1. Introduction to PharmaKB API

PharmaKB offers a robust API for financial analysts and data scientists in the pharmaceutical sector, providing in-depth data on drugs, companies, and indications. This tutorial demonstrates using PharmaKB's GraphQL queries to correlate drug approval dates with stock performance, a key aspect of market analysis in this industry.

2. Understanding GraphQL Queries in PharmaKB

PharmaKB utilizes GraphQL, allowing users to craft queries for specific data needs. Two primary queries are highlighted in this tutorial: one for collecting stock data and another for product information.

Collect Stock Data Query:

```
graphql
{
  companies(offset:0, limit: 10) {
    result {
      name
      stockData {
        equities(start:"1900-01-01", end: "2999-12-31") {
          open
          tradeDate
        }}}}
}
```

This query fetches the opening stock prices and trade dates for companies. The `offset` and `limit` parameters facilitate pagination, crucial for handling large datasets.
A lot of fields include optional pagination parameters. If your request fails without them try splitting your query into multiple requests using pagination.

Collect Products Query:

```
graphql
{
  companies(offset:0, limit: 10) {
    result {
      name
      info {
        result {
          tradename
          marketedSince
        }}}}
}
```

This query retrieves product names and their market introduction dates from the `info` field. The `info` field contains a list of company products, which is paginated as well. For this example we query all of them, but if you are trying to fetch a lot of products you might want to paginate this field as well.
Separating these queries enhances performance and allows splitting data collection into multiple separate requests.

3. Collecting and Analyzing Data

Executing these queries in PharmaKB yields two datasets: one detailing stock performance and another outlining product launch timelines. Analysts can merge these datasets to examine how a new drug's market entry impacts a company's stock performance.

For instance, analysts might compare the stock price trends before and after a drug's launch date to gauge investor reactions or predict future stock movements based on similar past events.

4. Optimizing Queries for Performance

Efficient use of PharmaKB involves not just understanding GraphQL syntax but also strategic query construction. While GraphQL's flexibility is advantageous, it requires careful management to avoid overloading the system, especially when dealing with large amounts of interconnected data.

Key tips include:

- Use pagination through `offset` and `limit`.
- Split complex data requests into multiple queries to avoid performance bottlenecks.
- Initially fetch key identifiers (like company or drug IDs) and then detail queries around these identifiers.

5. Python Example

The following Python example demonstrates basic API calls to PharmaKB, handling authentication and utilizing the GraphQL queries discussed in this tutorial.

Python Code Example:

```python
import requests
from string import Template
token = "your-token-here"

def fetch_data(query):
  """ Fetch data using the provided GraphQL query. """
  headers = {"Authorization": token}
```

```
response = requests.post("https://app.pharmakb.com/api/v1/graphql/", json={'query':
query}, headers=headers)
    return response.json()

def collect_data(query_template, count_query):
    """ Collect data in a paginated fashion. """
    data = []
    total_count = fetch_data(count_query)['data']['companies']['matchedCount']
    limit = 10
    for offset in range(0, total_count, limit):
        query = query_template.substitute(offset=offset, limit=limit)
        page_data = fetch_data(query)['data']['companies']['result']
        data.extend(page_data)
    return data

# Queries
count_query = "{ companies { matchedCount } }"

stock_query_template = Template("""
{
  companies(offset: $offset, limit: $limit ) {
    result {
      name
      stockData {
        equities(start:"1900-01-01", end: "2999-12-31") {
          open
          tradeDate
        } } } } } """)

product_query_template = Template("""
{
  companies(offset: $offset, limit: $limit ) {
    result {
      name
      info {
        result {
          tradename
          marketedSince
        } } } } } """)

# Collecting Stock and Product Data
stock_data = collect_data(stock_query_template, count_query)
product_data = collect_data(product_query_template, count_query)

#  Process and analyze the data as needed
```