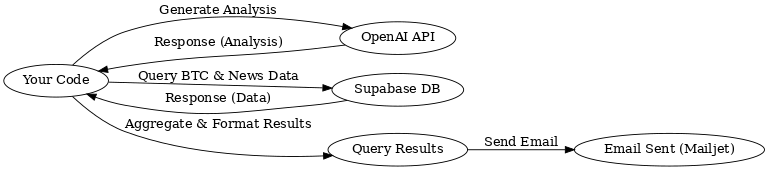
# Technical Documentation: Automated Analysis and Email Workflow

This document outlines the technical workflow for an automated system that retrieves data from Supabase, generates an analysis using OpenAI, and sends the results via Mailjet. The diagram below represents the API call flow and data handling process.

## Workflow Diagram

The following diagram illustrates the process:



## Code Implementation

The Python script below implements the described workflow.

import os  
 from datetime import datetime, timedelta  
 from dotenv import load\_dotenv  
 from supabase import create\_client  
 from openai import OpenAI  
 from mailjet\_rest import Client  
   
 # Load environment variables  
 load\_dotenv(override=True)  
   
 # Initialize clients  
 try:  
 # Supabase setup  
 supabase\_url = os.getenv('SUPABASE\_URL')  
 supabase\_key = os.getenv('SUPABASE\_KEY')  
 if not supabase\_url or not supabase\_key:  
 raise ValueError("Supabase credentials not found")  
 supabase = create\_client(supabase\_url, supabase\_key)  
   
 # OpenAI setup  
 openai\_key = os.getenv('OPENAI\_API\_KEY')  
 if not openai\_key:  
 raise ValueError("OpenAI key not found")  
 client = OpenAI(api\_key=openai\_key)  
   
 # Mailjet setup  
 mailjet\_api\_key = os.getenv('MAILJET\_API\_KEY')  
 mailjet\_api\_secret = os.getenv('MAILJET\_API\_SECRET')  
 if not mailjet\_api\_key or not mailjet\_api\_secret:  
 raise ValueError("Mailjet credentials not found")  
 mailjet = Client(auth=(mailjet\_api\_key, mailjet\_api\_secret), version='v3.1')  
   
 print("All API connections established successfully")  
   
 except Exception as e:  
 print(f"Error initializing clients: {e}")  
 exit(1)

def get\_recent\_data():  
 """  
 Fetch recent data from both tables with correct column names  
 """  
 try:  
 # Get last 24 hours of data  
 time\_threshold = (datetime.utcnow() - timedelta(hours=24)).isoformat()  
   
 # Fetch BTC prices with correct column names  
 btc\_prices = supabase.table('btc\_price') .select('id, price, timestamp') .gte('timestamp', time\_threshold) .order('timestamp', desc=True) .execute()  
   
 # Fetch economic news with correct column names  
 eco\_news = supabase.table('eco\_info') .select('id, created\_at, finance\_info') .gte('created\_at', time\_threshold) .order('created\_at', desc=True) .execute()  
   
 return btc\_prices.data, eco\_news.data  
   
 except Exception as e:  
 print(f"Error fetching data: {e}")  
 return None, None

def generate\_analysis(btc\_data, news\_data):  
 """  
 Generate analysis using OpenAI with a personalized signature  
 """  
 try:  
 # Prepare context with correct column names  
 btc\_context = "\n".join([  
 f"Bitcoin Price at {item['timestamp']}: ${item['price']}"  
 for item in btc\_data[:5]  
 ])  
   
 news\_context = "\n".join([  
 f"News at {item['created\_at']}: {item['finance\_info']}"  
 for item in news\_data  
 ])  
   
 prompt = f"""  
 As a professional financial analyst, write a concise email analyzing the following data:  
  
 Recent Bitcoin Prices:  
 {btc\_context}  
  
 Recent Financial News:  
 {news\_context}  
  
 Focus on:  
 1. Key price movements and their correlation with news  
 2. Important market trends  
 3. Potential short-term implications  
  
 Keep the analysis professional, concise, and actionable.  
 End the email with:  
   
 Best regards,  
 Ambika  
 """  
   
 completion = client.chat.completions.create(  
 model="gpt-3.5-turbo",  
 messages=[  
 {"role": "system", "content": "You are a professional financial analyst writing a concise email update."},  
 {"role": "user", "content": prompt}  
 ],  
 max\_tokens=500  
 )  
   
 return completion.choices[0].message.content  
   
 except Exception as e:  
 print(f"Error generating analysis: {e}")  
 return None

def send\_email(analysis):  
 """  
 Send email using Mailjet  
 """  
 try:  
 current\_time = datetime.utcnow().strftime("%Y-%m-%d %H:%M UTC")  
   
 data = {  
 'Messages': [  
 {  
 "From": {  
 "Email": "cvijay1116@gmail.com", # Replace with your verified sender  
 "Name": "Finance Agent"  
 },  
 "To": [  
 {  
 "Email": "ambikaleads@gmail.com",  
 "Name": "Ambika"  
 }  
 ],  
 "Subject": f"Financial Market Analysis - {current\_time}",  
 "TextPart": analysis,  
 "HTMLPart": analysis.replace('\n', '<br>') # Basic HTML formatting  
 }  
 ]  
 }  
   
 result = mailjet.send.create(data=data)  
   
 if result.status\_code == 200:  
 print("Email sent successfully!")  
 return True  
 else:  
 print(f"Failed to send email. Status code: {result.status\_code}")  
 return False  
   
 except Exception as e:  
 print(f"Error sending email: {e}")  
 return False

def main():  
 try:  
 # Fetch data  
 btc\_data, news\_data = get\_recent\_data()  
 if not btc\_data or not news\_data:  
 print("No recent data found")  
 return  
   
 # Generate analysis  
 analysis = generate\_analysis(btc\_data, news\_data)  
 if not analysis:  
 print("Failed to generate analysis")  
 return  
   
 # Send email  
 if send\_email(analysis):  
 print("Process completed successfully")  
 else:  
 print("Failed to send email")  
   
 except Exception as e:  
 print(f"Error in main process: {e}")  
  
 if \_\_name\_\_ == "\_\_main\_\_":  
 main()