

Advanced Operators and Clauses

ISM 6218

Due on September 10

The Avengers Team

“We will avenge every problem on our way”

Aitemir Yeskenov (Team Lead)

Nagarjuna Kanneganti

Sai Suraj Argula

Vinay Kumar Reddy Baradi

Table of Contents

Business Process Supported	1
Requirements Described	2
Database Diagram	3
Use of “number of records” clause	3
Use of TABLESAMPLE	4
Use of TOP 100	5
Use of INTERSECT AND EXCEPT	6
Use of DATEPART	9
Use of DATEADD	10
Use of ROW_NUMBER	12

Business Process Supported

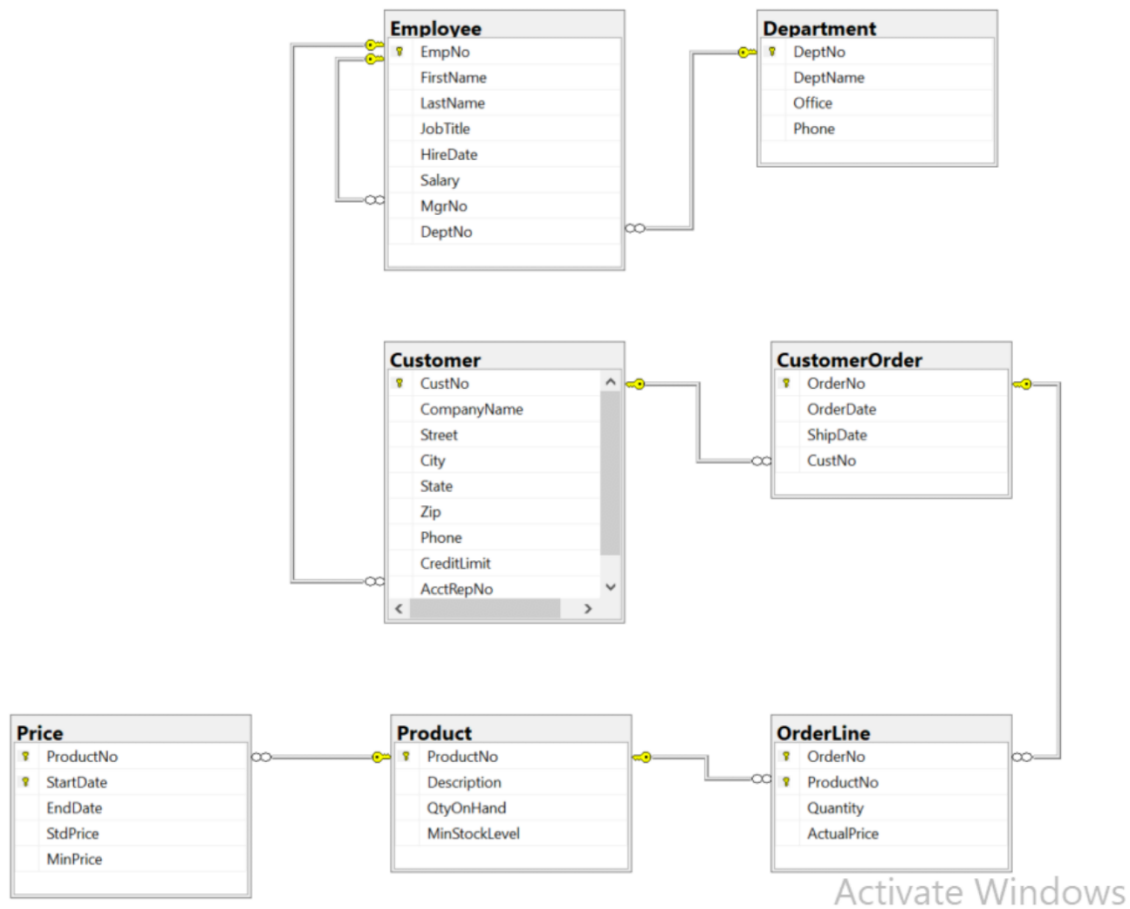
For this assignment, we decided to use the ACME database. The database has a variety of entities that have different relationships. The main two tables that we worked on are Customer and CustomerOrder. Both tables are connected to each other since based on the scenario a customer can place customer orders. We decided to run different experimental operations and find out how they can help us to achieve efficiency in performance and simplicity

Requirements Described

Using any table in Hospital or ACME databases, demonstrate use of:

1. 'number of records' clause;
 'Table Sample' percent and repeatable;
 Compare this with * top 100 (1000 rows).
 You will need to make use of data generating tool to create enough rows.
2. Demonstrate use of Intersect and Except operators with an experiment that compares each operation to a Join.
3. Create a user story for an experiment run comparing use of the three datepart functions to not using the three datepart functions.
4. Demonstrate use of AddDate.
5. Create a user story to demonstrate Row_Num, Over, partition by.

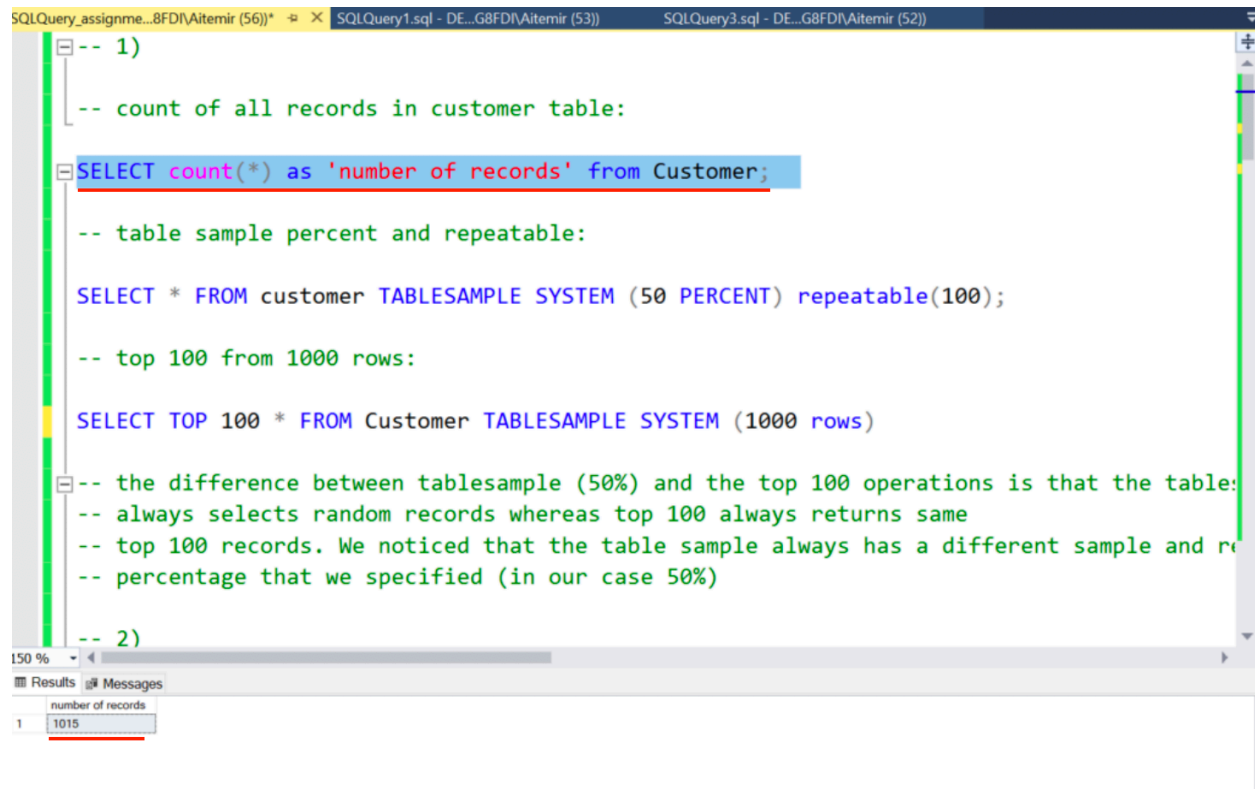
Below is the DB diagram of the ACME:



In order to run the experiments required for the assignment, we used **generatedata** tool to generate the data and insert it to the **Customer** table.

1) Use of “number of records” clause.

In order to count the number of records in the Customer table we used the count operation and showed it as “number of records” column:



```
-- 1)
-- count of all records in customer table:
SELECT count(*) as 'number of records' from Customer;

-- table sample percent and repeatable:
SELECT * FROM customer TABLESAMPLE SYSTEM (50 PERCENT) repeatable(100);

-- top 100 from 1000 rows:
SELECT TOP 100 * FROM Customer TABLESAMPLE SYSTEM (1000 rows)

-- the difference between tablesample (50%) and the top 100 operations is that the table:
-- always selects random records whereas top 100 always returns same
-- top 100 records. We noticed that the table sample always has a different sample and r
-- percentage that we specified (in our case 50%)

-- 2)
```

Results

	number of records
1	1015

The output that we received is 1015 and it is shown under the “number of records”, which is exactly the name column we stated in the operation.

We decided to run the **TABLESAMPLE** operation to return a random sample with records from the Customer table. We decided to run the operation to return only 50% of the random sample:

```
-- table sample percent and repeatable:

SELECT * FROM customer TABLESAMPLE SYSTEM (50 PERCENT) repeatable(100);
```

Results Messages									
	CustNo	CompanyName	Street	City	State	Zip	Phone	CreditLimit	AcctRepNo
1	100	Turner Sporting Goods	612 Sandstone St.	Ocala	FL	34481	(352) 751-8423	10000.00	1005
2	101	Ralph's Outdoor Emporium	3221 Oakdale Ln.	Palm Springs	FL	33461	(561) 324-9097	10000.00	1005
3	102	P & T Entertainment	51-A Lincoln St.	Bradenton	FL	34207	(941) 347-8787	5000.00	1007
4	103	Sports World	32190 Fresco Dr.	Tampa	FL	33629	(813) 842-1029	7500.00	1007
5	105	Fred's Funtime	932 Murray Blvd.	Atlanta	GA	30322	(404) 251-1000	10000.00	1010
6	106	Major League Sports	10 Bowdoin Rd.	Trenton	GA	30752	(706) 657-2223	10000.00	1010
7	107	Score-4 Sports	444 Windom Pl.	Lakeland	FL	33811	(863) 709-1486	7500.00	1005
8	109	Two Guys & A Gal Fitness Center	4 Branson St.	Baton Rouge	LA	70806	(225) 922-8777	5000.00	1018
9	110	The Sports Shoppe	2551 Richardson Dr.	Plano	TX	75023	(469) 241-0076	7500.00	1018
10	111	JRG Enterprises	43 Central Ave.	Tampa	FL	33615	(813) 885-1111	10000.00	1007
11	112	Bats, Balls, & Gloves	1500 Carroll Way	Tulsa	OK	74130	(918) 425-5005	5000.00	1018
12	113	Foster Sports Supply	87 Swanson Ln.	Lake City	FL	32024	(386) 755-3365	10000.00	1010
13	125	Nunc Associates	Ap #299-926 Eget, St.	Laramie	WY	34158	(411) 796-2266	445.00	1010
14	130	Nunc Associates	Ap #299-926 Eget, St.	Laramie	WY	34158	(411) 726-2266	445.00	1010
15	133	Non Cursus LLC	P.O. Box 904, 4799 Commodo Rd.	Gillette	WY	41836	(497) 299-7962	177.00	1005

Query executed successfully. DESKTOP-7NG8FDI (14.0 RTM) DESKTOP-7

(309 rows affected)

Completion time: 2019-09-10T03:06:54.2125495-04:00

As a result, 309 rows are returned. Whenever we run it again, we get the same results since we used “repeatable” and the seed number.

Our next task was to see the top 100 records from the sample of 1000 records. In order to retrieve the top 100 rows from the 1000 rows we used the following operation:

SELECT TOP 100 * FROM Customer TABLESAMPLE SYSTEM (1000 rows)

Results Messages									
	CustNo	CompanyName	Street	City	State	Zip	Phone	CreditLimit	AcctRepNo
1	100	Turner Sporting Goods	612 Sandstone St.	Ocala	FL	34481	(352) 751-8423	10000.00	1005
2	101	Ralph's Outdoor Emporium	3221 Oakdale Ln.	Palm Springs	FL	33461	(561) 324-9097	10000.00	1005
3	102	P & T Entertainment	51-A Lincoln St.	Bradenton	FL	34207	(941) 347-8787	5000.00	1007
4	103	Sports World	32190 Fresco Dr.	Tampa	FL	33629	(813) 842-1029	7500.00	1007
5	105	Fred's Funtime	932 Murray Blvd.	Atlanta	GA	30322	(404) 251-1000	10000.00	1010
6	106	Major League Sports	10 Bowdoin Rd.	Trenton	GA	30752	(706) 657-2223	10000.00	1010
7	107	Score-4 Sports	444 Windom Pl.	Lakeland	FL	33811	(863) 709-1486	7500.00	1005
8	109	Two Guys & A Gal Fitness Center	4 Branson St.	Baton Rouge	LA	70806	(225) 922-8777	5000.00	1018
9	110	The Sports Shoppe	2551 Richardson Dr.	Plano	TX	75023	(469) 241-0076	7500.00	1018
10	111	JRG Enterprises	43 Central Ave.	Tampa	FL	33615	(813) 885-1111	10000.00	1007
11	112	Bats, Balls, & Gloves	1500 Carroll Way	Tulsa	OK	74130	(918) 425-5005	5000.00	1018
12	113	Foster Sports Supply	87 Swanson Ln.	Lake City	FL	32024	(386) 755-3365	10000.00	1010
13	125	Nunc Associates	Ap #299-926 Eget, St.	Laramie	WY	34158	(411) 796-2266	445.00	1010
14	130	Nunc Associates	Ap #299-926 Eget, St.	Laramie	WY	34158	(411) 726-2266	445.00	1010
15	133	Non Cursus LLC	P.O. Box 904, 4799 Commodo Rd.	Gillette	WY	41836	(497) 299-7962	177.00	1005

Query executed successfully. DESKTOP-7NG8FDI (14.0 RTM) DESKTOP-7

Results		Messages							
	CustNo	CompanyName	Street	City	State	Zip	Phone	CreditLimit	AcctRepNo
86	208	Sed Incorporated	non enim commodo	Mesa	AZ	85683	(253) 824-2567	5748.00	1005
87	209	Mi Felis Adipiscing Institute	urna. Ut tincidunt	Aurora	CO	58786	(683) 961-7908	1921.00	1005
88	210	Suspendisse Aliquet Foundation	lectus pede, ultrices	Greet Falls	MT	25280	(701) 010-0274	6933.00	1005
89	211	Odio Ltd	ultrices, mauris ipsum	San Diego	CA	91169	(567) 294-7123	1172.00	1005
90	212	Tempor Erat Neque Foundation	nonummy ac, feugiat	Athens	GA	21189	(871) 713-1145	4677.00	1005
91	213	Venenatis Lacus PC	faucibus orci luctus	Jacksonville	FL	40925	(118) 159-7778	7797.00	1005
92	214	Sed Hendrerit Consulting	Aenean massa. Integer	Atlanta	GA	56250	(101) 491-3377	5822.00	1005
93	215	Scelerisque Neque Sed Corp.	Phasellus nulla. Integer	Grand Island	NE	74930	(094) 473-1079	7702.00	1005
94	216	Eget Tincidunt Dui Corp.	sit amet nulla.	Pocatello	ID	34925	(042) 920-9161	4797.00	1005
95	217	Donec Nibh Enim Inc.	vitae diam. Proin	New Orleans	LA	19441	(903) 810-7696	7770.00	1005
96	218	Est Nunc Laoreet Limited	nascetur ridiculus mus.	Cedar Rapi...	IA	48733	(004) 283-8845	6205.00	1005
97	219	Rhonus Id Institute	luctus aliquet odio.	Dover	DE	32959	(291) 882-9381	6033.00	1005
98	220	Nibh Aliquam Ornare Corp.	felis. Nulla tempor	Cheyenne	WY	80943	(334) 533-0007	3540.00	1005
99	221	Sed Hendrerit A Inc.	et risus. Quisque	Spokane	WA	21242	(056) 662-4522	78.00	1005
100	222	Nunc Ltd	nec urna suscipit	Chandler	AZ	86237	(273) 381-9699	1320.00	1005

(100 rows affected)

Completion time: 2019-09-10T03:16:48.0741304-04:00

The top 100 rows are retrieved successfully. After experimenting with these two operations we decided to compare them and find out - what is the difference?

The difference between the **tablesample** (50%) and the **top 100** operations is that the **tablesample** always selects different sample of data and returns specific percentage of that data, whereas the **top 100** always returns same top 100 records from the number of rows specified (1000 in our case). It is important to note that adding “repeatable” to the **tablesample** allows the DB user to retrieve same data over and over again.

2) Use of INTERSECT and EXCEPT.

In the next example we use **INTERSECT** operation and compare it with a left join operation.

We decided to experiment using two tables that are connected with each other through the relationship of the **customer** table primary key – **CustNo**. The two tables are **Customer** and **CustomerOrder**. First, we ran the select join operation:


```
SELECT Customer.CustNo, CustomerOrder.CustNo FROM Customer
LEFT JOIN CustomerOrder ON Customer.CustNo
= CustomerOrder.CustNo;
```

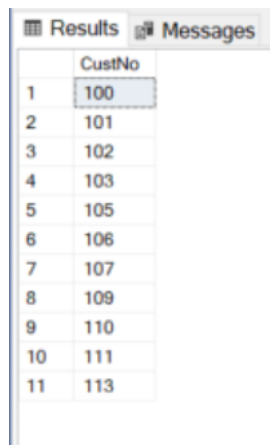
The operation retrieves two columns-- with the key value **CustNo**. We can see that the **CustomerOrder** table has way less records of **CustNo** than the **Customer** table. Hence, most of the values of the second columns are null:

	CustNo	CustNo
1	100	100
2	100	100
3	100	100
4	100	100
5	101	101
6	101	101
7	101	101
8	101	101
9	101	101
10	102	102
11	102	102
12	102	102
13	103	103
14	103	103
15	103	103

	CustNo	CustNo
10...	1458	NULL
10...	1459	NULL
10...	1460	NULL
10...	1461	NULL
10...	1462	NULL
10...	1463	NULL
10...	1464	NULL
10...	1465	NULL
10...	1466	NULL
10...	1467	NULL
10...	1468	NULL
10...	1469	NULL
10...	1470	NULL
10...	1471	NULL
10...	1472	NULL

INTERSECT, on the other hand, as it is supposed, only retrieves identical values that both the tables have and in our case shows the output with only 11 records:

```
|SELECT CustNo FROM customer  
INTERSECT  
SELECT CustNo FROM CustomerOrder;
```



	CustNo
1	100
2	101
3	102
4	103
5	105
6	106
7	107
8	109
9	110
10	111
11	113

In the next example we use **EXCEPT** operation and compare it with the same left join operation.

As in the previous example, we join two tables **customer** and **CustomerOrder**. Again, we can see that the **CustomerOrder** table has way less records of **CustNo** than the **Customer** table.

```
SELECT CustNo FROM customer  
EXCEPT  
SELECT CustNo FROM CustomerOrder;
```

Since the **EXCEPT** operation only returns unique values from the **left** table, we expect to see a big number of records the left table is the **Customer** table, in which we have 1015 records.

```
(1004 rows affected)
```

```
Completion time: 2019-09-10T03:45:09.4469683-04:00
```

Since the 11 records from the right table are not unique, we return the 1004 records that are unique and being stored in the **Customer** table. The difference between the join operation and except is that join returns every single record of **CustNo** for both tables whereas the **EXCEPT** allows us to return the exact unique values that we have in the left table that is in our case is the **Customer** table.

3) Practical use of the DATEPART operation.

There are many cases when we want to use the **DATEPART** as it has a great variety of keywords to run a certain operation. For instance, we may not only retrieve such basic time data as year, month, or day, but also minutes, seconds, and milliseconds without complex syntax:

```
SELECT DATEPART(M, getdate()); -- current minute
SELECT DATEPART(S, getdate()); -- current second
SELECT DATEPART(MS, getdate()); -- current millisecond
```

Days, months, and years oftentimes are returned by common operations such as **select year**, **select month**, and **select day**:

```
SELECT year('2019-01-10 00:00:01.1234567 +08:00')
SELECT month('2019-01-10 00:00:01.1234567 +08:00')
SELECT day('2019-01-10 00:00:01.1234567 +08:00')
```

Same results are shown by **DATEPART**:

```
SELECT DATEPART(YY, '2019-01-10 00:00:01.1234567 +08:00');  
SELECT DATEPART(M, '2019-01-10 00:00:01.1234567 +08:00');  
SELECT DATEPART(DD, '2019-01-10 00:00:01.1234567 +08:00');
```

However, there are operations at which **DATEPART** is the most convenient and effective operation to use. For example, when you are looking for a very specific timeframe in a given time period. This is an example of finding exact nanosecond in the given timeline that is '12:00:10.1234567'

```
SELECT DATEPART(nanosecond, '12:00:10.1234567'); -- returns 123456700
```

With the help of predefined keywords we could easily return a specific nanosecond. This eventually saves memory and improves the DB user's efficiency.

4) Practical use of DATEADD operation.

In scenarios when we have to estimate a delivery date, we should use the order table –

CustomerOrder. Imagine that there is a standard delivery option that normally takes 3 days. In this example we added 3 more days to the **ship date** to show the estimated delivery date:

```
select CustNo, OrderNo, OrderDate, ShipDate, DATEADD(day, 3, ShipDate) as  
'estimated standard delivery date' from CustomerOrder;
```

	CustNo	OrderNo	OrderDate	ShipDate	estimated standard delivery date
1	100	10000	2011-05-11	2011-05-16	2011-05-19
2	100	10001	2011-06-09	2011-06-13	2011-06-16
3	101	10002	2011-07-15	2011-07-22	2011-07-25
4	100	10003	2011-07-29	2011-08-02	2011-08-05
5	102	10004	2011-08-01	2011-08-04	2011-08-07
6	101	10005	2011-08-15	2011-08-19	2011-08-22
7	102	10006	2011-08-31	2011-09-05	2011-09-08
8	103	10007	2011-09-29	2011-10-03	2011-10-06
9	100	10008	2011-10-21	2011-10-26	2011-10-29
10	105	10010	2011-10-31	2011-11-04	2011-11-07
11	101	10011	2011-11-18	2011-11-22	2011-11-25
12	102	10012	2011-11-21	2011-11-28	2011-12-01
13	103	10013	2011-12-05	2011-12-08	2011-12-11
14	105	10014	2011-12-20	2011-12-22	2011-12-25
15	106	10015	2012-01-06	2012-01-12	2012-01-15

The output shows the estimated standard delivery date as a date that has 3 additional days to the **ShipDate** for every record. For the sake of experiment, we decided to add one more similar operation. The next example demonstrates express premium delivery that only takes one day from the ship date:

```
select CustNo, OrderNo, OrderDate, ShipDate, DATEADD(day, 1, ShipDate) as
'estimated express delivery date' from CustomerOrder;
```

	CustNo	OrderNo	OrderDate	ShipDate	estimated express delivery date
1	100	10000	2011-05-11	2011-05-16	2011-05-17
2	100	10001	2011-06-09	2011-06-13	2011-06-14
3	101	10002	2011-07-15	2011-07-22	2011-07-23
4	100	10003	2011-07-29	2011-08-02	2011-08-03
5	102	10004	2011-08-01	2011-08-04	2011-08-05
6	101	10005	2011-08-15	2011-08-19	2011-08-20
7	102	10006	2011-08-31	2011-09-05	2011-09-06
8	103	10007	2011-09-29	2011-10-03	2011-10-04
9	100	10008	2011-10-21	2011-10-26	2011-10-27
10	105	10010	2011-10-31	2011-11-04	2011-11-05
11	101	10011	2011-11-18	2011-11-22	2011-11-23
12	102	10012	2011-11-21	2011-11-28	2011-11-29
13	103	10013	2011-12-05	2011-12-08	2011-12-09
14	105	10014	2011-12-20	2011-12-22	2011-12-23
15	106	10015	2012-01-06	2012-01-12	2012-01-13

5) ROW_NUMBER function use.

Imagine that there is a requirement to use a ROW_NUMBER() OVER (PARTITION BY) based on each customer within the **CustomerOrder** table. The idea is to sort by **CustNo** attribute and assign a sequence record (row_number operation) to each customer, so that every customer has its own first row, second row, third row, etc... depending on how many times it appears on the table. So, we decided to run this operation:

```
SELECT ROW_NUMBER() OVER (PARTITION BY CustNo ORDER BY CustNo)
AS orderPerCustomer, OrderNo, OrderDate, ShipDate, CustNo FROM CustomerOrder;
```

So, the output returned the new column with the sequence of the numbers as expected:

	orderPerCustomer	OrderNo	OrderDate	ShipDate	CustNo
1	1	10000	2011-05-11	2011-05-16	100
2	2	10001	2011-06-09	2011-06-13	100
3	3	10003	2011-07-29	2011-08-02	100
4	4	10008	2011-10-21	2011-10-26	100
5	1	10002	2011-07-15	2011-07-22	101
6	2	10005	2011-08-15	2011-08-19	101
7	3	10011	2011-11-18	2011-11-22	101
8	4	10019	2012-01-31	2012-02-03	101
9	5	10027	2012-05-02	2012-05-05	101
10	1	10004	2011-08-01	2011-08-04	102
11	2	10006	2011-08-31	2011-09-05	102
12	3	10012	2011-11-21	2011-11-28	102
13	1	10007	2011-09-29	2011-10-03	103
14	2	10013	2011-12-05	2011-12-08	103
15	3	10020	2012-02-20	2012-02-24	103

That way we can quickly scroll and see how many and which customers placed more than one order.