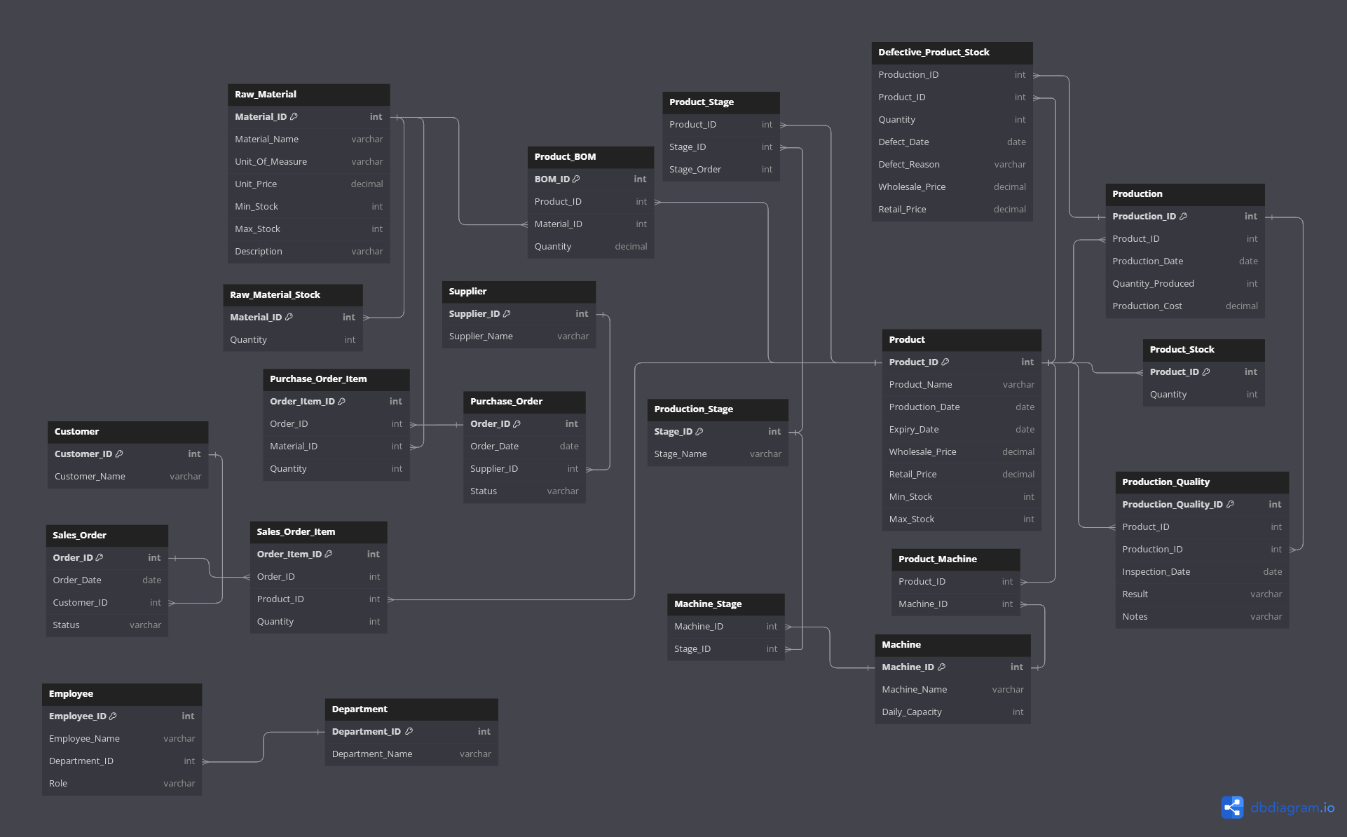
|  |  |  |
| --- | --- | --- |
| الجامعة الافتراضية السورية | logo4 | Syrian Virtual University |

المادة: مخبر نظم قواعد البيانات BDBL501

الوظيفة F23

الطلاب المشاركون

|  |  |  |
| --- | --- | --- |
| اسم الطالب | الرقم الجامعي | الصف |
| يزن جوريه | Yazan\_174681 | *C1* |

مخطط ERD :

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

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\_IDمُتغيّر لحفظ مُعرّف جودة الإنتاج.

\* v\_Production\_Quality\_IDمُتغيّر 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\_quantityمُتغيّر لحفظ الكمية المتاحة من المادة الخام في المخزون.

\* v\_required\_quantityمُتغيّر لحفظ الكمية المطلوبة من المادة الخام.

\* v\_material\_nameمُتغيّر 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

1- فتح المؤشر material\_cursor\_1 يتم فتح المؤشر للبدء في جلب البيانات.

2. 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 وUnit\_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.

3. إغلاق المؤشر material\_cursor\_1 بعد الانتهاء من التحقق من جميع المواد الخام، يتم إغلاق المؤشر material\_cursor\_1

4. حساب Production\_ID التالي: يتم جلب أحدث مُعرّف Production\_ID من جدول Production وإضافة 1 إليه للحصول على مُعرّف جديد لعملية الإنتاج الحالية.

5. حساب Production\_Quality\_ID التالي: يتم جلب أحدث مُعرّف Production\_Quality\_ID من جدول Production\_Quality وإضافة 1 إليه للحصول على مُعرّف جديد لجودة الإنتاج الحالية.

6. إنشاء سجل جديد في جدول Production يتم إدراج سجل جديد في جدول Productionبتفاصيل عملية الإنتاج الجديدة: مُعرّف `Production\_ID` ، مُعرّف المنتج `Product\_ID`، تاريخ الإنتاج `Production\_Date`، و الكمية المنتجة `Quantity\_Produced`.

7- 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 المستخدمةmaterial\_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.

8. 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`.

9. تحديث المخزون بناءً على نتيجة الفحص:

- إذا نجح الفحصQuality\_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.

10. النهاية:

- عرض التكلفة: يتم عرض التكلفة الإجمالية للإنتاج (`total\_cost`).

11. 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.