From Oops To Ops:

Smart Task Failure Diagnosis with OpenAI

Nathan Hadfield - King

Making the World Playful

King – cross platform casual games

Candy Crush Saga

Candy Crush Soda Saga

Farm Heroes Saga

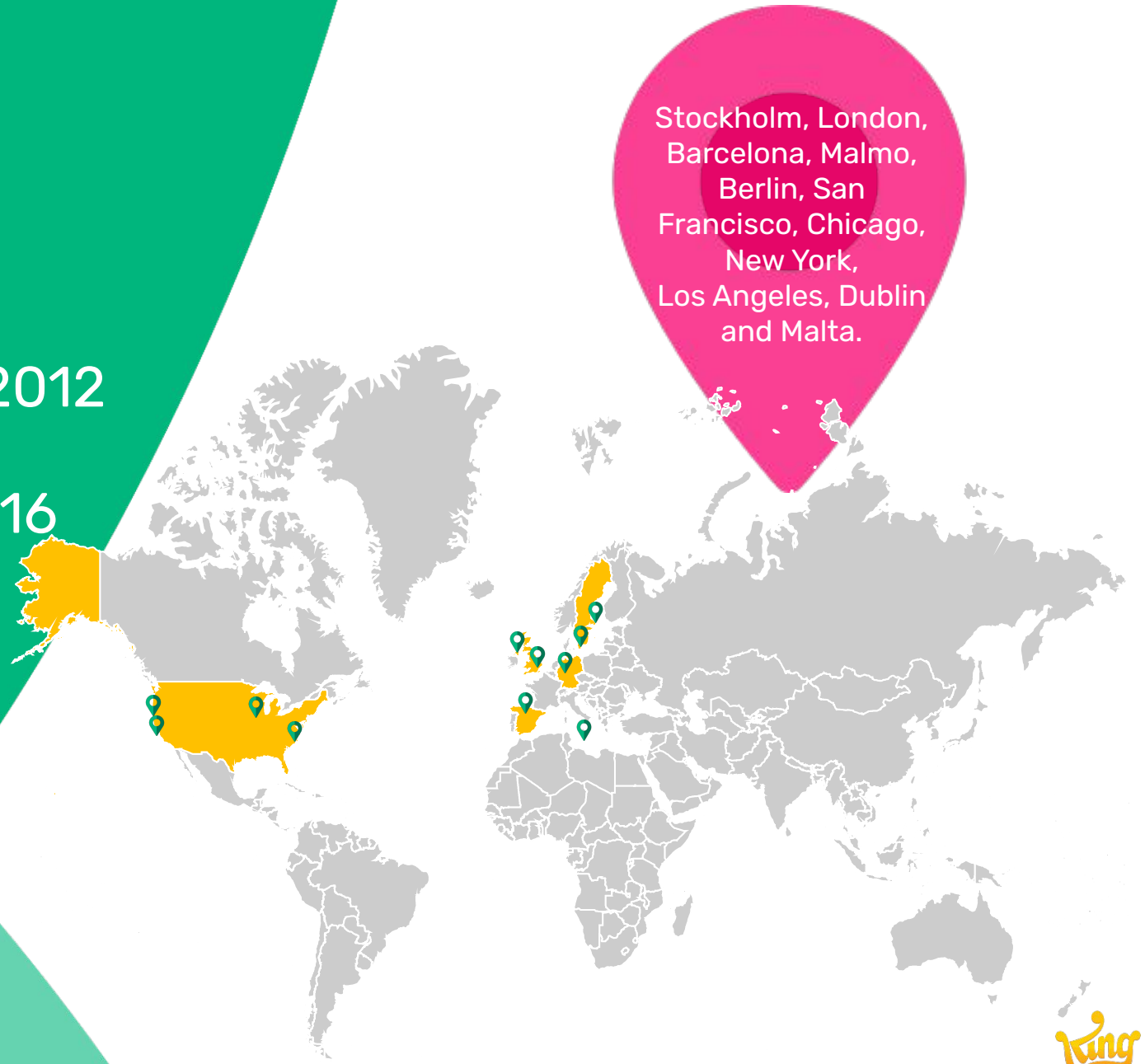
Large user base
over
200M
monthly players

Growing content
>17K
levels in Candy Crush Saga alone

King
Making the World Playful

A brief history of King…

- Founded in 2003
- Candy Crush Saga released 2012
- Part of Activision Blizzard 2016
- Acquired by Microsoft 2023
- ~2000 employees currently



Airflow & Gen AI

- Everyone at King is being encouraged to use AI tools
  - ChatGPT Enterprise – day-to-day help and assistance
  - GitHub CoPilot – coding assistance
  - OpenAI Platform – application integration
- No native Airflow features for Gen AI (yet)
  - Airflow v3?
- Several LLM Airflow providers already exist:
  - OpenAI
  - Cohere
  - Google Gemini (Vertex)
- How can we run our pipelines more efficiently?
  - Improve Data Ops through better failure analysis

Image source: https://markmcneilly.substack.com/p/the-best-memes-about-ai

Airflow Task Failure Diagnoser

```
[2024-07-30, 10:35:18 BST] {taskinstance.py:2905} ERROR - Task failed with exception
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/site-packages/airflow/models/taskinstance.py", line 465, in _execute_task
    result = _execute_callable(context=context, **execute_callable_kwargs)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/site-packages/airflow/models/taskinstance.py", line 432, in _execute_callable
    return execute_callable(context=context, **execute_callable_kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/site-packages/airflow/models/baseoperator.py", line 401, in wrapper
    return func(self, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/airflow/include/operators/data_ai.py", line 531, in execute
    data, stats = data_ai_hook.download_channel(
                  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/site-packages/backoff/_sync.py", line 105, in retry
    ret = target(*args, **kwargs)
          ^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/airflow/include/hooks/data_ai.py", line 423, in download_channel
    data = self._get_data(endpoint=endpoint)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/site-packages/backoff/_sync.py", line 105, in retry
    ret = target(*args, **kwargs)
          ^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/site-packages/backoff/_sync.py", line 105, in retry
    ret = target(*args, **kwargs)
          ^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/airflow/include/hooks/data_ai.py", line 77, in _get_data
    raise err
  File "/usr/local/airflow/include/hooks/data_ai.py", line 65, in _get_data
    response.raise_for_status()
  File "/usr/local/lib/python3.11/site-packages/requests/models.py", line 1021, in raise_for_status
    raise HTTPError(http_error_msg, response=self)
requests.exceptions.HTTPError: 400 Client Error: Bad Request for url: https://api.data.ai/v2.0/portfolio/download-channel?company_id=None&start_date=2024-07-22&end_date=2024-07-22&granularity=daily&countries=AU%2CBR%2CCA%2CDE%2CDK%2CES%2CFI%2CFR%2CGB%2CHK%2CID%2CIN%2CIT%2CJP%2CKR%2CMX%2CNO%2CRU%2CSE%2CTH%2CTR%2CTW%2CUS%2CWW&devices=ios-phone%2Cios-tablet%2Candroid-phone%2Candroid-tablet
[2024-07-30, 10:35:18 BST] {taskinstance.py:1206} INFO - Marking task as FAILED. dag_id=dataai-download-channel, task_id=get-download-channel, run_id=scheduled__2024-07-25T09:00:00+00:00, map_index=221, execution_date=20240725T090000, start_date=20240730T093123, end_date=20240730T093518
```

Assess and describe issues

Identify and explain the root cause

Suggest possible solutions

Provide rewritten SQL

Airflow Task Failure Diagnoser

Methodology

OpenAI Platform

Invoked via failure callbacks

Provide exception message, DAG code and SQL for context

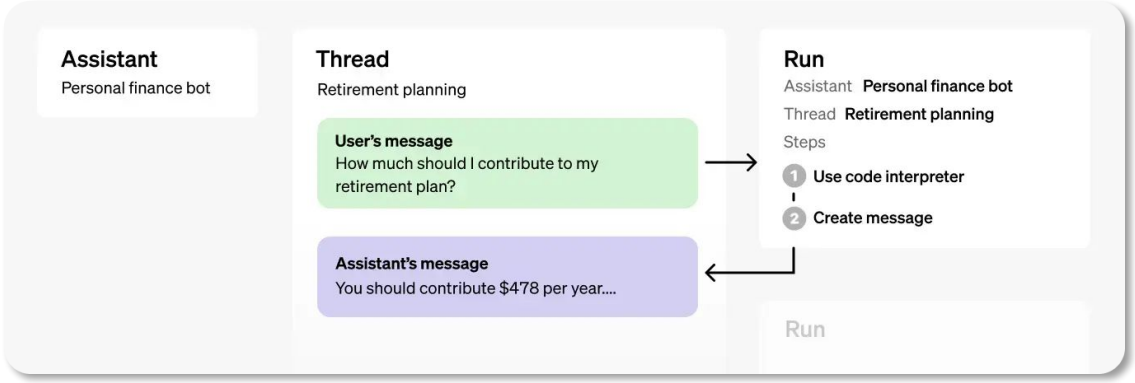
Output results to tasks logs and alerting channels

OpenAI Platform

	Chat Completion API	Assistants API
Functionality	Generates responses based on input, designed for conversational agents.	Advanced features for creating and managing custom AI assistants.
Use Cases	Chatbots, customer service, interactive fiction, natural language understanding.	Advanced customer support, personal assistants, workflow automation.
Customization	System messages, prompt engineering to control tone and style.	Sophisticated behaviour and memory customization, workflow definition.
Complexity and Customization	Simpler and more focused on immediate context responses.	Offers advanced capabilities, including state management and external system integration.
Integration	Mainly focuses on generating text responses.	Interacts with other APIs and data sources.
Memory	Does not natively support memory between interactions.	Can maintain state and remember previous interactions.

Assistants API

- Enables the building of AI assistants within applications
- An assistant has instructions and can leverage models, tools and files to respond to user queries
- Three types of tools:
  - **Code Interpreter** – Write and run Python code
  - **File Search** – Augment with knowledge outside the model
  - **Function Calling** – Describe custom functions for the assistant to use



Assistant	• Purpose built AI that uses OpenAI models and can call tools
Thread	• Conversation session between an Assistant and a user • Threads store messages • Threads persist over time and can be appended to
Message	• A message created by an assistant or a user • Can include text, images and other files • Stored as a list on the thread
Run	• Invocation of an Assistant on a Thread • Uses its configuration and Messages to perform tasks by calling models and tools • Appends Messages to the Thread
Run Step	• Detailed list of steps the Assistant took as part of a Run

Airflow Task Failure Diagnoser

Build Context

Use DAG context on failure to collate:

- DAG ID
- Task ID
- Exception
- Location of the DAG code
- SQL query

```python
from airflow.models.serialized_dag import SerializedDagModel

dag_id, task_id = context['task'].dag_id, context['task'].task_id
serialised_dag = SerializedDagModel.get(context['task_instance'].dag_id)
file_loc = serialised_dag.dag.fileloc

content_dict = {'dag_id': dag_id, 'task_id': task_id, 'exception_msg': context.get('exception')}
if task_operator == 'BigQueryInsertJobOperator':
    content_dict['sql'] = context['task'].configuration.get('query').get('query')

diagnosis = openai_assistant_diagnose(content_dict=content_dict, file_loc=file_loc)
```

Airflow Task Failure Diagnoser

Create Assistant

- **instructions**
  - Prompt the assistant on how to behave
- **gpt-4o-mini**
  - Cheaper and faster than gpt-4o
  - Smarter and cheaper than gpt-3.5-turbo
- **temperature**
  - Controls determinism
  - Values between 0 and 2
- **file_search**
  - Gives the assistant the ability to use files we have uploaded

```python
def openai_assistant_diagnose(
    content_dict: dict,
    file_loc: str,
) -> dict:
    model = 'gpt-4o-mini'
    temperature = 0.0
    tools = [{'type': 'file_search'}]
    openai_hook = OpenAIHook()
    assistant_name = 'Airflow Task Failure Diagnoser'
    assistant = get_assistant_by_name(openai_hook=openai_hook, assistant_name=assistant_name)

    if assistant:
        logging.info(f"An assistant {assistant.id} with the name '{assistant_name}' already exists.")
        assistant = openai_hook.modify_assistant(
            assistant_id=assistant.id,
            model=model,
            instructions=PROMPT,
            name=assistant_name,
            tools=tools,
            temperature=temperature,
        )
    else:
        assistant = openai_hook.create_assistant(
            name=assistant_name,
            model=model,
            instructions=PROMPT,
            tools=tools,
            temperature=temperature,
        )
```

Airflow Task Failure Diagnoser

Crafting a Prompt

Clarity & Precision
- Avoid ambiguity

Specificity
- Clearly state the question or topic

Context
- Provide background information

Examples
- Give example of the type of response

Instructions
- Outline specific instructions or constraints

Poor Prompt: “Tell me about dogs.”

Good Prompt: “Can you provide detailed information on the different breeds of dogs, focusing on their temperament, size, and common health issues?”

```
Role: Airflow Task Failure Diagnoser

Objective:
Diagnose Airflow task failures based on provided details and suggest solutions.

Instructions:
You will receive a JSON object containing the following:
- dag_id: The ID of the Directed Acyclic Graph (DAG).
- task_id: The ID of the specific task that failed.
- failure_exception: The exception message or error log describing the failure.
- query (if applicable): The SQL query involved in the task.

Your tasks are:
1. Assess and describe the issue causing the failure.
2. Identify and explain the root cause of the failure.
3. Provide three potential solutions to resolve the issue.
4. If the task involves SQL, suggest a rewritten version of the query to address the problem.

Additionally, refer to the DAG code to help pinpoint and understand the problem more accurately.

Output Format:
Your response should only consist of a JSON object structured as follows:

{
  "issue": "ISSUE_DESCRIPTION",
  "root_cause": "ROOT_CAUSE_DESCRIPTION",
  "suggested_solutions": ["SOLUTION_1", "SOLUTION_2", "SOLUTION_3"],
  "query": "REWRITTEN_SQL_QUERY"
}
```

Airflow Task Failure Diagnoser

Create Thread, Add Messages, Run Assistant

- Upload the DAG code
  - File is automatically parsed and chunked
  - Creates and stores the embeddings
- Create Thread with Message
  - Provide context
  - Attach file
- Create a Run
  - Uses the model and tools to generate a response
  - Added to the thread as `assistant` messages

```
message_file = openai_hook.upload_file(file=file_loc, purpose='assistants')
thread = openai_hook.create_thread(
    messages=[
        {
            'role': 'user',
            'content': f'{content_dict}',
            'attachments': [{'file_id': message_file.id, 'tools': tools}],
        }
    ]
)

run = openai_hook.create_run_and_poll(
    thread_id=thread.id,
    assistant_id=assistant.id,
)
```

Airflow Task Failure Diagnoser

Output Diagnosis

- Get messages from the Thread
- Cleanup response & return diagnosis

```
messages = openai_hook.get_messages(thread_id=thread.id, run_id=run.id)
openai_hook.delete_thread(thread_id=thread.id)

message_content = messages[0].content[0].text

diagnosis = json.loads(message_content.value.strip('```json\n').strip('\n```'))
openai_hook.delete_file(file_id=message_file.id)
return diagnosis
```

- Write diagnosis to Task Log

```
diagnosis = openai_assistant_diagnose(content_dict=content_dict, file_loc=file_loc)

logging.info(f'OpenAI - Diagnosis: {diagnosis.get("issue")}')
logging.info(f'OpenAI - Root Cause: {diagnosis.get("root_cause")}')

for fix in diagnosis.get('suggested_solutions'):
    logging.info(f'OpenAI - Suggested Fixes: {fix}')

if diagnosis.get('query'):
    logging.info(f'OpenAI - Updated SQL:\n{diagnosis.get("query")}')
```

Airflow Task Failure Diagnoser

In Action

```python
create_staging = BigQueryCreateEmptyTableOperator(
    task_id='create-staging-raw_fx_rate',
    dataset_id='external_data_kitche',
    table_id='raw_fx_rate',
    schema_fields=[
        {'name': 'base_currency', 'type': 'STRING', 'mode': 'NULLABLE'},
        {'name': 'currency', 'type': 'STRING', 'mode': 'NULLABLE'},
        {'name': 'dt', 'type': 'DATE', 'mode': 'NULLABLE'},
        {'name': 'bid', 'type': 'NUMERIC', 'mode': 'NULLABLE'},
        {'name': 'ask', 'type': 'NUMERIC', 'mode': 'NULLABLE'},
    ],
    time_partitioning={'field': 'dt'},
)
```

[2024-08-01, 12:05:32 BST] {bigquery.py:1635} INFO - Creating table
[2024-08-01, 12:05:32 BST] {taskinstance.py:2905} ERROR - Task failed with exception
File "/usr/local/lib/python3.11/site-packages/google/cloud/_http/__init__.py", line 494, in api_request
raise exceptions.from_http_response(response)
google.api_core.exceptions.NotFound: 404 POST https://bigquery.googleapis.com/bigquery/v2/projects/my-project/datasets/external_data_kitche/tables?prettyPrint=false: Not found: Dataset 'king-coredatasets-sandbox:external_data_kitche
[2024-08-01, 12:05:32 BST] {taskinstance.py:1206} INFO - Marking task as FAILED.
[2024-08-01, 12:05:45 BST] {alerting.py:216} INFO - OpenAI - Diagnosis: The task 'create-staging-raw_fx_rate' failed due to a 'NotFound' error when attempting to create a table in a non-existent BigQuery dataset.
[2024-08-01, 12:05:45 BST] {alerting.py:217} INFO - OpenAI - Root Cause: The dataset 'external_data_kitche' does not exist in the 'king-coredatasets-sandbox' project.
[2024-08-01, 12:05:45 BST] {alerting.py:222} INFO - OpenAI - Suggested Fixes: Verify the dataset name 'external_data_kitche' and ensure it is correctly spelled.
[2024-08-01, 12:05:45 BST] {alerting.py:222} INFO - OpenAI - Suggested Fixes: Create the dataset 'external_data_kitche' in the 'king-coredatasets-sandbox' project before running the DAG.
[2024-08-01, 12:05:45 BST] {alerting.py:222} INFO - OpenAI - Suggested Fixes: Update the DAG to use an existing dataset or create a new dataset with the correct name.

Airflow Task Failure Diagnoser

Example 1

Error:

google.api_core.exceptions.BadRequest: The field specified for partitioning cannot be found in the schema.

```
create_staging = BigQueryCreateEmptyTableOperator(
    task_id='create-staging-raw_fx_rate',
    dataset_id='external_data_kitchen',
    table_id='raw_fx_rate',
    schema_fields=[
        {'name': 'base_currency', 'type': 'STRING', 'mode': 'NULLABLE'},
        {'name': 'currency', 'type': 'STRING', 'mode': 'NULLABLE'},
        {'name': 'dt', 'type': 'DATE', 'mode': 'NULLABLE'},
        {'name': 'bid', 'type': 'NUMERIC', 'mode': 'NULLABLE'},
        {'name': 'ask', 'type': 'NUMERIC', 'mode': 'NULLABLE'},
    ],
    time_partitioning={'field': 'date'},
)
```

Diagnosis:

OpenAI - Diagnosis: The task 'create-staging-raw_fx_rate' failed due to a missing field in the schema for partitioning.
OpenAI - Root Cause: The field specified for time partitioning ('date') does not exist in the schema. The schema defines a field 'dt' of type 'DATE', but the partitioning field is incorrectly specified as 'date'.
OpenAI - Suggested Fixes: Update the time partitioning field to 'dt' in the 'create-staging-raw_fx_rate' task.
OpenAI - Suggested Fixes: Ensure that the schema and partitioning field names match exactly.
OpenAI - Suggested Fixes: Review the schema definition and the partitioning field to ensure consistency.

Airflow Task Failure Diagnoser

Example 2

Error:

airflow.exceptions.AirflowException: {'reason': 'invalidQuery', 'location': 'query', 'message': 'No matching signature for operator = for argument types: STRING, INT64. Supported signature: ANY = ANY at [13:5]'}

```
SELECT
  fx.base_currency AS base,
  fx.currency AS quote,
  COALESCE(SAFE_CAST(fx.ask AS NUMERIC), 1.0) AS ask,
  COALESCE(SAFE_CAST(fx.bid AS NUMERIC), 1.0) AS bid,
  CAST(NULL AS NUMERIC) AS midspread,
  CAST(NULL AS NUMERIC) AS prev_month_close_ask,
  CAST(NULL AS NUMERIC) AS prev_month_close_bid,
  CAST(NULL AS NUMERIC) AS prev_month_close_midspread,
  fx.dt,
FROM `{{ params.cdp_project }}.external_data_kitchen.raw_fx_rate` AS fx
WHERE dt = "{{ ds }}"
and 'foo' = 123
```

Diagnosis:

OpenAI - Diagnosis: SQL query contains a type mismatch in the WHERE clause.
OpenAI - Root Cause: The SQL query is attempting to compare a string ('foo') with an integer (123), which is not allowed in BigQuery.
OpenAI - Suggested Fixes: Ensure that both sides of the comparison in the WHERE clause are of the same type. For example, change 'foo' to an integer or 123 to a string.
OpenAI - Suggested Fixes: Use CAST or SAFE_CAST to explicitly convert one of the values to the type of the other.
OpenAI - Suggested Fixes: Remove or correct the erroneous condition in the WHERE clause if it is not needed.

Airflow Task Failure Diagnoser

Example 3

Error:

airflow.exceptions.AirflowException: 400:Bad Request

```
upload_fb_file_to_gcs = HttpToGoogleCloudStorageOperator(
    task_id=f'upload_fb_file_to_gcs_day{day}',
    destination_bucket=dag.params['airflow_bucket'],
    destination_object=f"{dag.params['gcs_staging_location_folder']}/{destination_gcs_file}",
    http_conn_id='facebook_paymentreports_http',
    endpoint='{{ var.json.facebook_paymentreports_http.client_id }}/repor',
    headers={'user-agent': 'King'},
    data={
        'date': '{{ macros.ds_add(ds, -%s) }}' % day,
        'type': dag.params['fb_report_type'],
        'access_token': '{{ var.json.facebook_paymentreports_http.client_id }}|{{ var.json.fac
    },
    compression='zip',
    retries=0
)
```

Diagnosis:

OpenAI - Diagnosis: The task 'upload_fb_file_to_gcs_day0' failed with a 400: Bad Request error.
OpenAI - Root Cause: The root cause of the failure is likely due to a typo in the endpoint URL. The endpoint is set to '{{ var.json.facebook_paymentreports_http.client_id }}/repor' instead of '{{ var.json.facebook_paymentreports_http.client_id }}/report'.
OpenAI - Suggested Fixes: Correct the endpoint URL from '{{ var.json.facebook_paymentreports_http.client_id }}/repor' to '{{ var.json.facebook_paymentreports_http.client_id }}/report'.
OpenAI - Suggested Fixes: Verify that the access token and other request parameters are correct and valid.
OpenAI - Suggested Fixes: Check the Facebook API documentation to ensure that the request parameters and headers are correctly formatted and meet the API requirements.

Airflow Task Failure Diagnoser

Example 4

Error:

requests.exceptions.HTTPError: 400 Client Error: Bad Request for url:
https://api.data.ai/v2.0/portfolio/download-channel?company_id=None&st
art_date=2024-07-22&end_date=2024-07-22&granularity=daily&countries
=AU%2CBR%2CCA%2CDE%2CES%2CFR%2CGB%2CJP%2CKR%2CMX%2CNO%2CRU%2CSE%2CTW%2CUS%2Cdevices=phone%2Ctablet&id=phone%2Candroid-tablet

```
download_channel = DataAIDownloadChannelOperator.partial(
    task_id='get-download-channel',
    dataai_conn_id='appannie_http',
    from_date='{{ prev_week(data_interval_end_date())[0] }}',
    to_date='{{ macros.ds_add(prev_week(data_interval_end_date())[1], 1) }}',
    countries=countries,
    devices=devices,
    granularity='daily',
    gcs_bucket=dag.params['airflow_bucket'],
    gcs_path='/'.join(('data_ai', 'download_channel', '{{ ds_nodash }}')),
).expand(company_id=XComArg(operator=get_companies))
```

Diagnosis:

OpenAI - Diagnosis: The task 'get-download-channel' failed due to an HTTP 400 Client Error: Bad Request.
OpenAI - Root Cause: The error is caused by an invalid or missing 'company_id' parameter in the API request URL.
OpenAI - Suggested Fixes: Ensure that the 'company_id' parameter is correctly set and not None before making the API request.
OpenAI - Suggested Fixes: Validate the 'company_id' values retrieved from the 'get-companies' task to ensure they are not empty or null.
OpenAI - Suggested Fixes: Add error handling to check for missing or invalid 'company_id' values and log appropriate error messages before making the API request.

Airflow Task Failure Diagnoser

Example 5

Error:

airflow.exceptions.AirflowException: Bad int64 value: 7.1.36

```
SELECT
  a.product_id,
  CASE WHEN a.market_code = "apple-store" THEN "ios" ELSE "google-play" END AS market_code,
  CAST(a.version AS INT64) AS version,
FROM `{{ params.cdp_project }}.data_ai.d_product` AS a
INNER JOIN `{{ params.cdp_project }}.data_ai.d_publisher` AS b
  ON a.publisher_id = b.publisher_id
INNER JOIN `{{ params.cdp_project }}.data_ai.d_company` AS c
  ON b.company_id = c.company_id
```

Diagnosis:

OpenAI - Diagnosis: The task 'create-products' failed due to a type casting error when attempting to cast a string value '7.1.36' to an INT64.
OpenAI - Root Cause: The SQL query attempts to cast a version string that contains non-integer characters (e.g., '7.1.36') to an INT64, which is not possible.
OpenAI - Suggested Fixes: Modify the SQL query to handle non-integer version strings appropriately, such as by extracting the major version number.
OpenAI - Suggested Fixes: Ensure that the data in the 'version' column only contains integer values before running the query.
OpenAI - Suggested Fixes: Use a different data type that can handle version strings, such as STRING, and perform any necessary parsing in subsequent steps.
OpenAI - Updated SQL:

```
SELECT
  a.product_id,
  CASE WHEN a.market_code = 'apple-store' THEN 'ios' ELSE 'google-play' END AS market_code,
  SAFE_CAST(REGEXP_EXTRACT(a.version, r'^[0-9]+') AS INT64) AS version
FROM `my-project.data_ai.d_product` AS a
INNER JOIN `my-project.data_ai.d_publisher` AS b
  ON a.publisher_id = b.publisher_id
INNER JOIN `my-project.data_ai.d_company` AS c
  ON b.company_id = c.company_id
```

Airflow Task Failure Diagnoser

Example 6

Error:

google.api_core.exceptions.BadRequest: 400 Query error: Array index 1 is out of bounds (overflow) at [5:1]

```
BEGIN TRANSACTION;

DELETE FROM `{{ project_id }}.staging_persistent.staging_selected_ab_test` WHERE kingappid = {{ params.kingappid }};

INSERT INTO `{{ project_id }}.staging_persistent.staging_selected_ab_test`
(
  kingappid,
  ab_test_name,
  ab_test_version
)
SELECT DISTINCT
  {{ params.kingappid }} AS kingappid,
  SPLIT(ab_test, ", version ")[OFFSET(0)] AS ab_test_name,
  CAST(SPLIT(ab_test, ", version ")[OFFSET(1)] AS INT64) AS ab_test_version,
FROM
  `{{ project_id }}.staging_temp.{{ params.temp_table_selected_test }}_{{ ds_nodash }}`
WHERE
  COALESCE(ab_test, "") != ""
;

COMMIT;
```

Diagnosis:

OpenAI - Updated SQL:

```
BEGIN TRANSACTION;
DELETE FROM `my-project.staging_persistent.staging_selected_ab_test` WHERE kingappid = 26;
INSERT INTO `my-project.staging_persistent.staging_selected_ab_test`
(
  kingappid,
  ab_test_name,
  ab_test_version
)
SELECT DISTINCT
  26 AS kingappid,
  SPLIT(ab_test, ", version ")[OFFSET(0)] AS ab_test_name,
  CASE
    WHEN ARRAY_LENGTH(SPLIT(ab_test, ", version ")) > 1 THEN CAST(SPLIT(ab_test, ", version ")[OFFSET(1)] AS INT64)
    ELSE NULL
  END AS ab_test_version
FROM
  `my-project.staging_temp.staging_selected_ab_test_26_20240822`
WHERE
  COALESCE(ab_test, "") != "";
COMMIT;
```

Diagnosis:

OpenAI - Diagnosis: The SQL query failed due to an array index out of bounds error when trying to access elements from the result of the SPLIT function.

OpenAI - Root Cause: The error indicates that the SPLIT function is returning an array with fewer elements than expected. Specifically, it seems that for some rows in the `ab_test` column, there is no version information present, leading to an attempt to access an index that does not exist.

OpenAI - Suggested Fixes: Add a condition to ensure that the `ab_test` string contains the expected delimiter before attempting to split it.

OpenAI - Suggested Fixes: Use a CASE statement to handle cases where the split does not yield the expected number of elements, providing a default value for `ab_test_version`.

OpenAI - Suggested Fixes: Validate the data in the `ab_test` column to ensure it conforms to the expected format before running the query.

Airflow Task Failure Diagnoser

Example 7

Error:

```
[2024-08-07, 09:25:23 BST] {taskinstance.py:2905} ERROR - Task failed with exception
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/site-packages/airflow/models/taskinstance.py", line 465, in _execute_task
    result = _execute_callable(context=context, **execute_callable_kwargs)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/site-packages/airflow/models/taskinstance.py", line 432, in _execute_callable
    return execute_callable(context=context, **execute_callable_kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/site-packages/airflow/models/baseoperator.py", line 401, in wrapper
    return func(self, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/airflow/include/operators/apptweak.py", line 88, in execute
    data = apptweak_hook.run_query(
           ^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/airflow/include/hooks/apptweak.py", line 164, in run_query
    api_account_id = self._get_api_account_id(app_store=app_store)
                     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/airflow/include/hooks/apptweak.py", line 79, in _get_api_account_id
    for account in self._accounts():
TypeError: 'NoneType' object is not iterable
```

Diagnosis:

OpenAI - Diagnosis: The task 'itunes-connect-channels.load-iap-to-gcs' failed due to a TypeError: 'NoneType' object is not iterable.
OpenAI - Root Cause: The error indicates that the code is attempting to iterate over a NoneType object, which is not allowed. This typically happens when a variable expected to be a list or other iterable is actually None.
OpenAI - Suggested Fixes: Check if the variable being iterated over is properly initialized and not None. Ensure that the configuration or data source providing this variable is correctly set up.
OpenAI - Suggested Fixes: Add a condition to check if the variable is None before attempting to iterate over it. If it is None, handle this case appropriately, perhaps by logging an error or skipping the iteration.
OpenAI - Suggested Fixes: Review the DAG configuration and the specific task parameters to ensure that all required fields are correctly populated and that no None values are being passed where a list or iterable is expected.

In Summary

- 👍 It works!
- 👍 Can provide clear and specific answers
- 👍 Running in live
- 👍 Great learning experience
- 👍 Provided contribution opportunities

- 👎 Common errors can be self-explanatory
- 👎 Not having error traceback limits deeper diagnosis
- 👎 Creating/modifying the Assistant and uploading files as part of the callback is not desirable
- 👎 Not compatible with other LLMs
- 👎 It's not free

GPT-4o

*** APPROXIMATED USAGE COST ***
Input Tokens: 4192, with a cost of: 0.02096
Output Tokens: 239, with a cost of: 0.003585
TOTAL COST: $0.024545

GPT-4o-mini

*** APPROXIMATED USAGE COST ***
Input Tokens: 4179, with a cost of: 0.000627
Output Tokens: 219, with a cost of: 0.000131
TOTAL COST: $0.000758

Thank you!

King

Making the World Playful

Questions?

nathan.hadfield@king.com

linkedin.com/in/nathanhadfield/

github.com/nathadfield

careers.king.com

Making the World Playful