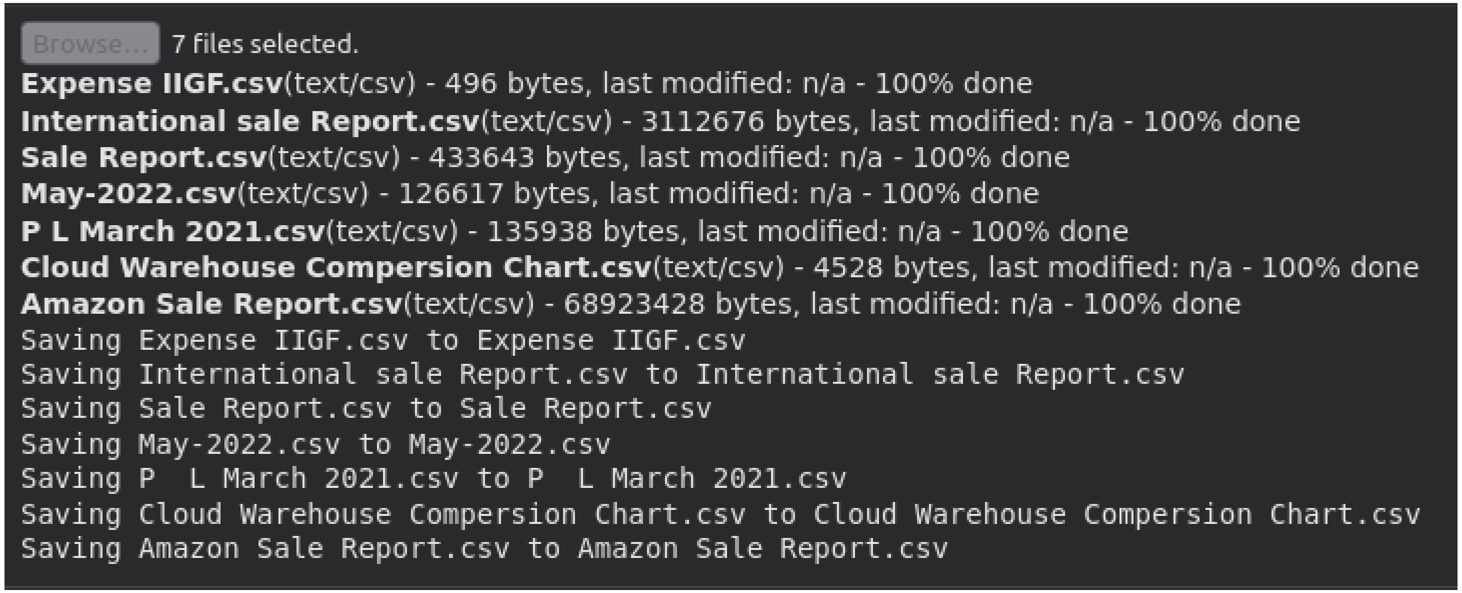
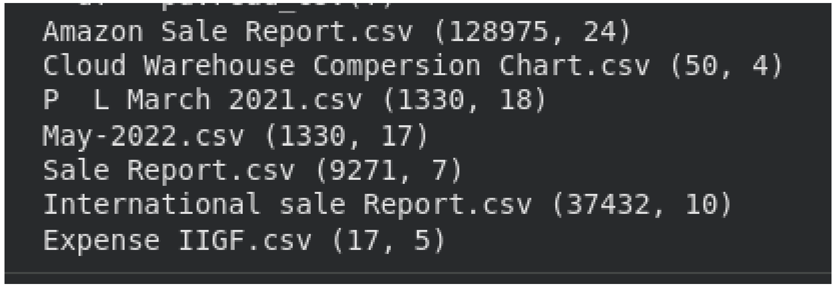
GenAI-Powered Retail Sales Insight Assistant

Data Loading

Loaded multiple raw sales CSV files into the workspace for further processing.



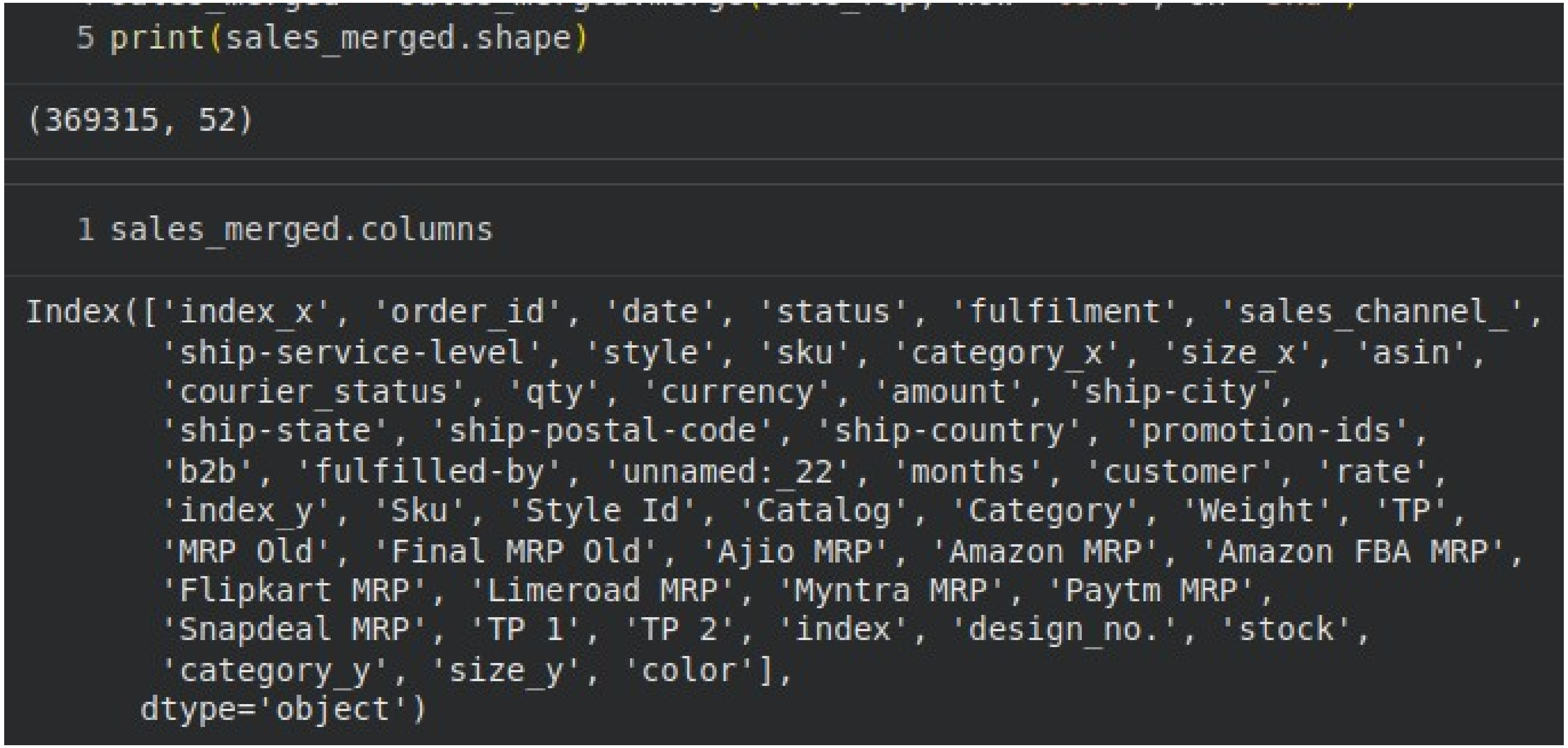
```
7 files selected.
Expense IIGF.csv(text/csv) - 496 bytes, last modified: n/a - 100% done
International sale Report.csv(text/csv) - 3112676 bytes, last modified: n/a - 100% done
Sale Report.csv(text/csv) - 433643 bytes, last modified: n/a - 100% done
May-2022.csv(text/csv) - 126617 bytes, last modified: n/a - 100% done
P L March 2021.csv(text/csv) - 135938 bytes, last modified: n/a - 100% done
Cloud Warehouse Compersion Chart.csv(text/csv) - 4528 bytes, last modified: n/a - 100% done
Amazon Sale Report.csv(text/csv) - 68923428 bytes, last modified: n/a - 100% done
Saving Expense IIGF.csv to Expense IIGF.csv
Saving International sale Report.csv to International sale Report.csv
Saving Sale Report.csv to Sale Report.csv
Saving May-2022.csv to May-2022.csv
Saving P  L March 2021.csv to P  L March 2021.csv
Saving Cloud Warehouse Compersion Chart.csv to Cloud Warehouse Compersion Chart.csv
Saving Amazon Sale Report.csv to Amazon Sale Report.csv
```



```
Amazon Sale Report.csv (128975, 24)
Cloud Warehouse Compersion Chart.csv (50, 4)
P  L March 2021.csv (1330, 18)
May-2022.csv (1330, 17)
Sale Report.csv (9271, 7)
International sale Report.csv (37432, 10)
Expense IIGF.csv (17, 5)
```

Data Consolidation

Merged sales datasets into a unified dataframe aligned by SKU, style, and order attributes.



```
print(sales_merged.shape)
```

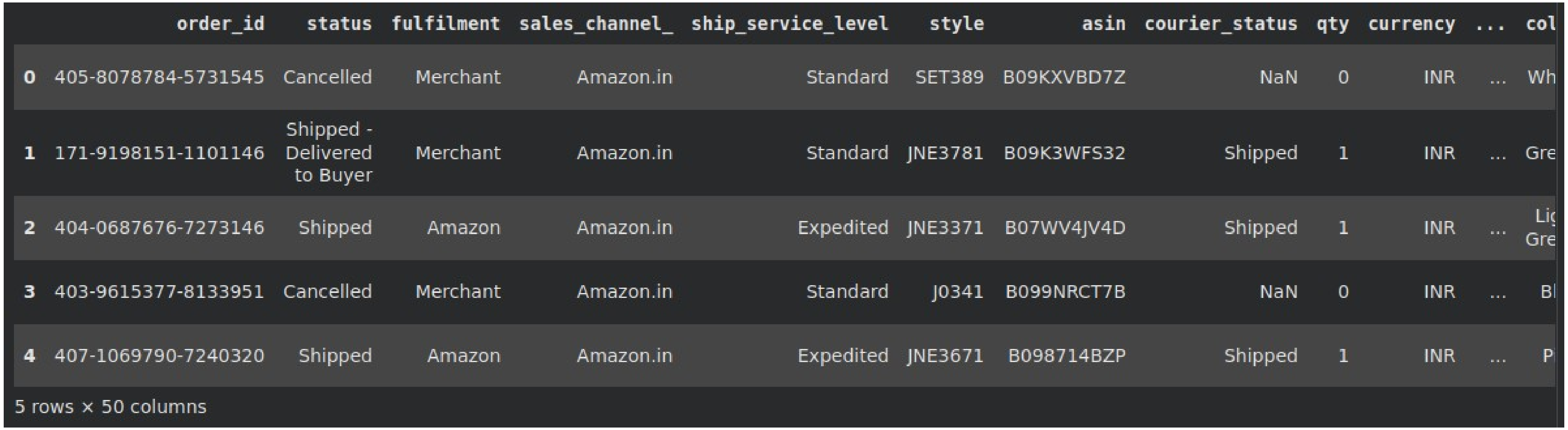
```
(369315, 52)
```

```
sales_merged.columns
```

```
Index(['index_x', 'order_id', 'date', 'status', 'fulfilment', 'sales_channel_',
       'ship-service-level', 'style', 'sku', 'category_x', 'size_x', 'asin',
       'courier_status', 'qty', 'currency', 'amount', 'ship-city',
       'ship-state', 'ship-postal-code', 'ship-country', 'promotion-ids',
       'b2b', 'fulfilled-by', 'unnamed:_22', 'months', 'customer', 'rate',
       'index_y', 'Sku', 'Style Id', 'Catalog', 'Category', 'Weight', 'TP',
       'MRP Old', 'Final MRP Old', 'Ajio MRP', 'Amazon MRP', 'Amazon FBA MRP',
       'Flipkart MRP', 'Limeroad MRP', 'Myntra MRP', 'Paytm MRP',
       'Snapdeal MRP', 'TP 1', 'TP 2', 'index', 'design_no.', 'stock',
       'category_y', 'size_y', 'color'],
      dtype='object')
```

Feature Engineering

Derived time-based fields like year, month, quarter, and day for analytical querying.

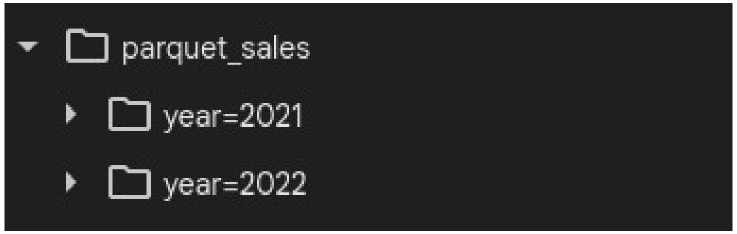


	order_id	status	fulfilment	sales_channel_	ship_service_level	style	asin	courier_status	qty	currency	...	col
0	405-8078784-5731545	Cancelled	Merchant	Amazon.in	Standard	SET389	B09KXVBD7Z	NaN	0	INR	...	Wh
1	171-9198151-1101146	Shipped - Delivered to Buyer	Merchant	Amazon.in	Standard	JNE3781	B09K3WFS32	Shipped	1	INR	...	Gre
2	404-0687676-7273146	Shipped	Amazon	Amazon.in	Expedited	JNE3371	B07WV4JV4D	Shipped	1	INR	...	Lig Gre
3	403-9615377-8133951	Cancelled	Merchant	Amazon.in	Standard	J0341	B099NRCT7B	NaN	0	INR	...	Bl
4	407-1069790-7240320	Shipped	Amazon	Amazon.in	Expedited	JNE3671	B098714BZP	Shipped	1	INR	...	P

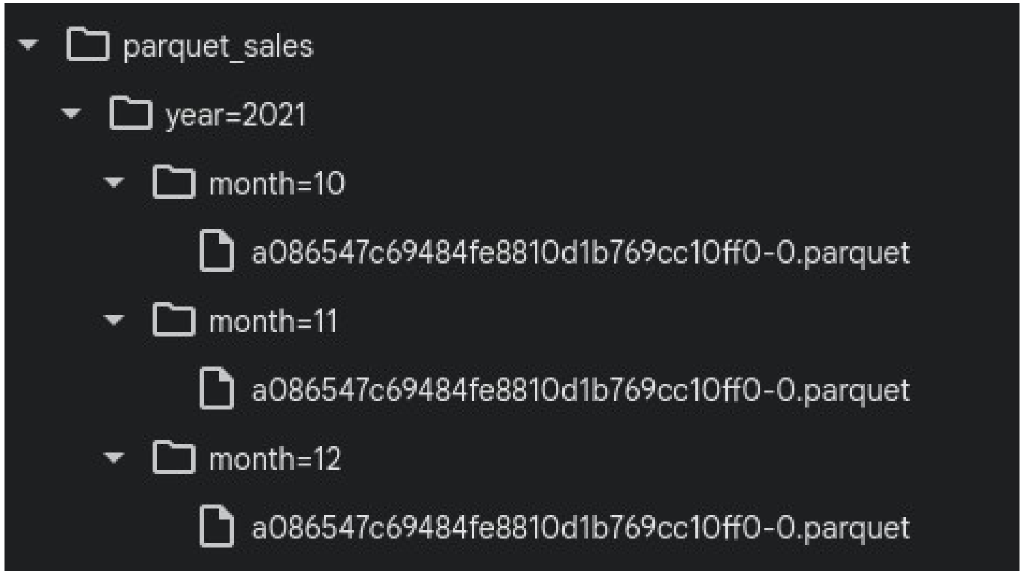
5 rows × 50 columns

Data Storage Optimization (Parquet Partitioning)

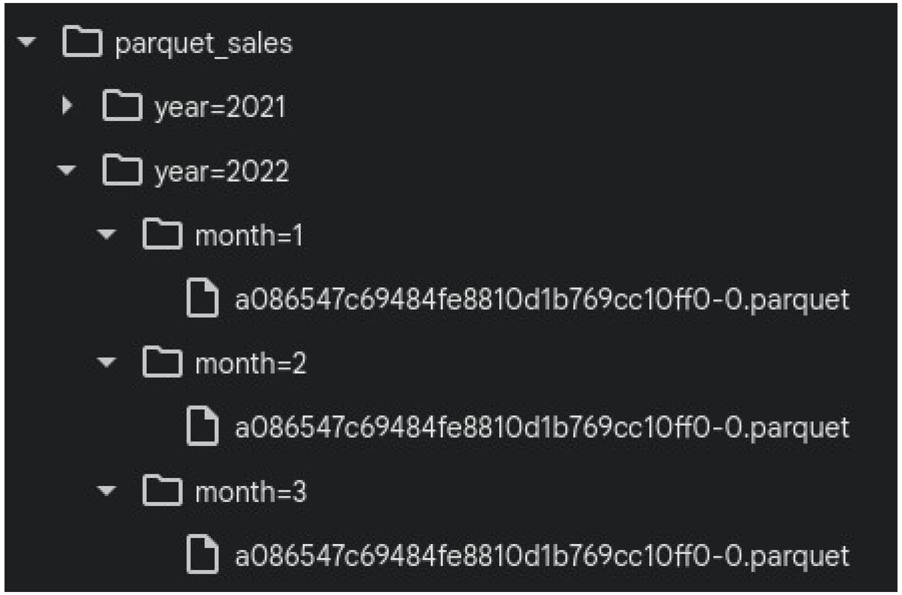
Partitioned the dataset into Parquet format by year and month for efficient access.



```
parquet_sales
  year=2021
  year=2022
```



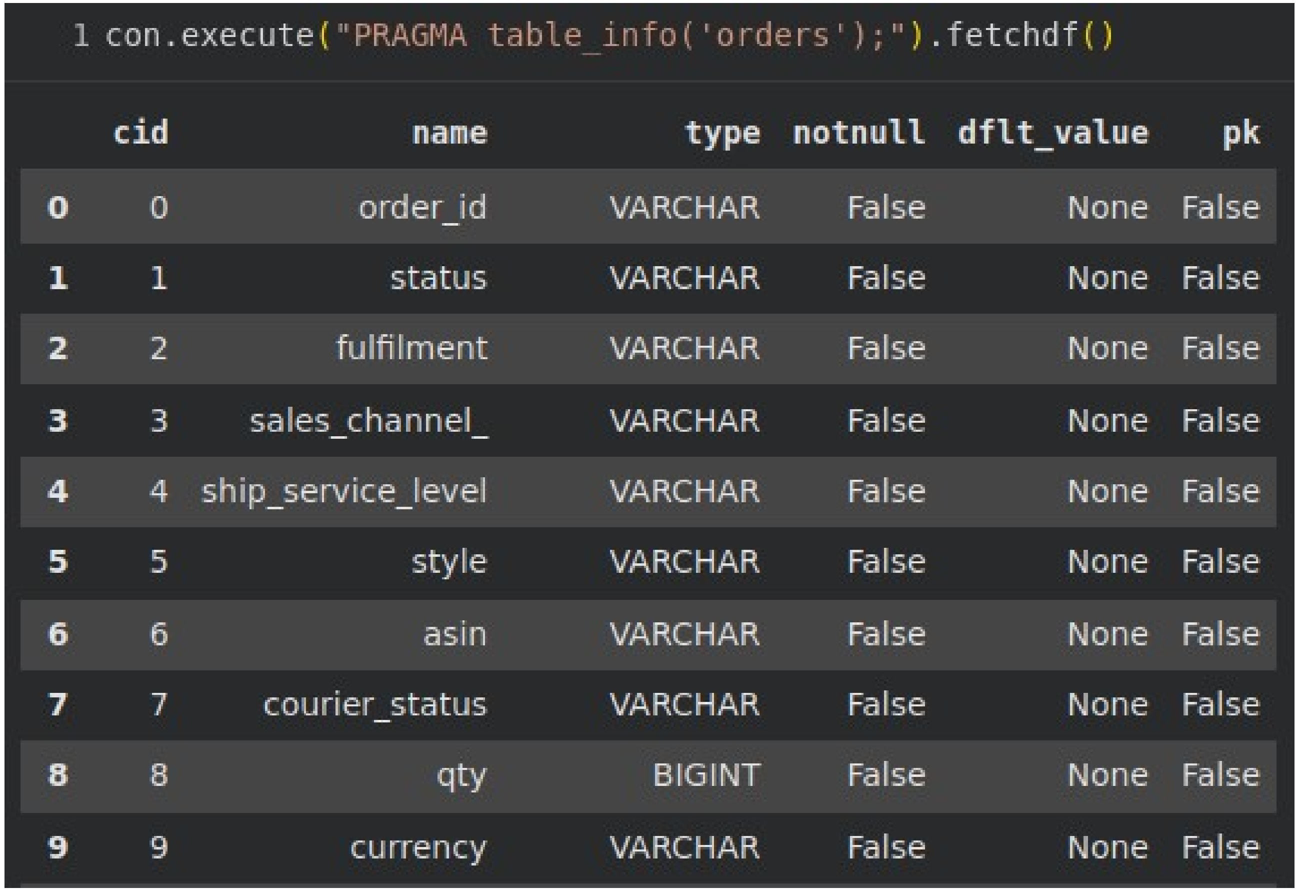
```
parquet_sales
  year=2021
    month=10
      a086547c69484fe8810d1b769cc10ff0-0.parquet
    month=11
      a086547c69484fe8810d1b769cc10ff0-0.parquet
    month=12
      a086547c69484fe8810d1b769cc10ff0-0.parquet
```



```
parquet_sales
  year=2021
  year=2022
    month=1
      a086547c69484fe8810d1b769cc10ff0-0.parquet
    month=2
      a086547c69484fe8810d1b769cc10ff0-0.parquet
    month=3
      a086547c69484fe8810d1b769cc10ff0-0.parquet
```

DuckDB Analytical Store Creation

Loaded the optimized dataset into DuckDB to enable fast SQL analytical queries.

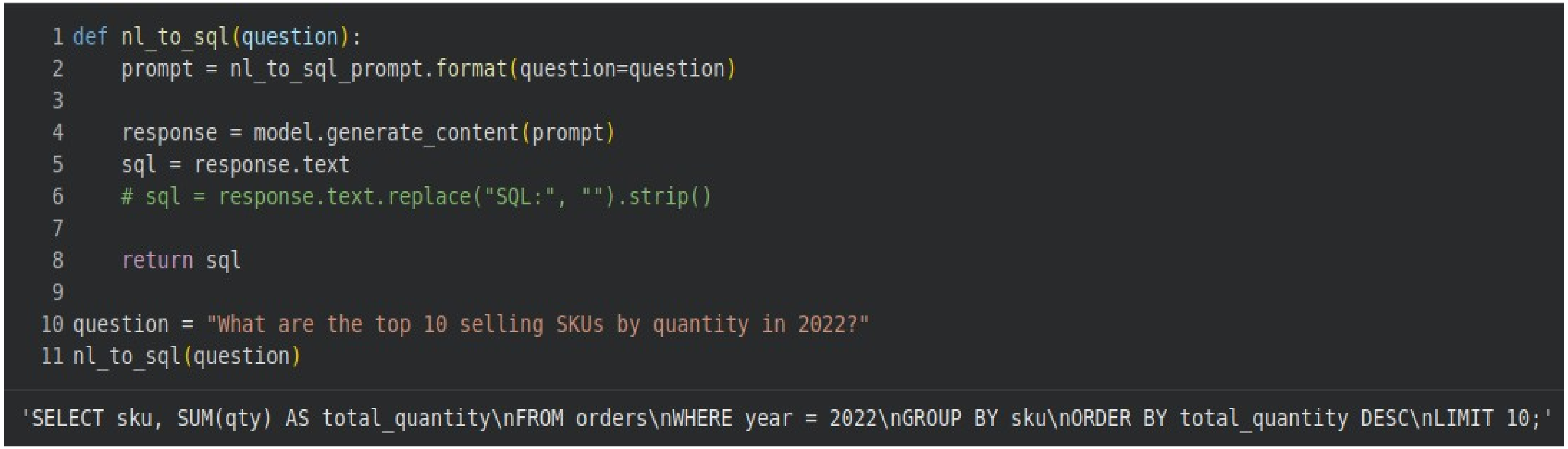


```
con.execute("PRAGMA table_info('orders');").fetchdf()
```

```
   cid               name     type  notnull  dflt_value     pk
0    0           order_id  VARCHAR    False        None  False
1    1             status  VARCHAR    False        None  False
2    2         fulfilment  VARCHAR    False        None  False
3    3     sales_channel_  VARCHAR    False        None  False
4    4 ship_service_level  VARCHAR    False        None  False
5    5              style  VARCHAR    False        None  False
6    6               asin  VARCHAR    False        None  False
7    7     courier_status  VARCHAR    False        None  False
8    8                qty   BIGINT    False        None  False
9    9           currency  VARCHAR    False        None  False
```

Natural Language to SQL Pipeline

Converted user questions into validated SQL queries using Gemini and prompt templates.

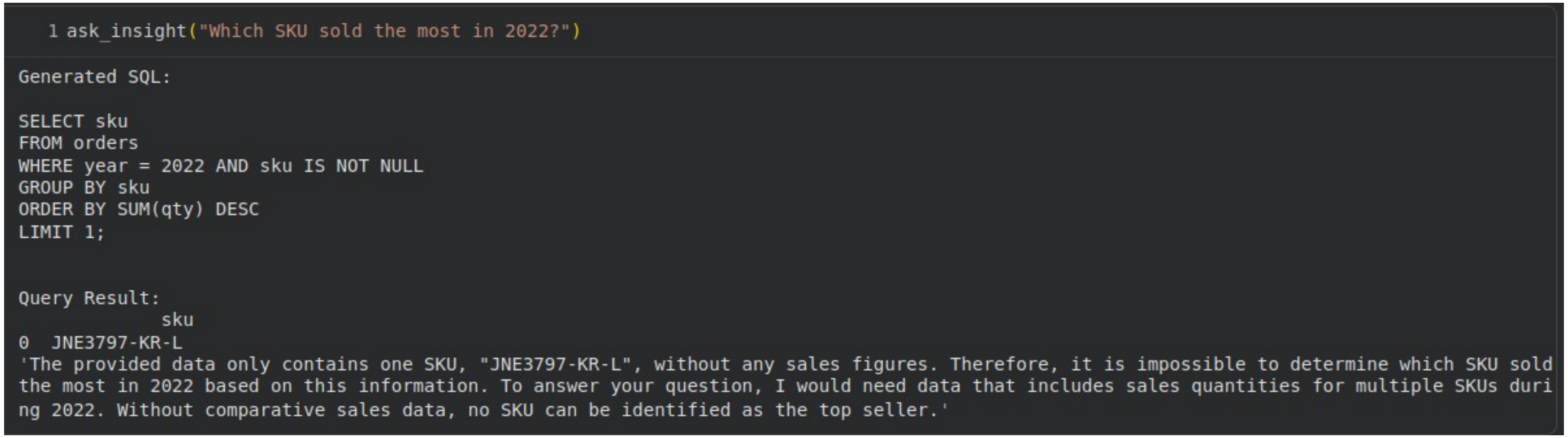


```python
def nl_to_sql(question):
    prompt = nl_to_sql_prompt.format(question=question)

    response = model.generate_content(prompt)
    sql = response.text
    # sql = response.text.replace("SQL:", "").strip()

    return sql

question = "What are the top 10 selling SKUs by quantity in 2022?"
nl_to_sql(question)
```

```
'SELECT sku, SUM(qty) AS total_quantity\nFROM orders\nWHERE year = 2022\nGROUP BY sku\nORDER BY total_quantity DESC\nLIMIT 10;'
```

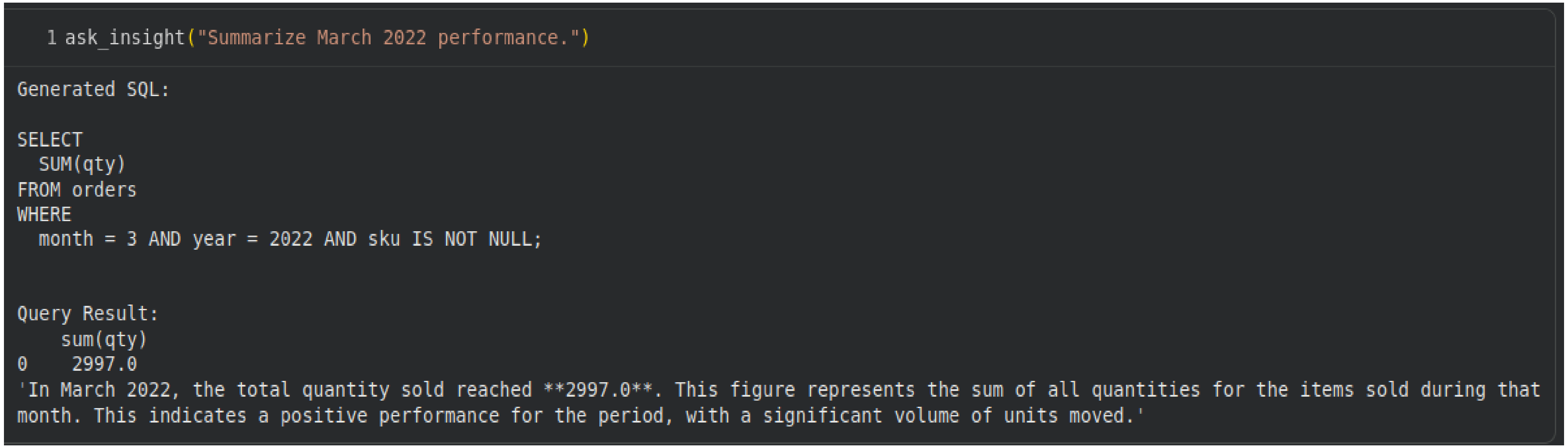
Result Execution & Insight Generation

Executed SQL in DuckDB and generated analytical business insights from results.



```
ask_insight("Which SKU sold the most in 2022?")
```

```
Generated SQL:

SELECT sku
FROM orders
WHERE year = 2022 AND sku IS NOT NULL
GROUP BY sku
ORDER BY SUM(qty) DESC
LIMIT 1;


Query Result:
            sku
0  JNE3797-KR-L
'The provided data only contains one SKU, "JNE3797-KR-L", without any sales figures. Therefore, it is impossible to determine which SKU sold
the most in 2022 based on this information. To answer your question, I would need data that includes sales quantities for multiple SKUs duri
ng 2022. Without comparative sales data, no SKU can be identified as the top seller.'
```

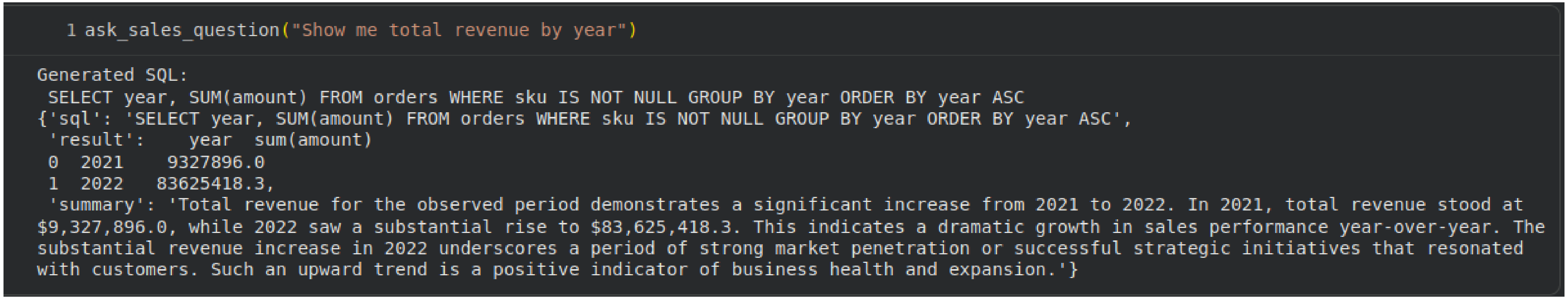


```
ask_insight("Summarize March 2022 performance.")
```

```
Generated SQL:

SELECT
  SUM(qty)
FROM orders
WHERE
  month = 3 AND year = 2022 AND sku IS NOT NULL;


Query Result:
    sum(qty)
0    2997.0
'In March 2022, the total quantity sold reached **2997.0**. This figure represents the sum of all quantities for the items sold during that
month. This indicates a positive performance for the period, with a significant volume of units moved.'
```

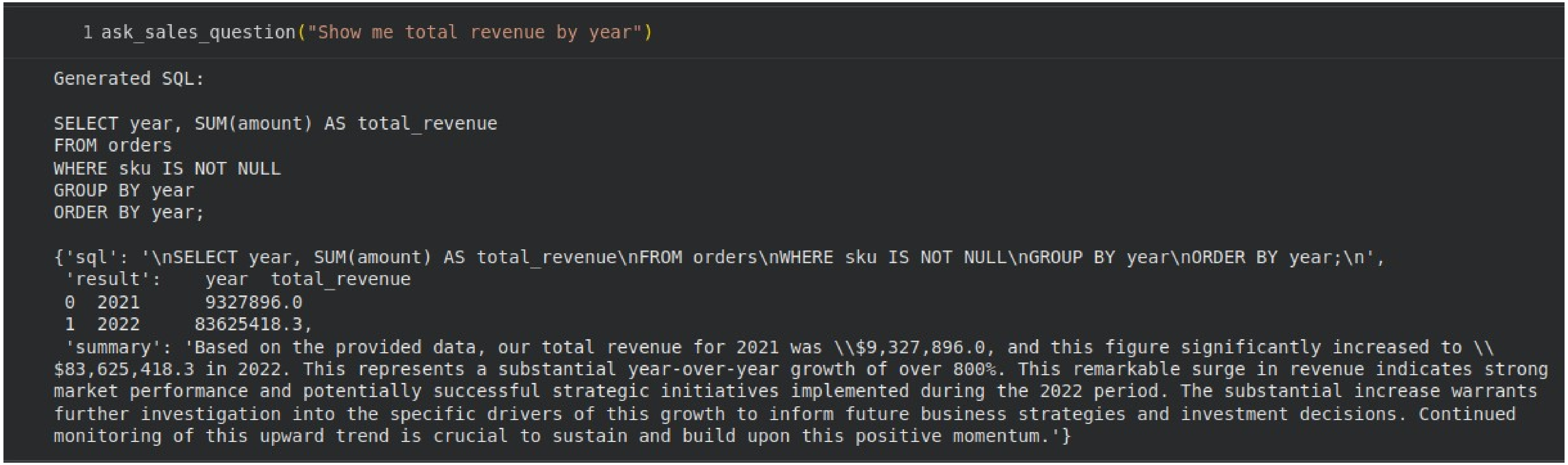
Enhanced NL and SQL Usability

Improved wording, structure, and formatting to deliver clearer, analyst-friendly responses.



```
ask_sales_question("Show me total revenue by year")
```

```
Generated SQL:
 SELECT year, SUM(amount) FROM orders WHERE sku IS NOT NULL GROUP BY year ORDER BY year ASC
{'sql': 'SELECT year, SUM(amount) FROM orders WHERE sku IS NOT NULL GROUP BY year ORDER BY year ASC',
 'result':    year  sum(amount)
 0  2021    9327896.0
 1  2022   83625418.3,
 'summary': 'Total revenue for the observed period demonstrates a significant increase from 2021 to 2022. In 2021, total revenue stood at
$9,327,896.0, while 2022 saw a substantial rise to $83,625,418.3. This indicates a dramatic growth in sales performance year-over-year. The
substantial revenue increase in 2022 underscores a period of strong market penetration or successful strategic initiatives that resonated
with customers. Such an upward trend is a positive indicator of business health and expansion.'}
```

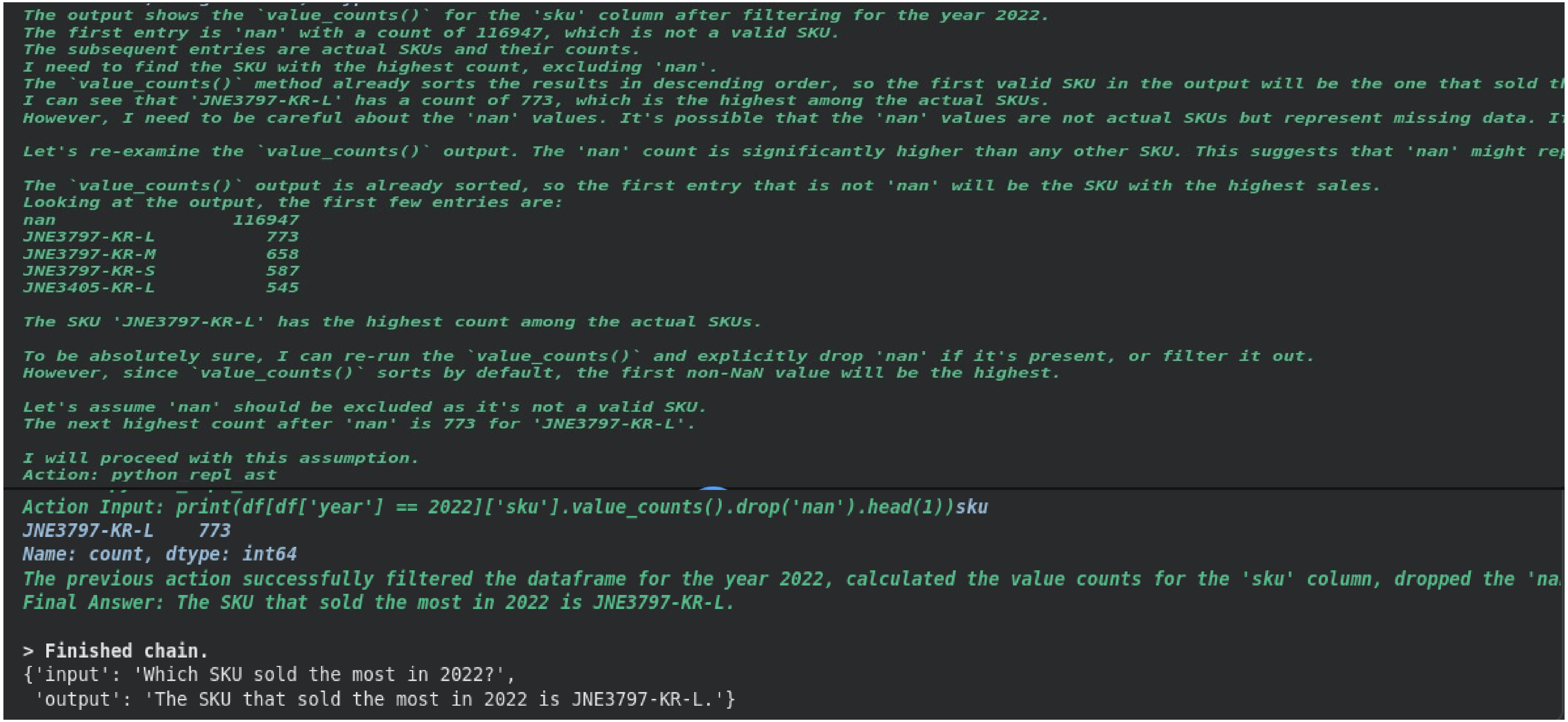


```
ask_sales_question("Show me total revenue by year")
```

```
Generated SQL:

SELECT year, SUM(amount) AS total_revenue
FROM orders
WHERE sku IS NOT NULL
GROUP BY year
ORDER BY year;

{'sql': '\nSELECT year, SUM(amount) AS total_revenue\nFROM orders\nWHERE sku IS NOT NULL\nGROUP BY year\nORDER BY year;\n',
 'result':    year  total_revenue
 0  2021      9327896.0
 1  2022     83625418.3,
 'summary': 'Based on the provided data, our total revenue for 2021 was \\$9,327,896.0, and this figure significantly increased to \\
$83,625,418.3 in 2022. This represents a substantial year-over-year growth of over 800%. This remarkable surge in revenue indicates strong
market performance and potentially successful strategic initiatives implemented during the 2022 period. The substantial increase warrants
further investigation into the specific drivers of this growth to inform future business strategies and investment decisions. Continued
monitoring of this upward trend is crucial to sustain and build upon this positive momentum.'}
```

LangChain Agent Integration

Integrated a dataframe agent to autonomously analyze data and produce accurate answers.

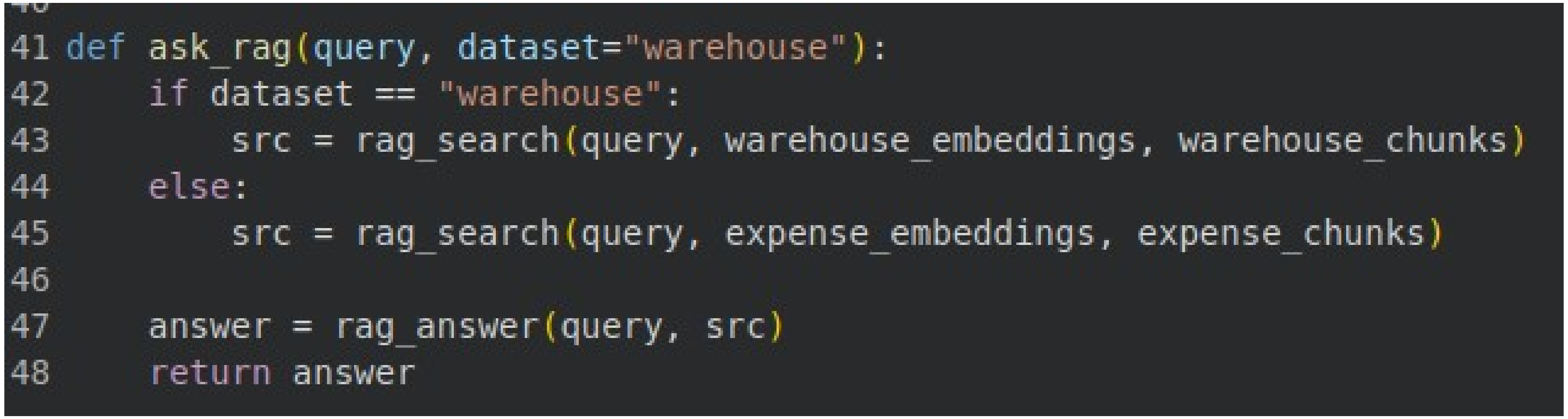


```
The output shows the `value_counts()` for the 'sku' column after filtering for the year 2022.
The first entry is 'nan' with a count of 116947, which is not a valid SKU.
The subsequent entries are actual SKUs and their counts.
I need to find the SKU with the highest count, excluding 'nan'.
The `value_counts()` method already sorts the results in descending order, so the first valid SKU in the output will be the one that sold th
I can see that 'JNE3797-KR-L' has a count of 773, which is the highest among the actual SKUs.
However, I need to be careful about the 'nan' values. It's possible that the 'nan' values are not actual SKUs but represent missing data. It

Let's re-examine the `value_counts()` output. The 'nan' count is significantly higher than any other SKU. This suggests that 'nan' might rep

The `value_counts()` output is already sorted, so the first entry that is not 'nan' will be the SKU with the highest sales.
Looking at the output, the first few entries are:
nan                 116947
JNE3797-KR-L           773
JNE3797-KR-M           658
JNE3797-KR-S           587
JNE3405-KR-L           545

The SKU 'JNE3797-KR-L' has the highest count among the actual SKUs.

To be absolutely sure, I can re-run the `value_counts()` and explicitly drop 'nan' if it's present, or filter it out.
However, since `value_counts()` sorts by default, the first non-NaN value will be the highest.

Let's assume 'nan' should be excluded as it's not a valid SKU.
The next highest count after 'nan' is 773 for 'JNE3797-KR-L'.

I will proceed with this assumption.
Action: python_repl_ast
```

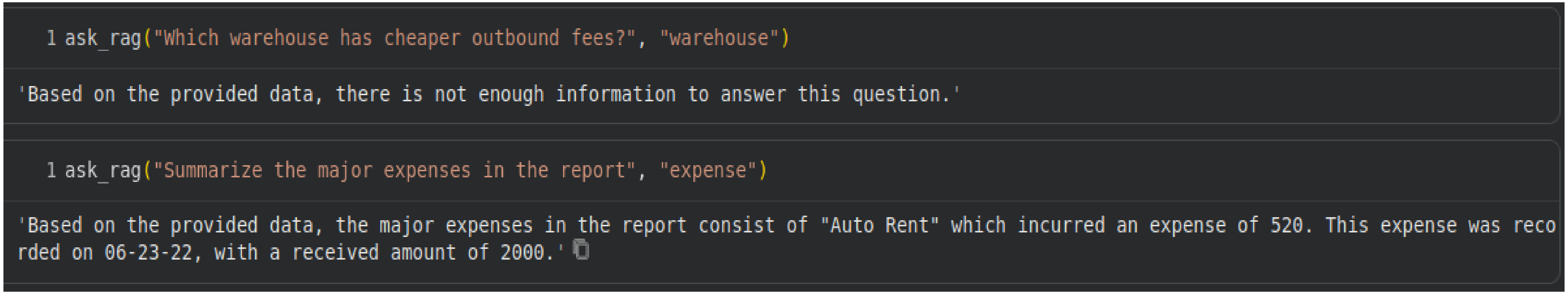
```
Action Input: print(df[df['year'] == 2022]['sku'].value_counts().drop('nan').head(1))sku
JNE3797-KR-L    773
Name: count, dtype: int64
The previous action successfully filtered the dataframe for the year 2022, calculated the value counts for the 'sku' column, dropped the 'na
Final Answer: The SKU that sold the most in 2022 is JNE3797-KR-L.

> Finished chain.
{'input': 'Which SKU sold the most in 2022?',
 'output': 'The SKU that sold the most in 2022 is JNE3797-KR-L.'}
```

RAG Contextual Enrichment

Used relevant reference text to enrich responses with context while avoiding hallucination.



```python
def ask_rag(query, dataset="warehouse"):
    if dataset == "warehouse":
        src = rag_search(query, warehouse_embeddings, warehouse_chunks)
    else:
        src = rag_search(query, expense_embeddings, expense_chunks)

    answer = rag_answer(query, src)
    return answer
```



```
ask_rag("Which warehouse has cheaper outbound fees?", "warehouse")
```

```
'Based on the provided data, there is not enough information to answer this question.'
```

```
ask_rag("Summarize the major expenses in the report", "expense")
```

```
'Based on the provided data, the major expenses in the report consist of "Auto Rent" which incurred an expense of 520. This expense was reco
rded on 06-23-22, with a received amount of 2000.'
```