**Полная шпаргалка для выполнения задания**

**1. Подготовка базы данных (Модуль 1)**

**Что нужно сделать:**

* Создать базу данных в СУБД (например, SQLite, MySQL или PostgreSQL).
* Наполнить её данными из предоставленных Excel-файлов.
* Построить ER-диаграмму.

**Файлы и скрипты:**

1. **database\_script.sql** – SQL-скрипт для создания таблиц.
2. **import\_data.py** – скрипт для загрузки данных из Excel в базу.
3. **generate\_er\_diagram.py** – скрипт для генерации ER-диаграммы (если СУБД не поддерживает автоматическое создание).

**Пример SQL-скрипта (database\_script.sql):**

sql

Copy

Download

-- Таблица "Типы материалов"

CREATE TABLE MaterialTypes (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

loss\_percentage REAL NOT NULL

);

-- Таблица "Материалы"

CREATE TABLE Materials (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

type\_id INTEGER NOT NULL,

price REAL NOT NULL,

unit TEXT NOT NULL,

quantity\_in\_pack REAL NOT NULL,

quantity\_in\_stock REAL NOT NULL,

min\_quantity REAL NOT NULL,

FOREIGN KEY (type\_id) REFERENCES MaterialTypes(id)

);

-- Таблица "Типы продукции"

CREATE TABLE ProductTypes (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

coefficient REAL NOT NULL

);

-- Таблица "Продукция"

CREATE TABLE Products (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

article TEXT NOT NULL,

type\_id INTEGER NOT NULL,

min\_partner\_price REAL NOT NULL,

FOREIGN KEY (type\_id) REFERENCES ProductTypes(id)

);

-- Таблица "Материалы для продукции"

CREATE TABLE MaterialProducts (

material\_id INTEGER NOT NULL,

product\_id INTEGER NOT NULL,

required\_quantity REAL NOT NULL,

PRIMARY KEY (material\_id, product\_id),

FOREIGN KEY (material\_id) REFERENCES Materials(id),

FOREIGN KEY (product\_id) REFERENCES Products(id)

);

**Скрипт для импорта данных (import\_data.py):**

python

Copy

Download

import sqlite3

import pandas as pd

# Подключение к базе данных

conn = sqlite3.connect("materials.db")

cursor = conn.cursor()

# Загрузка данных из Excel в таблицы

def import\_excel\_to\_db():

# Импорт типов материалов

df = pd.read\_excel("Material\_type\_import.xlsx")

df.to\_sql("MaterialTypes", conn, if\_exists="replace", index=False)

# Импорт материалов

df = pd.read\_excel("Materials\_import.xlsx")

df.to\_sql("Materials", conn, if\_exists="replace", index=False)

# Импорт типов продукции

df = pd.read\_excel("Product\_type\_import.xlsx")

df.to\_sql("ProductTypes", conn, if\_exists="replace", index=False)

# Импорт продукции

df = pd.read\_excel("Products\_import.xlsx")

df.to\_sql("Products", conn, if\_exists="replace", index=False)

# Импорт связи материалов и продукции

df = pd.read\_excel("Material\_products\_\_import.xlsx")

df.to\_sql("MaterialProducts", conn, if\_exists="replace", index=False)

import\_excel\_to\_db()

conn.close()

**2. Разработка модуля для работы с материалами (Модуль 2)**

**Что нужно сделать:**

* Реализовать просмотр списка материалов.
* Добавить расчёт требуемого количества материала для производства.

**Файлы и классы:**

1. **material.py** – класс для работы с материалами.
2. **database\_handler.py** – класс для работы с базой данных.
3. **main\_app.py** – главное приложение с интерфейсом.

**Пример кода (material.py):**

python

Copy

Download

class Material:

def \_\_init\_\_(self, id, name, type, price, unit, quantity\_in\_pack, quantity\_in\_stock, min\_quantity):

self.id = id

self.name = name

self.type = type

self.price = price

self.unit = unit

self.quantity\_in\_pack = quantity\_in\_pack

self.quantity\_in\_stock = quantity\_in\_stock

self.min\_quantity = min\_quantity

def calculate\_required\_quantity(self, db\_handler):

query = """

SELECT SUM(required\_quantity)

FROM MaterialProducts

WHERE material\_id = ?

"""

result = db\_handler.execute\_query(query, (self.id,))

return result[0][0] if result[0][0] else 0.0

**Пример кода (database\_handler.py):**

python

Copy

Download

import sqlite3

class DatabaseHandler:

def \_\_init\_\_(self, db\_path):

self.connection = sqlite3.connect(db\_path)

self.cursor = self.connection.cursor()

def execute\_query(self, query, params=None):

if params:

self.cursor.execute(query, params)

else:

self.cursor.execute(query)

return self.cursor.fetchall()

def close(self):

self.connection.close()

**Пример кода (main\_app.py):**

python

Copy

Download

from database\_handler import DatabaseHandler

from material import Material

def display\_materials():

db = DatabaseHandler("materials.db")

query = "SELECT \* FROM Materials"

materials = db.execute\_query(query)

for mat in materials:

material = Material(\*mat)

required = material.calculate\_required\_quantity(db)

print(f"{material.name}: {material.quantity\_in\_stock} {material.unit}, требуется: {required:.2f} {material.unit}")

db.close()

if \_\_name\_\_ == "\_\_main\_\_":

display\_materials()

**3. Интерфейс для добавления/редактирования материалов (Модуль 3)**

**Что нужно сделать:**

* Создать форму для добавления и редактирования материалов.
* Реализовать валидацию данных (цена не отрицательная и т. д.).

**Файлы и классы:**

1. **material\_form.py** – форма для работы с материалами.
2. **main\_window.py** – главное окно приложения.

**Пример кода (material\_form.py):**

python

Copy

Download

import tkinter as tk

from tkinter import messagebox

class MaterialForm:

def \_\_init\_\_(self, parent, db\_handler, material\_id=None):

self.parent = parent

self.db = db\_handler

self.material\_id = material\_id

self.window = tk.Toplevel(parent)

self.window.title("Добавить материал" if not material\_id else "Редактировать материал")

# Поля формы

tk.Label(self.window, text="Наименование:").grid(row=0, column=0)

self.name\_entry = tk.Entry(self.window)

self.name\_entry.grid(row=0, column=1)

# Кнопка сохранения

tk.Button(self.window, text="Сохранить", command=self.save).grid(row=7, column=1)

def save(self):

name = self.name\_entry.get()

if not name:

messagebox.showerror("Ошибка", "Наименование не может быть пустым!")

return

# Дополнительная логика сохранения...

**Пример кода (main\_window.py):**

python

Copy

Download

import tkinter as tk

from material\_form import MaterialForm

from database\_handler import DatabaseHandler

class MainWindow:

def \_\_init\_\_(self):

self.root = tk.Tk()

self.root.title("Система учёта материалов")

self.db = DatabaseHandler("materials.db")

# Кнопка "Добавить материал"

tk.Button(self.root, text="Добавить материал", command=self.open\_add\_form).pack()

def open\_add\_form(self):

MaterialForm(self.root, self.db)

if \_\_name\_\_ == "\_\_main\_\_":

app = MainWindow()

app.root.mainloop()

app.db.close()

**4. Модуль расчёта продукции (Модуль 4)**

**Что нужно сделать:**

* Реализовать метод для расчёта количества продукции из сырья.

**Файлы и классы:**

1. **production\_calculator.py** – класс для расчётов.

**Пример кода:**

python

Copy

Download

class ProductionCalculator:

@staticmethod

def calculate\_products(product\_type\_id, material\_type\_id, raw\_quantity, param1, param2, db\_handler):

# Проверка существования типов

product\_type = db\_handler.execute\_query(

"SELECT coefficient FROM ProductTypes WHERE id = ?",

(product\_type\_id,)

)

material\_type = db\_handler.execute\_query(

"SELECT loss\_percentage FROM MaterialTypes WHERE id = ?",

(material\_type\_id,)

)

if not product\_type or not material\_type:

return -1

# Расчёт

required\_per\_unit = param1 \* param2 \* product\_type[0][0]

total\_required = required\_per\_unit \* (1 + material\_type[0][0])

if total\_required <= 0:

return -1

return int(raw\_quantity / total\_required)

**Итоговая структура проекта**

Copy

Download

/project

│

├── /data

│ ├── Materials\_import.xlsx

│ ├── Material\_type\_import.xlsx

│ ├── Material\_products\_\_import.xlsx

│ ├── Product\_type\_import.xlsx

│ └── Products\_import.xlsx

│

├── /src

│ ├── database\_script.sql

│ ├── import\_data.py

│ ├── material.py

│ ├── database\_handler.py

│ ├── main\_app.py

│ ├── material\_form.py

│ ├── main\_window.py

│ └── production\_calculator.py

│

└── materials.db (создаётся после запуска import\_data.py)

**Что делать, если вы не умеете писать код?**

1. **Используйте готовые скрипты** из этой шпаргалки.
2. **Запускайте их по порядку**:
   * Сначала database\_script.sql (в SQLite или другой СУБД).
   * Затем import\_data.py для загрузки данных.
   * Потом main\_app.py для просмотра материалов.
3. **Для интерфейса** скопируйте код из main\_window.py и material\_form.py.
4. **Для расчётов** используйте production\_calculator.py.

Если что-то не работает, пишите в комментариях — помогу адаптировать код под ваши нужды!