```
#importing all the necessary libraries
import os
import pandas as pd
import re
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from transformers import BertTokenizerFast, BertForSequenceClassification, Trainer, TrainingArguments, logging
import torch


#Disabling weights & biases logging
os.environ['WANDB_DISABLED'] = "true"
#Disabling Transformers advisory and info logs
os.environ['TRANSFORMERS_NO_ADVISORY_WARNINGS'] = "true"
logging.set_verbosity_error()


#loading the dataset
df = pd.read_csv('https://raw.githubusercontent.com/psabhay2003/BCGX-GenAI/refs/heads/main/financial_chatbot_data.csv')
df
```

	question	response
0	What is Apple's revenue in 2022?	3.943280e+11
1	What is Apple's net income in 2022?	9.980300e+10
2	What are Apple's total assets in 2022?	3.527550e+11
3	What are Apple's total liabilities in 2022?	3.020830e+11
4	What is Apple's operating cashflow in 2022?	1.221510e+11
...	...	...
142	What is Tesla's investing cashflow growth in 2...	2.055313e+01
143	What is Tesla's gross margin growth in 2024?	-1.189128e+00
144	What is Tesla's profit margin in 2024?	7.322141e+00
145	What is Tesla's return on assets in 2024?	5.859753e+00
146	What is Tesla's return on equity in 2024?	9.708198e+00

147 rows × 2 columns

Next steps: Generate code with df View recommended plots New interactive sheet

```
#Defining intent mapping based on keywords
intent_keywords = {
    'revenue': ['revenue'],
    'net_income': ['net income'],
    'assets': ['total assets'],
    'liabilities': ['total liabilities'],
    'operating_cashflow': ['operating cashflow'],
    'financing_cashflow': ['financing cashflow'],
    'investing_cashflow': ['investing cashflow'],
    'profit_margin': ['profit margin'],
    'gross_margin': ['gross margin'],
}

def assign_intent(question):
    q = question.lower() #lowercasing 'question' column so that it is not case-sensitive
    for intent, keywords in intent_keywords.items():
        for kw in keywords:
            if kw in q:
                return intent
    return 'other'

#Applying intent labels on dataframe
df['intent'] = df['question'].apply(assign_intent)
intents = sorted(df['intent'].unique()) #extracting all unique names from the new column, and sorting them alphabetically
label2id = {label: i for i, label in enumerate(intents)} #mapping each intent name to a unique integer index (0, 1, 2, …)
id2label = {i: label for label, i in label2id.items()} #invert mapping
df['label'] = df['intent'].map(label2id)
df
```

	question	response	intent	label
0	What is Apple's revenue in 2022?	3.943280e+11	revenue	9
1	What is Apple's net income in 2022?	9.980300e+10	net_income	5
2	What are Apple's total assets in 2022?	3.527550e+11	assets	0
3	What are Apple's total liabilities in 2022?	3.020830e+11	liabilities	4
4	What is Apple's operating cashflow in 2022?	1.221510e+11	operating_cashflow	6
...	...	...	...	...
142	What is Tesla's investing cashflow growth in 2...	2.055313e+01	investing_cashflow	3
143	What is Tesla's gross margin growth in 2024?	-1.189128e+00	gross_margin	2
144	What is Tesla's profit margin in 2024?	7.322141e+00	profit_margin	8
145	What is Tesla's return on assets in 2024?	5.859753e+00	other	7
146	What is Tesla's return on equity in 2024?	9.708198e+00	other	7

147 rows × 4 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
#Train-test split
train_df, test_df = train_test_split(df, test_size=0.2, random_state=42, stratify=df['label'])
#stratify ensures each label class appears in the train and test sets in roughly the same proportions as in the full dataset.

#Tokenization using BertTokenizer
tokenizer = BertTokenizerFast.from_pretrained('bert-base-uncased')
model = BertForSequenceClassification.from_pretrained(
    'bert-base-uncased',
    num_labels=len(label2id),
    id2label=id2label,
    label2id=label2id
)

#Dataset class
class QADataset(torch.utils.data.Dataset):
    def __init__(self, questions, labels, tokenizer):
        self.encodings = tokenizer(questions.tolist(), truncation=True, padding=True)
        self.labels = labels.tolist()

    def __len__(self):
        return len(self.labels)

    def __getitem__(self, idx):
        item = {key: torch.tensor(val[idx]) for key, val in self.encodings.items()}
        item['labels'] = torch.tensor(self.labels[idx])
        return item

train_dataset = QADataset(train_df['question'], train_df['label'], tokenizer)
test_dataset = QADataset(test_df['question'], test_df['label'], tokenizer)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as s
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 100%                                          48.0/48.0 [00:00<00:00, 2.39kB/s]
vocab.txt: 100%                                          232k/232k [00:00<00:00, 2.02MB/s]
tokenizer.json: 100%                                          466k/466k [00:00<00:00, 7.01MB/s]
config.json: 100%                                          570/570 [00:00<00:00, 28.2kB/s]
Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back to regular HTTP download. For better p
WARNING:huggingface_hub.file_download:Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back
model.safetensors: 100%                                          440M/440M [00:08<00:00, 56.3MB/s]
```

```
#Defining the training arguments
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=3,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    eval_strategy='epoch',
    save_strategy='epoch',
    logging_dir='./logs',
    logging_steps=10,
    load_best_model_at_end=True,
    metric_for_best_model='accuracy',
    report_to=[]  # disable all logging integrations, including wandb
)

#Defining a A callback function that the hugging face trainer will use to compute evaluation metrics
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    preds = np.argmax(logits, axis=1)
    return {'accuracy': accuracy_score(labels, preds)}

#Defining the trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    compute_metrics=compute_metrics
)

#Training the model
trainer.train()
```

```
{'loss': 1.4462, 'grad_norm': 8.962411880493164, 'learning_rate': 4e-05, 'epoch': 0.6666666666666666}
{'eval_loss': 0.878197431564331, 'eval_accuracy': 1.0, 'eval_runtime': 1.6621, 'eval_samples_per_second': 18.05, 'eval_steps_per_sec
{'loss': 0.992, 'grad_norm': 6.452608108520508, 'learning_rate': 2.8888888888888888e-05, 'epoch': 1.3333333333333333}
{'loss': 0.7008, 'grad_norm': 4.687027454376221, 'learning_rate': 1.7777777777777778e-05, 'epoch': 2.0}
{'eval_loss': 0.5285896062850952, 'eval_accuracy': 1.0, 'eval_runtime': 1.6319, 'eval_samples_per_second': 18.384, 'eval_steps_per_s
{'loss': 0.5505, 'grad_norm': 4.756647109985352, 'learning_rate': 6.666666666666667e-06, 'epoch': 2.6666666666666665}
{'eval_loss': 0.4308786392211914, 'eval_accuracy': 1.0, 'eval_runtime': 1.5817, 'eval_samples_per_second': 18.967, 'eval_steps_per_s
{'train_runtime': 214.8508, 'train_samples_per_second': 1.634, 'train_steps_per_second': 0.209, 'train_loss': 0.879258378346761, 'ep
TrainOutput(global_step=45, training_loss=0.879258378346761, metrics={'train_runtime': 214.8508, 'train_samples_per_second': 1.634,
'train_steps_per_second': 0.209, 'train_loss': 0.879258378346761, 'epoch': 3.0})
```

```
#Model Evaluation
eval_results = trainer.evaluate()
print("Evaluation results:\n", eval_results)
print("Classification report on test set:")
preds_output = trainer.predict(test_dataset)
preds = np.argmax(preds_output.predictions, axis=1)
print(classification_report(test_df['label'], preds, target_names=intents))
```

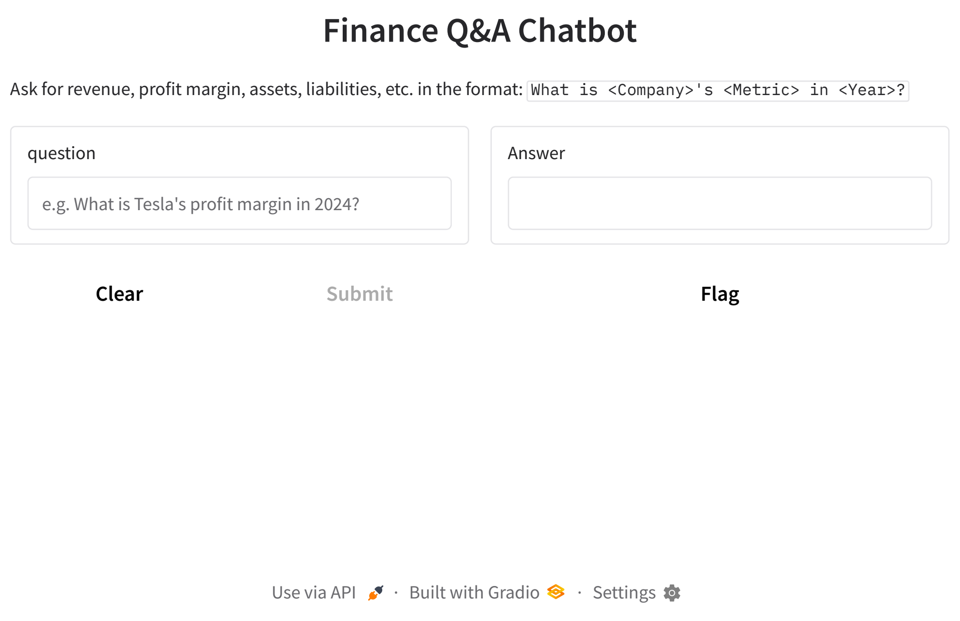
```
{'eval_loss': 0.878197431564331, 'eval_accuracy': 1.0, 'eval_runtime': 3.132, 'eval_samples_per_second': 9.578, 'eval_steps_per_seco
Evaluation results:
 {'eval_loss': 0.878197431564331, 'eval_accuracy': 1.0, 'eval_runtime': 3.132, 'eval_samples_per_second': 9.578, 'eval_steps_per_sec
Classification report on test set:
                    precision    recall  f1-score   support

            assets       1.00      1.00      1.00         3
financing_cashflow       1.00      1.00      1.00         3
      gross_margin       1.00      1.00      1.00         3
investing_cashflow       1.00      1.00      1.00         3
       liabilities       1.00      1.00      1.00         3
        net_income       1.00      1.00      1.00         3
operating_cashflow       1.00      1.00      1.00         3
             other       1.00      1.00      1.00         4
     profit_margin       1.00      1.00      1.00         2
           revenue       1.00      1.00      1.00         3

          accuracy                           1.00        30
         macro avg       1.00      1.00      1.00        30
      weighted avg       1.00      1.00      1.00        30
```

```
#Defining extraction & retrieval pipeline
class FinanceQA:
    def __init__(self, df, model, tokenizer, label2intent_map):
        self.df = df
        self.model = model
        self.tokenizer = tokenizer
        self.label2intent = label2intent_map
        self.device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
        self.model.to(self.device)
        #Precompile regex
        self.pattern = re.compile(r"What (?:is|are) (?P<company>.+?)'s (?P<entity>.+?) in (?P<year>\d{4})\?")

    def predict_intent(self, question):
        inputs = self.tokenizer(question, return_tensors='pt', truncation=True, padding=True).to(self.device)
        outputs = self.model(**inputs)
        logits = outputs.logits.detach().cpu().numpy()
        intent_id = np.argmax(logits, axis=1)[0]
        return self.label2intent[intent_id]

    def extract_slots(self, question):
        match = self.pattern.match(question)
        if not match:
            raise ValueError("Could not extract slots from question: {}".format(question))
        return match.group('company'), match.group('entity'), match.group('year')

    def lookup(self, intent, company, year):
        intent_key = intent
        subset = self.df[
            (self.df['intent'] == intent_key) &
            (self.df['question'].str.contains(company, case=False)) &
            (self.df['question'].str.contains(year))
        ]
        if subset.empty:
            raise ValueError("No data found for {}, {}, {}".format(company, intent, year))
        return subset['response'].values[0]

    def answer(self, question):
        intent = self.predict_intent(question)
        company, entity, year = self.extract_slots(question)
        value = self.lookup(intent, company, year)
        return value

#Testing the pipeline
pipeline = FinanceQA(df, model, tokenizer, id2label)

sample_questions = [
    "What is Microsoft's operating cashflow in 2023?",
    "What is Apple's net income in 2024?",
]

for q in sample_questions:
    try:
        ans = pipeline.answer(q)
        print(f"Q: {q}\nA: {ans}\n")
    except Exception as e:
        print(f"Error for question '{q}': {e}")
```

```
Q: What is Microsoft's operating cashflow in 2023?
A: 87582000000.0

Q: What is Apple's net income in 2024?
A: 93736000000.0
```

```
#Deploying the model using Gradio for quick demo of chatbot
!pip install gradio
import gradio as gr

def gradio_answer_fn(question: str) -> str:
    try:
        return pipeline.answer(question)
    except Exception as e:
        return f"Error: {e}"

iface = gr.Interface(
    fn=gradio_answer_fn,
    inputs=gr.Textbox(lines=1, placeholder="e.g. What is Tesla's profit margin in 2024?"),
    outputs=gr.Textbox(label="Answer"),
    title="Finance Q&A Chatbot",
    description="Ask for revenue, profit margin, assets, liabilities, etc. in the format:\n"
                "`What is <Company>'s <Metric> in <Year>?`"
)

#Launching the UI
iface.launch(share=True)
```

```
Requirement already satisfied: gradio in /usr/local/lib/python3.11/dist-packages (5.25.2)
Requirement already satisfied: aiofiles<25.0,>=22.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (24.1.0)
Requirement already satisfied: anyio<5.0,>=3.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.9.0)
Requirement already satisfied: fastapi<1.0,>=0.115.2 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.115.12)
Requirement already satisfied: ffmpy in /usr/local/lib/python3.11/dist-packages (from gradio) (0.5.0)
Requirement already satisfied: gradio-client==1.8.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (1.8.0)
Requirement already satisfied: groovy~=0.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.1.2)
Requirement already satisfied: httpx>=0.24.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.28.1)
Requirement already satisfied: huggingface-hub>=0.28.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.30.2)
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.1.6)
Requirement already satisfied: markupsafe<4.0,>=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.0.2)
Requirement already satisfied: numpy<3.0,>=1.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.0.2)
Requirement already satisfied: orjson~=3.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.10.16)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from gradio) (24.2)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.2.2)
Requirement already satisfied: pillow<12.0,>=8.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (11.1.0)
Requirement already satisfied: pydantic<2.12,>=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.11.3)
Requirement already satisfied: pydub in /usr/local/lib/python3.11/dist-packages (from gradio) (0.25.1)
Requirement already satisfied: python-multipart>=0.0.18 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.0.20)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (6.0.2)
Requirement already satisfied: ruff>=0.9.3 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.11.6)
Requirement already satisfied: safehttpx<0.2.0,>=0.1.6 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.1.6)
Requirement already satisfied: semantic-version~=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.10.0)
Requirement already satisfied: starlette<1.0,>=0.40.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.46.2)
Requirement already satisfied: tomlkit<0.14.0,>=0.12.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.13.2)
Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.15.2)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.13.2)
Requirement already satisfied: uvicorn>=0.14.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.34.2)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.8.0->gradio) (2025.3.2)
Requirement already satisfied: websockets<16.0,>=10.0 in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.8.0->gradio
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (3.10)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (1.3.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (2025.1.31)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (1.0.8)
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.11/dist-packages (from httpcore==1.*->httpx>=0.24.1->gradi
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (3.18.0)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (4.67
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio)
Requirement already satisfied: pydantic-core==2.33.1 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio)
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradi
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (8.1.8)
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (1.5.4
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (13.9.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas<3.0,>=1.0->
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0,>=0.
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0,>=
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.28.
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich>=10.11.0->ty
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://c53efb18cfead221c3.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working
```



Finance Q&A Chatbot

Ask for revenue, profit margin, assets, liabilities, etc. in the format: `What is <Company>'s <Metric> in <Year>?`

question

e.g. What is Tesla's profit margin in 2024?

Answer