Detailed Explanation of `AI_Interview_Question_Generator.ipynb`

1. Installing Required Libraries

```
!pip install transformers accelerate ipywidgets --quiet
```

This command installs three essential libraries:

- `transformers`: For loading and using large language models (LLMs).
- `accelerate`: For performance improvements on different hardware setups.
- `ipywidgets`: For creating interactive elements inside Jupyter notebooks.

---

2. Importing Required Libraries

```
import ipywidgets as widgets
from IPython.display import display, Markdown
from transformers import pipeline, AutoTokenizer, AutoModelForCausalLM
import torch
```

- `ipywidgets` is used to create text boxes, buttons, and numeric input for interaction.
- `display`, `Markdown` are used to output styled content in the notebook.
- `transformers`:
  - `pipeline`: A high-level API for various NLP tasks like text generation.
  - `AutoTokenizer` and `AutoModelForCausalLM` load a compatible tokenizer and causal language model.
- `torch` is used by the model backend.

---

3. Load the Model

```
model_id = "TinyLlama/TinyLlama-1.1B-Chat-v1.0"
tokenizer = AutoTokenizer.from_pretrained(model_id)
model = AutoModelForCausalLM.from_pretrained(model_id, torch_dtype="auto", device_map="cpu")
generator = pipeline("text-generation", model=model, tokenizer=tokenizer)
```

- Loads a small open-source model `TinyLlama`.
- Tokenizer converts text into tokens.
- Model is loaded and assigned to CPU.

- The `generator` pipeline is created for text generation.

---

4. Create UI Widgets

role_input = widgets.Text(...)

num_questions_input = widgets.BoundedIntText(...)

generate_button = widgets.Button(...)

output = widgets.Output()

These widgets take user input:

- Job title or resume
- Number of questions to generate (1 to 10)
- A generate button
- An output area to display results

---

5. Prompt Builders

def build_questions_prompt(...)

def build_answer_prompt(...)

These helper functions return formatted prompts that will be fed to the model:

- To get interview questions
- To get sample answers

---

6. Generate Questions with the Model

```
def get_questions(role_or_resume, num_questions):
    ...
```

- Calls the generator with a prompt.
- Extracts questions using regex.
- Handles both numbered and unnumbered output from the model.

---

7. Generate Answer with the Model

def get_answer(question, role_or_resume):

    ...

- Calls the generator to get a model-generated answer.

- Strips out the prompt to get clean output.

---

8. Callback Function

def on_submit(sender):

    ...

- Runs when the "Generate" button is clicked.

- Calls question and answer functions.

- Displays output using `Markdown`.

---

9. Link Button to Callback

generate_button.on_click(on_submit)

---

10. Display the Interface

display(role_input, num_questions_input, generate_button, output)

- This sets up the visible input/output fields in the notebook.