In [1]:
```
import pandas as pd
import sqlite3
import matplotlib.pyplot as plt
import seaborn as sns
import google.generativeai as genai
from datetime import datetime
import re
from flask import Flask, request, jsonify
import json
```

In [2]:
```
app = Flask(__name__)
DATABASE_NAME = 'ecommerce_data.db'
```

In [3]:
```
# --- Gemini Configuration ---
genai.configure(api_key="AIzaSyCtnWAGeFFd04JLSfqkQle4ioJDevrGrRI")  # Replace with your actual key
model = genai.GenerativeModel("gemini-1.5-pro")
```

In [4]:
```
# --- Fix time format for eligibility table ---
def fix_time_format(dt_str):
    if not isinstance(dt_str, str):
        return None
    try:
        if ' ' in dt_str and '.' in dt_str.split(' ')[1] and dt_str.count(':') < 2:
            date_part, time_part = dt_str.split(' ')
            time_parts = time_part.split('.')
            time_part_fixed = ':'.join(f"{int(t):02d}" for t in time_parts)
            return f"{date_part} {time_part_fixed}"
        return dt_str
    except Exception as e:
        print(f"Error fixing time format for '{dt_str}': {e}")
        return None
```

In [5]:
```
# --- Load and process CSV file ---
def load_and_process_data(file_path, table_name, conn):
    print(f"Loading {file_path}...")
    try:
        df = pd.read_csv(file_path, encoding='iso-8859-1')
    except Exception as e:
        print(f"Error loading {file_path}: {e}")
        return

    if 'eligibility_datetime_utc' in df.columns:
        df['eligibility_datetime_utc'] = df['eligibility_datetime_utc'].apply(fix_time_format)
        df['eligibility_datetime_utc'] = pd.to_datetime(df['eligibility_datetime_utc'], errors='coerce')
        df['date_only'] = df['eligibility_datetime_utc'].dt.date
        df['hour'] = df['eligibility_datetime_utc'].dt.hour

    for col in df.columns:
        if df[col].dtype == 'object':
            try:
                df[col] = pd.to_numeric(df[col], errors='coerce')
            except ValueError:
                pass

    try:
```

```
        df.to_sql(table_name, conn, if_exists='replace', index=False)
        print(f"Successfully saved {table_name}.")
    except Exception as e:
        print(f"Error saving {table_name}: {e}")
```

In [6]:
```
# --- Create combined view ---
def create_combined_view(conn):
    cursor = conn.cursor()
    cursor.execute("DROP VIEW IF EXISTS combined_metrics")
    cursor.execute("""
        CREATE VIEW combined_metrics AS
        SELECT
            e.product_id,
            e.eligibility_datetime_utc,
            e.date_only,
            e.hour,
            a.ad_sales,
            a.roas,
            a.cpc,
            t.total_sales,
            t.units_sold
        FROM eligibility_table e
        LEFT JOIN ad_sales_metrics a ON e.product_id = a.product_id
        LEFT JOIN total_sales_metrics t ON e.product_id = t.product_id;
    """)
    conn.commit()
```

In [7]:
```
# --- Get schema from SQLite ---
def get_database_schema(conn):
    cursor = conn.cursor()
    cursor.execute("SELECT name FROM sqlite_master WHERE type IN ('table', 'view');")
    tables = cursor.fetchall()
    all_schemas = {}

    for table_name_tuple in tables:
        table_name = table_name_tuple[0]
        cursor.execute(f"PRAGMA table_info({table_name});")
        columns = cursor.fetchall()
        all_schemas[table_name] = "\n".join([f"    {col[1]} ({col[2]})" for col in columns])

    schema_string = ""
    for table, schema in all_schemas.items():
        schema_string += f"Table: {table}\n{schema}\n\n"
    return schema_string.strip()
```

In [8]:
```
# --- Extract SQL from Gemini response ---
def extract_sql_query(response_text):
    sql_match = re.search(r"```sql\s*(.*?)\s*```", response_text, re.DOTALL)
    if sql_match:
        return sql_match.group(1).strip()
    if response_text.strip().upper().startswith(("SELECT", "PRAGMA", "WITH")):
        return response_text.strip()
    return None
```

In [9]:
```
# --- Ask Gemini for SQL ---
def ask_gemini_for_sql(question, schema_description):
    prompt = f"""You are an AI agent working with the following SQLite database schema.
```

```
{schema_description}

Based on the schema, write an appropriate SQLite SQL query to answer this question.
Respond ONLY with the SQL query enclosed in a markdown code block (```sql...```).

Question: {question}
"""
    try:
        response = model.generate_content(prompt)
        return response.text
    except Exception as e:
        print(f"Error calling Gemini API: {e}")
        return None
```

In [10]:
```
# --- Execute SQL and return result ---
def execute_sql_and_get_results(sql_query, conn):
    try:
        df = pd.read_sql(sql_query, conn)
        return df, None
    except Exception as e:
        return None, str(e)

# --- Main Agent Logic ---
def answer_question(question, conn, db_schema):
    gemini_response = ask_gemini_for_sql(question, db_schema)
    if not gemini_response:
        return {"error": "Gemini response failed"}, 500

    sql_query = extract_sql_query(gemini_response)
    if not sql_query:
        return {"error": f"Invalid SQL response:\n{gemini_response}"}, 400

    print(f"\n🧠 Question: {question}")
    print(f"📜 SQL Generated:\n{sql_query}\n")

    result_df, error = execute_sql_and_get_results(sql_query, conn)
    if error:
        return {"error": error, "sql_query": sql_query}, 500

    if result_df is not None and not result_df.empty:
        return {
            "question": question,
            "sql_query": sql_query,
            "results": result_df.to_dict(orient='records')
        }, 200
    else:
        return {
            "question": question,
            "sql_query": sql_query,
            "explanation": "The query returned no results.",
            "results": []
        }, 200
```

In [11]:
```
# --- API Endpoint ---
@app.route('/ask', methods=['POST'])
def ask_api():
```

```
    question = request.json.get('question')
    if not question:
        return jsonify({"error": "No question provided"}), 400

    conn = sqlite3.connect(DATABASE_NAME)
    schema = get_database_schema(conn)
    response, status = answer_question(question, conn, schema)
    conn.close()
    return jsonify(response), status
```

In [12]:

```
# --- API Endpoint ---
# Avoid re-registering the route if running in an interactive environment
app.view_functions.pop('ask_api', None)  # <-- ADD THIS LINE

@app.route('/ask', methods=['POST'])
def ask_api():
    question = request.json.get('question')
    if not question:
        return jsonify({"error": "No question provided"}), 400

    conn = sqlite3.connect(DATABASE_NAME)
    schema = get_database_schema(conn)
    response, status = answer_question(question, conn, schema)
    conn.close()
    return jsonify(response), status
```

In [28]:
```python
# --- Initialize DB ---
def initialize_database():
    conn = sqlite3.connect(DATABASE_NAME)
    load_and_process_data("D:\\DAA\\CDC\\ANARIX\\Product-Level Eligibility Table (mapped).csv", 'eligibility_table', conn)
    load_and_process_data("D:\\DAA\\CDC\\ANARIX\\Product-Level Ad Sales and Metrics (mapped).csv", 'ad_sales_metrics', conn)
    load_and_process_data("D:\\DAA\\CDC\\ANARIX\\Product-Level Ad Sales and Metrics (mapped).csv", 'total_sales_metrics', conn)
    create_combined_view(conn)
    conn.close()
    print("✅ Database initialized.")

# --- Main Entrypoint ---
if __name__ == '__main__':
    initialize_database()
    print("🚀 AI Agent running at http://127.0.0.1:5000/ask")
    print("Example: curl -X POST -H \"Content-Type: application/json\" -d '{\"question\": \"What is my total sales?\"}' http://127.0.0.1:5000/ask")
    app.run(debug=False, port=5000)
```

```
Loading D:\DAA\CDC\ANARIX\Product-Level Eligibility Table (mapped).csv...
Successfully saved eligibility_table.
Loading D:\DAA\CDC\ANARIX\Product-Level Ad Sales and Metrics (mapped).csv...
Successfully saved ad_sales_metrics.
Loading D:\DAA\CDC\ANARIX\Product-Level Ad Sales and Metrics (mapped).csv...
Successfully saved total_sales_metrics.
✅ Database initialized.
🚀 AI Agent running at http://127.0.0.1:5000/ask
Example: curl -X POST -H "Content-Type: application/json" -d '{"question": "What is my total sales?"}' http://127.0.0.1:5000/ask
 * Serving Flask app '__main__'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [22/Jul/2025 18:14:39] "GET /ask HTTP/1.1" 405 -
```

In [29]:
```python
import requests

url = "http://127.0.0.1:5000/ask"
payload = {"question": "Which product had the highest ROAS?"}
response = requests.post(url, json=payload)

print(response.json())
```

```
---------------------------------------------------------------------------
ConnectionRefusedError                    Traceback (most recent call last)
File ~\anaconda3\Lib\site-packages\urllib3\connection.py:198, in HTTPConnection._new_conn(self)
    197 try:
--> 198     sock = connection.create_connection(
    199         (self._dns_host, self.port),
    200         self.timeout,
    201         source_address=self.source_address,
    202         socket_options=self.socket_options,
    203     )
    204 except socket.gaierror as e:

File ~\anaconda3\Lib\site-packages\urllib3\util\connection.py:85, in create_connection(address, timeout, source_address, socket_options)
     84 try:
---> 85     raise err
     86 finally:
     87     # Break explicitly a reference cycle

File ~\anaconda3\Lib\site-packages\urllib3\util\connection.py:73, in create_connection(address, timeout, source_address, socket_options)
     72     sock.bind(source_address)
---> 73 sock.connect(sa)
     74 # Break explicitly a reference cycle

ConnectionRefusedError: [WinError 10061] No connection could be made because the target machine actively refused it

The above exception was the direct cause of the following exception:

NewConnectionError                        Traceback (most recent call last)
File ~\anaconda3\Lib\site-packages\urllib3\connectionpool.py:787, in HTTPConnectionPool.urlopen(self, method, url, body, headers, retries, redirect, assert_same_host, timeout, pool_timeout
, release_conn, chunked, body_pos, preload_content, decode_content, **response_kw)
    786 # Make the request on the HTTPConnection object
--> 787 response = self._make_request(
    788     conn,
    789     method,
    790     url,
    791     timeout=timeout_obj,
    792     body=body,
    793     headers=headers,
    794     chunked=chunked,
    795     retries=retries,
    796     response_conn=response_conn,
    797     preload_content=preload_content,
    798     decode_content=decode_content,
    799     **response_kw,
    800 )
    802 # Everything went great!

File ~\anaconda3\Lib\site-packages\urllib3\connectionpool.py:493, in HTTPConnectionPool._make_request(self, conn, method, url, body, headers, retries, timeout, chunked, response_conn, prel
oad_content, decode_content, enforce_content_length)
    492 try:
--> 493     conn.request(
    494         method,
    495         url,
    496         body=body,
    497         headers=headers,
    498         chunked=chunked,
    499         preload_content=preload_content,
    500         decode_content=decode_content,
```

```
    501         enforce_content_length=enforce_content_length,
    502     )
    504 # We are swallowing BrokenPipeError (errno.EPIPE) since the server is
    505 # legitimately able to close the connection after sending a valid response.
    506 # With this behaviour, the received response is still readable.

File ~\anaconda3\Lib\site-packages\urllib3\connection.py:445, in HTTPConnection.request(self, method, url, body, headers, chunked, preload_content, decode_content, enforce_content_length)
    444     self.putheader(header, value)
--> 445 self.endheaders()
    447 # If we're given a body we start sending that in chunks.

File ~\anaconda3\Lib\http\client.py:1298, in HTTPConnection.endheaders(self, message_body, encode_chunked)
   1297     raise CannotSendHeader()
-> 1298 self._send_output(message_body, encode_chunked=encode_chunked)

File ~\anaconda3\Lib\http\client.py:1058, in HTTPConnection._send_output(self, message_body, encode_chunked)
   1057 del self._buffer[:]
-> 1058 self.send(msg)
   1060 if message_body is not None:
   1061
   1062     # create a consistent interface to message_body

File ~\anaconda3\Lib\http\client.py:996, in HTTPConnection.send(self, data)
    995 if self.auto_open:
--> 996     self.connect()
    997 else:

File ~\anaconda3\Lib\site-packages\urllib3\connection.py:276, in HTTPConnection.connect(self)
    275 def connect(self) -> None:
--> 276     self.sock = self._new_conn()
    277     if self._tunnel_host:
    278         # If we're tunneling it means we're connected to our proxy.

File ~\anaconda3\Lib\site-packages\urllib3\connection.py:213, in HTTPConnection._new_conn(self)
    212 except OSError as e:
--> 213     raise NewConnectionError(
    214         self, f"Failed to establish a new connection: {e}"
    215     ) from e
    217 sys.audit("http.client.connect", self, self.host, self.port)

NewConnectionError: <urllib3.connection.HTTPConnection object at 0x000001AD687A10D0>: Failed to establish a new connection: [WinError 10061] No connection could be made because the target machine actively refused it

The above exception was the direct cause of the following exception:

MaxRetryError                             Traceback (most recent call last)
File ~\anaconda3\Lib\site-packages\requests\adapters.py:667, in HTTPAdapter.send(self, request, stream, timeout, verify, cert, proxies)
    666 try:
--> 667     resp = conn.urlopen(
    668         method=request.method,
    669         url=url,
    670         body=request.body,
    671         headers=request.headers,
    672         redirect=False,
    673         assert_same_host=False,
    674         preload_content=False,
    675         decode_content=False,
    676         retries=self.max_retries,
```

```
    677         timeout=timeout,
    678         chunked=chunked,
    679     )
    681 except (ProtocolError, OSError) as err:

File ~\anaconda3\Lib\site-packages\urllib3\connectionpool.py:841, in HTTPConnectionPool.urlopen(self, method, url, body, headers, retries, redirect, assert_same_host, timeout, pool_timeout
, release_conn, chunked, body_pos, preload_content, decode_content, **response_kw)
    839     new_e = ProtocolError("Connection aborted.", new_e)
--> 841 retries = retries.increment(
    842     method, url, error=new_e, _pool=self, _stacktrace=sys.exc_info()[2]
    843 )
    844 retries.sleep()

File ~\anaconda3\Lib\site-packages\urllib3\util\retry.py:519, in Retry.increment(self, method, url, response, error, _pool, _stacktrace)
    518     reason = error or ResponseError(cause)
--> 519     raise MaxRetryError(_pool, url, reason) from reason  # type: ignore[arg-type]
    521 log.debug("Incremented Retry for (url='%s'): %r", url, new_retry)

MaxRetryError: HTTPConnectionPool(host='127.0.0.1', port=5000): Max retries exceeded with url: /ask (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x000001AD68
7A10D0>: Failed to establish a new connection: [WinError 10061] No connection could be made because the target machine actively refused it'))

During handling of the above exception, another exception occurred:

ConnectionError                           Traceback (most recent call last)
Cell In[29], line 5
      3 url = "http://127.0.0.1:5000/ask"
      4 payload = {"question": "Which product had the highest ROAS?"}
----> 5 response = requests.post(url, json=payload)
      7 print(response.json())

File ~\anaconda3\Lib\site-packages\requests\api.py:115, in post(url, data, json, **kwargs)
    103 def post(url, data=None, json=None, **kwargs):
    104     r"""Sends a POST request.
    105
    106     :param url: URL for the new :class:`Request` object.
   (...)
    112     :rtype: requests.Response
    113     """
--> 115     return request("post", url, data=data, json=json, **kwargs)

File ~\anaconda3\Lib\site-packages\requests\api.py:59, in request(method, url, **kwargs)
     55 # By using the 'with' statement we are sure the session is closed, thus we
     56 # avoid leaving sockets open which can trigger a ResourceWarning in some
     57 # cases, and look like a memory leak in others.
     58 with sessions.Session() as session:
---> 59     return session.request(method=method, url=url, **kwargs)

File ~\anaconda3\Lib\site-packages\requests\sessions.py:589, in Session.request(self, method, url, params, data, headers, cookies, files, auth, timeout, allow_redirects, proxies, hooks, st
ream, verify, cert, json)
    584 send_kwargs = {
    585     "timeout": timeout,
    586     "allow_redirects": allow_redirects,
    587 }
    588 send_kwargs.update(settings)
--> 589 resp = self.send(prep, **send_kwargs)
    591 return resp

File ~\anaconda3\Lib\site-packages\requests\sessions.py:703, in Session.send(self, request, **kwargs)
```

```
    700 start = preferred_clock()
    702 # Send the request
--> 703 r = adapter.send(request, **kwargs)
    705 # Total elapsed time of the request (approximately)
    706 elapsed = preferred_clock() - start

File ~\anaconda3\Lib\site-packages\requests\adapters.py:700, in HTTPAdapter.send(self, request, stream, timeout, verify, cert, proxies)
    696     if isinstance(e.reason, _SSLError):
    697         # This branch is for urllib3 v1.22 and later.
    698         raise SSLError(e, request=request)
--> 700     raise ConnectionError(e, request=request)
    702 except ClosedPoolError as e:
    703     raise ConnectionError(e, request=request)

ConnectionError: HTTPConnectionPool(host='127.0.0.1', port=5000): Max retries exceeded with url: /ask (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x000001AD
687A10D0>: Failed to establish a new connection: [WinError 10061] No connection could be made because the target machine actively refused it'))
```

In [ ]: