Use your Open Api key in this code to get the result:

(I have tested many models but Only Open API Models are Capable of correctly doing this Task correctly)
(I am Submitting an example of Json that would be Extracted from it because I had limited tokens to use it)

```
import openai
import os
import json
import fitz  # PyMuPDF for text extraction
import pdfplumber  # For table extraction
import pytesseract  # OCR for images
import cv2
import numpy as np
import re
from PIL import Image

# Set your OpenAI API key
openai.api_key = "your-openai-api-key"

# Function to extract text from PDFs efficiently
def extract_text_from_pdf(pdf_path):
    """Extracts text, tables, and OCR-processed text from a PDF file."""
    text = ""
```

```
# Extract plain text using PyMuPDF
with fitz.open(pdf_path) as doc:
    for page in doc:
        text += page.get_text("text") + "\n"

# Extract tables using pdfplumber
table_text = ""
with pdfplumber.open(pdf_path) as pdf:
    for page in pdf.pages:
        tables = page.extract_table()
        if tables:
            for row in tables:
                clean_row = [cell if cell is not None else "" for cell in row]
                table_text += " | ".join(clean_row) + "\n"

# Extract images and use OCR (if needed)
ocr_text = ""
if len(text.strip()) == 0:  # Run OCR only if no text was extracted
    with fitz.open(pdf_path) as doc:
        for page in doc:
            for img in page.get_images(full=True):
                xref = img[0]
                base_image = doc.extract_image(xref)
                img_bytes = base_image["image"]
```

```
                # Convert image bytes to OpenCV format
                image_np = np.frombuffer(img_bytes, np.uint8)
                image = cv2.imdecode(image_np, cv2.IMREAD_GRAYSCALE)

                # Apply thresholding for better OCR
                _, image = cv2.threshold(image, 128, 255, cv2.THRESH_BINARY)

                # Convert to PIL format for OCR
                pil_image = Image.fromarray(image)

                # Run OCR with timeout
                try:
                    ocr_text += pytesseract.image_to_string(pil_image, timeout=5) +
"\n"
                except RuntimeError:
                    print(f"⚠ Skipping slow OCR image in {pdf_path}")

    full_text = text + "\n[TABLE DATA]\n" + table_text + "\n[OCR DATA]\n" +
ocr_text
    return full_text

# Function to process financial data using OpenAI API
def extract_financial_data_with_gpt(text):
    """Uses OpenAI GPT model to extract structured financial data."""

    extraction_prompt = """
```

```
    Extract the following financial entities from the financial report and return the result in JSON format:
    {
        "Company Name": "",
        "Report Date": "",
        "Revenue from Operations": "",
        "Profit Before Tax": "",
        "Net Profit": "",
        "Expenses Breakdown": [
            {
                "name": "",
                "value": ""
            }
        ],
        "Additional Financial Figures": []
    }
    Ensure that the output is valid JSON and do not include any extra text.
    """

    response = openai.ChatCompletion.create(
        model="gpt-4-turbo",  # You can replace this with a different model like "gpt-3.5-turbo"
        messages=[
            {"role": "system", "content": "You are a financial data extraction assistant."},
            {"role": "user", "content": extraction_prompt + "\n\n" + text[:5000]}  # Limit input text to fit model
```

```
        ],
        temperature=0.2  # Ensure deterministic output
    )

    # Extract JSON output
    result = response["choices"][0]["message"]["content"]
    try:
        json_data = json.loads(result)
    except json.JSONDecodeError:
        print("⚠ GPT model output is not valid JSON. Returning raw output.")
        return result

    return json_data

# Define PDF file paths
pdf_files = [
    "/mnt/data/Amaar raja Earnings Summary.pdf",
    "/mnt/data/1_FinancialResults_05022025142214.pdf"
]

# Process PDFs and Extract Data
extracted_data = {}

for pdf_path in pdf_files:
    print(f"📄 Processing {pdf_path}...")
    extracted_text = extract_text_from_pdf(pdf_path)
```

```
    extracted_info = extract_financial_data_with_gpt(extracted_text)
    extracted_data[os.path.basename(pdf_path)] = extracted_info

# Display the extracted JSON data
import ace_tools as tools
tools.display_dataframe_to_user(name="Extracted Financial Data",
dataframe=json.dumps(extracted_data, indent=4, ensure_ascii=False))
```