High Frequency Finance Coursework I

K23023533 Jhao-Wei Chen

Part I: Compare SQL and NoSQL

To compare SQL and NoSQL databases, it is essential to understand what these terms represent.

SQL databases, which interact through SQL, store data in tables and use keys for data recognition and connection. These databases are typically ordered, structured and relational. It is worth noting that SQL databases have three vital characteristics: standardization, ease of use, and stability. Firstly, standardization refers to the consistent use of the same standardized language, SQL across different roles and different SQL databases. As for ease of use, it means that SQL is a simple language which does not have complex structures and can be easily learned. Lastly, stability indicates that it follows the ACID principle (Atomicity, Consistency, Isolation, and Durability). For instance, Oracle, MySQL, and Microsoft SQL Server are all well-known SQL databases.

On the other hand, NoSQL databases store data in formats like JSON, rather than in tables, providing multiple types of data storage including key-value pairs, documents, and graphs. Compared with SQL databases, NoSQL databases have four advantages: flexibility, scalability, high performance, and high functionality. First and foremost, flexibility refers to the ability to accommodate many data types and models, which makes it more elastic. Secondly, scalability indicates that users can horizontally expand databases. Regarding high performance, this advantage represents improved efficiency in optimizing specific data models. Last but not least, high functionality denotes that various APIs and data for data models are available in NoSQL databases.

Considering the characteristics of SQL and NoSQL databases mentioned above, it can be concluded that there are three main differences between them. First, SQL databases are structured and follow a regulated schema, enabling users to search data by identifying relationships within the databases. In contrast, this approach is not applicable in NoSQL databases, which do not support searching data as trading factors in the same way. However, NoSQL databases can store a wider variety of data types, including those not encountered before. Second, scalability marks a significant difference between them. NoSQL databases, which support horizontal scalability, can manage high volumes and velocities of data by adding more servers. On the other hand, SQL databases usually increase capacity through significant investments in hardware upgrades. Third, the query language constitutes another key difference. SQL databases use SQL as their query language, making it easy and intuitive for users to search data. Nevertheless, NoSQL databases employ various query languages across different data models, which can confuse traders and slow down decision-making processes.

In conclusion, based on the differences in structures, scalability, and query languages, SQL databases are structured and intuitive, while NoSQL databases are flexible and scalable.

Comparative Table:

Aspect	SQL Databases	NoSQL Databases
Data Storage	Store data in tables	Store data in formats like JSON
Schema	Structured and regulated	elastic, supporting various data types
Scalability	vertical, hard to handle high-frequency finance data	horizontal, easier to handle high-frequency finance data
Query Language	only SQL, easy and intuitive	various query languages, hard and complicated

Part II: Create a database for high-frequency data

1. Data Analysis and Pre-processing

(1) OrderDetail

In [46]:
```
import pandas as pd
import pandas as pd
import sqlite3
from sqlalchemy import create_engine

# Convert allGlaxoOrderDetail into a DataFrame
column_names = ['OrderCode', 'MarketSegmentCode', 'MarketSectorCode', 'TICode',
                'CountryOfRegister', 'CurrencyCode', 'ParticipantCode', 'BuySellInd
                'MarketMechanismGroup', 'MarketMechanismType', 'Price', 'AggregateS
                'SingleFillInd', 'BroadcastUpdateAction', 'Date', 'Time',
                'MessageSequenceNumber']

Detail = pd.read_csv('allGlaxoOrderDetail.csv', header=None, names=column_names)

# Show allGlaxoOrderDetail
Detail.iloc[:10, :8]
```

Out[46]:

	OrderCode	MarketSegmentCode	MarketSectorCode	TICode	CountryOfRegister	Curre
0	709ENVUN07	SET1	FE10	GB0009252882	GB	
1	208ATNHG07	SET1	FE10	GB0009252882	GB	
2	006D95WX07	SET1	FE10	GB0009252882	GB	
3	006D94UH07	SET1	FE10	GB0009252882	GB	
4	709FJNIR07	SET1	FE10	GB0009252882	GB	
5	709EPA1T07	SET1	FE10	GB0009252882	GB	
6	709EODFR07	SET1	FE10	GB0009252882	GB	
7	609JJXF807	SET1	FE10	GB0009252882	GB	
8	5099MB5A07	SET1	FE10	GB0009252882	GB	
9	308PVR0Q07	SET1	FE10	GB0009252882	GB	

(2) OrderHistory

In [47]:

```
# Convert allGlaxoOrderHistory into a DataFrame
column_names = ['OrderCode', 'OrderActionType', 'MatchingOrderCode', 'TradeSize',
                'TradeCode', 'TICode', 'CountryOfRegister', 'CurrencyCode',
                'MarketSegmentCode', 'AggregateSize', 'BuySellInd',
                'MarketMechanismType', 'MessageSequenceNumber', 'Date', 'Time']

History = pd.read_csv('allGlaxoOrderHistory.csv', header=None, names=column_names)

# Show allGlaxoOrderHistory
History.iloc[:10, :8]
```

Out[47]:

	OrderCode	OrderActionType	MatchingOrderCode	TradeSize	TradeCode	TICode	C
0	208VSG5Q07	M	709JKPU707	500	709JKPUL07	GB0009252882	
1	609NJZ0107	M	308XOPF507	243	308XOPF807	GB0009252882	
2	308WLUQQ07	M	609MGG5Y07	1680	609MGG5Z07	GB0009252882	
3	208SG8CP07	M	509ABGBD07	3288	509ABGBI07	GB0009252882	
4	609MYSWA07	M	208V4A1707	12148	208V4A1807	GB0009252882	
5	609R1IHO07	M	408LULF907	392	408LULFE07	GB0009252882	
6	006QWO5E07	D	NaN	0	NaN	GB0009252882	
7	3095JP3907	D	NaN	0	NaN	GB0009252882	
8	509JJ8KI07	D	NaN	0	NaN	GB0009252882	
9	20938C5X07	D	NaN	0	NaN	GB0009252882	

(3) TradeReport

In [48]:

```
# Convert allGlaxoTradeReport into a DataFrame
column_names = ['MessageSequenceNumber', 'TICode', 'MarketSegmentCode',
                'CountryOfRegister', 'CurrencyCode',
                'TradeCode', 'TradePrice', 'TradeSize', 'TradeDate',
```

```
                'TradeTime', 'BroadcastUpdateAction', 'TradeTypeInd',
                'TradeTimeInd', 'BargainConditions', 'ConvertedPriceInd',
                'PublicationDate', 'PublicationTime']

TradeReport = pd.read_csv('allGlaxoTradeReport.csv', header=None, names=column_name

# Show allGlaxoTradeReport
TradeReport.iloc[:10, :8]
```

Out[48]:

	MessageSequenceNumber	TICode	MarketSegmentCode	CountryOfRegister	CurrencyCod
0	1114866	GB0009252882	SET1	GB	GB
1	438192	GB0009252882	SET1	GB	GB
2	1285577	GB0009252882	SET1	GB	GB
3	736925	GB0009252882	SET1	GB	GB
4	1137035	GB0009252882	SET1	GB	GB
5	900321	GB0009252882	SET1	GB	GB
6	601775	GB0009252882	SET1	GB	GB
7	355811	GB0009252882	SET1	GB	GB
8	664624	GB0009252882	SET1	GB	GB
9	1191567	GB0009252882	SET1	GB	GB

2. Create a SQLite Database and Add Tables

In [49]:
```
# Connect to the SQLite database (create one if it does not exist)
conn = sqlite3.connect('high_frequency_finance.db')

# Create a cursor object using the cursor method
cursor = conn.cursor()
```

(1) OrderDetail

In [50]:
```
# Create the 'Detail' table
create_detail_query = """
CREATE TABLE IF NOT EXISTS Detail (
    OrderCode CHAR(10) NOT NULL,
    MarketSegmentCode CHAR(4) NOT NULL,
    MarketSectorCode CHAR(4) NOT NULL,
    TICode CHAR(12) NOT NULL,
    CountryOfRegister CHAR(2) NOT NULL,
    CurrencyCode CHAR(3) NOT NULL,
    ParticipantCode CHAR(11),
    BuySellInd CHAR(1) NOT NULL,
    MarketMechanismGroup CHAR(1) NOT NULL,
    MarketMechanismType CHAR(2) NOT NULL,
    Price DECIMAL(18,8) NOT NULL,
    AggregateSize DECIMAL(12) NOT NULL,
    SingleFillInd CHAR(1) NOT NULL,
    BroadcastUpdateAction CHAR(1) NOT NULL,
    Date TEXT NOT NULL,
    Time TEXT NOT NULL,
    MessageSequenceNumber INTEGER(10) NOT NULL,
    PRIMARY KEY (OrderCode)
```

```
);
"""
# Execute the create table query
cursor.execute(create_detail_query)
```

Out[50]:
```
<sqlite3.Cursor at 0x215c3283140>
```

(2) OrderHistory

In [51]:
```
# Create the 'Hisory' table
create_Hisory_query = """
CREATE TABLE IF NOT EXISTS History (
    OrderCode CHAR(10) NOT NULL,
    OrderActionType CHAR(1) NOT NULL,
    MatchingOrderCode CHAR(10),
    TradeSize DECIMAL(8),
    TradeCode CHAR(10),
    TICode CHAR(12) NOT NULL,
    CountryOfRegister CHAR(2) NOT NULL,
    CurrencyCode CHAR(3) NOT NULL,
    MarketSegmentCode CHAR(4) NOT NULL,
    AggregateSize DECIMAL(12) NOT NULL,
    BuySellInd CHAR(1) NOT NULL,
    MarketMechanismType CHAR(2) NOT NULL,
    MessageSequenceNumber INTEGER(10) NOT NULL,
    Date TEXT NOT NULL,
    Time TEXT NOT NULL,
    PRIMARY KEY (OrderCode, MatchingOrderCode)
);
"""
# Execute the create table query
cursor.execute(create_Hisory_query)
```

Out[51]:
```
<sqlite3.Cursor at 0x215c3283140>
```

(3) TradeReport

In [52]:
```
# Create the 'TradeReport' table
create_tradereport_query = """
CREATE TABLE IF NOT EXISTS TradeReport (
    MessageSequenceNumber INTEGER(10) NOT NULL,
    TICode CHAR(12) NOT NULL,
    MarketSegmentCode CHAR(4) NOT NULL,
    CountryOfRegister CHAR(2) NOT NULL,
    CurrencyCode CHAR(3) NOT NULL,
    TradeCode CHAR(10) NOT NULL,
    TradePrice DECIMAL(18,8) NOT NULL,
    TradeSize DECIMAL(12) NOT NULL,
    TradeDate TEXT NOT NULL,
    TradeTime TEXT NOT NULL,
    BroadcastUpdateAction CHAR(1) NOT NULL,
    TradeTypeInd CHAR(2) NOT NULL,
    TradeTimeInd CHAR(1) NOT NULL,
    BargainConditions CHAR(1) NOT NULL,
    ConvertedPriceInd CHAR(1) NOT NULL,
    PublicationDate TEXT NOT NULL,
    PublicationTime TEXT NOT NULL,
    PRIMARY KEY (TradeCode,MessageSequenceNumber)
);
"""
# Execute the create table query
cursor.execute(create_tradereport_query)
```

Out[52]:
```
<sqlite3.Cursor at 0x215c3283140>
```

In [53]:
```
# Commit the changes and close the connection
conn.commit()
conn.close()
```

3. Convert DataFrames into Tables in the SQLite Database

(1) OrderDetail

In [54]:
```
# Put DataFrames into tables
# Create a SQLite database connection
engine = create_engine('sqlite:///high_frequency_finance.db')

# Insert the DataFrame into the 'Detail' table
Detail.to_sql('Detail', con=engine, if_exists='append', index=False)
```

Out[54]:
```
274322
```

	OrderCode	MarketSegmentCode	MarketSectorCode	TICode	CountryOfRegister	CurrencyCode	ParticipantCode	BuySellInd	MarketMechanismGroup	MarketMechanismType	Price	AggregateSize	SingleFillInd	Broad
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	709ENVUN07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1510	173	N	F
2	208ATNHG07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1550	1800	N	F
3	006D95WX07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1485	700	N	F
4	006D94UH07	SET1	FE10	GB0009252882	GB	GBX	NULL	B	O	LO	1300	230	N	F
5	709FJNIR07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1449	1100	N	F
6	709EPA1T07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1438	20000	N	F
7	709EODFR07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1464	1100	N	F
8	609JJXF807	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1448	38600	N	F
9	5099MB5A07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1438	12867	N	F
10	308PVR0Q07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1570	55	N	F
11	408DJNTU07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1550	1100	N	F
12	208RQR0707	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1473	1000	N	F
13	006D93VE07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1700	1011	N	F
14	006D9R8R07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1436	275000	N	F
15	709FFKRL07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1472	8196	N	F
16	006E2XR807	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1441	134	N	F
17	709C5XFD07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1650	30	N	F
18	709FJQ6K07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1437	20000	N	F
19	609JAOH307	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1480	5100	N	F
20	208QX6TK07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1650	168	N	F
21	208P7EHX07	SET1	FE10	GB0009252882	GB	GBX	NULL	B	O	LO	1401	332	N	F
22	709FJQ9F07	SET1	FE10	GB0009252882	GB	GBX	NULL	B	O	LO	1424	49540	N	F
23	308TQKCJ07	SET1	FE10	GB0009252882	GB	GBX	NULL	B	O	LO	1420	10000	N	F
24	208KD5SH07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1555	4000	N	F

(2) OrderHistory

In [55]:
```
# Insert the DataFrame into the 'History' table
History.to_sql('History', con=engine, if_exists='append', index=False)
```

Out[55]:
```
322208
```

	OrderCode	OrderActionType	MatchingOrderCode	TradeSize	TradeCode	TICode	CountryOfRegister	CurrencyCode	MarketSegmentCode	AggregateSize	BuySellInd	MarketMechanismType	MessageSequenceNumber
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	208VSG5Q07	M	709JKPU707	500	709JKPUL07	GB0009252882	GB	GBX	SET1	0	S	LO	364284
2	609NJZ0107	M	308XOPF507	243	308XOPF807	GB0009252882	GB	GBX	SET1	0	S	LO	221138
3	308WLUQQ07	M	609MGG5Y07	1680	609MGG5Z07	GB0009252882	GB	GBX	SET1	0	S	LO	661284
4	208SG8CP07	M	509ABGBD07	3288	509ABGBI07	GB0009252882	GB	GBX	SET1	0	S	LO	1114862
5	609MYSWA07	M	208V4A1707	12148	208V4A1807	GB0009252882	GB	GBX	SET1	0	B	LO	359978
6	609R1IHO07	M	408LULF907	392	408LULFE07	GB0009252882	GB	GBX	SET1	0	B	LO	286944
7	006QWO5E07	D	NULL	0	NULL	GB0009252882	GB	GBX	SET1	3656	B	LO	915280
8	3095JP3907	D	NULL	0	NULL	GB0009252882	GB	GBX	SET1	5000	B	LO	595012
9	509JJ8KI07	D	NULL	0	NULL	GB0009252882	GB	GBX	SET1	8419	S	LO	820174
10	20938C5X07	D	NULL	0	NULL	GB0009252882	GB	GBX	SET1	375	B	LO	57596
11	408QDOMV07	D	NULL	0	NULL	GB0009252882	GB	GBX	SET1	1100	B	LO	814708
12	408O19PG07	D	NULL	0	NULL	GB0009252882	GB	GBX	SET1	27300	S	LO	155498
13	709QQWQK07	D	NULL	0	NULL	GB0009252882	GB	GBX	SET1	4830	B	LO	943908
14	006Q67CR07	D	NULL	0	NULL	GB0009252882	GB	GBX	SET1	2073	B	LO	955727
15	609TI31U07	D	NULL	0	NULL	GB0009252882	GB	GBX	SET1	780	S	LO	737141
16	408PKZ9107	D	NULL	0	NULL	GB0009252882	GB	GBX	SET1	9750	B	LO	634596
17	609TEB7707	D	NULL	0	NULL	GB0009252882	GB	GBX	SET1	6000	B	LO	533415
18	006Q3VVB07	D	NULL	0	NULL	GB0009252882	GB	GBX	SET1	8267	B	LO	871242
19	2092ZZ2N07	D	NULL	0	NULL	GB0009252882	GB	GBX	SET1	3190	B	LO	988114
20	509KH67507	D	NULL	0	NULL	GB0009252882	GB	GBX	SET1	2247	S	LO	270040
21	006Q4LAA07	D	NULL	0	NULL	GB0009252882	GB	GBX	SET1	2244	S	LO	841680
22	709R30SF07	D	NULL	0	NULL	GB0009252882	GB	GBX	SET1	100	B	LO	171238
23	408OC2P707	D	NULL	0	NULL	GB0009252882	GB	GBX	SET1	5000	B	LO	664070
24	3094R1F507	D	NULL	0	NULL	GB0009252882	GB	GBX	SET1	3603	B	LO	392380

(3) TradeReport

In [56]:
```
# Insert the DataFrame into the 'TradeReport' table
TradeReport.to_sql('TradeReport', con=engine, if_exists='append', index=False)
```

Out[56]:
```
130136
```

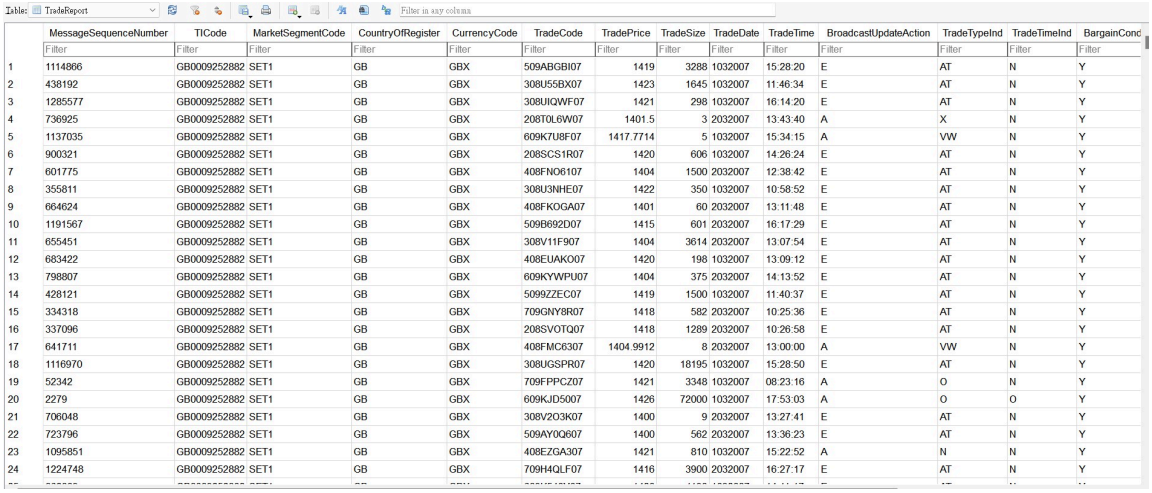


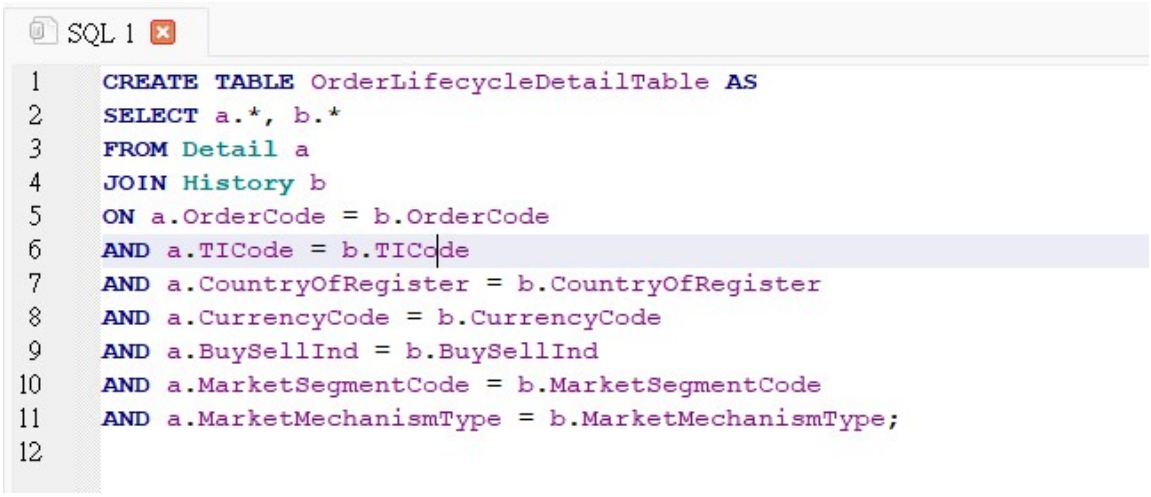
Table: TradeReport

	MessageSequenceNumber	TICode	MarketSegmentCode	CountryOfRegister	CurrencyCode	TradeCode	TradePrice	TradeSize	TradeDate	TradeTime	BroadcastUpdateAction	TradeTypeInd	TradeTimeInd	BargainCond
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1114866	GB0009252882	SET1	GB	GBX	509ABGBI07	1419	3288	1032007	15:28:20	E	AT	N	Y
2	438192	GB0009252882	SET1	GB	GBX	308U55BX07	1423	1645	1032007	11:46:34	E	AT	N	Y
3	1285577	GB0009252882	SET1	GB	GBX	308UIQWF07	1421	298	1032007	16:14:20	E	AT	N	Y
4	736925	GB0009252882	SET1	GB	GBX	208T0L6W07	1401.5	3	2032007	13:43:40	A	X	N	Y
5	1137035	GB0009252882	SET1	GB	GBX	609K7U8F07	1417.7714	5	1032007	15:34:15	A	VW	N	Y
6	900321	GB0009252882	SET1	GB	GBX	208SCS1R07	1420	606	1032007	14:26:24	E	AT	N	Y
7	601775	GB0009252882	SET1	GB	GBX	408FNO6107	1404	1500	2032007	12:38:42	E	AT	N	Y
8	355811	GB0009252882	SET1	GB	GBX	308U3NHE07	1422	350	1032007	10:58:52	E	AT	N	Y
9	664624	GB0009252882	SET1	GB	GBX	408FKOGA07	1401	60	2032007	13:11:48	E	AT	N	Y
10	1191567	GB0009252882	SET1	GB	GBX	509B692D07	1415	601	2032007	16:17:29	E	AT	N	Y
11	655451	GB0009252882	SET1	GB	GBX	308V11F907	1404	3614	2032007	13:07:54	E	AT	N	Y
12	683422	GB0009252882	SET1	GB	GBX	408EUAKO07	1420	198	1032007	13:09:12	E	AT	N	Y
13	798807	GB0009252882	SET1	GB	GBX	609KYWPU07	1404	375	2032007	14:13:52	E	AT	N	Y
14	428121	GB0009252882	SET1	GB	GBX	5099ZZEC07	1419	1500	1032007	11:40:37	E	AT	N	Y
15	334318	GB0009252882	SET1	GB	GBX	709GNY8R07	1418	582	2032007	10:25:36	E	AT	N	Y
16	337096	GB0009252882	SET1	GB	GBX	208SVOTQ07	1418	1289	2032007	10:26:58	E	AT	N	Y
17	641711	GB0009252882	SET1	GB	GBX	408FMC6307	1404.9912	8	2032007	13:00:00	A	VW	N	Y
18	1116970	GB0009252882	SET1	GB	GBX	308UGSPR07	1420	18195	1032007	15:28:50	E	AT	N	Y
19	52342	GB0009252882	SET1	GB	GBX	709FPPCZ07	1421	3348	1032007	08:23:16	A	O	N	Y
20	2279	GB0009252882	SET1	GB	GBX	609KJD5007	1426	72000	1032007	17:53:03	A	O	O	Y
21	706048	GB0009252882	SET1	GB	GBX	308V2O3K07	1400	9	2032007	13:27:41	E	AT	N	Y
22	723796	GB0009252882	SET1	GB	GBX	509AY0Q607	1400	562	2032007	13:36:23	E	AT	N	Y
23	1095851	GB0009252882	SET1	GB	GBX	408EZGA307	1421	810	1032007	15:22:52	A	N	N	Y
24	1224748	GB0009252882	SET1	GB	GBX	709H4QLF07	1416	3900	2032007	16:27:17	E	AT	N	Y

4. Merge Tables

(1) OrderLifecycleDetailTable:

Merge OrderDetail and OrderHistory as a new table



SQL 1

```
1  CREATE TABLE OrderLifecycleDetailTable AS
2  SELECT a.*, b.*
3  FROM Detail a
4  JOIN History b
5  ON a.OrderCode = b.OrderCode
6  AND a.TICode = b.TICode
7  AND a.CountryOfRegister = b.CountryOfRegister
8  AND a.CurrencyCode = b.CurrencyCode
9  AND a.BuySellInd = b.BuySellInd
10 AND a.MarketSegmentCode = b.MarketSegmentCode
11 AND a.MarketMechanismType = b.MarketMechanismType;
12
```

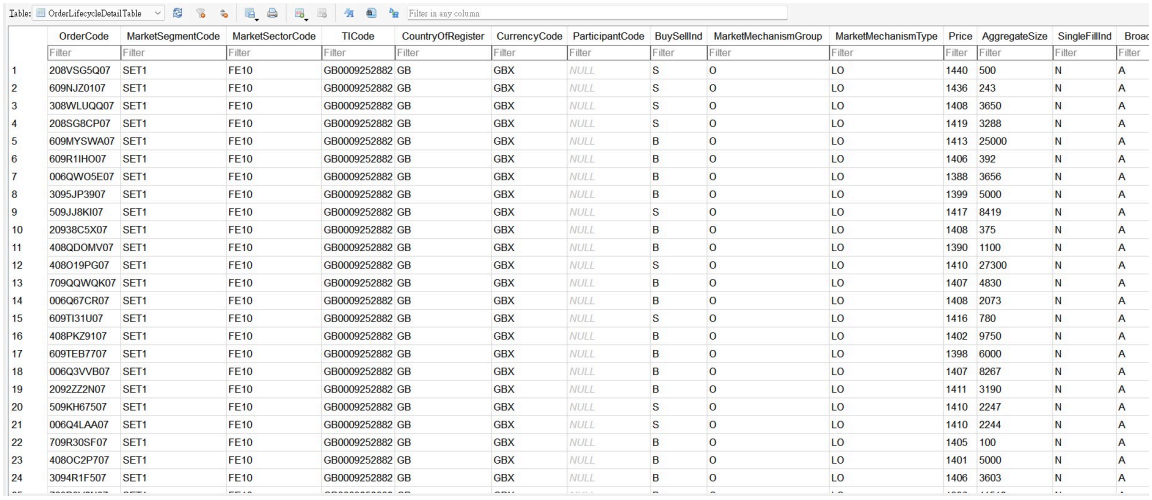


Table: OrderLifecycleDetailTable

	OrderCode	MarketSegmentCode	MarketSectorCode	TICode	CountryOfRegister	CurrencyCode	ParticipantCode	BuySellInd	MarketMechanismGroup	MarketMechanismType	Price	AggregateSize	SingleFillInd	Broac
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	208VSG5Q07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1440	500	N	A
2	609NJZ0107	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1436	243	N	A
3	308WLUQQ07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1408	3650	N	A
4	208SG8CP07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1419	3288	N	A
5	609MYSWA07	SET1	FE10	GB0009252882	GB	GBX	NULL	B	O	LO	1413	25000	N	A
6	609R1IHO07	SET1	FE10	GB0009252882	GB	GBX	NULL	B	O	LO	1406	392	N	A
7	006QWO5E07	SET1	FE10	GB0009252882	GB	GBX	NULL	B	O	LO	1388	3656	N	A
8	3095JP3907	SET1	FE10	GB0009252882	GB	GBX	NULL	B	O	LO	1399	5000	N	A
9	509JJ8KI07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1417	8419	N	A
10	20938C5X07	SET1	FE10	GB0009252882	GB	GBX	NULL	B	O	LO	1408	375	N	A
11	408QDOMV07	SET1	FE10	GB0009252882	GB	GBX	NULL	B	O	LO	1390	1100	N	A
12	408O19PG07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1410	27300	N	A
13	709QQWQK07	SET1	FE10	GB0009252882	GB	GBX	NULL	B	O	LO	1407	4830	N	A
14	006Q67CR07	SET1	FE10	GB0009252882	GB	GBX	NULL	B	O	LO	1408	2073	N	A
15	609TI31U07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1416	780	N	A
16	408PKZ9107	SET1	FE10	GB0009252882	GB	GBX	NULL	B	O	LO	1402	9750	N	A
17	609TEB7707	SET1	FE10	GB0009252882	GB	GBX	NULL	B	O	LO	1398	6000	N	A
18	006Q3VVB07	SET1	FE10	GB0009252882	GB	GBX	NULL	B	O	LO	1407	8267	N	A
19	2092ZZ2N07	SET1	FE10	GB0009252882	GB	GBX	NULL	B	O	LO	1411	3190	N	A
20	509KH67507	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1410	2247	N	A
21	006Q4LAA07	SET1	FE10	GB0009252882	GB	GBX	NULL	S	O	LO	1410	2244	N	A
22	709R30SF07	SET1	FE10	GB0009252882	GB	GBX	NULL	B	O	LO	1405	100	N	A
23	408OC2P707	SET1	FE10	GB0009252882	GB	GBX	NULL	B	O	LO	1401	5000	N	A
24	3094R1F507	SET1	FE10	GB0009252882	GB	GBX	NULL	B	O	LO	1406	3603	N	A

(2) TradeExecutionHistoryTable:

Merge OrderHistory and TradeReport as a new table

SQL 1

```sql
CREATE TABLE TradeExecutionHistoryTable AS
SELECT a.*, b.*
FROM History a
JOIN TradeReport b
ON a.TradeCode = b.TradeCode
AND a.TICode = b.TICode
AND a.CountryOfRegister = b.CountryOfRegister
AND a.CurrencyCode = b.CurrencyCode
AND a.TradeSize = b.TradeSize
AND a.MarketSegmentCode = b.MarketSegmentCode;
```

Table: TradeExecutionHistoryTable

	OrderCode	OrderActionType	MatchingOrderCode	TradeSize	TradeCode	TICode	CountryOfRegister	CurrencyCode	MarketSegmentCode	AggregateSize	BuySellInd	MarketMechanismType	MessageSequenceNumb
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	208VSG5Q07	M	709JKPU707	500	709JKPUL07	GB0009252882	GB	GBX	SET1	0	S	LO	3642
2	609NJZ0107	M	308XOPF507	243	308XOPF807	GB0009252882	GB	GBX	SET1	0	S	LO	2211
3	308WLUQQ07	M	609MGG5Y07	1680	609MGG5Z07	GB0009252882	GB	GBX	SET1	0	S	LO	6612
4	208SG8CP07	M	509ABGBD07	3288	509ABGBI07	GB0009252882	GB	GBX	SET1	0	S	LO	11148
5	609MYSWA07	M	208V4A1707	12148	208V4A1807	GB0009252882	GB	GBX	SET1	0	B	LO	3599
6	609R1IHO07	M	408LULF907	392	408LULFE07	GB0009252882	GB	GBX	SET1	0	B	LO	2869
7	2092OUV707	M	408PDTTT07	567	408PDTTU07	GB0009252882	GB	GBX	SET1	0	B	LO	3073
8	408ONYWA07	M	408OHHLK07	3056	408OHHLL07	GB0009252882	GB	GBX	SET1	0	B	LO	902
9	208Y1WP307	M	609PVK0Z07	524	609PVK1007	GB0009252882	GB	GBX	SET1	0	B	LO	8097
10	709TMPB907	M	006TAZVV07	812	006TAZW007	GB0009252882	GB	GBX	SET1	0	B	LO	3388
11	006SPAWD07	M	609X4QL807	100	609X4QLD07	GB0009252882	GB	GBX	SET1	0	B	LO	4892
12	209499V907	M	20949HFW07	2660	20949HGG07	GB0009252882	GB	GBX	SET1	0	S	LO	7366
13	408EP4MC07	M	5099YNQJ07	3000	5099YNQO07	GB0009252882	GB	GBX	SET1	0	S	LO	3619
14	006F0H1P07	M	509ABNET07	1544	509ABNEU07	GB0009252882	GB	GBX	SET1	0	S	LO	12625
15	006PPB2J07	M	408P35TS07	25000	408P4N3207	GB0009252882	GB	GBX	SET1	0	S	LO	73
16	006GQKTE07	M	308W2O7R07	7761	609L8XHV07	GB0009252882	GB	GBX	SET1	0	S	LO	12071
17	208WNE1I07	M	006JFEJU07	212	006JFEJV07	GB0009252882	GB	GBX	SET1	0	B	LO	7570
18	609JM0FT07	M	006EUBMT07	326	006EUBMU07	GB0009252882	GB	GBX	SET1	0	S	LO	9818
19	709H2HDM07	M	308V8NHL07	2774	308V8NHU07	GB0009252882	GB	GBX	SET1	0	S	LO	10776
20	609OJ3EK07	M	709KJI9X07	5500	103Z5P3M07	GB0009252882	GB	GBX	SET1	0	B	LO	9768
21	408LEU0M07	M	609QLT6I07	1500	609QLT6P07	GB0009252882	GB	GBX	SET1	0	S	LO	9729
22	408QR62807	M	408QR78307	9000	408QR78607	GB0009252882	GB	GBX	SET1	0	B	LO	2733
23	709HBHIZ07	M	709HPBX307	286	709HPBX407	GB0009252882	GB	GBX	SET1	0	S	LO	7977
24	609LSJTF07	M	709HSESL07	309	709HSESO07	GB0009252882	GB	GBX	SET1	0	B	LO	9292

Database Structure

Name	Type	Schema
Tables (6)		
Detail		CREATE TABLE Detail (OrderCode CHAR(10) NOT
History		CREATE TABLE History (OrderCode CHAR(10) NO
OrderLifecycleDetailTable		CREATE TABLE OrderLifecycleDetailTable(OrderCo
TradeExecutionHistoryTable		CREATE TABLE TradeExecutionHistoryTable(OrderC
TradeReport		CREATE TABLE TradeReport (MessageSequenceNum
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
Indices (0)		
Views (0)		
Triggers (0)		

Part III: Establish a connection to the database in Python

---

1. Execute a Query

In [57]:

```sql
# Connect to the SQLite database
conn = sqlite3.connect('high_frequency_finance.db')
```

```
# Create a cursor object using the cursor method
cursor = conn.cursor()

# Choose the order code you want to query
sql_query = "SELECT * FROM History WHERE OrderCode = '006E6OQQ07'"

# Execute the SQL query
cursor.execute(sql_query)

# Fetch all rows from the query result
rows = cursor.fetchall()

# Show the query
for row in rows:
    print(row)

# Close the connection
conn.close()
```

```
('006E6OQQ07', 'M', '006ECHAX07', 4910, '006ECHAY07', 'GB0009252882', 'GB', 'GBX',
'SET1', 0, 'B', 'LO', 12320, '1032007', '08:02:42')
('006E6OQQ07', 'P', '308TX55U07', 8309, '308TX55V07', 'GB0009252882', 'GB', 'GBX',
'SET1', 8257, 'B', 'LO', 12295, '1032007', '08:02:41')
('006E6OQQ07', 'P', '609JQE9H07', 3347, '609JQE9I07', 'GB0009252882', 'GB', 'GBX',
'SET1', 4910, 'B', 'LO', 12303, '1032007', '08:02:41')
('006E6OQQ07', 'P', '709FQ38N07', 3434, '709FQ38O07', 'GB0009252882', 'GB', 'GBX',
'SET1', 16566, 'B', 'LO', 12276, '1032007', '08:02:41')
```

2. Count the Number of Cancelled Orders

In [58]:

```
# Connect to the SQLite database
conn = sqlite3.connect('high_frequency_finance.db')

# Create a cursor object
cursor = conn.cursor()

# Count the number of cancelled orders
sql_query = "SELECT * FROM History WHERE OrderActionType = 'D'"
# Execute the query
cursor.execute(sql_query)

# Fetch all rows from the query result
rows = cursor.fetchall()


# Count the number of cancelled orders
count=0
for row in rows:
    count+=1

# Print the result
print(f"Number of canceled orders: {count}")

# Close the connection
conn.close()
```

```
Number of canceled orders: 201675
```