```
# ================================
# ■ StudyMate - AI Study Assistant
# Google Colab Compatible Code
# ================================

# Install required libraries
!pip install transformers sentencepiece datasets pdfplumber python-docx

# Import libraries
import pdfplumber
import docx
from transformers import pipeline
from google.colab import files
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

# ================================
# ■ Step 1: Upload Study Material
# ================================
uploaded = files.upload()

# Read uploaded file
file_name = list(uploaded.keys())[0]

text = ""

if file_name.endswith(".txt"):
    with open(file_name, "r", encoding="utf-8") as f:
        text = f.read()

elif file_name.endswith(".pdf"):
    with pdfplumber.open(file_name) as pdf:
        for page in pdf.pages:
            text += page.extract_text() + "\n"

elif file_name.endswith(".docx"):
    doc = docx.Document(file_name)
    for para in doc.paragraphs:
        text += para.text + "\n"

else:
    print("■ Unsupported file format. Please upload .txt, .pdf, or .docx")

# Show first 500 characters of input text
print("■ Extracted Text Preview:\n", text[:500])


# ================================
# ■ Step 2: Summarization
# ================================
print("\n■ Generating Summary...")

def summarize_text(text, max_chunk_length=1000):
    summarizer = pipeline("summarization", model="facebook/bart-large-cnn")
    chunks = [text[i:i+max_chunk_length] for i in range(0, len(text), max_chunk_length)]
    summaries = []
    for chunk in chunks:
        summary = summarizer(chunk, max_length=200, min_length=40, do_sample=False)
        summaries.append(summary[0]['summary_text'])
    return " ".join(summaries)

summary = summarize_text(text)
print("\n■ Summary:\n", summary)


# ================================
# ■ Step 3: Question Generation
# ================================
print("\n■ Generating Questions...")

import re

def split_text(text, max_chunk_length=500):
```

```
    sentences = re.split(r'(?<=[.!?]) +', text)
    chunks, chunk = [], ""
    for sentence in sentences:
        if len(chunk) + len(sentence) <= max_chunk_length:
            chunk += " " + sentence
        else:
            chunks.append(chunk.strip())
            chunk = sentence
    if chunk:
        chunks.append(chunk.strip())
    return chunks

qg_model_name = "iarfmoose/t5-base-question-generator"
tokenizer = AutoTokenizer.from_pretrained(qg_model_name)
model = AutoModelForSeq2SeqLM.from_pretrained(qg_model_name)

def generate_questions(text, num_questions=5, max_chunk_length=500):
    chunks = split_text(text, max_chunk_length)
    questions = set()
    for chunk in chunks:
        input_text = "generate questions: " + chunk
        inputs = tokenizer.encode(input_text, return_tensors="pt", max_length=512, truncation=True)
        outputs = model.generate(
            inputs, max_length=64, num_beams=6,
            early_stopping=True, num_return_sequences=num_questions
        )
        for out in outputs:
            q = tokenizer.decode(out, skip_special_tokens=True)
            questions.add(q)
    return list(questions)

questions = generate_questions(text, num_questions=5, max_chunk_length=500)

print("\n■ Sample Questions:")
for i, q in enumerate(questions[:10], 1):
    print(f"{i}. {q}")


# ================================
# ■ Step 4: Save Summary & Questions
# ================================
with open("study_summary.txt", "w", encoding="utf-8") as f:
    f.write("■ Summary:\n" + summary + "\n\n")
    f.write("■ Questions:\n")
    for i, q in enumerate(questions, 1):
        f.write(f"{i}. {q}\n")

files.download("study_summary.txt")
```