Team 20 Code Sample doc

Environment setup

1. Spark installation: MAC:
https://github.com/charlie-ph/BigDataAnalytics/blob/master/Installations-HowTos/How-To-Install-Spark-On-MACOS.md

Windows:
https://github.com/charlie-ph/BigDataAnalytics/blob/master/Installations-HowTos/How-To-Install-Spark-On-Windows.md

2. PostgreSQL installation:
https://www.postgresql.org/docs/current/tutorial-install.html

How to run the code

Part 1: PostgreSQL
1) run code/Postgre/create_file.sql to create Data-warehouse in Postgre
2) import all csv files in /data into your database
3) run code/Postgre/SQL_basic_search.sql to get SQL results

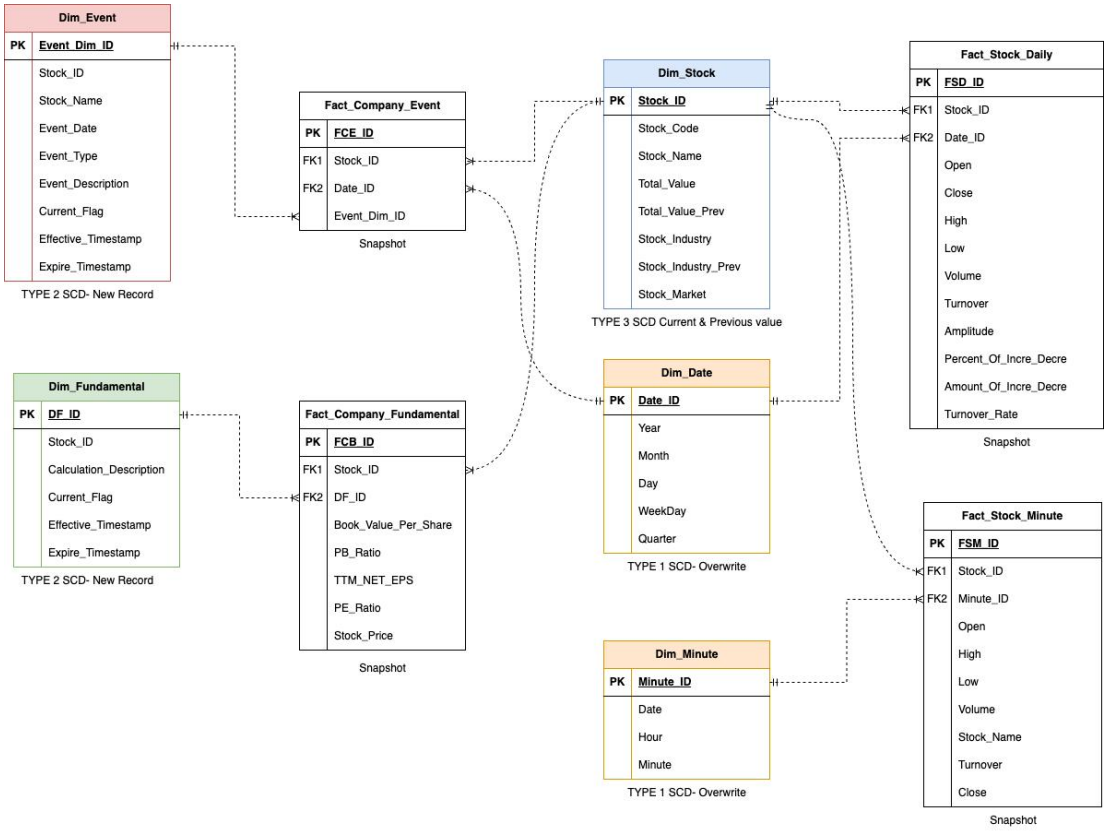
Part2: Spark
1) run code/Spark/Example.ipynb in your notebook.
2) make sure you have correct address to access all csv files.

Dataset Explanation

I designed this database according to some basic requirements of my own trading system.

Req1 Data for all Instruments	Minute and day stock data, including open, close, high, low, etc.
Req 2 Company Basic Info	Basic information about the company including market capitalization, the company's industry, etc.
Req 3 Significant events occurring in the company(CN Market)	Significant events occurring in the company include asset reorganization, external guarantees, share pledges, asset acquisitions, etc.
Req 4 Company Earnings Data(US Market)	Earnings data released by the company each quarter, including PB, PE, etc.

Database ERD:



For each table, we provide one notebook to finish the ETL process(code/Database/*.ipynb). All the data is from this API: akshare(https://akshare.akfamily.xyz/)

Results of running the code with data

Seq1: select 10 stocks from Shanghai Stock Exchange(SSE)

PostgreSQL:

```sql
-- select any stock from Shanghai Stock Exchange(SSE)
SELECT ds.stock_id, ds.total_value from "Trading".Dim_Stock ds
where stock_id>600000
order by stock_id
limit 10;
```

Data Output | Messages | Explain | Notifications

	stock_id [PK] integer	total_value numeric (20,2)
1	600004	25820896467.53
2	600006	12820000000.00
3	600007	21102569087.30
4	600008	20113218454.98
5	600009	91352149991.40
6	600010	71739808751.84
7	600011	116165890856.60
8	600012	17150027400.00
9	600015	90715092267.60
10	600016	167686662862.66

Spark:

```
dim_stock_df = spark.read.csv('Dim_Stock.csv', header=True, inferSchema=True)
sse_stocks_df = dim_stock_df.filter(col("stock_id") > 600000).orderBy("stock_id").limit(10)
sse_stocks_df.select('Stock_ID', 'Total_Value').show()
```

Select Cell Language Mode

4] ✓ 0.3s Python

```
+--------+--------------------+
|Stock_ID|         Total_Value|
+--------+--------------------+
|  600004|   2.582089646753E10|
|  600006|            1.282E10|
|  600007|    2.11025690873E10|
|  600008|2.011321845498000...|
|  600009|9.135214999140001E10|
|  600010|   7.173980875184E10|
|  600011|   1.161658908566E11|
|  600012|       1.71500274E10|
|  600015|    9.07150922676E10|
|  600016|  1.6768666286266E11|
+--------+--------------------+
```

Seq2: select all stocks that rise over 9.9% on 2024-03-11

PostgreSQL:

```sql
--select all stocks that rise over 9.9% on 2024-03-11
SELECT * from "Trading".Fact_Stock_Daily
where Date_ID = 20240311 and percent_of_incre_decre>9.9
order by percent_of_incre_decre desc;

--select infomation from 20231120 9:30-9:40 on stock SH600004
```

Data Output | Messages | Explain | Notifications

	fsd_id [PK] bigint	stock_id integer	date_id integer	open numeric (10,2)	high numeric (10,2)	low numeric (10,2)	volume integer	turn
1	20240311300317	300317	20240311	4.04	4.84	4.04	972902	
2	20240311300324	300324	20240311	2.98	3.29	2.88	1842843	
3	20240311300530	300530	20240311	17.80	21.40	17.80	94640	
4	20240311300438	300438	20240311	22.01	25.91	22.01	425121	
5	20240311301202	301202	20240311	28.70	34.42	27.66	136343	
6	20240311301292	301292	20240311	14.16	16.85	14.16	50492	
7	20240311300890	300890	20240311	23.37	27.95	23.37	76936	
8	20240311301349	301349	20240311	28.89	34.66	28.89	30457	
9	20240311300712	300712	20240311	21.48	25.14	21.37	92134	
10	20240311688063	688063	20240311	85.00	99.60	85.00	136891	
11	20240311301205	301205	20240311	91.00	116.62	91.00	133827	

Spark:

```python
fact_stock_daily_df = spark.read.csv('Fact_Stock_Daily_2024.csv', header=True, inferSchema=True)
rising_stocks_df = fact_stock_daily_df.filter((col("Date_ID") == 20240311) & (col("percent_of_incre_decre") > 9.9))\
                   .orderBy(col("percent_of_incre_decre").desc())
rising_stocks_df.show()
```

0.9s Python

```
+--------------+--------+--------+-----+-----+-----+-------+--------------+---------+--------------------+--------------------+
|        FSD_ID|Stock_ID| Date_ID| Open| High|  Low| Volume|      Turnover|Amplitude|Percent_Of_Incre_Decre|Amount_Of_Incre_Decre|
+--------------+--------+--------+-----+-----+-----+-------+--------------+---------+--------------------+--------------------+
|20240311300317|  300317|20240311| 4.04| 4.84| 4.04| 972902|  4.60648926E8|    19.85|                20.1|                0.81|
|20240311300324|  300324|20240311| 2.98| 3.29| 2.88|1842843|  5.68963029E8|    14.96|               20.07|                0.59|
|20240311300530|  300530|20240311| 17.8| 21.4| 17.8|  94640|  1.91813994E8|    20.19|               20.02|                3.57|
|20240311301292|  301292|20240311|14.16|16.85|14.16|  50492|   8.4292834E7|    19.16|               20.01|                2.81|
|20240311301202|  301202|20240311| 28.7|34.42|27.66| 136343|  4.30519547E8|    23.57|               20.01|                5.74|
|20240311300890|  300890|20240311|23.37|27.95|23.37|  76936|  2.03484159E8|    19.67|               20.01|                4.66|
|20240311301349|  301349|20240311|28.89|34.66|28.89|  30457|  1.01086693E8|    19.98|               20.01|                5.78|
|20240311300438|  300438|20240311|22.01|25.91|22.01| 425121| 1.021499595E9|    18.06|               20.01|                4.32|
|20240311688063|  688063|20240311| 85.0| 99.6| 85.0| 136891|  1.26584976E9|    17.59|                20.0|                16.6|
|20240311300712|  300712|20240311|21.48|25.14|21.37|  92134|  2.19739943E8|     18.0|                20.0|                4.19|
|20240311301205|  301205|20240311| 91.0|116.62| 91.0| 133827| 1.369576675E9|    26.36|               17.31|                16.8|
|20240311300025|  300025|20240311| 8.54| 10.08| 8.38| 306597|  2.85149119E8|    20.09|               17.14|                1.49|
|20240311300769|  300769|20240311|41.01|46.89|41.01| 302334| 1.342825861E9|    14.71|               16.59|                6.63|
|20240311300953|  300953|20240311| 45.8|  53.0| 45.8|  31804|  1.59406988E8|    15.72|               15.31|                7.01|
|20240311688248|  688248|20240311|27.78|31.31|27.78| 171501|  5.19394678E8|    13.07|               15.07|                4.07|
|20240311300750|  300750|20240311|165.0|181.65|164.1| 763350|1.3248182793E10|    11.11|               14.46|               22.85|
|20240311301236|  301236|20240311|46.95| 55.88| 46.8|1141301|  5.880365894E9|    18.92|                14.4|                6.91|
|20240311300709|  300709|20240311| 38.8| 44.98|36.89| 446070|  1.808201963E9|     21.3|               14.01|                5.32|
|20240311300014|  300014|20240311| 38.6|42.76|38.55| 648947|  2.651262664E9|    11.19|                13.0|                4.89|
|20240311300411|  300411|20240311| 8.12|  9.49| 8.12| 962847|  8.43633186E8|    17.19|               12.92|                1.03|
+--------------+--------+--------+-----+-----+-----+-------+--------------+---------+--------------------+--------------------+
only showing top 20 rows
```

Seq3: select information from 20231120 9:30-9:40 on stock SH600004

PostgreSQL:

```
19  --select infomation from 20231120 9:30-9:40 on stock SH600004
20  SELECT * from "Trading".Fact_Stock_Minute
21  where Minute_ID>202311200930 and Minute_ID<202311200940 and stock_id = 600004;
22
23  --show total minutes in 2024-03-11
```

Data Output | Messages | Explain | Notifications

	fsm_id [PK] bigint	stock_id integer	minute_id bigint	open numeric (10,2)	high numeric (10,2)	low numeric (10,2)	volume integer
1	600004202311200931	600004	202311200931	0.00	10.92	10.89	647
2	600004202311200932	600004	202311200932	0.00	10.90	10.89	401
3	600004202311200933	600004	202311200933	0.00	10.91	10.86	2568
4	600004202311200934	600004	202311200934	0.00	10.91	10.90	310
5	600004202311200935	600004	202311200935	0.00	10.95	10.91	1727
6	600004202311200936	600004	202311200936	0.00	10.94	10.92	202
7	600004202311200937	600004	202311200937	0.00	10.94	10.93	1230
8	600004202311200938	600004	202311200938	0.00	10.93	10.93	211
9	600004202311200939	600004	202311200939	0.00	10.94	10.93	86

Time Cost for PostgreSQL:

#	Node	Timings	
		Exclusive	Inclusive
1.	→ Bitmap Heap Scan on Trading.fact_stock… **Recheck Cond**: ((fact_stock_minute.stock_id = **Heap Blocks**: exact=2	0.489 ms	7.586 ms
2.	→ Bitmap AND (cost=644.38..644.38 r…	0.004 ms	7.098 ms
3.	→ Bitmap Index Scan using idx_fs… **Index Cond**: (fact_stock_minute.stoc	2.269 ms	2.269 ms
4.	→ Bitmap Index Scan using idx_fs… **Index Cond**: ((fact_stock_minute.min	4.825 ms	4.825 ms

Spark:

```
import time
start_time=  time.time()
fact_stock_minute_df = spark.read.csv('Fact_Stock_Minute.csv', header=True, inferSchema=True)
stock_600004_info_df = fact_stock_minute_df.filter((col("Minute_ID") > 202311200930) \
                                                   & (col("Minute_ID") < 202311200940) \
                                                    & (col("Stock_ID") == 600004))
stock_600004_info_df.show()
print(f"Time used: {time.time() - start_time}")
```

✓ 4.8s Python

```
[Stage 29:>                                                         (0 + 3) / 3]
+-----------------+--------+-------------+----+-----+-----+------+----------+--------+-----+
|           FSM_ID|Stock_ID|   Minute_ID|Open| High|  Low|Volume|Stock_Name| Turnover|Close|
+-----------------+--------+-------------+----+-----+-----+------+----------+--------+-----+
|600004202311200931|  600004|202311200931| 0.0|10.92|10.89|   647|   ▯▯▯2▯▯▯| 705561.0|10.91|
|600004202311200932|  600004|202311200932| 0.0| 10.9|10.89|   401|   ▯▯▯2▯▯▯| 436909.0|10.89|
|600004202311200933|  600004|202311200933| 0.0|10.91|10.86|  2568|   ▯▯▯2▯▯▯|2794153.0| 10.9|
|600004202311200934|  600004|202311200934| 0.0|10.91| 10.9|   310|   ▯▯▯2▯▯▯| 338083.0| 10.9|
|600004202311200935|  600004|202311200935| 0.0|10.95|10.91|  1727|   ▯▯▯2▯▯▯|1886825.0|10.94|
|600004202311200936|  600004|202311200936| 0.0|10.94|10.92|   202|   ▯▯▯2▯▯▯| 220870.0|10.93|
|600004202311200937|  600004|202311200937| 0.0|10.94|10.93|  1230|   ▯▯▯2▯▯▯|1345150.0|10.93|
|600004202311200938|  600004|202311200938| 0.0|10.93|10.93|   211|   ▯▯▯2▯▯▯| 230657.0|10.93|
|600004202311200939|  600004|202311200939| 0.0|10.94|10.93|    86|   ▯▯▯2▯▯▯|  94037.0|10.94|
+-----------------+--------+-------------+----+-----+-----+------+----------+--------+-----+

Time used: 4.820951700210571
```

Seq4: show total minutes in 2024-03-11

PostgreSQL:

```
23  --show total minutes in 2024-03-11
24  SELECT count(Minute_ID) AS Total_Minutes
25  from "Trading".Dim_Minute
26  where DATE_TRUNC('day',CAST(Date AS timestamp))='2024-03-11';
27
```

Data Output | Messages | Explain ✕ | Notifications

	total_minutes bigint
1	241

Spark:

```
dim_minute_df = spark.read.csv('Dim_Minute_2024.csv', header=True, inferSchema=True)
total_minutes_df = dim_minute_df.filter(col("Date").cast("date") == "2024-03-11")\
                                .agg({"Minute_ID": "count"})\
                                .withColumnRenamed("count(Minute_ID)", "Total_Minutes")
total_minutes_df.show()
```

✓ 0.9s Python

```
+-------------+
|Total_Minutes|
+-------------+
|          241|
+-------------+
```

Seq5: show total trading days in October

PostgreSQL:

```
28  --show total trading days in Feburary 2024
29  SELECT count(Date_ID) AS Total_Trading_Days
30  from "Trading".Dim_Date
31  where year = 2024 and month = 2;
32
```

Data Output | Messages | Explain ✕ | Notifications

	total_trading_days bigint
1	16

Spark:

```
dim_date_df = spark.read.csv('Dim_Date_2024.csv', header=True, inferSchema=True)
total_trading_days_df = dim_date_df.filter((col("year") == 2024) & (col("month") == 2))\
    .agg({"Date_ID": "count"})\
    .withColumnRenamed("count(Date_ID)", "Total_Trading_Days")
total_trading_days_df.show()
```

✓ 0.4s Python

```
+------------------+
|Total_Trading_Days|
+------------------+
|                16|
+------------------+
```

Seq6: recent company events from September

PostgreSQL:

```
--show recent company events from September
SELECT de.Stock_ID, de.Event_Date, de.Event_Type, de.Current_Flag,
       de.Effective_Timestamp, de.Expire_Timestamp
FROM "Trading".Fact_Company_Event fce
JOIN "Trading".Dim_Event de
ON fce.Event_Dim_ID = de.Event_Dim_ID
where fce.Date_ID>20230901 and fce.Stock_ID>600000 and fce.Stock_ID<600010;

--select Minute Level Data from 2024-03-11 9:30 to 2024-03-11 9:40
```

Data Output | Messages | Explain | Notifications

stock_id integer	event_date date	event_type character varying (50)	current_flag boolean	effective_timestamp timestamp without time zone	expire_timestamp timestamp without
600006	2023-09-15	Asset Restructuring	true	2023-09-15 00:00:00	[null]
600008	2023-11-14	Asset Restructuring	true	2023-11-14 00:00:00	[null]

Spark:

```
fact_company_event_df = spark.read.csv('Fact_Company_Event.csv', header=True, inferSchema=True).withColumnRenamed("Stock_ID","FCE
dim_event_df = spark.read.csv('Dim_Event.csv', header=True, inferSchema=True).withColumnRenamed("Stock_ID","DE_Stock_ID")

recent_events_df = fact_company_event_df.join(dim_event_df, fact_company_event_df.Event_Dim_ID == dim_event_df.Event_Dim_ID, "inn
                   .select("FCE_ID","FCE_Stock_ID","Date_ID","Event_Type","Current_Flag","Effective_Timestamp","Expire_Timestamp
                   .filter((col("Date_ID") > 20230901) & (col("FCE_Stock_ID") > 600000) & (col("FCE_Stock_ID") < 600010))

recent_events_df.show()
```

✓ 0.4s Python

```
+------------------+------------+--------+------------------+------------+-------------------+----------------+
|            FCE_ID|FCE_Stock_ID| Date_ID|        Event_Type|Current_Flag|Effective_Timestamp|Expire_Timestamp|
+------------------+------------+--------+------------------+------------+-------------------+----------------+
|20230915008260006|      600006|20230915|Asset Restructuring|           1|         2023-09-15|            NULL|
|20231114006360008|      600008|20231114|Asset Restructuring|           1|         2023-11-14|            NULL|
+------------------+------------+--------+------------------+------------+-------------------+----------------+
```

Seq7 select Minute Level Data from 2023-11-21 9:30 to 2023-11-21 9:40 where the company is in CN market and its Total_Value is in top 10

PostgreSQL:

```sql
SELECT fsm.FSM_ID, fsm.Stock_ID, fsm.Minute_ID, fsm.Close, fsm.High, fsm.Low, fsm.volu
FROM "Trading".Fact_Stock_Minute fsm
WHERE fsm.Stock_ID in (
    SELECT Stock_ID
    FROM "Trading".Dim_Stock
    WHERE Stock_Market = 'CN'
    ORDER BY Total_Value DESC
    LIMIT 10
)
AND fsm.Minute_ID >= 202311210930 AND fsm.Minute_ID <= 202311210940;
```

Data Output | Messages | Explain | Notifications

	fsm_id [PK] bigint	stock_id integer	minute_id bigint	close numeric (10,2)	high numeric (10,2)	low numeric (10,2)	volume integer
1	600938202311210930	600938	202311210930	19.12	19.12	19.12	
2	600938202311210931	600938	202311210931	19.06	19.14	19.05	4
3	600938202311210932	600938	202311210932	19.05	19.09	19.04	3
4	600938202311210933	600938	202311210933	19.01	19.05	19.00	3
5	600938202311210934	600938	202311210934	19.04	19.04	18.99	3
6	600938202311210935	600938	202311210935	19.05	19.05	19.04	3
7	600938202311210936	600938	202311210936	19.01	19.04	19.01	2
8	600938202311210937	600938	202311210937	19.02	19.02	19.01	2
9	600938202311210938	600938	202311210938	19.02	19.03	19.01	1
10	600938202311210939	600938	202311210939	19.02	19.03	19.01	1

Time Cost: 119.978ms

#	Node	Timings		Rows		
		Exclusive	Inclusive	Rows X	Actual	Plan
1.	→ Nested Loop Inner Join (cost=1371.75...	95.093 ms	119.978 ms	↓ 1.05	110	105
2.	→ Aggregate (cost=581.38..581.48 r... Buckets: Batches: Memory Usage: 24 k	0.157 ms	14.365 ms	↑ 1	10	10
3.	→ Limit (cost=581.23..581.25 r...	0.001 ms	14.209 ms	↑ 1	10	10
4.	→ Sort (cost=581.23..594....	1.151 ms	14.208 ms	↑ 530.9	10	5309
5.	→ Seq Scan on Tradin... Filter: ((dim_stock.stoc Rows Removed by Filte	13.057 ms	13.057 ms	↑ 1	5309	5309

Spark:

```python
start_time=  time.time()
top_10_cn_stocks_cached = dim_stock_df.filter(col("Stock_Market") == "CN") \
    .orderBy(col("Total_Value").desc()) \
    .limit(10) \
    .select("Stock_ID") \
    .withColumnRenamed("Stock_ID","top10_Stock_ID")\
    .cache()


minute_level_data_df = fact_stock_minute_df.join(top_10_cn_stocks_cached, \
                        fact_stock_minute_df.Stock_ID == top_10_cn_stocks_cached.top10_Stock_ID, "inner") \
    .filter((col("Minute_ID") >= 202311210930) & (col("Minute_ID") <= 202311210940)) \
    .select("FSM_ID", "Stock_ID", "Minute_ID", "Close", "High", "Low", "volume")

minute_level_data_df.show()

top_10_cn_stocks_cached.unpersist()
print(f"Time used: {time.time() - start_time}")
```

✓ 0.6s Python

```
+---------------+--------+------------+-----+-----+-----+------+
|            FSM_ID|Stock_ID|   Minute_ID|Close|High|  Low|volume|
+---------------+--------+------------+-----+-----+-----+------+
|601288202311210930|  601288|202311210930| 3.66|3.66| 3.66| 11861|
|601288202311210931|  601288|202311210931| 3.66|3.67| 3.65|155799|
|601288202311210932|  601288|202311210932| 3.66|3.67| 3.65|116492|
|601288202311210933|  601288|202311210933| 3.67|3.67| 3.66| 11392|
|601288202311210934|  601288|202311210934| 3.66|3.67| 3.66| 14070|
|601288202311210935|  601288|202311210935| 3.67|3.67| 3.66| 28420|
|601288202311210936|  601288|202311210936| 3.66|3.67| 3.66| 14610|
|601288202311210937|  601288|202311210937| 3.67|3.67| 3.66|  9740|
|601288202311210938|  601288|202311210938| 3.66|3.67| 3.66|  9401|
|601288202311210939|  601288|202311210939| 3.66|3.67| 3.66| 10829|
|601288202311210940|  601288|202311210940| 3.67|3.67| 3.66| 38237|
|601857202311210930|  601857|202311210930| 7.14|7.14| 7.14|  4878|
|601857202311210931|  601857|202311210931| 7.15|7.15| 7.12| 23906|
|601857202311210932|  601857|202311210932| 7.12|7.14| 7.12|  9806|
|601857202311210933|  601857|202311210933| 7.12|7.13| 7.11|  4412|
|601857202311210934|  601857|202311210934| 7.14|7.14| 7.12|  5834|
|601857202311210935|  601857|202311210935| 7.15|7.15| 7.13| 14307|
|601857202311210936|  601857|202311210936| 7.14|7.15| 7.14|  7101|
|601857202311210937|  601857|202311210937| 7.14|7.15| 7.14|  2217|
|601857202311210938|  601857|202311210938| 7.15|7.15| 7.14|  4517|
+---------------+--------+------------+-----+-----+-----+------+
only showing top 20 rows

Time used: 0.6944880485534668
```

Time Cost: 0.69ms