GPT-3.5 vs GPT-4

Task

GPT-3.5 and GPT-4 are OpenAI's foundational models trained on plethora of data of different varieties and can be prompted to help accomplish various types of tasks.

Here I am going to try `Text to SQL` task i.e given a text command containing the info about a SQL Database and an instruction, the model is prompted to output an SQL query to accomplish the task.

I used Yahoo-Finance library to retrieve some stock data from an index and stored it in a SQL Database and got the table-info which i use later to create the prompt. Here is an example:

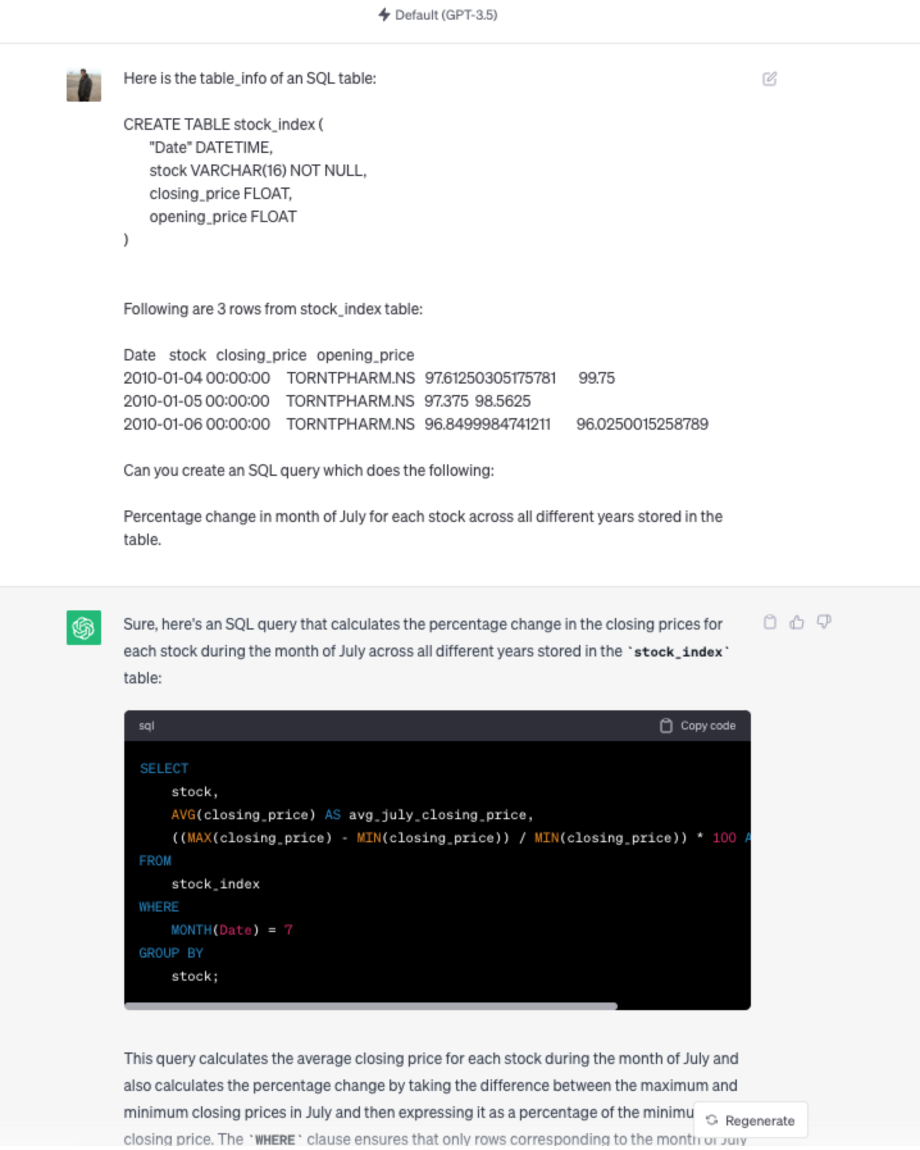
```
import os
import json
import pandas as pd
import numpy as np
import yfinance as yf
import sqlalchemy
import math
import guardrails as gd
from datetime import datetime
from datetime import timedelta
from sqlalchemy import create_engine, insert, MetaData, Table, Column, String, Integer, select, column

from llama_index import SQLDatabase, ServiceContext

from rich import print
```

```
def tickers_in_index():
    df = pd.read_csv('data.csv')
    return np.array(df['SYMBOL \n'][1:].values).tolist()

def get_pandas_frame(start_year, end_year):
    stock_tickers = tickers_in_index()
    stock_tickers = list(map(lambda x : x +'.NS' , stock_tickers))
    return stock_tickers, yf.download(" ".join(stock_tickers), start=f"{start_year}-01-01", end=f"{end_year}-12-31")

def make_sql_db(db_name):
    engine = create_engine('sqlite:///:memory:', echo=False)
    tickers, stock_frame = get_pandas_frame(start_year=2010, end_year=2023)
    stock_frame.to_sql(db_name, con=engine)
    return stock_frame, engine
```

```
stock_tickers, stock_frame = get_pandas_frame(start_year=2010, end_year=2023)

engine = create_engine("sqlite:///:memory:")
metadata_obj = MetaData()

# create city SQL table
table_name = "stock_index"
stock_index_table = Table(
    table_name,
    metadata_obj,
    Column("Date", sqlalchemy.DateTime, default=datetime.utcnow),
    Column("stock", String(16), nullable=False),
    Column("closing_price", sqlalchemy.Float),
    Column("opening_price", sqlalchemy.Float),
)
opening_price_table = Table(
    "first_valid_date",
    metadata_obj,
    Column("Date", sqlalchemy.DateTime, default=datetime.utcnow),
)
last_price_table = Table(
    "last_valid_date",
    metadata_obj,
    Column("Date", sqlalchemy.DateTime, default=datetime.utcnow),
)
metadata_obj.create_all(engine)

with engine.connect() as connection:
    for ticker in ['TORNTPHARM.NS', 'CUMMINSIND.NS', 'ATUL.NS', 'RELIANCE.NS']:
        for key, value in stock_frame['Close'][ticker].items():
            open_value = stock_frame['Open'][ticker][key]
            if not math.isnan(value) and not math.isnan(open_value):
                stmt = insert(stock_index_table).values({"Date": key.to_pydatetime(), "stock" : ticker, "closing_price": value, "opening_pr
                cursor = connection.execute(stmt)
    connection.commit()
```

```
from llama_index import SQLDatabase
sql_database = SQLDatabase(engine, include_tables=["stock_index"])
print(sql_database.table_info)


'''
CREATE TABLE stock_index (
        "Date" DATETIME,
        stock VARCHAR(16) NOT NULL,
        closing_price FLOAT,
        opening_price FLOAT
)

/*
3 rows from stock_index table:
Date    stock   closing_price   opening_price
2010-01-04 00:00:00     TORNTPHARM.NS   97.61250305175781       99.75
2010-01-05 00:00:00     TORNTPHARM.NS   97.375  98.5625
2010-01-06 00:00:00     TORNTPHARM.NS   96.8499984741211        96.0250015258789
*/
'''
```

GPT-3.5 👎



Here is the table_info of an SQL table:

```
CREATE TABLE stock_index (
    "Date" DATETIME,
    stock VARCHAR(16) NOT NULL,
    closing_price FLOAT,
    opening_price FLOAT
)
```

Following are 3 rows from stock_index table:

```
Date   stock  closing_price  opening_price
2010-01-04 00:00:00    TORNTPHARM.NS  97.61250305175781      99.75
2010-01-05 00:00:00    TORNTPHARM.NS  97.375  98.5625
2010-01-06 00:00:00    TORNTPHARM.NS  96.8499984741211      96.0250015258789
```

Can you create an SQL query which does the following:

Percentage change in month of July for each stock across all different years stored in the table.

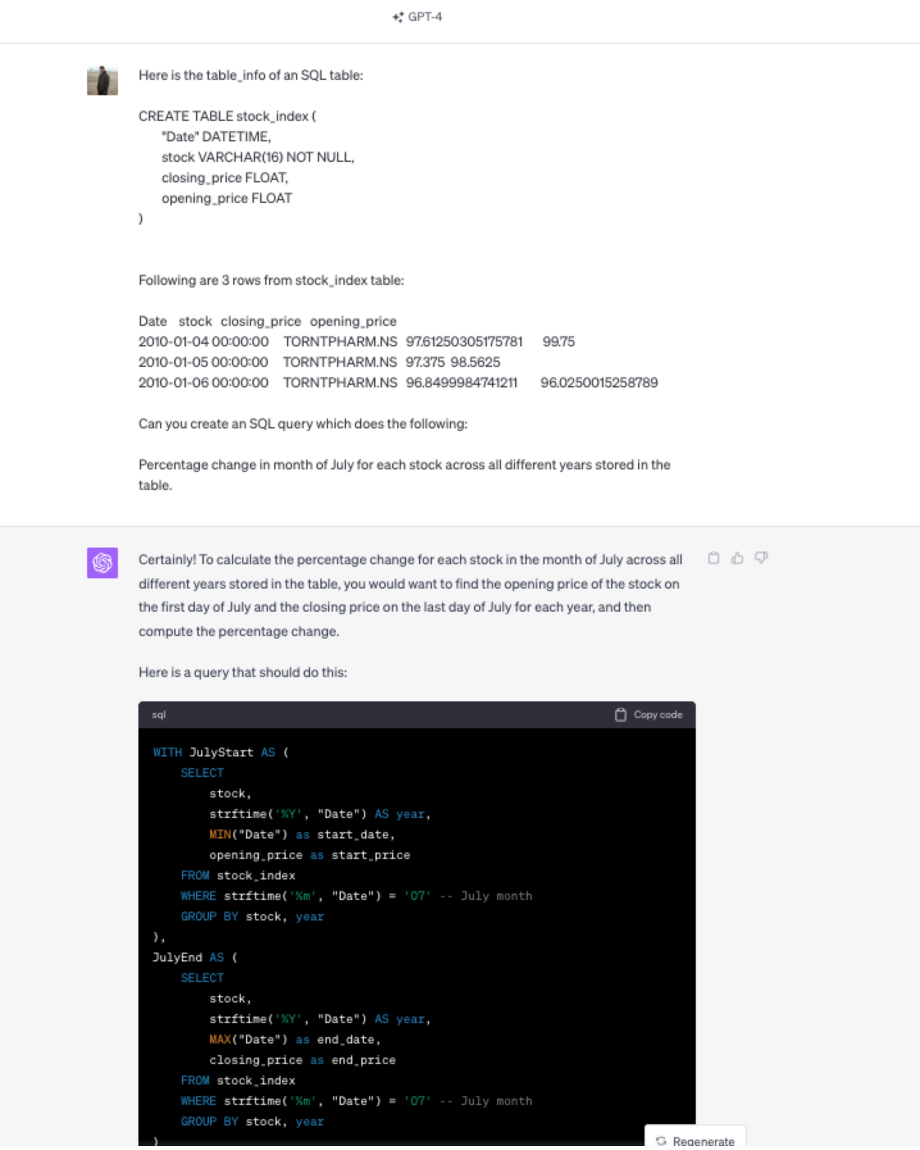
Sure, here's an SQL query that calculates the percentage change in the closing prices for each stock during the month of July across all different years stored in the `stock_index` table:

```sql
SELECT
    stock,
    AVG(closing_price) AS avg_july_closing_price,
    ((MAX(closing_price) - MIN(closing_price)) / MIN(closing_price)) * 100 A
FROM
    stock_index
WHERE
    MONTH(Date) = 7
GROUP BY
    stock;
```

This query calculates the average closing price for each stock during the month of July and also calculates the percentage change by taking the difference between the maximum and minimum closing prices in July and then expressing it as a percentage of the minimu closing price. The `WHERE` clause ensures that only rows corresponding to the month of July

- Not able to comprehend the instruction properly: I asked for % change in month of July for all stocks for each year but it returned assimilated result across all years.
- Could try other prompts to guide the model better i.e task splitting.

GPT-4 👍



Here is the table_info of an SQL table:

```
CREATE TABLE stock_index (
    "Date" DATETIME,
    stock VARCHAR(16) NOT NULL,
    closing_price FLOAT,
    opening_price FLOAT
)
```

Following are 3 rows from stock_index table:

```
Date   stock  closing_price  opening_price
2010-01-04 00:00:00    TORNTPHARM.NS  97.61250305175781      99.75
2010-01-05 00:00:00    TORNTPHARM.NS  97.375  98.5625
2010-01-06 00:00:00    TORNTPHARM.NS  96.8499984741211      96.0250015258789
```

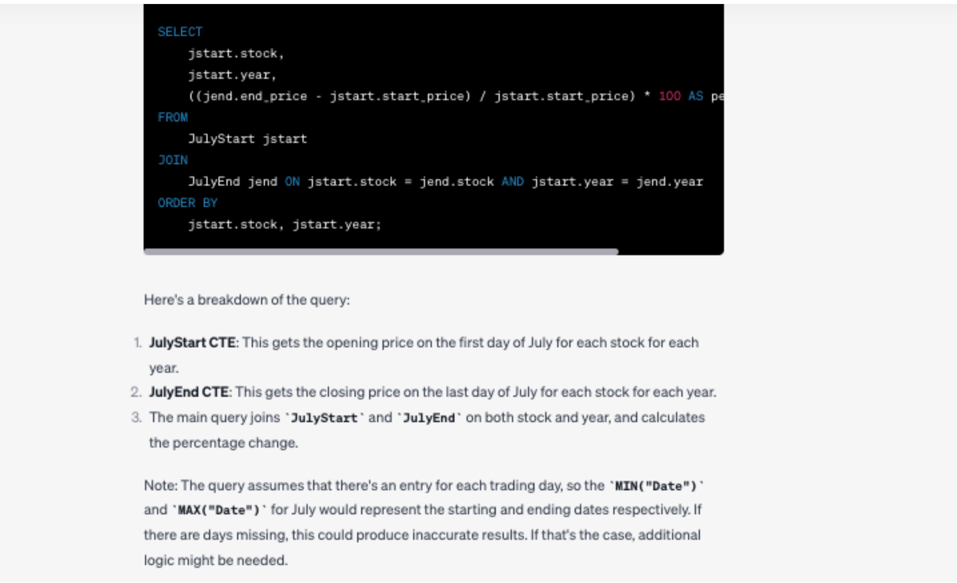
Can you create an SQL query which does the following:

Percentage change in month of July for each stock across all different years stored in the table.

Certainly! To calculate the percentage change for each stock in the month of July across all different years stored in the table, you would want to find the opening price of the stock on the first day of July and the closing price on the last day of July for each year, and then compute the percentage change.

Here is a query that should do this:

```sql
WITH JulyStart AS (
    SELECT
        stock,
        strftime('%Y', "Date") AS year,
        MIN("Date") as start_date,
        opening_price as start_price
    FROM stock_index
    WHERE strftime('%m', "Date") = '07' -- July month
    GROUP BY stock, year
),
JulyEnd AS (
    SELECT
        stock,
        strftime('%Y', "Date") AS year,
        MAX("Date") as end_date,
        closing_price as end_price
    FROM stock_index
    WHERE strftime('%m', "Date") = '07' -- July month
    GROUP BY stock, year
)
```



```
SELECT
    jstart.stock,
    jstart.year,
    ((jend.end_price - jstart.start_price) / jstart.start_price) * 100 AS pe
FROM
    JulyStart jstart
JOIN
    JulyEnd jend ON jstart.stock = jend.stock AND jstart.year = jend.year
ORDER BY
    jstart.stock, jstart.year;
```

Here's a breakdown of the query:

1. **JulyStart CTE**: This gets the opening price on the first day of July for each stock for each year.
2. **JulyEnd CTE**: This gets the closing price on the last day of July for each stock for each year.
3. The main query joins `` `JulyStart` `` and `` `JulyEnd` `` on both stock and year, and calculates the percentage change.

Note: The query assumes that there's an entry for each trading day, so the `` `MIN("Date")` `` and `` `MAX("Date")` `` for July would represent the starting and ending dates respectively. If there are days missing, this could produce inaccurate results. If that's the case, additional logic might be needed.

- Made two tables referring the opening and closing price of each stock in the month of July and returned % change for each stock for each year.
- Able to execute the generated query successfully:

```
# SQL query as given above
sql_query = """
WITH JulyStart AS (
    SELECT
        stock,
        strftime('%Y', "Date") AS year,
        MIN("Date") as start_date,
        opening_price as start_price
    FROM stock_index
    WHERE strftime('%m', "Date") = '07' -- July month
    GROUP BY stock, year
),
JulyEnd AS (
    SELECT
        stock,
        strftime('%Y', "Date") AS year,
        MAX("Date") as end_date,
        closing_price as end_price
    FROM stock_index
    WHERE strftime('%m', "Date") = '07' -- July month
    GROUP BY stock, year
)

SELECT
    jstart.stock,
    jstart.year,
    ((jend.end_price - jstart.start_price) / jstart.start_price) * 100 AS percentage_change
FROM
    JulyStart jstart
JOIN
    JulyEnd jend ON jstart.stock = jend.stock AND jstart.year = jend.year

ORDER BY
    jstart.stock, jstart.year;
"""

# Use the engine to execute the query
with engine.connect() as connection:
    result = connection.execute(text(sql_query))
    rows = result.fetchall()

# Print the results
```

```
for row in rows:
    print(row)
```