BDBL501
S23

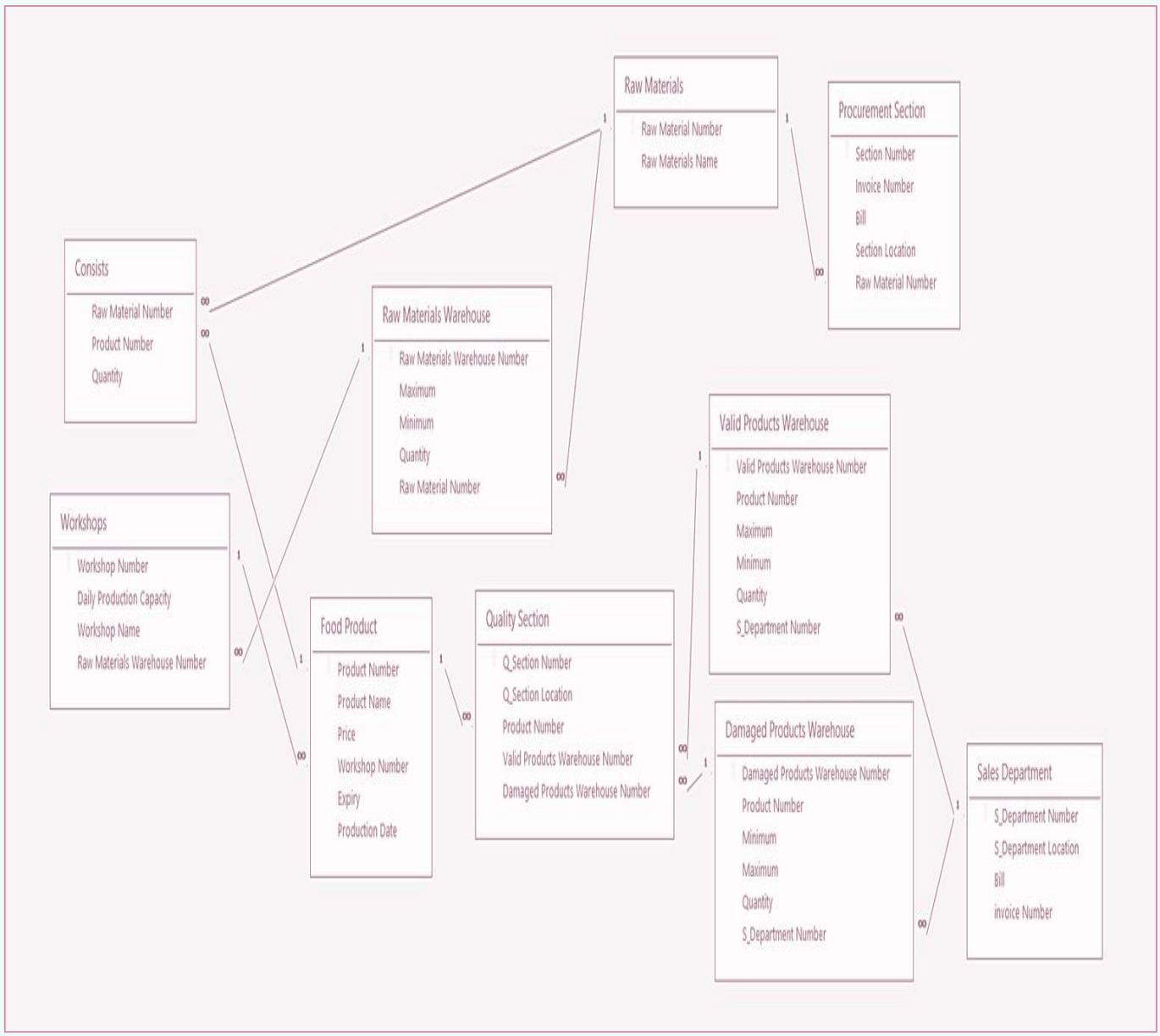
الطلاب المشاركين:

Haneen_197899_C3	حنين الأصفر
Yaman_199273_C3	يمان حامد
Amena_187866_C3	أمينة النعيمي
Lubna_175170_C3	لبنى بكداش
Huda_203500_C3	هدى المصري

بإشراف الدكتورة: ألاء الناطور.

الطلب الأول:

- تصميم مخطط الـ ERD للقاعدة. || تم رسم المخطط بالاستعانة ببرنامج Access ||.



Relationships Between Entities:

First Entity	Relationship	Second Entity	Relationship Type
Raw_Materials	Include	Procurement_Section	1 to M
المادة الأولية تنتمي لقسم مشتريات واحد – قسم المشتريات يضم العديد من المواد الأولية.			
Raw_Materials	Consist	Food_Product	M to M
المنتج الغذائي يمكن أن يتألف من أكثر من مادة أولية –المادة الأولية يمكن أن تصنع أكثر من منتج غذائي تم إنشاء جدول العلاقة. **(Consists)**			
Raw_Materials	Include	Raw_Materials_Warehouse	1 to M
المادة الأولية تنتمي لمستودع مواد أولية واحد – مستودع المواد الأولية يمكن أن يحتوي أكثر من مادة أولية.			
Raw_Materials_Warehouse	Consume	WorkShops	1 to M
ورشة العمل تستهلك من مستودع مواد أولية واحد.–مستودع المواد الأولية يُستهلك من أكثر من ورشة عمل			
WorkShops	Consist	Food_Product	1 to M
المنتج الغذائي يمكن أن يتبع إلى مجموعة من –الورشة يمكن أن تنتج منتج غذائي واحد الآلات.			
Food_Product	Include	Quality_Section	1 to M
قسم الجودة يمكن أن يضم أكثر من منتج غذائي.–المنتج الغذائي يتبع لقسم جودة وحيد			
Quality_Section	Include	Valid_Products_Warehouse	1 to M
مستودع المنتجات الصالحة يمكن أن يضم –المخرجات من قسم الجودة تنتمي لمستودع منتجات صالحة وحيد أكثر من مخرج من قسم الجودة.			

First Entity	Relationship	Second Entity	Relationship Type
Quality_Section	Include	Damaged_Products_Warehouse	1 to M
المخرجات من قسم الجودة تنتمي لمستودع منتجات تالفة وحيد – مستودع المنتجات التالفة يمكن أن يضم أكثر من مخرج من قسم الجودة.			
Valid_Products_Warehouse	Include	Sales_Department	M to M
المخرجات من مستودع المنتجات الصالحة تنتمي لقسم مبيعات وحيد – قسم المبيعات يمكن أن يضم أكثر من مخرج من مستودع المنتجات الصالحة.			
Damaged_Products_ Warehouse	Include	Sales_Department	1 to M
المخرجات من مستودع المنتجات التالفة تنتمي لقسم مبيعات وحيد – قسم المبيعات يمكن أن يضم أكثر من مخرج من مستودع المنتجات التالفة.			

الطلب الثاني:

- إنشاء ملف دفعي لإنشاء القاعدة مع قيودها.

```
CREATE TABLE Food_Product (
  Product_Number INT NOT NULL,
  Product_Name VARCHAR2(100),
  Expiry NUMBER,
  Production_Date DATE,
  Wholesale_Price NUMBER,
  Retail_Price NUMBER,
  WorkShops_Number INT NOT NULL,
  CONSTRAINT Food_Product_PK
  PRIMARY KEY(Product_Number),
  CONSTRAINT Food_Product_FK
  FOREIGN KEY (WorkShops_Number)
  REFERENCES WorkShops(WorkShops_Number) );
```

```
CREATE TABLE Valid_Products_Warehouse (
  Valid_Products_Warehouse_Number INT NOT NULL,
  Maximum NUMBER,
  Minimum NUMBER,
  Quentity NUMBER,
  Product_Number INT NOT NULL,
  Department_S_Number INT NOT NULL,
  CONSTRAINT Valid_Products_Warehouse_PK
  PRIMARY KEY(Valid_Products_Warehouse_Number),
  CONSTRAINT Valid_Products_Warehouse1_FK
  FOREIGN KEY (Product_Number)
  REFERENCES Food_Product(Product_Number),
  CONSTRAINT Valid_Products_Warehouse2_FK
  FOREIGN KEY (Department_S_Number)
  REFERENCES Sales_Department(Department_S_Number) );
```

```
CREATE TABLE Sales_Department (
  Department_S_Number INT NOT NULL,
  Department_S_Location VARCHAR2 (100),
  Bill NUMBER,
  Invoice_Number INT,
  CONSTRAINT Sales_Department_PK PRIMARY KEY(Department_S_Number),
  CONSTRAINT UC_InvoNumSales UNIQUE (Invoice_Number) );
```

```
CREATE TABLE Raw_Materials (
  Raw_Materials_Number INT NOT NULL,
  Raw_Materials_Name VARCHAR2(100),
  CONSTRAINT Raw_Materials_PK PRIMARY KEY (Raw_Materials_Number) );
```

```
CREATE TABLE Raw_Materials_Warehouse (
  Warehouse_Number INT NOT NULL,
  Maximum NUMBER,
  Minimum NUMBER,
  Quentity NUMBER,
  Raw_Materials_Number INT NOT NULL,
  CONSTRAINT Raw_Materials_Warehouse_PK
  PRIMARY KEY(Warehouse_Number),
  CONSTRAINT Raw_Materials_Warehouse_FK
  FOREIGN KEY (Raw_Materials_Number)
  REFERENCES Raw_Materials(Raw_Materials_Number) );
```

```
CREATE TABLE WorkShops (
  WorkShops_Number INT NOT NULL,
  Daily_Production_Capacity INT,
  WorkShops_Name VARCHAR2(100),
  Warehouse_Number INT NOT NULL,
  CONSTRAINT WorkShops_PK
  PRIMARY KEY (WorkShops_Number),
  CONSTRAINT WorkShops2_FK
  FOREIGN KEY (Warehouse_Number)
  REFERENCES Raw_Materials_Warehouse(Warehouse_Number) );
```

```
CREATE TABLE Damaged_Products_Warehouse (
  Damaged_Products_Warehouse_Number INT NOT NULL,
  Maximum NUMBER,
  Minimum NUMBER,
  Quentity NUMBER,
  Product_Number INT NOT NULL,
  Department_S_Number INT NOT NULL,
  CONSTRAINT Damaged_Products_Warehouse_PK
  PRIMARY KEY (Damaged_Products_Warehouse_Number),
  CONSTRAINT Damaged_Products_Warehouse1_FK
  FOREIGN KEY (Product_Number)
  REFERENCES Food_Product(Product_Number),
  CONSTRAINT Damaged_Products_Warehouse2_FK
  FOREIGN KEY (Department_S_Number)
  REFERENCES Sales_Department(Department_S_Number) );
```

```
CREATE TABLE Quality_Section (
  Q_Section_Number INT NOT NULL,
  Q_Section_Location VARCHAR2(100),
  Damaged_Products_Warehouse_Number INT NOT NULL,
  Valid_Products_Warehouse_Number INT NOT NULL,
  Product_Number INT NOT NULL,
  CONSTRAINT Quality_Section_PK
  PRIMARY KEY (Q_Section_Number),
  CONSTRAINT Quality_Section1_FK
  FOREIGN KEY (Product_Number)
  REFERENCES Food_Product(Product_Number),
  CONSTRAINT Quality_Section2_FK
  FOREIGN KEY (Damaged_Products_Warehouse_Number)
  REFERENCES Damaged_Products_Warehouse
  (Damaged_Products_Warehouse_Number),
  CONSTRAINT Quality_Section3_FK
  FOREIGN KEY (Valid_Products_Warehouse_Number)
  REFERENCES Valid_Products_Warehouse
  (Valid_Products_Warehouse_Number) );
```

```
CREATE TABLE Procurement_Section (
  Section_Number INT NOT NULL,
  Invoice_Number INT,
  Raw_Materials_Number INT NOT NULL,
  Bill NUMBER,
  Section_Location VARCHAR2(100),
  CONSTRAINT Procurement_Section_PK
  PRIMARY KEY (Section_Number),
  CONSTRAINT UC_InvoNumProc
  UNIQUE (Invoice_Number),
  CONSTRAINT Procurement_Section1_FK
  FOREIGN KEY (Raw_Materials_Number)
  REFERENCES Raw_Materials(Raw_Materials_Number) );
```

```
CREATE TABLE Consists (
  Raw_Materials_Number INT NOT NULL,
  Product_Number INT NOT NULL,
  Quentity NUMBER,
  CONSTRAINT Consists1_FK
  FOREIGN KEY (Product_Number)
  REFERENCES Food_Product(Product_Number),
  CONSTRAINT Consists2_FK
  FOREIGN KEY (Raw_Materials_Number)
  REFERENCES Raw_Materials(Raw_Materials_Number) );
```

الطلب الثالث:

- 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 PROCEDURE Production_Food (
 P_Product_Number IN NUMBER,
 P_Quantity IN NUMBER )
AS
BEGIN
  DECLARE
    Product_name VARCHAR2(50);
    SearchProduct INT;
    Material_Product Consists% ROWTYPE;
    l_count INT := 0;
 BEGIN
    SELECT COUNT(*) INTO SearchProduct
    FROM Food_Product
    WHERE Product_Number = P_Product_Number;
    IF SearchProduct = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Product not found');
    END IF;


FOR Material_Product IN
    ( SELECT * FROM Consists
     WHERE Product_Number = P_Product_Number )
     LOOP
      SELECT Quentity INTO l_count
      FROM Raw_Materials_Warehouse
      WHERE Raw_Materials_Number
      = Material_Product.Raw_Materials_Number;
      IF l_count < (P_Quantity * Material_Product.Quentity)
      THEN
       RAISE_APPLICATION_ERROR(-20002,
       'Required quantity not available');
      END IF;
      UPDATE Raw_Materials_Warehouse
      SET Quentity = Quentity - P_Quantity * Material_Product.Quentity
      WHERE Raw_Materials_Number
      = Material_Product.Raw_Materials_Number;
    END LOOP;
```

```
UPDATE Valid_Products_Warehouse
    SET Quentity = Quentity + P_Quantity
    WHERE Product_Number = P_Product_Number;

    UPDATE Food_Product
    SET Production_Date = SYSDATE
    WHERE Product_Number = P_Product_Number;

    SELECT PRODUCT_NAME INTO Product_name
    FROM Food_Product
    WHERE product_number = P_Product_Number;
    DBMS_OUTPUT.PUT_LINE('~ '||Product_name||'
    ~ manufacturing has been completed successfully.');
  END;
EXCEPTION
 WHEN OTHERS THEN
  ROLLBACK;
  RAISE;
END;
```

➢ فيما يلي شرح آلية عمل الإجرائية:

بدايةً تقوم الإجرائية باستقبال وسيطين رقم المنتج المراد تصنيعه والكمية المراد تصنيعها.

في البلوك Declare تم تعريف بعض المتغيرات المراد استخدامها داخل الإجرائية.

في البلوك Body تبدأ الإجرائية بجملة SELECT تقوم بالتحقق ما إذا كان رقم المنتج موجود ضمن قائمة المصنع لدينا أم لا.

في حال لم يكن لدينا رقم المنتج ضمن قائمة المصنع تعيد 0 وتظهر Exception يحوي رسالة Product not found.

في حال كان موجود تعيد 1 وتبدأ الإجرائية بتنفيذ التصنيع من حلقة الفور التي تقوم بدايةً بأخذ جميع أسطر الجدول Consists التي تطابق رقم المنتج وتضعها بالمتغير Material_Product الذي تم تعريفه مسبقًا بالبلوك Declare والذي يأخذ نوع الجدول Consists.

لدينا داخل الحلقة جملة SELECT تقوم بأخذ الكمية للمادة الأولية الواحدة من جدول مستودع المواد الأولية Raw_Materials_Number ووضعها في المتغير l_count الذي تم تعريفه مسبقًا، ثم في جملة IF شرطية نقوم بالتحقق مما إذا كان لدينا مواد أولية كافية لتصنيع الكمية المطلوبة من المنتج الغذائي وفي حال لم يكن لدينا يظهر رسالة Exception تحوي الجملة Required quantity not available.

أما في حال لدينا؛ فنأتي للجملة Update التي تقوم بتحديث عدد المواد الأولية داخل المستودع التي تم استعمالها لتصنيع المنتج حسب الكمية المطلوبة للمنتج.

بعد الحلقة نقوم بتحديث كمية المنتج في مستودع المنتجات الصالحة.

<< قد فرضنا أن كل المنتجات صالحة فتذهب لمستودع المنتجات الصالحة لا التالفة >>

ثم نقوم بتحديث تاريخ تصنيع المنتج الأخير ووضعه باستخدام الدالة SYSDATE المبنية في النظام.

ثم نقوم باستعمال جملة SELECT لجلب اسم المنتج المراد تصنيعه ووضعه في المتغير الذي تم تعريفه مسبقًا أيضًا واستعماله في أمر الطباعة الذي يشير لانتهاء عملية تصنيع المنتج.

لسلامة البيانات في القاعدة لدينا تم معالجة الاستثناء في حال ظهر أي خطأ غير متوقع أثناء عمل الإجرائية وتم وضع ROLLBACK بحيث إذا ظهر خطأ يتم التراجع عن أي عمل تم فعله من بداية تنفيذ الإجرائية وسيتم طرح الخطأ باستعمال RAISE.

الطلب الرابع:

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

قمنا بكتابة ثلاثة قوادح، الأول لمستودع المواد الأولية، الثاني لمستودع المواد الصالحة، الثالث لمستودع المواد التالفة:

القادح الأول:

```
CREATE OR REPLACE TRIGGER trigger_for_quantity_checks
BEFORE UPDATE OR INSERT ON Raw_Materials_Warehouse
REFERENCING OLD AS old NEW AS new
FOR EACH ROW
DECLARE
Material_name VARCHAR2(50);
BEGIN
SELECT Raw_Materials_Name INTO Material_name
FROM Raw_Materials
WHERE Raw_Materials_Number =: NEW.Raw_Materials_Number;

IF :NEW.Quentity <: NEW.Minimum THEN
DBMS_OUTPUT.PUT_LINE('Warning ..Your quantity of ~
'||Material_name||' ~ is less than ' :NEW.Minimum);
INSERT INTO stock_alerts (Quentity,message)
VALUES(:NEW.Quentity,'Warning ..Your quantity of ~
'||Material_name||' ~ is less than ' :NEW.Minimum);

ELSIF :NEW.Quentity >: NEW.Maximum THEN
DBMS_OUTPUT.PUT_LINE('Warning ..Your quantity of ~
'||Material_name||' ~ is greater than ' :NEW.Maximum);
INSERT INTO stock_alerts (Quentity,message)
VALUES (:NEW.Quentity,'Warning ..Your quantity of ~
'||Material_name||' ~ is greater than ' :NEW.Maximum);

ELSE
DBMS_OUTPUT.PUT_LINE('Quantity of ~'||Material_name'
~ in warehouse within natural limits.');

END IF;
DBMS_OUTPUT.PUT_LINE('-------------------');
END;
```

القادح الثاني:

```
CREATE OR REPLACE TRIGGER Trigger_For_VProducts_Quantity
BEFORE UPDATE OR INSERT ON Valid_Products_Warehouse
REFERENCING OLD AS old NEW AS new
FOR EACH ROW
DECLARE
  VProduct_name VARCHAR2(50);
BEGIN
  SELECT Product_Name INTO VProduct_name FROM Food_Product
  WHERE Product_Number =: NEW.Product_Number;

  IF :NEW.Quentity <: NEW.Minimum THEN
    DBMS_OUTPUT.PUT_LINE('Warning ..Your quantity of ~
    '||VProduct_name||' ~ is less than ' :NEW.Minimum);
    INSERT INTO stock_alerts (Quentity,message)
    VALUES(:NEW.Quentity,'Warning ..Your quantity of ~
    '||VProduct_name||' ~ is less than ' :NEW.Minimum);

  ELSIF :NEW.Quentity >: NEW.Maximum THEN
    DBMS_OUTPUT.PUT_LINE('Warning ..Your quantity of ~
    '||VProduct_name||' ~ is greater than ' :NEW.Maximum);
    INSERT INTO stock_alerts (Quentity,message)
    VALUES (:NEW.Quentity,'Warning ..Your quantity of ~
    '||VProduct_name||' ~ is greater than ' :NEW.Maximum);

  ELSE
    DBMS_OUTPUT.PUT_LINE('Quantity of ~
    '||VProduct_name||' ~ in warehouse within natural limits.');

  END IF;
  DBMS_OUTPUT.PUT_LINE('-------------------');
END;
```

القادح الثالث:

```
CREATE OR REPLACE TRIGGER Trigger_For_DProducts_Quantity
BEFORE UPDATE OR INSERT ON Damaged_Products_Warehouse
REFERENCING OLD AS old NEW AS new
FOR EACH ROW
DECLARE
  DProduct_name VARCHAR2(50);
BEGIN
  SELECT Product_Name INTO
  DProduct_name FROM Food_Product
  WHERE Product_Number =: NEW.Product_Number;

  IF :NEW.Quentity <: NEW.Minimum THEN
    DBMS_OUTPUT.PUT_LINE('Warning ..Your quantity of ~
    '||DProduct_name||' ~ is less than ' :NEW.Minimum);
    INSERT INTO stock_alerts (Quentity,message)
    VALUES(:NEW.Quentity,'Warning ..Your quantity of ~
    '||DProduct_name||' ~ is less than ' :NEW.Minimum);

  ELSIF :NEW.Quentity >: NEW.Maximum THEN
    DBMS_OUTPUT.PUT_LINE('Warning ..Your quantity of ~
    '||DProduct_name||' ~ is greater than ' :NEW.Maximum);
    INSERT INTO stock_alerts (Quentity,message)
    VALUES (:NEW.Quentity,'Warning ..Your quantity of ~
    '||DProduct_name||' ~ is greater than ' :NEW.Maximum);

  ELSE
    DBMS_OUTPUT.PUT_LINE('Quantity of ~
    '||DProduct_name||' ~ in warehouse within natural limits.');

  END IF;
  DBMS_OUTPUT.PUT_LINE('----------');
END;
```

➢ فيما يلي شرح آلية عمل القادح الأول:

يتم استخدام هذا القادح في قاعدة البيانات ويتم استدعاؤه عند حدوث عملية تحديث أو إدخال في جدول "Raw_Materials_Warehouse". يقوم بالتحقق من كمية المواد الأولية في المستودع وإصدار تحذير إذا كانت الكمية أقل من الحد الأدنى أو أكبر من الحد الأقصى المحددين.

يتم تعريف متغير Material_name من نوع VARCHAR2 بقيمة افتراضية 50..

يتم استعلام جدول "Raw_Materials" للحصول على اسم المادة الأولية باستخدام رقم المادة الأولية الجديد المدخل في جدول "Raw_Materials_Warehouse".

يتم فحص كمية المادة الأولية المدخلة في جدول "Raw_Materials_Warehouse" ومقارنتها بالحد الأدنى والحد الأقصى المحددين.

إذا كانت الكمية أقل من الحد الأدنى، يتم إظهار رسالة تحذيرية تفيد بأن الكمية أقل من الحد الأدنى ويتم إدخال تسجيل في جدول "stock_alerts" يحتوي على قيمة الكمية والرسالة التحذيرية.

إذا كانت الكمية أكبر من الحد الأقصى، يتم إظهار رسالة تحذيرية تفيد بأن الكمية أكبر من الحد الأقصى ويتم إدخال تسجيل في جدول "stock_alerts" يحتوي على قيمة الكمية والرسالة التحذيرية.

إذا كانت الكمية داخل الحدود المحددة، يتم إظهار رسالة تفيد بأن الكمية في المستودع ضمن الحدود الطبيعية.

- **فيما يلي شرح آلية عمل القادح الثاني:**

يتم استخدام هذا القادح لإنشاء أو تحديث حدث قبل الإدخال أو التحديث على جدول Valid_Products_Warehouse. يتم استدعاء الحدث عندما يتم تنفيذ أمر INSERT أو UPDATE على الجدول.

في البداية، يتم تعريف متغير VProduct_name من نوع VARCHAR2 بحجم 50.

ثم يتم استعلام من جدول Food_Product لتحديد قيمة VProduct_name باستخدام Product_Number من سجل العمود NEW.

إذا كانت قيمة NEW.Quentity أقل من NEW.Minimum، يتم طباعة رسالة تحذير تفيد أن الكمية لـ VProduct_name أقل من NEW.Minimum. ثم يتم إدخال سجل في جدول stock_alerts يحتوي على الكمية والرسالة.

إذا كانت قيمة NEW.Quentity أكبر من NEW.Maximum، يتم طباعة رسالة تحذير تفيد أن الكمية لـ VProduct_name أكبر من NEW.Maximum. ثم يتم إدخال سجل في جدول stock_alerts يحتوي على الكمية والرسالة.

إذا لم تتحقق أي من الشروط السابقة، يتم طباعة رسالة تفيد أن الكمية لـ VProduct_name في المستودع ضمن حدودها الطبيعية.

- **فيما يلي شرح آلية عمل القادح الثالث:**

لدينا قادح في قاعدة البيانات ويتم استدعاؤه عند حدوث عملية تحديث أو إدخال في جدول "Damaged_Products_Warehouse".

يقوم بالتحقق من كمية المنتج التالف في المستودع وإصدار تنبيه إذا كانت الكمية أقل من الحد الأدنى أو أكبر من الحد الأقصى المحددين.

يتم تعريف متغير DProduct_name من نوع VARCHAR2 بقيمة افتراضية 50.

يتم استعلام جدول "Food_Product" للحصول على اسم المنتج التالف باستخدام رقم المنتج الجديد المدخل في جدول "Damaged_Products_Warehouse".

يتم فحص كمية المنتج المدخلة في جدول "Damaged_Products_Warehouse" ومقارنتها بالحد الأدنى والحد الأقصى المحددين.

إذا كانت الكمية أقل من الحد الأدنى، يتم إظهار رسالة تحذيرية تفيد بأن الكمية أقل من الحد الأدنى ويتم إدخال تسجيل في جدول "stock_alerts" يحتوي على قيمة الكمية والرسالة التحذيرية.

إذا كانت الكمية أكبر من الحد الأقصى، يتم إظهار رسالة تحذيرية تفيد بأن الكمية أكبر من الحد الأقصى ويتم إدخال تسجيل في جدول "stock_alerts" يحتوي على قيمة الكمية والرسالة التحذيرية.

إذا كانت الكمية داخل الحدود المحددة، يتم إظهار رسالة تفيد بأن الكمية في المستودع ضمن الحدود الطبيعية.

➢ فيما يلي صور تنفيذ البرنامج لقيم مختلفة:

```
---------------------- EXECUTE PROCEDURE ----------------------
---------------------------------------------------------------
EXECUTE Production_Food(6, 2);
```

Script Output | Task completed in 0.039 seconds

```
PL/SQL procedure successfully completed.
```

Dbms Output | Buffer Size: 20000

YAMANONE

```
Quantity of ~ Milk ~ in warehouse within natural limits.
-----------------------------------------------
Quantity of ~ Sugar ~ in warehouse within natural limits.
-----------------------------------------------
Quantity of ~ Chocolate ~ in warehouse within natural limits.
-----------------------------------------------
~ Bakdash Ice Cream ~ manufacturing has been completed successfully.
```

```
---------------------- EXECUTE PROCEDURE ----------------------
---------------------------------------------------------------
EXECUTE Production_Food(6, 10);
```

Script Output | Task completed in 0.034 seconds

```
Error starting at line : 243 in command -
BEGIN Production_Food(6, 10); END;
Error report -
ORA-20002: Required quantity not available
ORA-06512: at "YAMANONE.PRODUCTION_FOOD", line 48
ORA-06512: at "YAMANONE.PRODUCTION_FOOD", line 26
ORA-06512: at "YAMANONE.PRODUCTION_FOOD", line 26
ORA-06512: at line 1
```

Dbms Output | Buffer Size: 20000

YAMANONE

```
------------------------- EXECUTE PROCEDURE -----------------------
-------------------------------------------------------------------
EXECUTE Production_Food(7, 10);
```

Script Output | Task completed in 0.036 seconds

```
Error starting at line : 243 in command -
BEGIN Production_Food(7, 10); END;
Error report -
ORA-20001: Product not found
ORA-06512: at "YAMANONE.PRODUCTION_FOOD", line 48
ORA-06512: at "YAMANONE.PRODUCTION_FOOD", line 17
ORA-06512: at line 1
```

Dbms Output | Buffer Size: 20000

YAMANONE

```
------------------------- EXECUTE PROCEDURE -----------------------
-------------------------------------------------------------------
EXECUTE Production_Food(2, 1);
```

Script Output | Task completed in 0.029 seconds

```
PL/SQL procedure successfully completed.
```

Dbms Output | Buffer Size: 20000

YAMANONE

```
Warning .. Your quantity of ~ Flour ~ is less than 400
-----------------------------------------------
Quantity of ~ Sugar ~ in warehouse within natural limits.
-----------------------------------------------
Warning .. Your quantity of ~ Cake ~ is greater than 300
-----------------------------------------------
~ Cake ~ manufacturing has been completed successfully.
```

Messages - Log