Project on FIX file parsing

CS 6083 - Principles of Database Systems

Aditya Rajmane (anr331@nyu.edu)
Janhavi S Bagwe (js643@nyu.edu)
Vaibhav Mahajan (vm1131@nyu.edu)

Acknowledgements

We would like to thank Professor Raman Kannan for his help throughout the various stages of this project, right from the project idea – all the way to providing use with FIX files to process. We would also like to thank IBM for providing the cloud resources which made doing this project possible.

1. Introduction

The FIX protocol is the primary protocol used for communication between various financial entities in the stock markets today. This involves the exchange of several messages using fixed tag pairs – tag pair values have distinct meanings as per the FIX protocol, but our focus is limited to the most important tag pair fields (as explained in Section 2). As such, we are not processing all the fields which the FIX protocol involves.
Another point to note is that the FIX protocol uses socket communication to exchange messages over real time. Our project does not involve socket communication, and instead works on FIX log files which contain details on previous message exchanges.
Our project works on the following 2 message types:
35 = D (Single Order)
35 = 8 (Execution Report)
35= D messages are normally sent from the buy/sell side of a transaction, whereas 35 = 8 messages are sent from the side which conducts these transactions. ie. 35=8 messages contain details about whether the transaction requested in an order message (35=D) was fulfilled, partially fulfilled or unfulfilled, along with other related details.

2. Normalization and Table Creation

Initially we created 2 tables, one containing details from the 35=D messages file and one containing details from the 35=8 messages file. Both tables use the underlying FIX idea of key-pair values. i.e. Fields correspond to certain values. The queries used to create the tables are as follows:-

Query for creating 35=D table:-

```
create table 35d(
        ID int NOT NULL AUTO_INCREMENT,
        d49 varchar(10) NOT NULL,
        d56 varchar(10) NOT NULL,
        d34 int NOT NULL,
        d11 varchar(32) NOT NULL,
        d21 int NOT NULL,
        d55 varchar(10) NOT NULL,
        d48 varchar(32),
        d22 int,
        d167 varchar(2),
        d207 varchar(10),
        d54 int NOT NULL,
        d60 varchar(60) NOT NULL,
        d38 int,
        d40 varchar(1) NOT NULL,
        d44 double,
        CONSTRAINT chkd21 CHECK (d21 in (1,2,3)),
        CONSTRAINT chkd22 CHECK (d22 <=9 && d22>100),
        PRIMARY KEY (ID),
        CONSTRAINT chkd54  CHECK (d54 in (1,2,3,4,5,6,7,8,9)),
        CONSTRAINT chkd40 CHECK (d40 in
(1,2,3,4,5,6,7,8,9,'A','B','C','D','E','F','G','H','I','P')));
```

```
mysql> desc 35d
    -> ;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| ID    | int(11)     | NO   | PRI | NULL    | auto_increment |
| d49   | varchar(10) | NO   |     | NULL    |                |
| d56   | varchar(10) | NO   |     | NULL    |                |
| d34   | int(11)     | NO   |     | NULL    |                |
| d11   | varchar(32) | NO   |     | NULL    |                |
| d21   | int(11)     | NO   |     | NULL    |                |
| d55   | varchar(10) | NO   |     | NULL    |                |
| d48   | varchar(32) | YES  |     | NULL    |                |
| d22   | int(11)     | YES  |     | NULL    |                |
| d167  | varchar(2)  | YES  |     | NULL    |                |
| d207  | varchar(10) | YES  |     | NULL    |                |
| d54   | int(11)     | NO   |     | NULL    |                |
| d60   | varchar(60) | NO   |     | NULL    |                |
| d38   | int(11)     | YES  |     | NULL    |                |
| d40   | varchar(1)  | NO   |     | NULL    |                |
| d44   | double      | YES  |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
16 rows in set (0.00 sec)
```

In table 35d, we created a field called ID which we set to auto increment and used this field as our primary key. For certain fields such as d21,d22,d54 and d40 - we had to create constraints. This was done because according to the FIX documentation, these fields were only permitted to have certain values. For example, the constraint for d21 is created to ensure this value is set to 1,2 or 3 as these are the only permitted values for this field.

Query for creating 35=8 table:-

```
create table 35_8(
        EID int NOT NULL AUTO_INCREMENT,
        PRIMARY KEY (EID),
        e34 int NOT NULL,
        e49 varchar(10) NOT NULL,
        e56 varchar(10) NOT NULL,
        e55 varchar(10) NOT NULL,
        e54 int NOT NULL,
        CONSTRAINT chke54 CHECK (e54 in (1,2,3,4,5,6,7,8,9)),
        e11 varchar(32),
        e38 int,
        e40 varchar(1),
        CONSTRAINT chke40 CHECK (e40 in
(1,2,3,4,5,6,7,8,9,'A','B','C','D','E','F','G','H','I','P')),
        e59 int,
        CONSTRAINT chke59 CHECK (e59 in (0,1,2,3,4,5,6)),
        e21 int,
        CONSTRAINT chke21 CHECK (e21 in (1,2,3)),
        e31 double,
        e32 int,
        e39 varchar(10) NOT NULL,
        CONSTRAINT chke39 CHECK (e39 in (0,1,2,3,4,5,6,7,8,9,'A','B','C','D','E')),
        e150 varchar(2) NOT NULL,
        CONSTRAINT chke150 CHECK (e150 in (0,1,2,3,4,5,6,7,8,9,'A','B','C','D','E')),
        e14 int NOT NULL,
        e6 double NOT NULL,
        e151 int NOT NULL,
        e20 int NOT NULL,
        CONSTRAINT chke20 CHECK (e20 in (1,2,3)),
        e17 varchar(32) NOT NULL

);
```

```
mysql> desc 35_8;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| EID   | int(11)     | NO   | PRI | NULL    | auto_increment |
| e34   | int(11)     | NO   |     | NULL    |                |
| e49   | varchar(10) | NO   |     | NULL    |                |
| e56   | varchar(10) | NO   |     | NULL    |                |
| e55   | varchar(10) | NO   |     | NULL    |                |
| e54   | int(11)     | NO   |     | NULL    |                |
| e11   | varchar(32) | YES  |     | NULL    |                |
| e38   | int(11)     | YES  |     | NULL    |                |
| e40   | varchar(1)  | YES  |     | NULL    |                |
| e59   | int(11)     | YES  |     | NULL    |                |
| e21   | int(11)     | YES  |     | NULL    |                |
| e31   | double      | YES  |     | NULL    |                |
| e32   | int(11)     | YES  |     | NULL    |                |
| e39   | varchar(10) | NO   |     | NULL    |                |
| e150  | varchar(2)  | NO   |     | NULL    |                |
| e14   | int(11)     | NO   |     | NULL    |                |
| e6    | double      | NO   |     | NULL    |                |
| e151  | int(11)     | NO   |     | NULL    |                |
| e20   | int(11)     | NO   |     | NULL    |                |
| e17   | varchar(32) | NO   |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
20 rows in set (0.00 sec)
```

Using similar logic as that for table 35d, we created table 35_8. Here, we created a new auto increment field EID which was set as the primary key for this table. Constraints for different fields were created as per the FIX documentation for permitted values.

After running the python script *populate35d.py*, we ran a simple select query to view the contents of the 35d table. Note that limit 20 was only used so as to fit the output of the query to the screen.

```
mysql> Select * from 35d limit 20;
+-----+------+-------+------+------+------+------+------+------+------+------+---------+------+
| ID  | e35  | d49   | d56  | d34  | d55  | d40  | d54  | d38  | d21  | d11  | d59     |      |
+-----+------+-------+------+------+------+------+------+------+------+------+---------+------+
| 521 | D    | HB    | NYSE | 1501 | DUK  | 1    |    1 | 1700 |    2 | 500001 |    1 |
| 522 | D    | FMCP  | BIDS | 1502 | SBS  | 1    |    2 | 2000 |    2 | 500002 |    1 |
| 523 | D    | CSEQ  | BIDS | 1503 | NOK  | 1    |    2 | 2800 |    2 | 500003 |    1 |
| 524 | D    | CSEQ  | BATS | 1504 | BA   | 1    |    1 | 1500 |    2 | 500004 |    1 |
| 525 | D    | OPHMR | BATS | 1505 | JPM  | 1    |    1 | 1900 |    2 | 500005 |    1 |
| 526 | D    | HB    | ARCA | 1506 | BAC  | 1    |    2 | 1100 |    2 | 500006 |    1 |
| 527 | D    | JANUS | NYSE | 1507 | BAC  | 1    |    2 |  800 |    2 | 500007 |    1 |
| 528 | D    | CPW   | ARCA | 1508 | JPM  | 1    |    2 | 1400 |    2 | 500008 |    1 |
| 529 | D    | GSAM  | ARCA | 1509 | NOK  | 1    |    1 | 2400 |    2 | 500009 |    1 |
| 530 | D    | OPHMR | BATS | 1510 | ALU  | 1    |    2 | 2500 |    2 | 500010 |    1 |
| 531 | D    | FID   | ARCA | 1511 | VZ   | 1    |    2 | 2700 |    2 | 500011 |    1 |
| 532 | D    | OPHMR | ARCA | 1512 | BP   | 1    |    2 | 1400 |    2 | 500012 |    1 |
| 533 | D    | FID   | NYSE | 1513 | SBS  | 1    |    1 | 2500 |    2 | 500013 |    1 |
| 534 | D    | FID   | BATS | 1514 | AAPL | 1    |    1 |  700 |    2 | 500014 |    1 |
| 535 | D    | FID   | BATS | 1515 | AAPL | 1    |    2 | 2700 |    2 | 500015 |    1 |
| 536 | D    | HB    | NDQ  | 1516 | INTX | 1    |    2 | 2300 |    2 | 500016 |    1 |
| 537 | D    | FMCP  | BATS | 1517 | AA   | 1    |    1 | 1300 |    2 | 500017 |    1 |
| 538 | D    | CPW   | NDQ  | 1518 | SD   | 1    |    1 |  200 |    2 | 500018 |    1 |
| 539 | D    | OPHMR | NYSE | 1519 | BHP  | 1    |    2 | 1900 |    2 | 500019 |    1 |
| 540 | D    | FID   | BATS | 1520 | FCX  | 1    |    2 | 1200 |    2 | 500020 |    1 |
+-----+------+-------+------+------+------+------+------+------+------+------+---------+------+
20 rows in set (0.00 sec)
```

As we see, the input from the 35d log file has been successfully parsed by the python script. This parsed data has been used to populate our mySQL table 35d.

Similarly, table 35_8 was also populated by running the script *populate35_8.py*. The screenshot below shows part of the populated 35_8 table:

```
mysql> Select * from 35_8 limit 10;
+------+------+------+------+------+------+------+--------+------+------+------+------+------+------+------+------+------+------------------+------+------+------------
----+
| EID  | e35  | e34  | e49  | e56  | e55  | e54  | e11    | e38  | e40  | e59  | e21  | e31  | e32  | e39  | e150 | e14  | e6               | e151 | e20  | e17
   |
+------+------+------+------+------+------+------+--------+------+------+------+------+------+------+------+------+------+------------------+------+------+------------
----+
| 8226 | 8    |    1 | NYSE | HB   | DUK  |    1 | 500001 | 1700 | 1    |    1 |    2 | 76.81 |   50 | 1    | 1    |   50 |            76.81 | 1650 |    0 | 500001-DUK-
1
 |
| 8227 | 8    |    2 | NYSE | HB   | DUK  |    1 | 500001 | 1700 | 1    |    1 |    2 | 76.74 |  100 | 1    | 1    |  150 | 76.7633333333333 | 1550 |    0 | 500001-DUK-
2
 |
| 8228 | 8    |    3 | NYSE | HB   | DUK  |    1 | 500001 | 1700 | 1    |    1 |    2 | 76.88 |  200 | 1    | 1    |  350 |            76.83 | 1350 |    0 | 500001-DUK-
3
 |
| 8229 | 8    |    4 | NYSE | HB   | DUK  |    1 | 500001 | 1700 | 1    |    1 |    2 | 76.88 |  300 | 1    | 1    |  650 | 76.8530769230769 | 1050 |    0 | 500001-DUK-
4
 |
| 8230 | 8    |    5 | NYSE | HB   | DUK  |    1 | 500001 | 1700 | 1    |    1 |    2 | 76.81 |  500 | 1    | 1    | 1150 |  76.834347826087 |  550 |    0 | 500001-DUK-
5
 |
| 8231 | 8    |    6 | NYSE | HB   | DUK  |    1 | 500001 | 1700 | 1    |    1 |    2 | 76.74 |  500 | 1    | 1    | 1650 | 76.8057575757576 |   50 |    0 | 500001-DUK-
6
 |
| 8232 | 8    |    7 | NYSE | HB   | DUK  |    1 | 500001 | 1700 | 1    |    1 |    2 | 76.74 |   50 | 2    | 1    | 1700 | 76.8038235294118 |    0 |    0 | 500001-DUK-
7
 |
| 8233 | 8    |    2 | BIDS | FMCP | SBS  |    2 | 500002 | 2000 | 1    |    1 |    2 |  6.48 |   50 | 1    | 1    |   50 |             6.48 | 1950 |    0 | 500002-SBS-
2
 |
| 8234 | 8    |    3 | BIDS | FMCP | SBS  |    2 | 500002 | 2000 | 1    |    1 |    2 |  6.48 |  100 | 1    | 1    |  150 |             6.48 | 1850 |    0 | 500002-SBS-
3
 |
| 8235 | 8    |    4 | BIDS | FMCP | SBS  |    2 | 500002 | 2000 | 1    |    1 |    2 |  6.48 |  200 | 1    | 1    |  350 |             6.48 | 1650 |    0 | 500002-SBS-
4
 |
+------+------+------+------+------+------+------+--------+------+------+------+------+------+------+------+------+------+------------------+------+------+------------
----+
10 rows in set (0.00 sec)
```

We have normalised the two tables into a single table and that's how we understand the flow of FIX messages.

- We have the **Sender_id** and **Receiver_id** to initiate and receive the order respectively.
- **e11** is the order number column, unique to an order.
- This order is of which type is specified by the **TYPE** column.
- **SYMBOL** also specifies the type of the message
- Quantity specifying the max no of shares for this order number.
- STATUS column defining status of the order, either partially filled/filled.
- order_px: price of this order
- avg_px: avg price for all the fills in the order.
- qty_left: number of fills left for this order to be completely filled.

```
mysql> Select * from normalized limit 20;
+-----------+-------------+--------+------+--------+----------+------------------+----------+------------------+----------+
| Sender_id | Receiver_id | e11    | type | symbol | quantity | status           | order_px | avg_px           | qty_left |
+-----------+-------------+--------+------+--------+----------+------------------+----------+------------------+----------+
| NYSE      | HB          | 500001 | BUY  | DUK    |     1700 | PARTIALLY FILLED |    76.81 |            76.81 |     1650 |
| NYSE      | HB          | 500001 | BUY  | DUK    |     1700 | PARTIALLY FILLED |    76.74 | 76.7633333333333 |     1550 |
| NYSE      | HB          | 500001 | BUY  | DUK    |     1700 | PARTIALLY FILLED |    76.88 |            76.83 |     1350 |
| NYSE      | HB          | 500001 | BUY  | DUK    |     1700 | PARTIALLY FILLED |    76.88 | 76.8530769230769 |     1050 |
| NYSE      | HB          | 500001 | BUY  | DUK    |     1700 | PARTIALLY FILLED |    76.81 |  76.834347826087 |      550 |
| NYSE      | HB          | 500001 | BUY  | DUK    |     1700 | PARTIALLY FILLED |    76.74 | 76.8057575757576 |       50 |
| NYSE      | HB          | 500001 | BUY  | DUK    |     1700 | FILLED           |    76.74 | 76.8038235294118 |        0 |
| BIDS      | FMCP        | 500002 | SELL | SBS    |     2000 | PARTIALLY FILLED |     6.48 |             6.48 |     1950 |
| BIDS      | FMCP        | 500002 | SELL | SBS    |     2000 | PARTIALLY FILLED |     6.48 |             6.48 |     1850 |
| BIDS      | FMCP        | 500002 | SELL | SBS    |     2000 | PARTIALLY FILLED |     6.48 |             6.48 |     1650 |
| BIDS      | FMCP        | 500002 | SELL | SBS    |     2000 | PARTIALLY FILLED |     6.34 | 6.41538461538462 |     1350 |
| BIDS      | FMCP        | 500002 | SELL | SBS    |     2000 | PARTIALLY FILLED |     6.34 | 6.38260869565217 |      850 |
| BIDS      | FMCP        | 500002 | SELL | SBS    |     2000 | PARTIALLY FILLED |     6.41 | 6.39090909090909 |      350 |
| BIDS      | FMCP        | 500002 | SELL | SBS    |     2000 | PARTIALLY FILLED |     6.34 | 6.38307692307692 |       50 |
| BIDS      | FMCP        | 500002 | SELL | SBS    |     2000 | FILLED           |     6.48 |           6.3855 |        0 |
| BIDS      | CSEQ        | 500003 | SELL | NOK    |     2800 | PARTIALLY FILLED |     6.79 |             6.79 |     2750 |
| BIDS      | CSEQ        | 500003 | SELL | NOK    |     2800 | PARTIALLY FILLED |     6.79 |             6.79 |     2650 |
| BIDS      | CSEQ        | 500003 | SELL | NOK    |     2800 | PARTIALLY FILLED |     6.65 |             6.71 |     2450 |
| BIDS      | CSEQ        | 500003 | SELL | NOK    |     2800 | PARTIALLY FILLED |     6.79 | 6.74692307692308 |     2150 |
| BIDS      | CSEQ        | 500003 | SELL | NOK    |     2800 | PARTIALLY FILLED |     6.65 | 6.70478260869565 |     1650 |
+-----------+-------------+--------+------+--------+----------+------------------+----------+------------------+----------+
20 rows in set (0.01 sec)
```

3. Scripts

The following were the php and html scripts which were created and run on the localhost:

php file:

```
<html>
<?php
$servername = "localhost:3307";
$username = "anr331";
$password = "anr331123";
$dbname = "anr331";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}


$sql = $_POST["query"].' where e11='.$_POST["custid"]." ; ";
//print $sql;
$result = $conn->query($sql);

//var_dump($result->num_rows);
//print $result;
if ($result->num_rows > 0) {
  // output data of each row
   while($row = $result->fetch_assoc()) {
          //while($row = $result) {
     echo "<br><i>EXCHANGE :</i>" . $row["Sender_id"]. "    | <i>CLIENT</i> :" .
$row["Receiver_id"]. "   | <i>ACTION :</i> <b>". $row["type"]."</b> ".$row["symbol"]."   |
<i>Shares Processed: </i>".($row["quantity"]-$row["qty_left"])."| <i>Average price recieved
for this iteration: </i>".($row["avg_px"]);
    }

} else {
```

```
    echo "0 results";
}

$conn->close();
?>
</html>
```

html file:

```
<html>
<body>

<form action="test.php" method="post">
enter query <input type="text" name="query"><br>
enter ORDER NUMBER<input type="text" name ="custid"><br>
<input type="submit">
</form>

</body>
</html>
```

The python script is placed in the projects folder of the IBM cloud account ‘anr331’. Script for this was as follows:

python file:

```
f=open('/home/CS6083/anr331/35-8.txt','r')
data = f.readlines()

for line in data:
        words= re.split('|',line)
        print words
        eight = words[0].split("=")[1]
        nine = words[1].split("=")[1]
        #if(words[33].split("=")[0]=='44'):
         #       ff=words[33].split("=")[1]
        #else:
         #       ff= '0'

        cursor.execute("insert into 35_8
values('"+words[4].split('=')[1]+"','"+words[6].split('=')[1]+"','"+words[3].split('=')[1]+"','"+
words[11].split('=')[1]+"','"+words[9].split('=')[1]+"','"+words[10].split('=')[1]+"','"+words[12].
split('=')[1]+"','"+words[13].split('=')[1]+"','"+words[14].split('=')[1]+"','"+words[16].split('=')[
1]+"','"+words[17].split('=')[1]+"','"+words[18].split('=')[1]+"','"+words[19].split('=')[1]+"','"+
words[21].split('=')[1]+"','"+words[22].split('=')[1]+"','"+words[20].split('=')[1]+"','"+words[31
].split('=')[1]+"','"+words[25].split('=')[1]+"','"+words[24].split('=')[1]+"','"+words[26].split('='
)[1]+"','"+words[29].split('=')[1]+"','"+words[30].split('=')[1]+"','");")
myDB.commit()
```

4. Queries:

uname for ibm:anr331 pwd: aditya1993

mysql -uanr331 -panr331123

Query 1. Give out the overall average cost at which he has received the entire stock:

mysql> select symbol,AVG(avg_px) from normalized group by symbol;

```
+--------+--------------------+
| symbol | AVG(avg_px)        |
+--------+--------------------+
| AA     | 13.722771986228564 |
| AAPL   | 125.26677454336024 |
| ALU    |  3.642086909592894 |
| BA     |   141.488878601264 |
| BAC    | 16.242873263275463 |
| BHP    |  50.27303207735124 |
| BP     | 41.831446858693965 |
| C      |  53.30628843506085 |
| CCJ    |  16.66827852565633 |
| CSCO   | 28.807009003750313 |
| DUK    |  76.81665474404178 |
| FB     |  78.43244972997019 |
| FCX    | 22.920180426454763 |
| GE     | 27.028505531676355 |
| IBM    |  170.9876732701961 |
| INTC   |  32.25304137835773 |
| INTX   |  3.486693878235626 |
| JPM    |  64.49407594277987 |
| MC     | 27.861301771768044 |
| MSFT   | 46.709107188564985 |
| MYL    |  70.39353009923934 |
| NOK    | 6.7396462818042675 |
| SBS    |  6.407668456436998 |
| SD     | 1.6263991072595385 |
| T      |   33.3745125546859 |
| V      |  66.57665464109384 |
| VZ     |  49.71461391663697 |
| XOM    |  87.59999862654587 |
+--------+--------------------+
28 rows in set (0.01 sec)
```

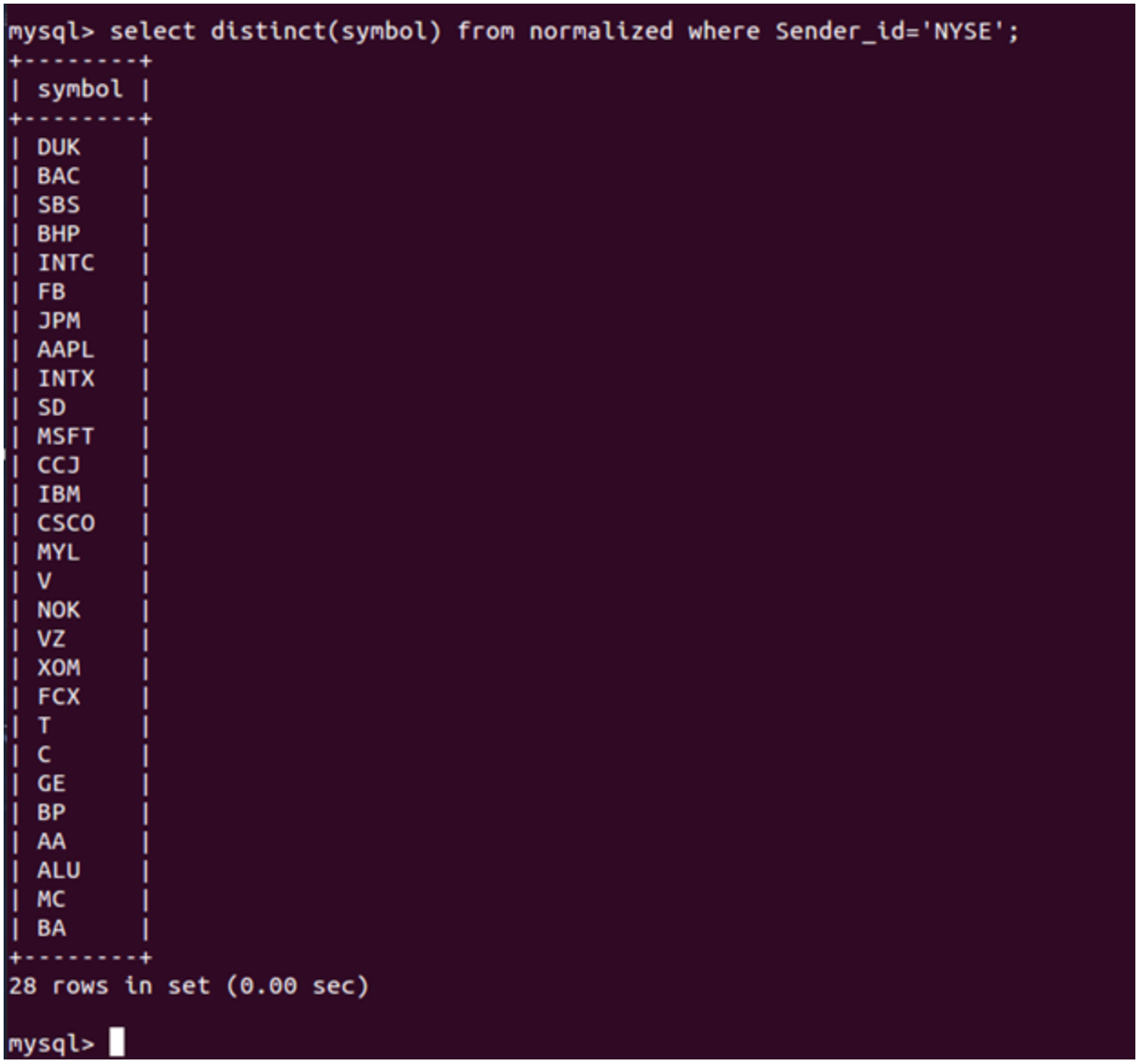
Query 2: Display the stock whose quantity was the most:

select symbol,MAX(quantity) from normalized;

```
+--------+---------------+
| symbol | MAX(quantity) |
+--------+---------------+
```

| DUK | 2800 |
+--------+---------------+
1 row in set (0.00 sec)

Query 3: To list the unique symbols which were traded by the NYSE:



Query4: Find the total number of transactions for each order:

mysql> Select e11,count(*) from normalized group by e11;

+--------+----------+

| e11 | count(*) |

+--------+----------+

500001	7
500002	8
500003	14
500004	7
500005	8
500006	5
500007	4
500008	7
500009	12
500010	13
500011	13
500012	7
500013	13
500014	4
500015	13
500016	10
500017	7
500018	3
500019	8
500020	6
500021	4
500022	5
500023	5
500024	7
500025	4
500026	3
500027	4
500028	4
500029	2
500030	8
500031	7
500032	9
500033	8

500034	5
500035	13
500036	3
500037	5
500038	8
500039	14
500040	13
500041	8
500042	2
500043	13
500044	6
500045	3
500046	3
500047	6
500048	8
500049	2
500050	7
500051	8
500052	8
500053	3
500054	7
500055	3
500056	10
500057	3
500058	9
500059	8
500060	12
500061	2
500062	13
500063	5
500064	8
500065	10
500066	7

500067	12
500068	7
500069	3
500070	8
500071	6
500072	5
500073	4
500074	9
500075	9
500076	4
500077	3
500078	5
500079	8
500080	14
500081	3
500082	3
500083	3
500084	5
500085	13
500086	3
500087	8
500088	7
500089	2
500090	5
500091	8
500092	9
500093	5
500094	4
500095	8
500096	3
500097	7
500098	13
500099	9

500100	14
500101	9
500102	3
500103	9
500104	4
500105	3
500106	14
500107	7
500108	4
500109	2
500110	3
500111	12
500112	9
500113	4
500114	7
500115	3
500116	9
500117	9
500118	8
500119	14
500120	10
500121	8
500122	3
500123	9
500124	9
500125	8
500126	9
500127	8
500128	8
500129	7
500130	3
500131	13
500132	10

500133	3
500134	4
500135	5
500136	13
500137	4
500138	4
500139	5
500140	5
500141	3
500142	12
500143	3
500144	8
500145	7
500146	14
500147	4
500148	7
500149	9
500150	8
500151	8
500152	8
500153	5
500154	3
500155	9
500156	3
500157	7
500158	7
500159	3
500160	3
500161	12
500162	14
500163	4
500164	7
500165	7

500166	6
500167	3
500168	8
500169	9
500170	13
500171	13
500172	7
500173	2
500174	13
500175	3
500176	8
500177	7
500178	8
500179	8
500180	14
500181	3
500182	9
500183	10
500184	8
500185	3
500186	4
500187	4
500188	13
500189	4
500190	4
500191	4
500192	6
500193	4
500194	8
500195	14
500196	9
500197	6
500198	7

500199	5
500200	7
500201	8
500202	12
500203	13
500204	7
500205	3
500206	8
500207	13
500208	7
500209	8
500210	7
500211	8
500212	8
500213	4
500214	7
500215	3
500216	13
500217	13
500218	8
500219	2
500220	5
500221	2
500222	9
500223	4
500224	7
500225	6
500226	13
500227	3
500228	6
500229	13
500230	14
500231	5

500232	8
500233	8
500234	8
500235	2
500236	2
500237	13
500238	4
500239	3
500240	3
500241	4
500242	8
500243	3
500244	4
500245	9
500246	8
500247	5
500248	3
500249	7
500250	4
500251	2
500252	12
500253	13
500254	4
500255	10
500256	9
500257	6
500258	8
500259	6
500260	8
500261	4
500262	3
500263	5
500264	8

500265	3
500266	3
500267	3
500268	9
500269	7
500270	3
500271	7
500272	3
500273	14
500274	9
500275	2
500276	4
500277	12
500278	7
500279	13
500280	3
500281	4
500282	9
500283	2
500284	3
500285	3
500286	3
500287	7
500288	4
500289	8
500290	12
500291	9
500292	10
500293	5
500294	3
500295	3
500296	2
500297	3

500298	9
500299	8
500300	8
500301	13
500302	5
500303	12
500304	14
500305	7
500306	3
500307	7
500308	8
500309	13
500310	8
500311	8
500312	8
500313	3
500314	8
500315	9
500316	7
500317	13
500318	4
500319	8
500320	7
500321	14
500322	8
500323	5
500324	8
500325	13
500326	4
500327	9
500328	8
500329	5
500330	8

500331	3
500332	9
500333	13
500334	4
500335	14
500336	4
500337	6
500338	3
500339	7
500340	7
500341	7
500342	7
500343	4
500344	14
500345	13
500346	3
500347	8
500348	8
500349	7
500350	3
500351	4
500352	13
500353	7
500354	9
500355	4
500356	4
500357	14
500358	3
500359	12
500360	9
500361	8
500362	5
500363	10

500364	14
500365	9
500366	8
500367	9
500368	4
500369	7
500370	4
500371	9
500372	7
500373	8
500374	14
500375	7
500376	13
500377	3
500378	6
500379	7
500380	5
500381	4
500382	13
500383	8
500384	4
500385	13
500386	3
500387	9
500388	3
500389	5
500390	8
500391	12
500392	4
500393	8
500394	4
500395	7
500396	3

500397	6
500398	14
500399	4
500400	10
500401	4
500402	3
500403	8
500404	4
500405	4
500406	5
500407	13
500408	13
500409	6
500410	4
500411	13
500412	4
500413	3
500414	4
500415	6
500416	12
500417	13
500418	4
500419	7
500420	4
500421	9
500422	10
500423	13
500424	3
500425	7
500426	8
500427	9
500428	3
500429	4

500430	4
500431	5
500432	3
500433	6
500434	3
500435	7
500436	14
500437	13
500438	10
500439	10
500440	8
500441	4
500442	10
500443	4
500444	3
500445	3
500446	3
500447	3
500448	3
500449	7
500450	14
500451	12
500452	7
500453	4
500454	3
500455	14
500456	12
500457	13
500458	3
500459	3
500460	7
500461	3
500462	13

500463	3
500464	6
500465	4
500466	4
500467	12
500468	9
500469	8
500470	9
500471	9
500472	8
500473	5
500474	4
500475	6
500476	2
500477	9
500478	3
500479	3
500480	8
500481	7
500482	4
500483	9
500484	2
500485	6
500486	3
500487	7
500488	3
500489	8
500490	6
500491	3
500492	3
500493	8
500494	3
500495	13

500496	8
500497	2
500498	5
500499	6
500500	8
+--------+----------+
500 rows in set (0.01 sec)

Query 5: Determine whether each order was BUY/SELL

mysql> select distinct e11, type from normalized where e11 in (Select distinct e11 from normalized);

+--------+------+
| e11 | type |
+--------+------+
500001	BUY
500002	SELL
500003	SELL
500004	BUY
500005	BUY
500006	SELL
500007	SELL
500008	SELL
500009	BUY
500010	SELL
500011	SELL
500012	SELL
500013	BUY
500014	BUY
500015	SELL
500016	SELL
500017	BUY

500018	BUY
500019	SELL
500020	SELL
500021	BUY
500022	BUY
500023	BUY
500024	BUY
500025	SELL
500026	BUY
500027	SELL
500028	BUY
500029	BUY
500030	BUY
500031	BUY
500032	SELL
500033	BUY
500034	SELL
500035	BUY
500036	SELL
500037	BUY
500038	BUY
500039	SELL
500040	BUY
500041	BUY
500042	SELL
500043	BUY
500044	BUY
500045	BUY
500046	SELL
500047	BUY
500048	SELL
500049	SELL
500050	BUY

500051	BUY
500052	BUY
500053	BUY
500054	BUY
500055	BUY
500056	BUY
500057	BUY
500058	BUY
500059	SELL
500060	SELL
500061	SELL
500062	BUY
500063	SELL
500064	SELL
500065	BUY
500066	BUY
500067	BUY
500068	SELL
500069	BUY
500070	BUY
500071	SELL
500072	BUY
500073	SELL
500074	SELL
500075	SELL
500076	SELL
500077	SELL
500078	BUY
500079	BUY
500080	BUY
500081	BUY
500082	SELL
500083	BUY

500084	BUY
500085	SELL
500086	SELL
500087	BUY
500088	BUY
500089	SELL
500090	SELL
500091	SELL
500092	BUY
500093	SELL
500094	SELL
500095	BUY
500096	BUY
500097	BUY
500098	SELL
500099	BUY
500100	SELL
500101	BUY
500102	BUY
500103	BUY
500104	BUY
500105	BUY
500106	BUY
500107	BUY
500108	BUY
500109	BUY
500110	SELL
500111	BUY
500112	SELL
500113	SELL
500114	SELL
500115	SELL
500116	BUY

500117	BUY
500118	SELL
500119	BUY
500120	BUY
500121	SELL
500122	BUY
500123	SELL
500124	BUY
500125	BUY
500126	BUY
500127	BUY
500128	BUY
500129	SELL
500130	BUY
500131	SELL
500132	BUY
500133	BUY
500134	BUY
500135	BUY
500136	BUY
500137	BUY
500138	SELL
500139	SELL
500140	SELL
500141	BUY
500142	SELL
500143	SELL
500144	SELL
500145	BUY
500146	SELL
500147	SELL
500148	SELL
500149	BUY

500150	BUY
500151	SELL
500152	BUY
500153	SELL
500154	SELL
500155	SELL
500156	BUY
500157	BUY
500158	BUY
500159	BUY
500160	SELL
500161	SELL
500162	BUY
500163	BUY
500164	BUY
500165	BUY
500166	BUY
500167	BUY
500168	BUY
500169	BUY
500170	SELL
500171	BUY
500172	SELL
500173	BUY
500174	BUY
500175	SELL
500176	SELL
500177	BUY
500178	SELL
500179	BUY
500180	SELL
500181	BUY
500182	BUY

500183	BUY
500184	SELL
500185	BUY
500186	BUY
500187	BUY
500188	BUY
500189	SELL
500190	BUY
500191	SELL
500192	SELL
500193	BUY
500194	BUY
500195	BUY
500196	BUY
500197	BUY
500198	BUY
500199	BUY
500200	SELL
500201	BUY
500202	BUY
500203	SELL
500204	BUY
500205	SELL
500206	SELL
500207	BUY
500208	SELL
500209	BUY
500210	BUY
500211	SELL
500212	SELL
500213	BUY
500214	BUY
500215	SELL

500216	BUY
500217	SELL
500218	SELL
500219	BUY
500220	SELL
500221	SELL
500222	SELL
500223	BUY
500224	BUY
500225	BUY
500226	BUY
500227	BUY
500228	BUY
500229	BUY
500230	SELL
500231	SELL
500232	SELL
500233	BUY
500234	BUY
500235	SELL
500236	BUY
500237	BUY
500238	SELL
500239	SELL
500240	SELL
500241	BUY
500242	SELL
500243	BUY
500244	BUY
500245	SELL
500246	SELL
500247	BUY
500248	BUY

500249	BUY
500250	BUY
500251	SELL
500252	SELL
500253	SELL
500254	BUY
500255	SELL
500256	BUY
500257	BUY
500258	SELL
500259	SELL
500260	SELL
500261	SELL
500262	SELL
500263	BUY
500264	BUY
500265	BUY
500266	BUY
500267	SELL
500268	BUY
500269	SELL
500270	BUY
500271	BUY
500272	SELL
500273	BUY
500274	BUY
500275	SELL
500276	BUY
500277	BUY
500278	BUY
500279	BUY
500280	BUY
500281	BUY

500282	**BUY**
500283	**BUY**
500284	**SELL**
500285	**BUY**
500286	**SELL**
500287	**BUY**
500288	**SELL**
500289	**BUY**
500290	**SELL**
500291	**BUY**
500292	**BUY**
500293	**BUY**
500294	**SELL**
500295	**SELL**
500296	**BUY**
500297	**BUY**
500298	**SELL**
500299	**BUY**
500300	**BUY**
500301	**BUY**
500302	**BUY**
500303	**BUY**
500304	**SELL**
500305	**BUY**
500306	**SELL**
500307	**SELL**
500308	**SELL**
500309	**SELL**
500310	**SELL**
500311	**SELL**
500312	**SELL**
500313	**SELL**
500314	**SELL**

500315	BUY
500316	BUY
500317	SELL
500318	SELL
500319	BUY
500320	BUY
500321	SELL
500322	BUY
500323	SELL
500324	BUY
500325	SELL
500326	BUY
500327	BUY
500328	SELL
500329	SELL
500330	SELL
500331	BUY
500332	BUY
500333	BUY
500334	BUY
500335	SELL
500336	BUY
500337	BUY
500338	BUY
500339	SELL
500340	BUY
500341	BUY
500342	SELL
500343	SELL
500344	SELL
500345	SELL
500346	BUY
500347	SELL

500348	**BUY**
500349	**SELL**
500350	**BUY**
500351	**BUY**
500352	**SELL**
500353	**SELL**
500354	**SELL**
500355	**BUY**
500356	**SELL**
500357	**BUY**
500358	**SELL**
500359	**BUY**
500360	**BUY**
500361	**BUY**
500362	**SELL**
500363	**BUY**
500364	**BUY**
500365	**BUY**
500366	**SELL**
500367	**BUY**
500368	**BUY**
500369	**SELL**
500370	**SELL**
500371	**BUY**
500372	**SELL**
500373	**SELL**
500374	**SELL**
500375	**SELL**
500376	**BUY**
500377	**SELL**
500378	**SELL**
500379	**BUY**
500380	**SELL**

500381	**SELL**
500382	**BUY**
500383	**SELL**
500384	**SELL**
500385	**SELL**
500386	**BUY**
500387	**BUY**
500388	**SELL**
500389	**BUY**
500390	**BUY**
500391	**SELL**
500392	**SELL**
500393	**SELL**
500394	**SELL**
500395	**SELL**
500396	**BUY**
500397	**SELL**
500398	**SELL**
500399	**SELL**
500400	**SELL**
500401	**BUY**
500402	**SELL**
500403	**BUY**
500404	**SELL**
500405	**BUY**
500406	**SELL**
500407	**SELL**
500408	**SELL**
500409	**BUY**
500410	**SELL**
500411	**BUY**
500412	**SELL**
500413	**SELL**

500414	**BUY**
500415	**BUY**
500416	**SELL**
500417	**SELL**
500418	**SELL**
500419	**BUY**
500420	**SELL**
500421	**BUY**
500422	**BUY**
500423	**SELL**
500424	**BUY**
500425	**BUY**
500426	**SELL**
500427	**BUY**
500428	**SELL**
500429	**BUY**
500430	**BUY**
500431	**SELL**
500432	**BUY**
500433	**SELL**
500434	**BUY**
500435	**SELL**
500436	**SELL**
500437	**SELL**
500438	**BUY**
500439	**BUY**
500440	**BUY**
500441	**SELL**
500442	**SELL**
500443	**BUY**
500444	**SELL**
500445	**BUY**
500446	**SELL**

500447	**SELL**
500448	**SELL**
500449	**SELL**
500450	**BUY**
500451	**BUY**
500452	**SELL**
500453	**BUY**
500454	**SELL**
500455	**BUY**
500456	**SELL**
500457	**SELL**
500458	**BUY**
500459	**BUY**
500460	**BUY**
500461	**BUY**
500462	**SELL**
500463	**SELL**
500464	**BUY**
500465	**SELL**
500466	**BUY**
500467	**SELL**
500468	**SELL**
500469	**BUY**
500470	**SELL**
500471	**SELL**
500472	**SELL**
500473	**BUY**
500474	**BUY**
500475	**BUY**
500476	**BUY**
500477	**BUY**
500478	**SELL**
500479	**BUY**

500480	BUY
500481	BUY
500482	SELL
500483	BUY
500484	SELL
500485	BUY
500486	BUY
500487	BUY
500488	SELL
500489	SELL
500490	SELL
500491	SELL
500492	SELL
500493	BUY
500494	SELL
500495	BUY
500496	SELL
500497	BUY
500498	BUY
500499	BUY
500500	SELL
+--------+------+
500 rows in set (3.77 sec)