import mysql.connector  
from fpdf import FPDF  
  
  
def get\_product\_data(codigo\_produto):  
 conn = mysql.connector.connect(  
 host="localhost",  
 user="root",  
 password="madmax00st1",  
 database="FICHAS\_TECNICAS"  
 )  
 cursor = conn.cursor()  
 query = ("SELECT tipo\_tubo, quantidade, tamanho FROM produtos WHERE codigo = %s")  
 cursor.execute(query, (codigo\_produto,))  
  
 tubos = {}  
 for (tipo\_tubo, quantidade, tamanho) in cursor:  
 if tipo\_tubo not in tubos:  
 tubos[tipo\_tubo] = []  
 tubos[tipo\_tubo].extend([tamanho + 2] \* quantidade) # Adiciona 2mm a cada tamanho  
  
 cursor.close()  
 conn.close()  
 return tubos  
  
  
def cortar\_barras(tubos, comprimento\_barra):  
 barras = []  
 tipo\_contador = {}  
  
 for tubo\_tipo, tamanhos in tubos.items():  
 tamanhos.sort(reverse=True) # Ordenar tamanhos em ordem decrescente  
 tipo\_contador[tubo\_tipo] = tipo\_contador.get(tubo\_tipo, 0) + 1  
 while tamanhos:  
 barra\_atual = []  
 comprimento\_restante = comprimento\_barra  
  
 for tamanho in tamanhos[:]:  
 if tamanho <= comprimento\_restante:  
 barra\_atual.append(tamanho)  
 comprimento\_restante -= tamanho  
 tamanhos.remove(tamanho)  
  
 barras.append((tubo\_tipo, barra\_atual, comprimento\_restante, tipo\_contador[tubo\_tipo]))  
 tipo\_contador[tubo\_tipo] += 1  
  
 return barras  
  
  
def calcular\_percentual\_sobra(comprimento\_barra, sobra):  
 return (sobra / comprimento\_barra) \* 100  
  
  
def calcular\_desperdicio\_total(barras\_cortadas, comprimento\_barra):  
 total\_sobra = sum(sobra for \_, \_, sobra, \_ in barras\_cortadas)  
 total\_barras = len(barras\_cortadas)  
 desperdicio\_total = (total\_sobra / (total\_barras \* comprimento\_barra)) \* 100  
 return desperdicio\_total  
  
  
def analisar\_intervalos(codigo\_produto, intervalos):  
 comprimento\_barra = 6000  
 produto\_data = get\_product\_data(codigo\_produto)  
 resultados = []  
  
 for (inicio, fim) in intervalos:  
 melhor\_quantidade = None  
 menor\_desperdicio = float('inf')  
  
 for quantidade in range(inicio, fim + 1):  
 tubos = {}  
 for tipo, tamanhos in produto\_data.items():  
 tubos[tipo] = tamanhos \* quantidade  
  
 barras\_cortadas = cortar\_barras(tubos, comprimento\_barra)  
 desperdicio\_total = calcular\_desperdicio\_total(barras\_cortadas, comprimento\_barra)  
  
 if desperdicio\_total < menor\_desperdicio:  
 menor\_desperdicio = desperdicio\_total  
 melhor\_quantidade = quantidade  
  
 resultados.append((inicio, fim, melhor\_quantidade, menor\_desperdicio))  
  
 return resultados  
  
  
def gerar\_pdf(resultados, codigo\_produto):  
 pdf = FPDF()  
 pdf.add\_page()  
 pdf.set\_font("Arial", size=12)  
  
 # Título  
 pdf.cell(200, 10, f"Análise de Produção para o Produto {codigo\_produto}", ln=True, align='C')  
 pdf.ln(10)  
  
 # Resultados  
 for inicio, fim, quantidade, desperdicio in resultados:  
 detalhes = f"Lote de {inicio} a {fim}: ideal de {quantidade} peças com desperdício de {desperdicio:.2f}%"  
 pdf.multi\_cell(0, 10, detalhes, border=0)  
  
 # Melhor resultado  
 melhor\_lote = min(resultados, key=lambda x: x[3])  
 melhor\_detalhes = (f"\nEm resumo, o ideal para produzir com menor desperdício é de "  
 f"{melhor\_lote[2]} peças com o percentual de {melhor\_lote[3]:.2f}%")  
 pdf.multi\_cell(0, 10, melhor\_detalhes, border=0)  
  
 # Nome do arquivo PDF  
 nome\_arquivo = f"analise\_{codigo\_produto}.pdf"  
 pdf.output(nome\_arquivo)  
 return nome\_arquivo  
  
  
def main():  
 codigo\_produto = input("Digite o código do produto: ")  
  
 intervalos = [(1, 25), (26, 50), (51, 75), (76, 100)]  
  
 resultados = analisar\_intervalos(codigo\_produto, intervalos)  
  
 nome\_arquivo = gerar\_pdf(resultados, codigo\_produto)  
 print(f"PDF gerado com sucesso: {nome\_arquivo}")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()