID`: مُعرّف العميل الذي قام بالشراء (مفتاح أجنبي مرتبط بجدول `Customer`).

` * Status`: حالة طلب البيع (مثال: جديد، قيد التنفيذ، مكتمل، ملغي).

` * Sales_Order_Item`:

تُخزّن تفاصيل الأصناف المدرجة في كل طلب بيع:

` * Order_Item_ID`: مُعرّف فريد لصنف في طلب البيع (مفتاح أساسي).

` * Order_ID`: مُعرّف طلب البيع (مفتاح أجنبي مرتبط بجدول `Sales_Order`).

` * Product_ID`: مُعرّف المنتج الذي تم طلبه (مفتاح أجنبي مرتبط بجدول `Product`).

` * Quantity`: كمية المنتج المطلوبة.

الأقسام والموظفون (Departments and Employees)

` *Department`: تُخزّن معلومات عن أقسام الشركة:

` * Department_ID`: مُعرّف فريد للقسم (مفتاح أساسي).

` * Department_Name`: اسم القسم (مثال: المشتريات، ضمان الجودة، المبيعات).

` * Employee`:

تُخزّن معلومات الموظفين:

` * Employee_ID`: مُعرّف فريد للموظف (مفتاح أساسي).

` * Employee_Name`: اسم الموظف.

` * Department_ID`: مُعرّف القسم الذي يعمل به الموظف (مفتاح أجنبي مرتبط بجدول `Department`).

` * Role`: دور الموظف أو منصبه في القسم (مثال: مدير، موظف مشتريات).

تصميم إجرائية لتصنيع كمية معينة من منتج معين مع الاستفادة من المناقلات لضمان الكميات مع معالجة الخطأ أو توليد خطأ في حال عدم توفر الكمية المطلوبة من المواد الأولية.

```
CREATE OR REPLACE FUNCTION Production_Process (
    PR_Product INT,
    PR_Quantity INT
)
```

```
RETURN NUMBER -- Returns 0 if production is successful, raises an error
otherwise
AS
    total_cost NUMBER := 0;
    Production_ID INT;
    v_Production_ID INT; -- Temporary variable to hold the generated
Production ID
    Production_Quality_ID INT := 0; -- Variable to store the next available
Production Quality ID
    v_Production_Quality_ID INT; -- Temporary variable to hold the generated
Production Quality ID
    Quality_Result varchar(255);
    material_qty_used  DECIMAL(10,2);
    BOM_Quantity Product_BOM.quantity%TYPE;
    Unit_cost Raw_Material.unit_price%TYPE;
    v_available_quantity Raw_Material_Stock.Quantity%TYPE;
    v_required_quantity Raw_Material_Stock.Quantity%TYPE;
    v_material_name Raw_Material.Material_Name%TYPE;
    material_id Product_BOM.material_id%TYPE; --Store mateial id


    -- Declare a cursor to fetch the required data for each material to
check if raw materials are sufficient for the production
CURSOR material_cursor_1 IS
    SELECT b.quantity, r.unit_price, s.Quantity, r.Material_Name
    FROM Product_BOM b
    JOIN Raw_Material r ON b.material_id = r.material_id
    JOIN Raw_Material_Stock s ON b.material_id = s.Material_ID
    WHERE b.product_id = 1;

-- Declare a cursor to fetch the required data for each material Calculate
the total cost of production and update the raw material stock
CURSOR material_cursor_2 IS
    SELECT b.material_id, b.quantity, r.unit_price
    FROM Product_BOM b
    JOIN Raw_Material r ON b.material_id = r.material_id
    WHERE b.product_id = PR_Product;



BEGIN
OPEN material_cursor_1;
LOOP
    FETCH material_cursor_1 INTO BOM_Quantity, Unit_cost,
v_available_quantity, v_material_name;
    EXIT WHEN material_cursor_1%NOTFOUND;
    -- Calculate the required quantity of material based on the production
quantity
    v_required_quantity := PR_Quantity * BOM_Quantity;
```

```
    -- Check if the available quantity is sufficient
    IF v_available_quantity < v_required_quantity THEN
        RAISE_APPLICATION_ERROR(-20001, 'Insufficient ' || v_material_name
|| ' in stock. Required: ' || v_required_quantity || ', Available: ' ||
v_available_quantity);
    END IF;
END LOOP;
CLOSE material_cursor_1;


        -- Get the next available Production ID
    SELECT NVL(MAX(Production_ID), 0) into Production_ID FROM Production;

        -- Calculate the new Production ID
    v_Production_ID := Production_ID + 1;

    -- Get the next available Production Quality ID
    SELECT NVL(MAX(Production_Quality_ID), 0) into Production_Quality_ID
FROM Production_Quality;

    -- Calculate the new Production Quality ID
    v_Production_Quality_ID := Production_Quality_ID +1;

        -- Create the production plan in the Production table
    INSERT INTO Production (Production_ID, Product_ID,
Production_Date,Quantity_Produced)
    VALUES (v_Production_ID,PR_Product,SYSDATE, PR_Quantity);


    -- Calculate the total cost of production and update the raw material
stock
OPEN material_cursor_2;
LOOP
    FETCH material_cursor_2 INTO material_id, BOM_Quantity, Unit_cost;
    EXIT WHEN material_cursor_2%NOTFOUND;
    -- Calculate the quantity of material used in this production run
    material_qty_used := PR_Quantity * BOM_Quantity;
    -- Update the Raw Material Stock table to reflect the material used
    BEGIN
        UPDATE Raw_Material_Stock
        SET Quantity = Quantity - material_qty_used
        WHERE Material_ID = material_id;
    EXCEPTION
        WHEN OTHERS THEN
            -- Rollback transaction and raise an error if the update fails
            ROLLBACK;
            RAISE_APPLICATION_ERROR(-20002, 'Error updating raw material
stock.');
```

```
    END;

    -- Calculate the total cost of the materials used
    total_cost := total_cost + (material_qty_used * Unit_cost);
END LOOP;
CLOSE material_cursor_2;

    -- Randomly determine the quality inspection result
    DBMS_RANDOM.SEED(DBMS_RANDOM.VALUE);
    Quality_Result := CASE when DBMS_RANDOM.VALUE() <0.8 THen 'Passed' Else
'Faild' END;

    -- Record the quality inspection result in the Production_Quality table
    INSERT into Production_Quality
(Production_Quality_ID,Production_ID,Product_ID,Inspection_Date,Result)
    Values
(v_Production_Quality_ID,v_Production_ID,PR_Product,sysdate,Quality_Result);

  -- If the quality inspection passes, update the product stock
  if Quality_Result ='Passed' then
    BEGIN
      Update Product_Stock
      Set Quantity=Quantity + PR_Quantity
      where Product_ID = PR_Product;
    EXCEPTION
        WHEN OTHERS THEN
            -- Rollback transaction and raise an error if the update fails
            ROLLBACK;
            RAISE_APPLICATION_ERROR(-20003, 'Error updating product
stock.');
    END;

  -- If the quality inspection fails, add the produced items to the
defective stock
  else
    BEGIN
        Insert Into Defective_Product_Stock
(Production_ID,Product_ID,Quantity,Defect_Date,Defect_Reason,Wholesale_Price
,Retail_Price) values
        (v_Production_ID,PR_Product,PR_Quantity,sysdate,'Random reason',
        ( (select max(Wholesale_Price) from Product where
Product_ID=PR_Product)/2),
        ((select max(Retail_Price) from Product where
Product_ID=PR_Product)/2));
        DBMS_OUTPUT.PUT_LINE('Defective' );
    EXCEPTION
        WHEN OTHERS THEN
            -- Rollback transaction and raise an error if the insert fails
```

```
            ROLLBACK;
            RAISE_APPLICATION_ERROR(-20004, 'Error inserting into defective
stock.');
    END;
  end if;
    -- Output the total production cost
    DBMS_OUTPUT.PUT_LINE('Productio Complete , Total Cost : ' ||
total_cost);
    return 0; -- Return 0 to indicate successful production
EXCEPTION
    WHEN OTHERS THEN
        -- Rollback transaction if any error occurs and re-raise the error
for handling
        ROLLBACK;
        RAISE;
END;
/
```

شرح الكود:

أولاً: تعريف المتغيرات

* *PR_Product مُدخل للدالة يحدد مُعرّف المنتج.*
* *PR_Quantity مُدخل للدالة يُحدد الكمية المطلوب إنتاجها من المنتج.*
* *total_cost مُتغيّر لحساب التكلفة الإجمالية لعملية الإنتاج.*
* *Production_ID مُتغيّر لحفظ مُعرّف عملية الإنتاج.*
* *v_Production_ID مُتغيّر مؤقت لحفظ مُعرّف عملية الإنتاج.*
* *Production_Quality_ID مُتغيّر لحفظ مُعرّف جودة الإنتاج.*
* *v_Production_Quality_ID مُتغيّر مؤقت لحفظ مُعرّف جودة الإنتاج.*
* *Quality_Result مُتغيّر لحفظ نتيجة فحص الجودة.*
* *material_qty_used مُتغيّر لحفظ كمية المواد المُستخدمة.*
* *BOM_Quantity مُتغيّر لحفظ كمية المواد المُحددة في قائمة المكونات*
* *Unit_cost مُتغيّر لحفظ سعر الوحدة للمادة الخام.*
* *v_available_quantity مُتغيّر لحفظ الكمية المتاحة من المادة الخام في المخزون.*
* *v_required_quantity مُتغيّر لحفظ الكمية المطلوبة من المادة الخام.*
* *v_material_name مُتغيّر لحفظ اسم المادة الخام.*
* *material_id مُتغيّر لحفظ مُعرّف المادة الخام.*

ثانياً: تعريف المؤشرات

* material_cursor_1 مؤشر لاسترداد بيانات كل مادة خام من جداول Product_BOM و Raw_Material و Raw_Material_Stock للتحقق من توافرها قبل بدء الإنتاج.

* material_cursor_2 مؤشر لاسترداد بيانات كل مادة خام من جداول Product_BOM و Raw_Material لحساب التكلفة الإجمالية للإنتاج وتحديث كميات المواد الخام في المخزون بعد الانتهاء من الإنتاج.

ثالثاً: التنفيذ

١- فتح المؤشر material_cursor_1 يتم فتح المؤشر للبدء في جلب البيانات.

٢. التحقق من كمية المواد الخام: يتم استخدام حلقة LOOP للتحقق من كمية كل مادة خام مطلوبة للإنتاج:

* جلب بيانات المواد الخام: يتم جلب بيانات BOM_Quantity و Unit_cost و v_available_quantity و v_material_name لكل مادة من خلال المؤشر material_cursor_1

* حساب الكمية المطلوبة: يتم حساب الكمية المطلوبة v_required_quantity من كل مادة خام.

* التحقق من الكمية المتاحة: يتم التحقق مما إذا كانت الكمية المتاحة v_available_quantity في المخزون كافية لتغطية الكمية المطلوبة

v_required_quantity

* في حالة عدم كفاية الكمية: إذا كانت الكمية المتاحة غير كافية، يتم إطلاق خطأ ` `RAISE_APPLICATION_ERROR مع رسالة توضح نقص المادة الخام المحددة.

٣. إغلاق المؤشر material_cursor_1 بعد الانتهاء من التحقق من جميع المواد الخام، يتم إغلاق المؤشر material_cursor_1

٤. حساب Production_ID التالي: يتم جلب أحدث مُعرّف Production_ID من جدول Production وإضافة ١ إليه للحصول على مُعرّف جديد لعملية الإنتاج الحالية.

٥. حساب Production_Quality_ID التالي: يتم جلب أحدث مُعرّف Production_Quality_ID من جدول Production_Quality وإضافة ١ إليه للحصول على مُعرّف جديد لجودة الإنتاج الحالية.

٦. إنشاء سجل جديد في جدول Production يتم إدراج سجل جديد في جدول Production بتفاصيل عملية الإنتاج الجديدة: مُعرّف ``Production_ID ، مُعرّف المنتج ``Product_ID، تاريخ الإنتاج ``Production_Date، و الكمية المنتجة ` .`Quantity_Produced

٧- حساب التكلفة وتحديث مخزون المواد الخام:

- فتح المؤشر ` :`material_cursor_2 يُفتح المؤشر للبدء في جلب بيانات المواد الخام.

- التكرار على المواد الخام: تُستخدم حلقة ` `LOOP لمعالجة كل مادة خام:

- جلب بيانات المادة: يتم جلب مُعرّف المادة material_id ، كمية المادة في قائمة مكونات الصنع `BOM_Quantity ، وسعر الوحدة Unit_cost من خلال المؤشر ` .`material_cursor_2

- حساب الكمية المُستخدمة: يتم حساب كمية المادة المستخدمة` material_qty_used بناءً على كمية الإنتاج.

- تحديث مخزون المواد الخام:

- يتم تحديث جدول ` `Raw_Material_Stock بتخفيض الكمية المتاحة من المادة الخام بمقدار الكمية المستخدمة .material_qty_used

- تم استخدام كتلة `BEGIN...EXCEPTION...END` لالتقاط أي أخطاء تحدث أثناء تحديث المخزون والتراجع عن العملية Rollback في حالة حدوث أي خطأ.

- حساب التكلفة الإجمالية: تُضاف تكلفة المواد المُستخدمة `material_qty_used * Unit_cost` إلى `total_cost`.

- إغلاق المؤشر `material_cursor_2`: يتم إغلاق المؤشر بعد الانتهاء من معالجة جميع المواد الخام.

٨. فحص الجودة:

- توليد نتيجة عشوائية: يتم استخدام `DBMS_RANDOM.VALUE()` لتوليد نتيجة عشوائية لفحص الجودة (نجاح أو فشل).

- تسجيل نتيجة الفحص: يتم إدراج نتيجة الفحص Quality_Result في جدول `Production_Quality`.

٩. تحديث المخزون بناءً على نتيجة الفحص:

- إذا نجح الفحص: `Quality_Result = 'Passed'`

- يتم تحديث جدول `Product_Stock` بزيادة كمية المنتج المُنتج.

- تم استخدام كتلة `BEGIN...EXCEPTION...END` لالتقاط أي أخطاء تحدث أثناء التحديث والتراجع عن العملية في حالة وجود خطأ.

- إذا فشل الفحص: `Quality_Result = 'Failed'`

- يتم إضافة المنتجات التالفة إلى جدول `Defective_Product_Stock`.

- تم استخدام كتلة `BEGIN...EXCEPTION...END` لالتقاط أي أخطاء تحدث أثناء الإدراج والتراجع عن العملية في حالة وجود خطأ.

١٠. النهاية:

- عرض التكلفة: يتم عرض التكلفة الإجمالية للإنتاج (`total_cost`).

١١. معالجة الأخطاء العامة:

- تم استخدام كتلة `EXCEPTION...WHEN OTHERS...` لالتقاط أي أخطاء غير مُعالجة والتراجع عن جميع العمليات التي تمت على قاعدة البيانات وإعادة إطلاق الخطأ ليتم التعامل معه بشكل صحيح.

كتابة قادح للتحذير من تناقص أو ازدياد كمية عن حدودها الطبيعية في المستودع

```
CREATE OR REPLACE TRIGGER RAW_MATERIAL_STOCK_WARNING
BEFORE INSERT OR UPDATE OF Quantity ON Raw_Material_Stock
FOR EACH ROW
```

```
BEGIN
  -- Get the Material Name and Stock Levels
  DECLARE
    material_name VARCHAR2(255);
    max_stock     INT;
    min_stock     INT;
  BEGIN
    SELECT Material_Name, Max_Stock, Min_Stock
    INTO material_name, max_stock, min_stock
    FROM Raw_Material
    WHERE Material_ID = :new.Material_ID;

    -- Check if the new quantity exceeds the maximum stock
    IF :new.Quantity > max_stock THEN
      DBMS_OUTPUT.PUT_LINE('WARNING: Raw Material ' || material_name || '
stock exceeds maximum allowed level.');

    END IF;

    -- Check if the new quantity falls below the minimum stock
    IF :new.Quantity < min_stock THEN
      DBMS_OUTPUT.PUT_LINE('WARNING: Raw Material ' || material_name || '
stock is below the minimum required level.');

    END IF;
  END;
END;

CREATE OR REPLACE TRIGGER Product_Stock_Warning
BEFORE INSERT OR UPDATE OF Quantity ON Product_Stock
FOR EACH ROW
BEGIN
  -- Get the Product Name and Stock Levels
  DECLARE
    product_name VARCHAR2(255);
    max_stock     INT;
    min_stock     INT;
  BEGIN
    SELECT Product_Name, Max_Stock, Min_Stock
    INTO product_name, max_stock, min_stock
    FROM Product
    WHERE Product_ID = :new.Product_ID;

    -- Check if the new quantity exceeds the maximum stock
    IF :new.Quantity > max_stock THEN
      DBMS_OUTPUT.PUT_LINE('WARNING: Product ' || product_name || ' stock
exceeds maximum allowed level.');
```

```
    END IF;

    -- Check if the new quantity falls below the minimum stock
    IF :new.Quantity < min_stock THEN
      DBMS_OUTPUT.PUT_LINE('WARNING: Product ' || product_name || ' stock is
below the minimum required level.');

    END IF;
  END;
END;
/
```

شرح الكود:

1- RAW_MATERIAL_STOCK_WARNING:

* يتم تشغيله قبل عملية الإدراج أو التحديث في حقل Quantity في جدول Raw_Material_Stock.

* يسترجع اسم المادة ومستوى المخزون الأقصى ومستوى المخزون الأدنى من جدول Raw_Material بناءً على Material_ID.

* يقارن الكمية الجديدة بمستويات المخزون القصوى والدنيا، ويصدر تحذيرات في حالة تجاوز العتبات أو عدم بلوغها.

2- Product_Stock_Warning:

* يتم تشغيله قبل عملية الإدراج أو التحديث في حقل Quantity في جدول Product_Stock.

* يسترجع اسم المنتج ومستوى المخزون الأقصى ومستوى المخزون الأدنى من جدول Product بناءً على Product_ID.

* يقارن الكمية الجديدة بمستويات المخزون القصوى والدنيا، ويصدر تحذيرات في حالة تجاوز العتبات أو عدم بلوغها.