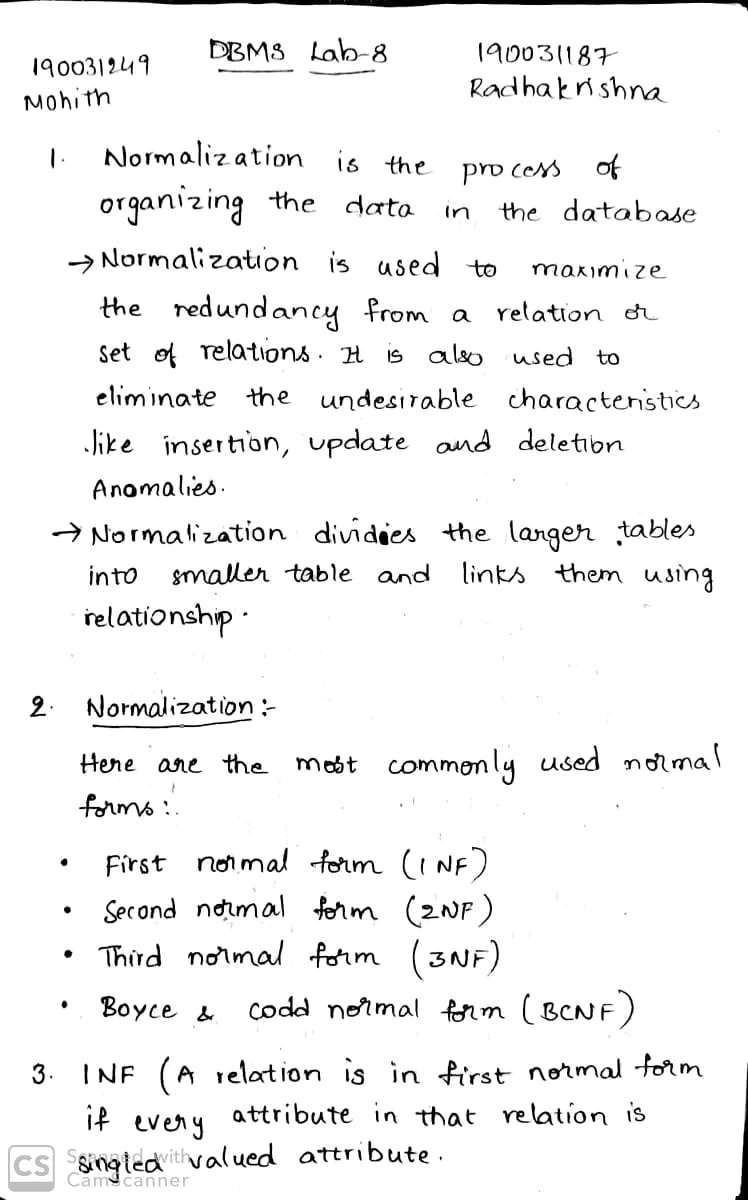
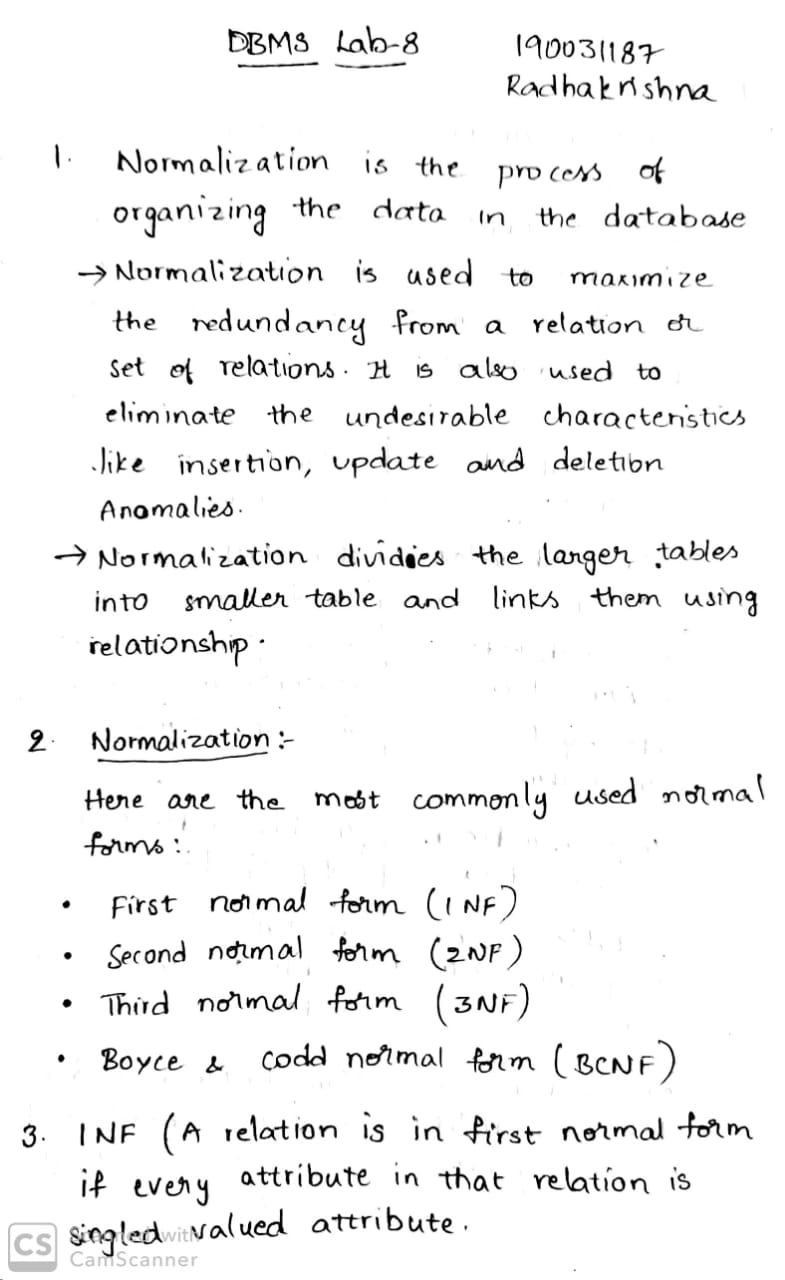
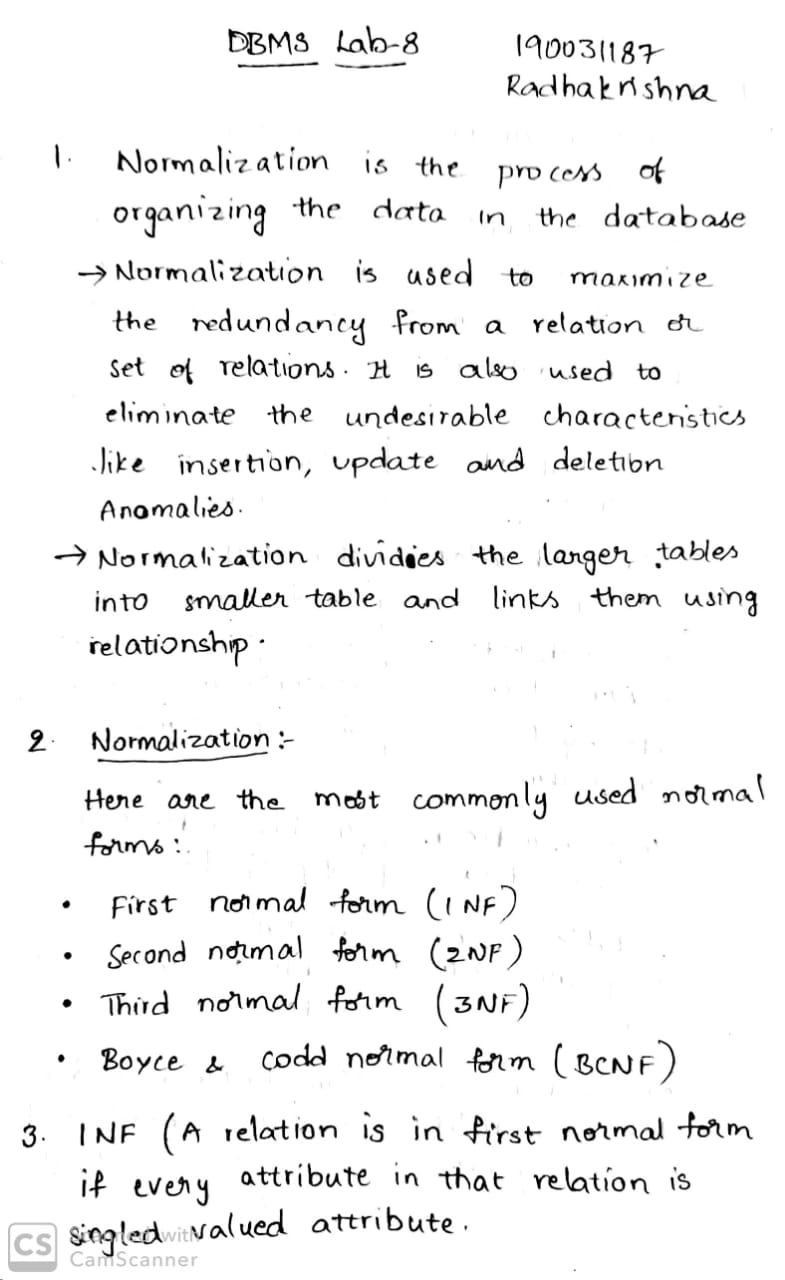
**DBMS PRACTICAL- 8 PRELAB**

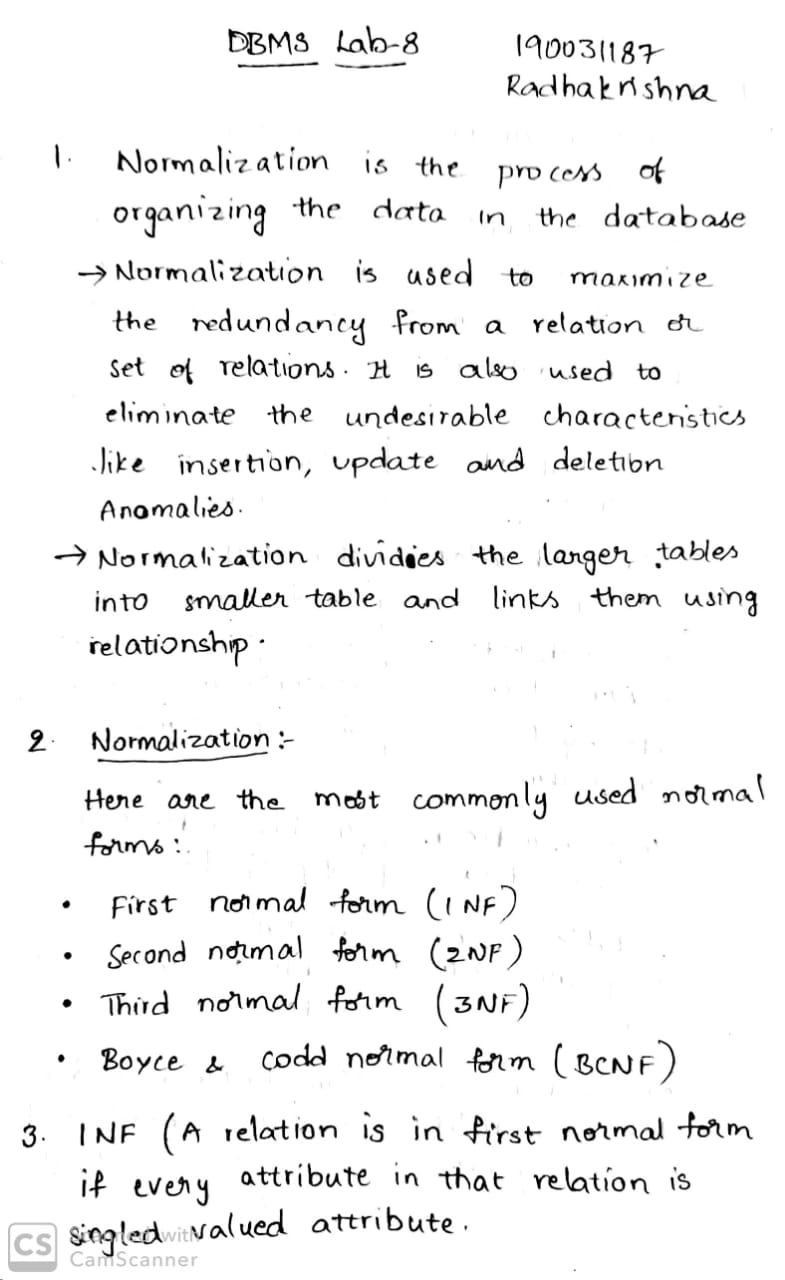
1.What is Normalization?

****

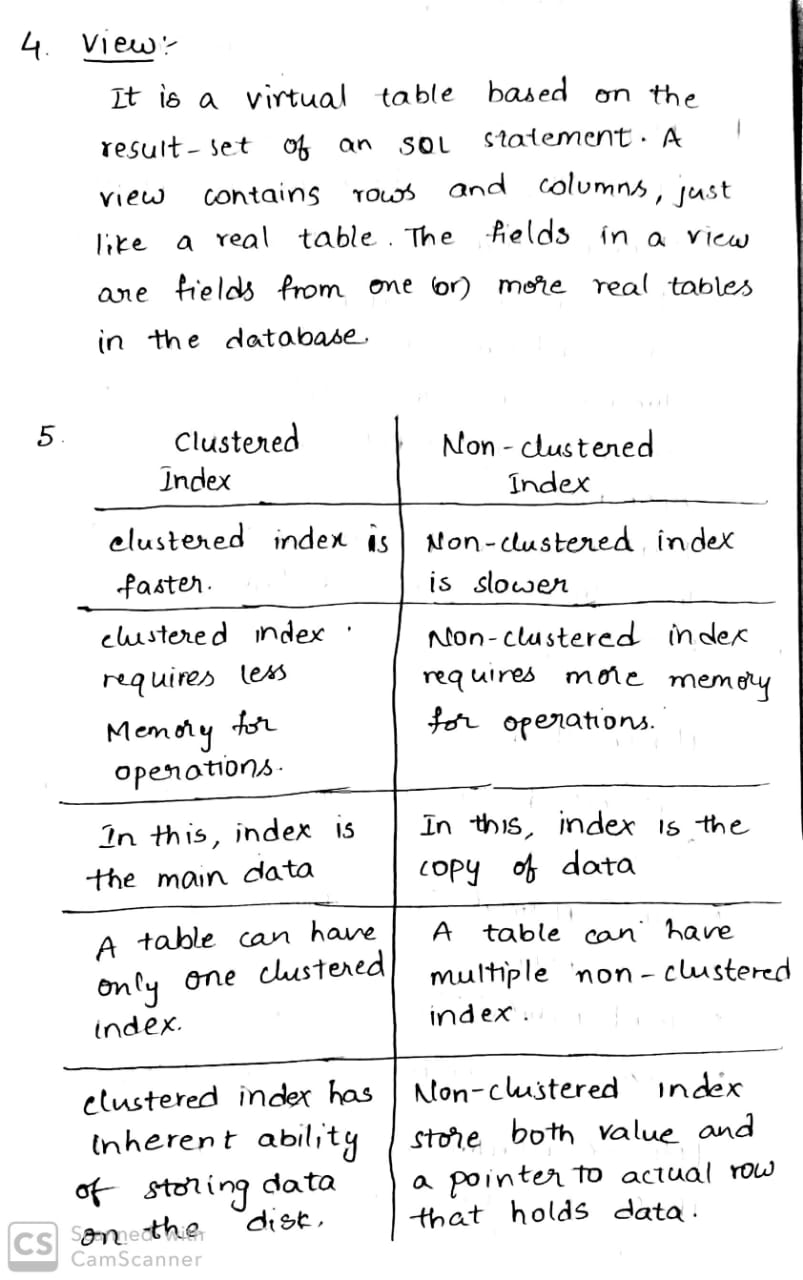
2.What are various forms of Normalization?



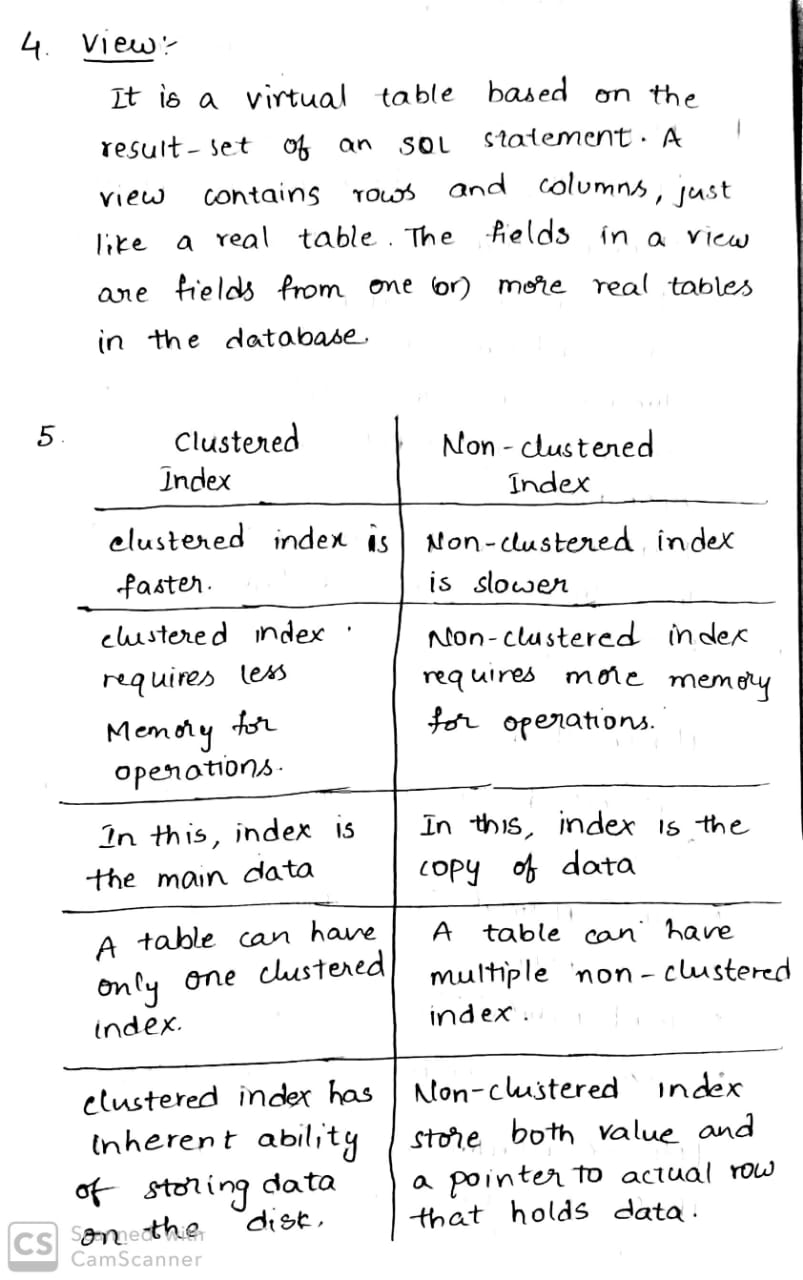
3.A table has fields F1, F2, F3, F4, F5 with the following functional dependencies   
F1 → F3   F2→ F4   (F1 . F2) → F5  
In terms of Normalization, this table is in which normal form?



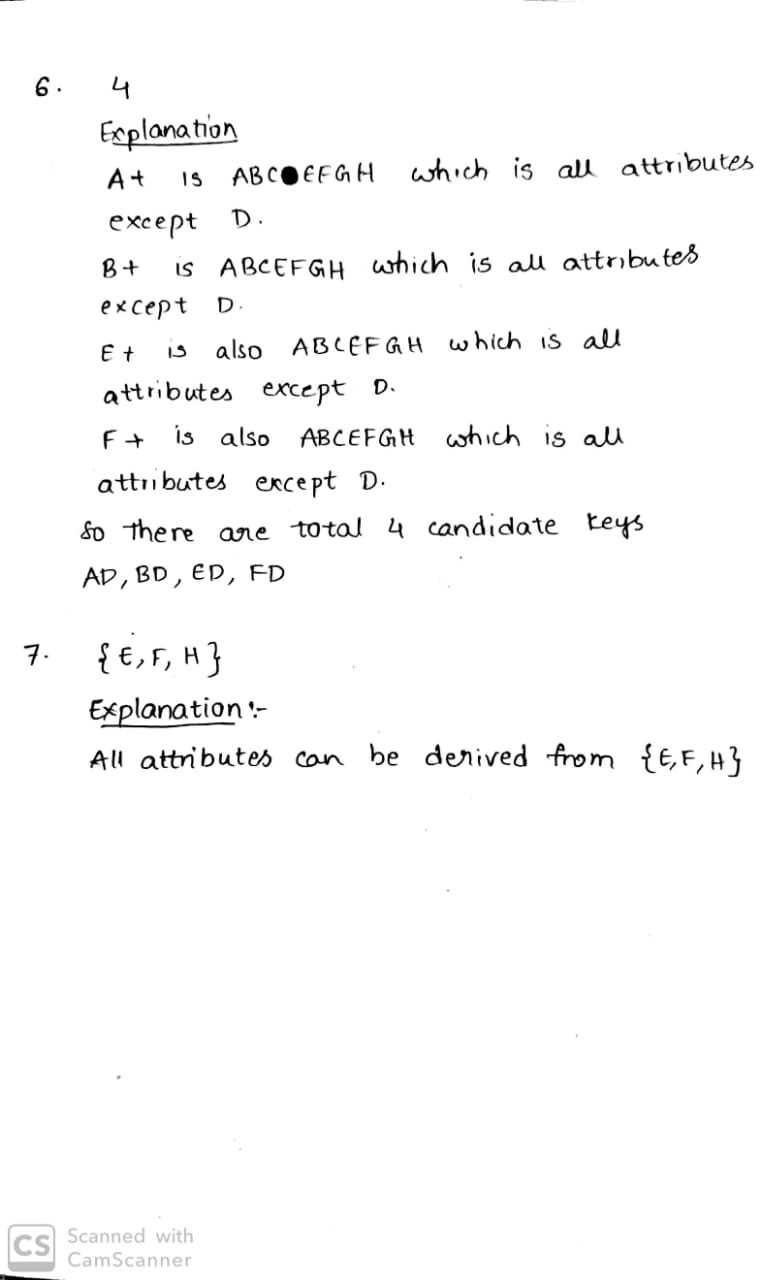
4.What is a view?



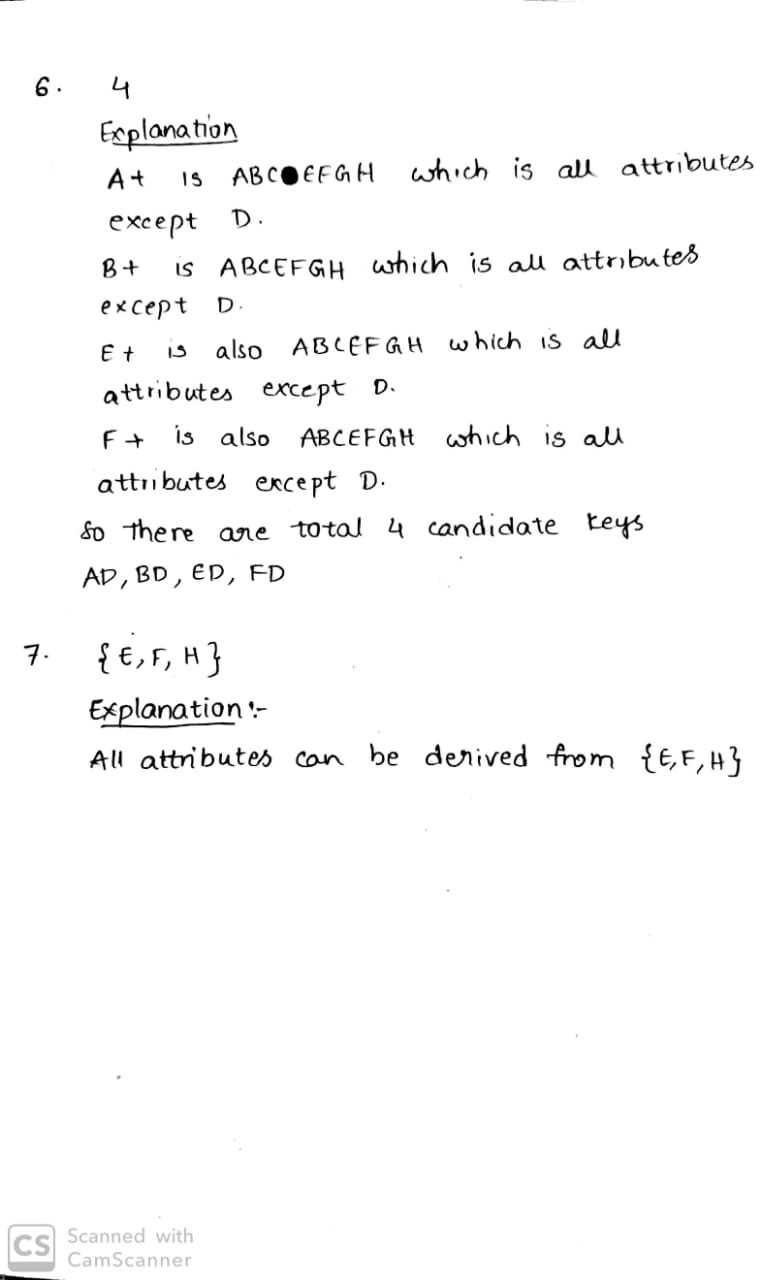
5.What is the difference between Clustered and Non-clustered index?



6.Relation R has eight attributes ABCDEFGH. Fields of R contain only atomic values. F = {CH -> G, A -> BC, B -> CFH, E -> A, F -> EG} is a set of functional dependencies (FDs) so that F+ is exactly the set of FDs that hold for R.How many candidate keys does the relation R have



7.Consider the relation scheme R = {E, F, G, H, I, J, K, L, M, M} and the set of functional dependencies {{E, F} -> {G}, {F} -> {I, J}, {E, H} -> {K, L}, K -> {M}, L -> {N} on R. What is the key for R?



**INLAB**

**Case Study 3 : WAREHOUSE SYSTEM**

## **employee** table: **manager** table:**worker** table: **phone** table:

|  |  |  |
| --- | --- | --- |
| **eno** | **area\_code** | **number** |
| 100 | 91 | 9678897435 |
| 101 | 81 | 9743523134 |
| 102 | 61 | 9808148833 |
| 103 | 91 | 9872774532 |
| 104 | 91 | 9937400231 |
| 105 | 91 | 9675453111 |
| 106 | 81 | 9413505991 |
| 107 | 81 | 9151558871 |
| 108 | 91 | 8889611751 |
| 109 | 91 | 8627664631 |

|  |  |
| --- | --- |
| **eno** | **ename** |
| 100 | Hari |
| 101 | Giri |
| 102 | Arun |
| 103 | Verma |
| 104 | Jaya |
| 105 | Kalyan |
| 106 | Krishna |
| 107 | Mohan |
| 108 | Bhasker |
| 109 | Arjun |

|  |
| --- |
| **eno** |
| 102 |
| 103 |
| 105 |
| 107 |
| 109 |

|  |  |
| --- | --- |
| **eno** | **mno** |
| 101 | 102 |
| 104 | 103 |
| 106 | 102 |
| 108 | 105 |

|  |
| --- |
| **partno** |
| 1000 |
| 1001 |
| 1002 |
| 1003 |
| 1004 |
| 1005 |
| 1006 |
| 1007 |
| 1008 |
| 1009 |

|  |  |  |  |
| --- | --- | --- | --- |
| **eno** | **street** | **city** | **state** |
| 100 | Ameerpet | Hyderabad | Telangana |
| 101 | Raju Nagar | Guntur | Andhra Pradesh |
| 102 | Chowadavaram | Guntur | Andhra Pradesh |
| 103 | Kukatpally | Hyderabad | Telangana |
| 104 | BHEL | Hyderabad | Telangana |
| 105 | Poranki | Vijayawada | Andhra Pradesh |
| 106 | Bachupally | Hyderabad | Telangana |
| 107 | Nizampet | Hyderabad | Telangana |
| 108 | ECIL X Road | Hyderabad | Telangana |
| 109 | Benz Circle | Vijayawada | Andhra Pradesh |

## **address** table: **part** table: **subpart** table:

|  |  |
| --- | --- |
| **partno** | **assmno** |
| 1000 | 2001 |
| 1001 | 2000 |
| 1002 | 2001 |
| 1003 | 2000 |
| 1004 | 2000 |
| 1005 | 2002 |
| 1006 | 2002 |
| 1007 | 2001 |
| 1008 | 2000 |
| 1009 | 2002 |

## **warehouse** table: **bin** table: **item** table:

|  |  |  |
| --- | --- | --- |
| **partno** | **itemno** | **batchno** |
| 1000 | 8000 | 7000 |
| 1005 | 8001 | 7002 |
| 1007 | 8002 | 7003 |
| 1008 | 8003 | 7004 |
| 1009 | 8004 | 7005 |

|  |  |  |
| --- | --- | --- |
| **warehouseid** | **binno** | **capacity** |
| 5000 | 6000 | 100 |
| 5001 | 6001 | 150 |
| 5002 | 6002 | 200 |
| 5003 | 6003 | 250 |
| 5004 | 6004 | 300 |
| 5005 | 6005 | 200 |

|  |
| --- |
| **warehouseid** |
| 5000 |
| 5001 |
| 5002 |
| 5003 |
| 5004 |
| 5005 |

**batch** table:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **partno** | **batchno** | **size** | **date\_in** | **mgrno** | **warehousid** | **binno** |
| 1000 | 7000 | 10 | 13-03-2020 | 102 | 5000 | 6000 |
| 1001 | 7000 | 20 | 25-02-2020 | 103 | 5000 | 6001 |
| 1002 | 7001 | 30 | 03-07-2020 | 105 | 5000 | 6002 |
| 1003 | 7002 | 40 | 15-04-2020 | 107 | 5001 | 6002 |
| 1004 | 7002 | 50 | 20-04-2020 | 109 | 5001 | 6003 |
| 1005 | 7003 | 60 | 07-08-2019 | 102 | 5002 | 6004 |
| 1006 | 7003 | 10 | 02-02-2020 | 105 | 5003 | 6001 |
| 1007 | 7004 | 20 | 10-02-2020 | 107 | 5003 | 6000 |
| 1008 | 7005 | 30 | 17-05-2020 | 105 | 5000 | 6003 |
| 1009 | 7005 | 30 | 20-07-2020 | 105 | 5004 | 6000 |

## **current\_backorder** table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **partno** | **orgqty** | **remqty** | **bodate** | **backorder** |
| 1000 | 1000 | 500 | 10-03-2020 | arun |
| 1005 | 500 | 300 | 15-02-2020 | kiran |
| 1007 | 700 | 400 | 12-03-2020 | raju |
| 1008 | 800 | 600 | 11-04-2020 | david |
| 1009 | 600 | 400 | 20-06-2020 | eswar |

## **old\_backorder** table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **partno** | **orgqty** | **bodate** | **backorder** | **fulfilled** |
| 1000 | 1000 | 10-03-2020 | arun | yes |
| 1005 | 500 | 15-02-2020 | kiran | no |
| 1007 | 700 | 12-03-2020 | raju | yes |
| 1008 | 800 | 11-04-2020 | david | yes |
| 1009 | 600 | 20-06-2020 | eswar | yes |

1. Create all the tables required for creating the database for the Warehouse System case study.

create table employee(eno int primary key,ename varchar(10));

create table manager(eno int,foreign key(eno) references employee(eno));

create table worker(eno int,mno int,foreign key(eno) references employee(eno),foreign key(mno) references employee(eno));

create table address(eno int,street varchar(20),city varchar(20),state varchar(20),foreign key(eno) references employee(eno));

create table part(partno int primary key);

create table subpart(partno int,assmno int,foreign key(partno) references part(partno));

create table warehouse(warehouseid int primary key);

create table bin(warehouseid int,binno int primary key,capacity int,foreign key(warehouseid) references warehouse(warehouseid));

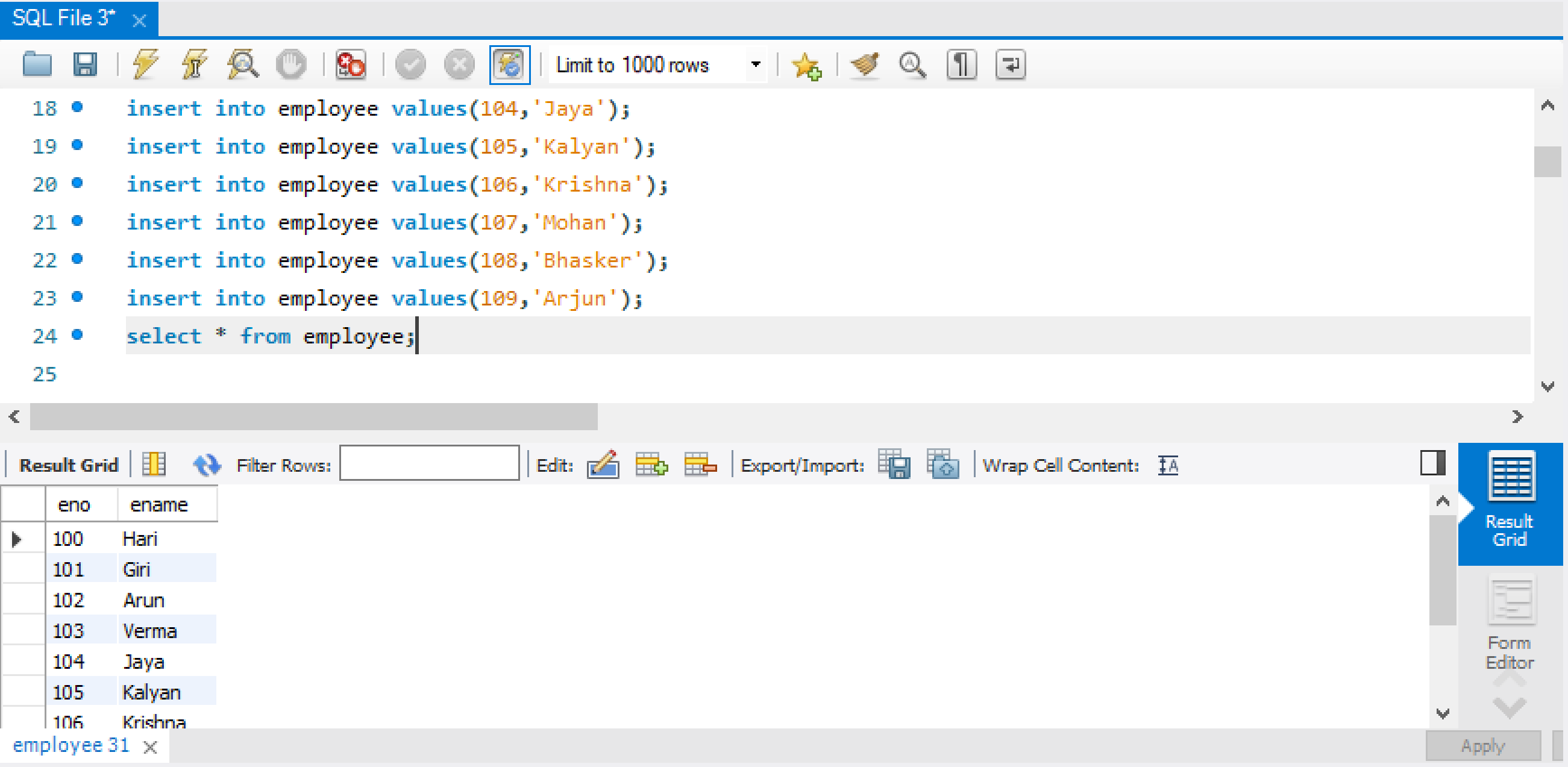
create table batch(partno int,batchno int,size int,date\_in date,mgrno int,warehouseid int,binno int,foreign key(partno) references part(partno),foreign key(mgrno) references manager(eno),foreign key(warehouseid) references warehouse(warehouseid),foreign key(binno) references bin(binno));

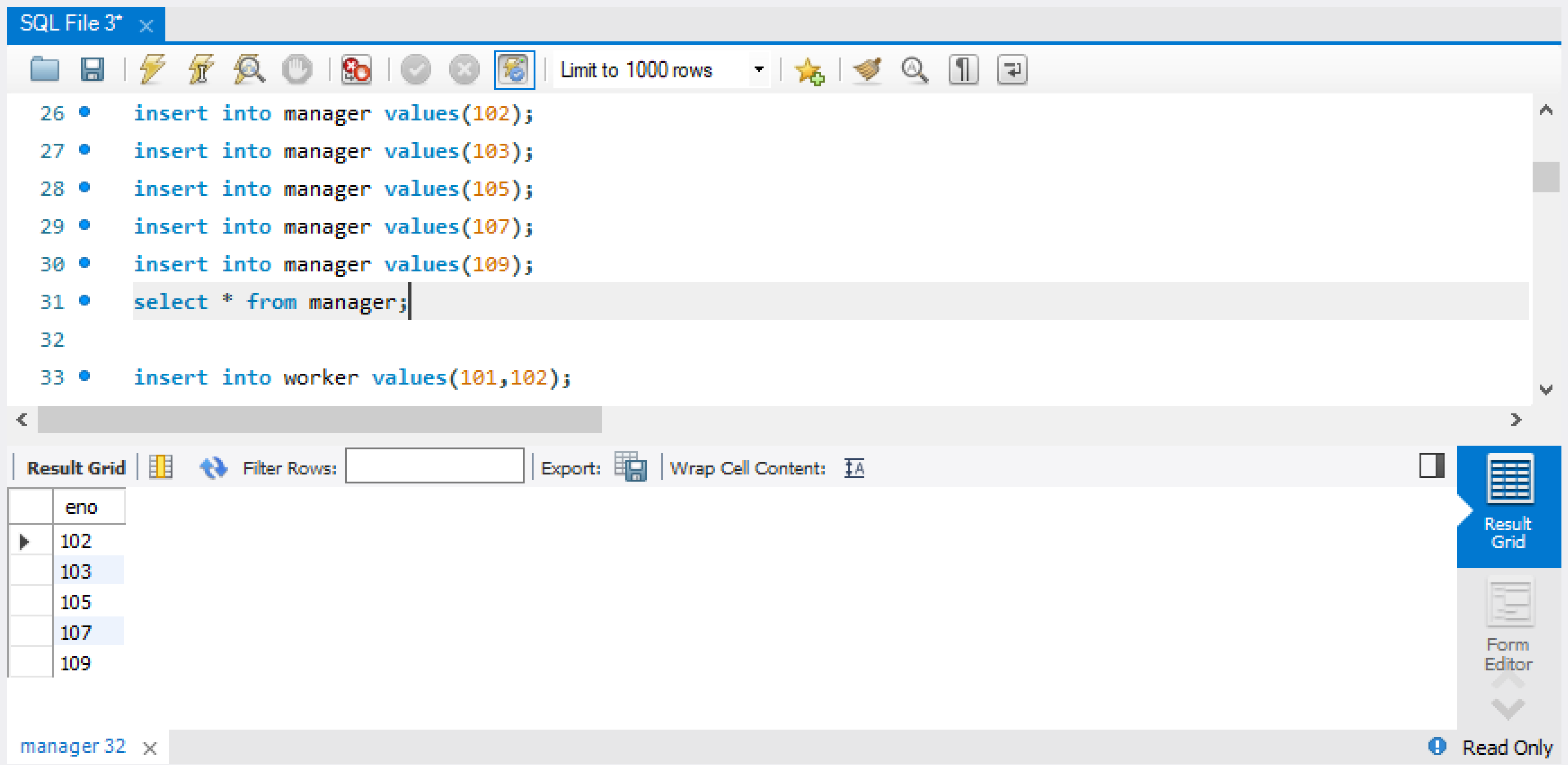
create table current\_backorder(partno int,orgqty int,remqty int,bodate date,backorder varchar(10),foreign key(partno) references part(partno));

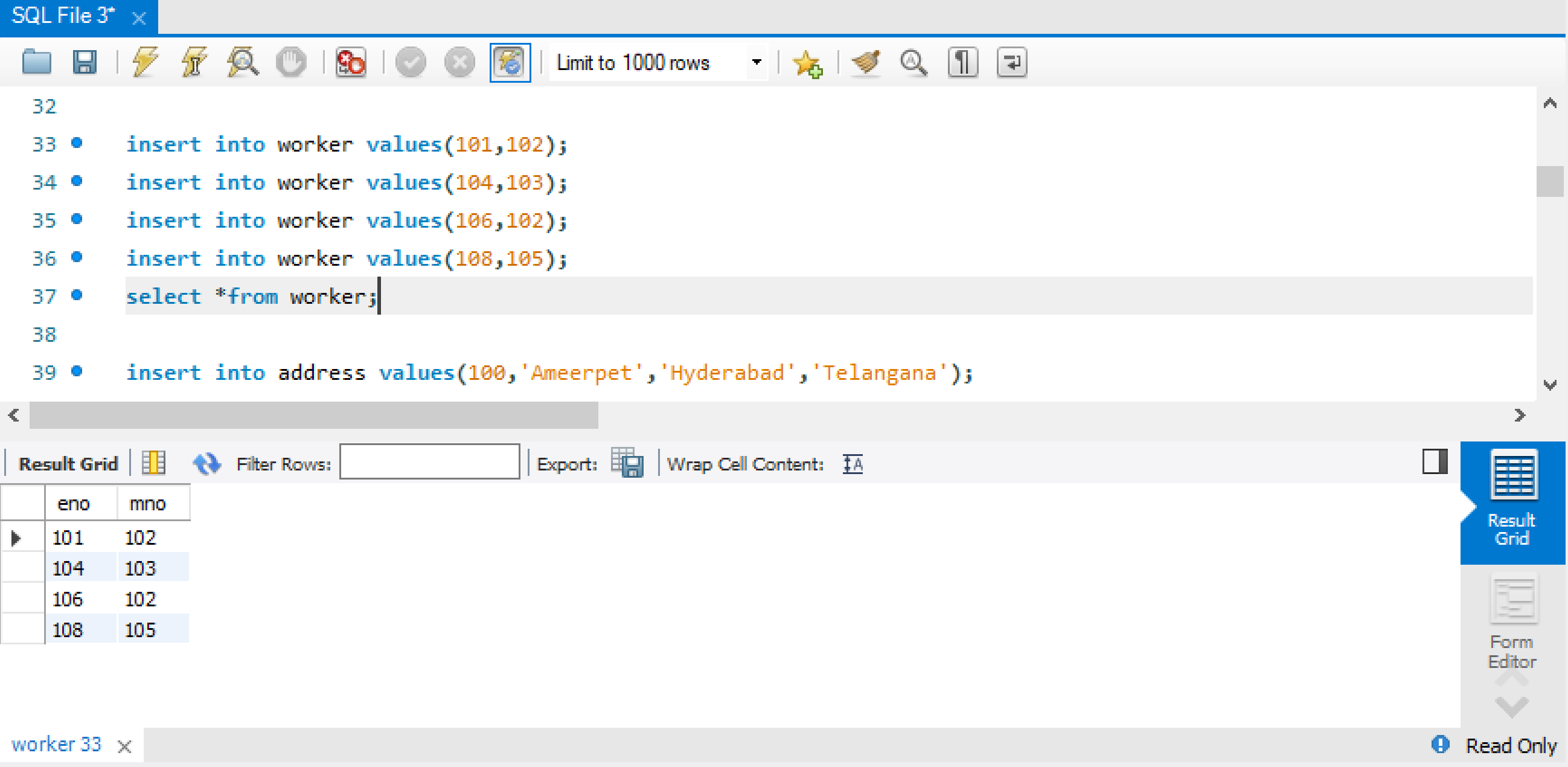
create table old\_backorder(partno int,orgqty int,bodate date,backorder varchar(10),fulfilled varchar(10),foreign key(partno) references part(partno));

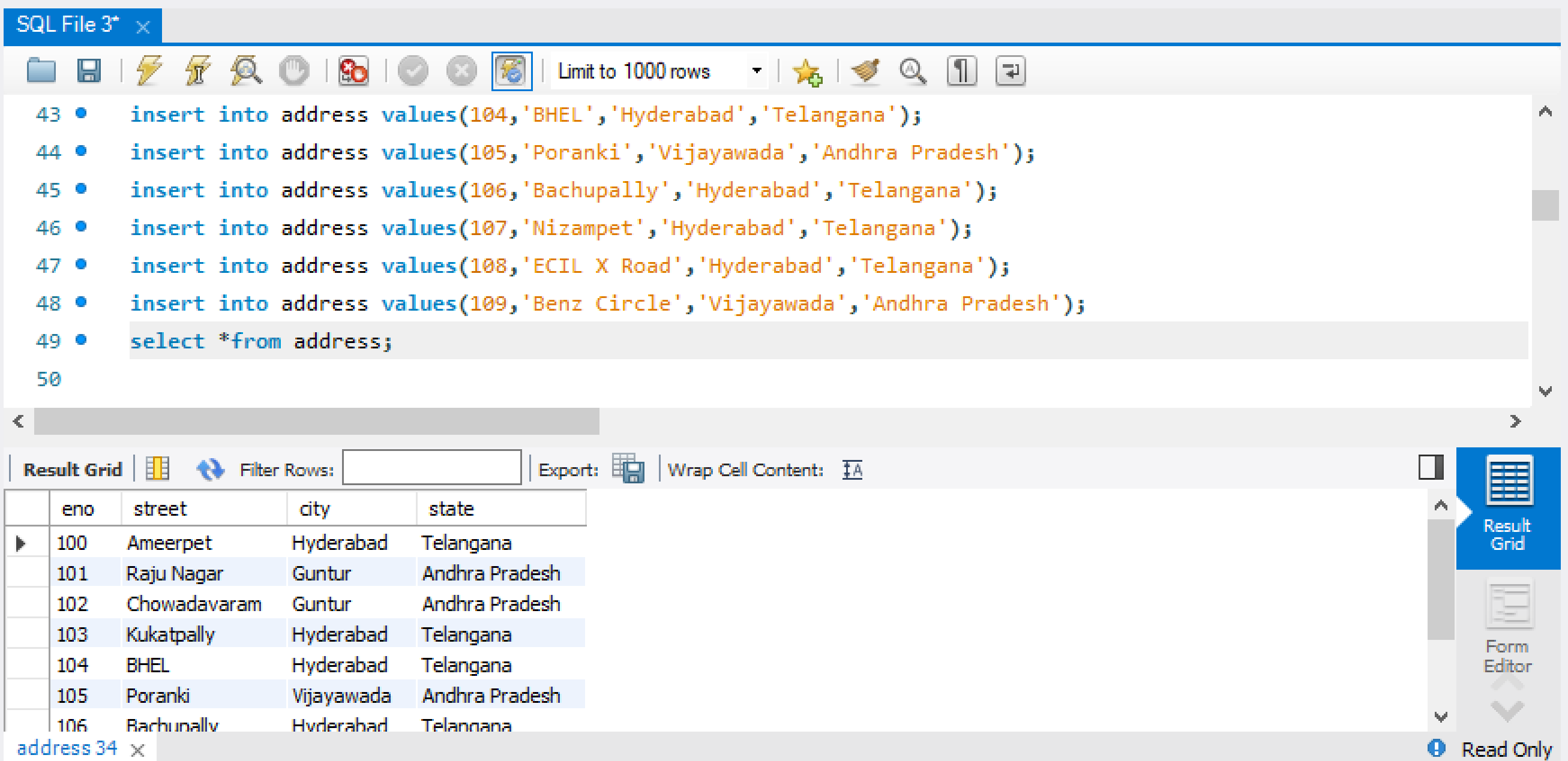
create table phone(eno int,area\_code int,number bigint,foreign key(eno) references employee(eno));

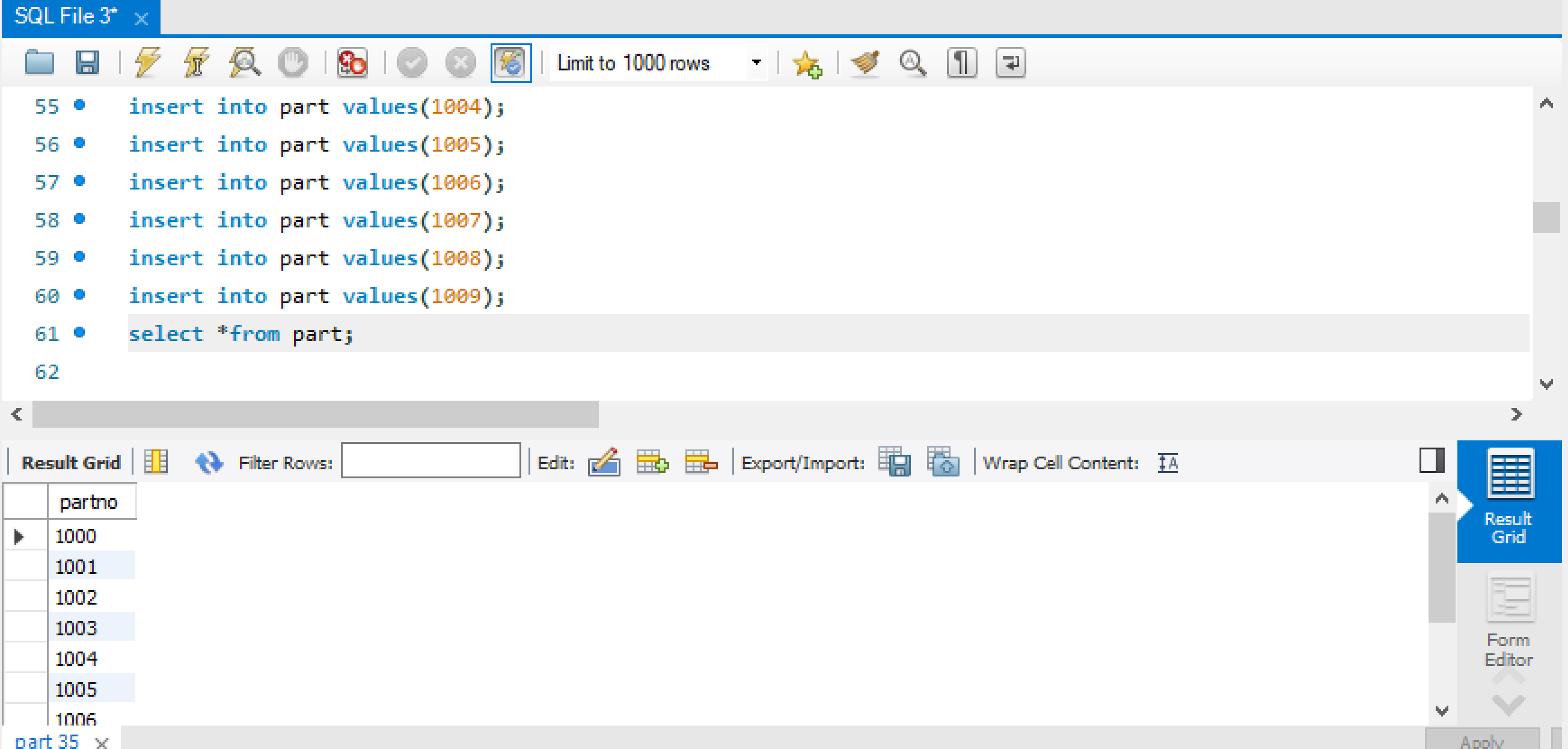
1. Insert atleast 10 records into all the tables

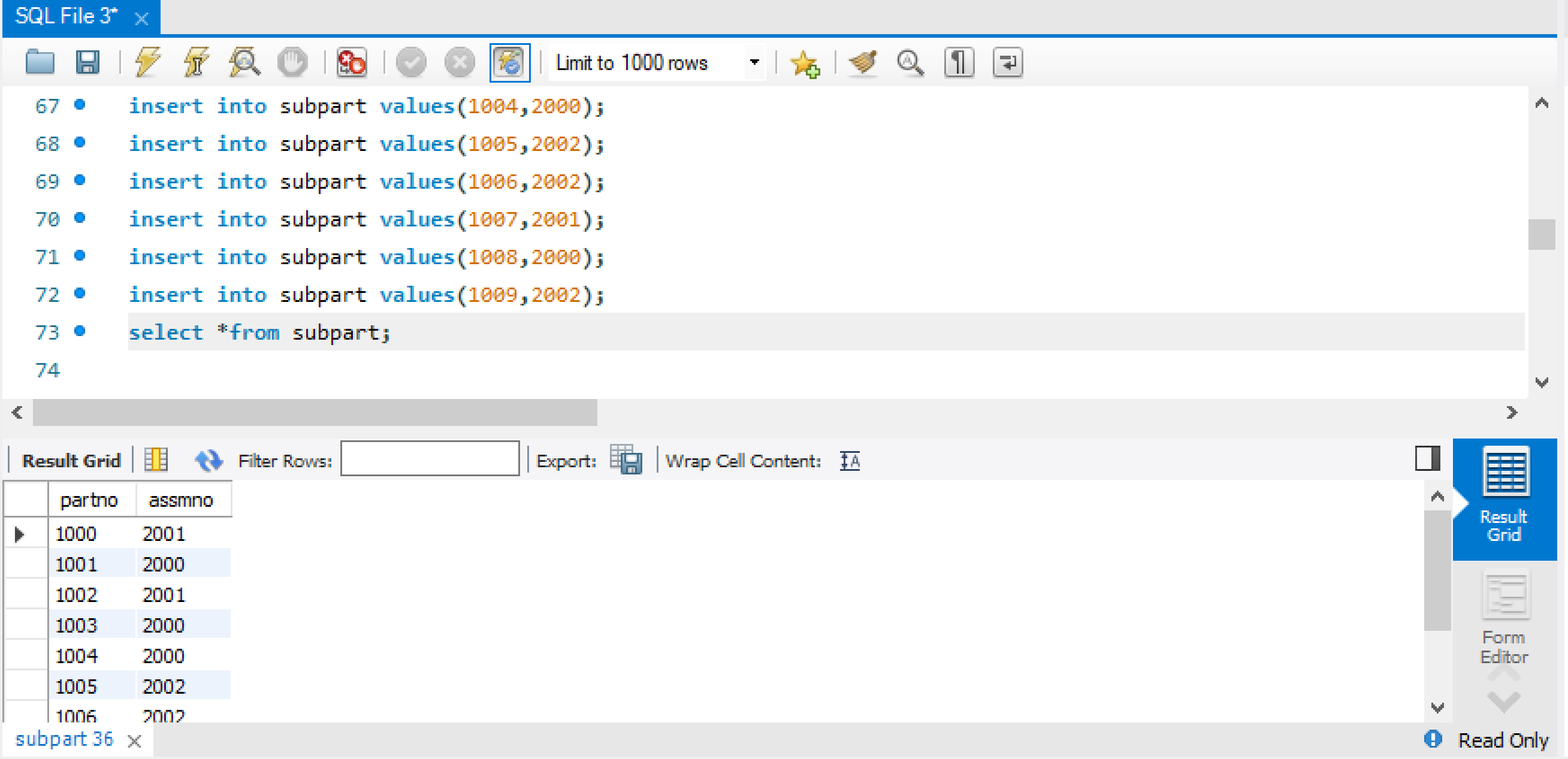


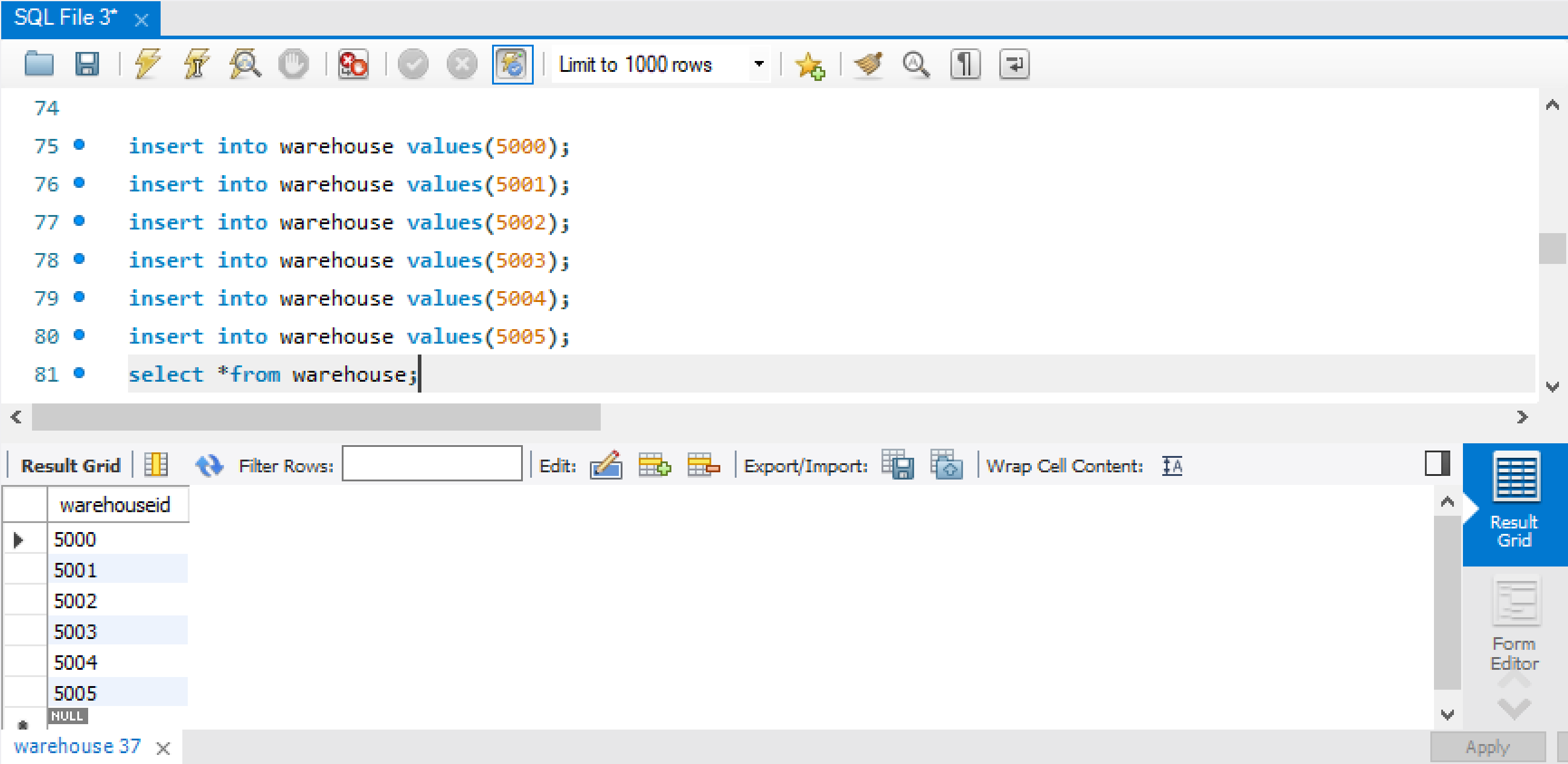


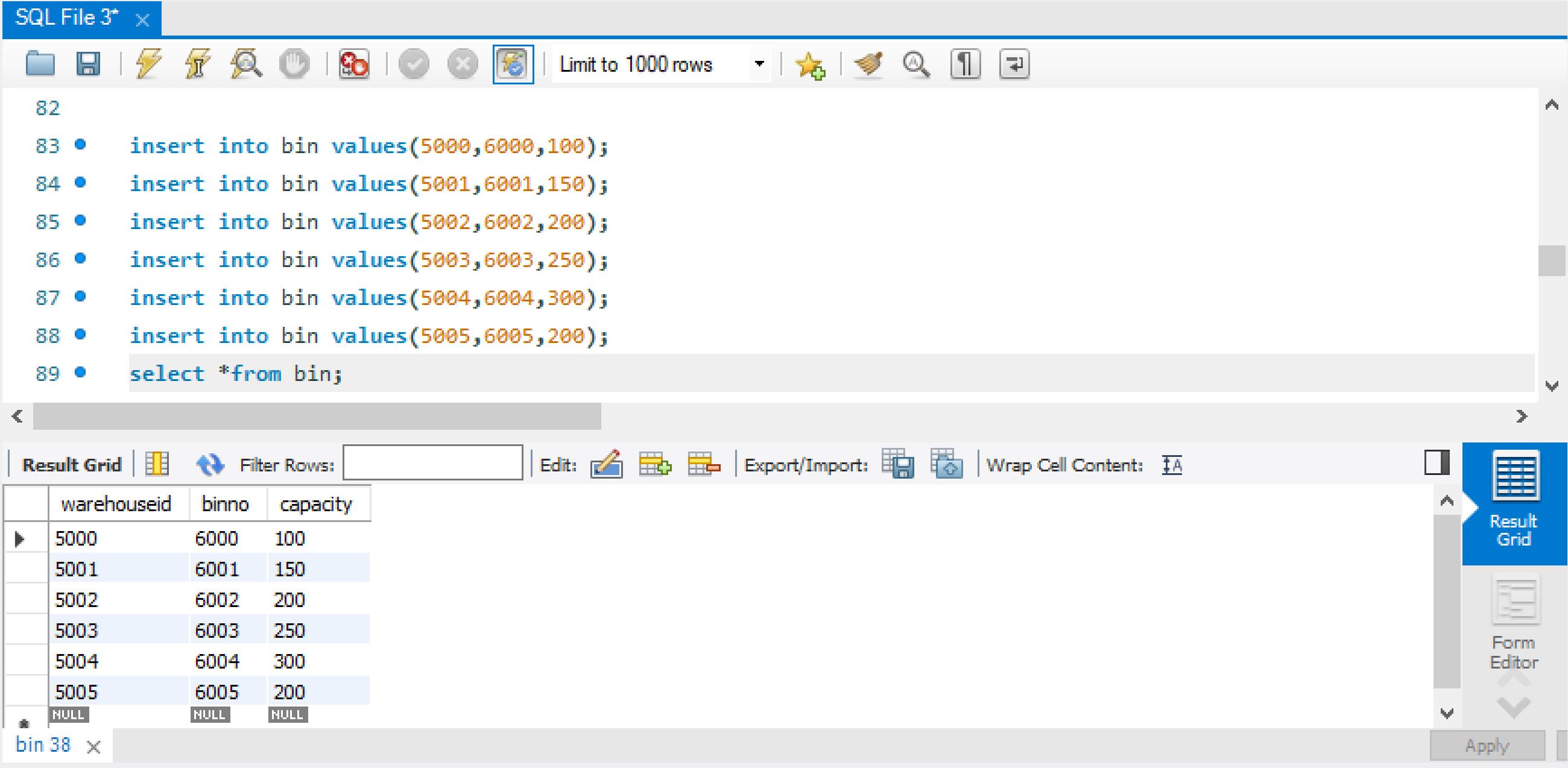


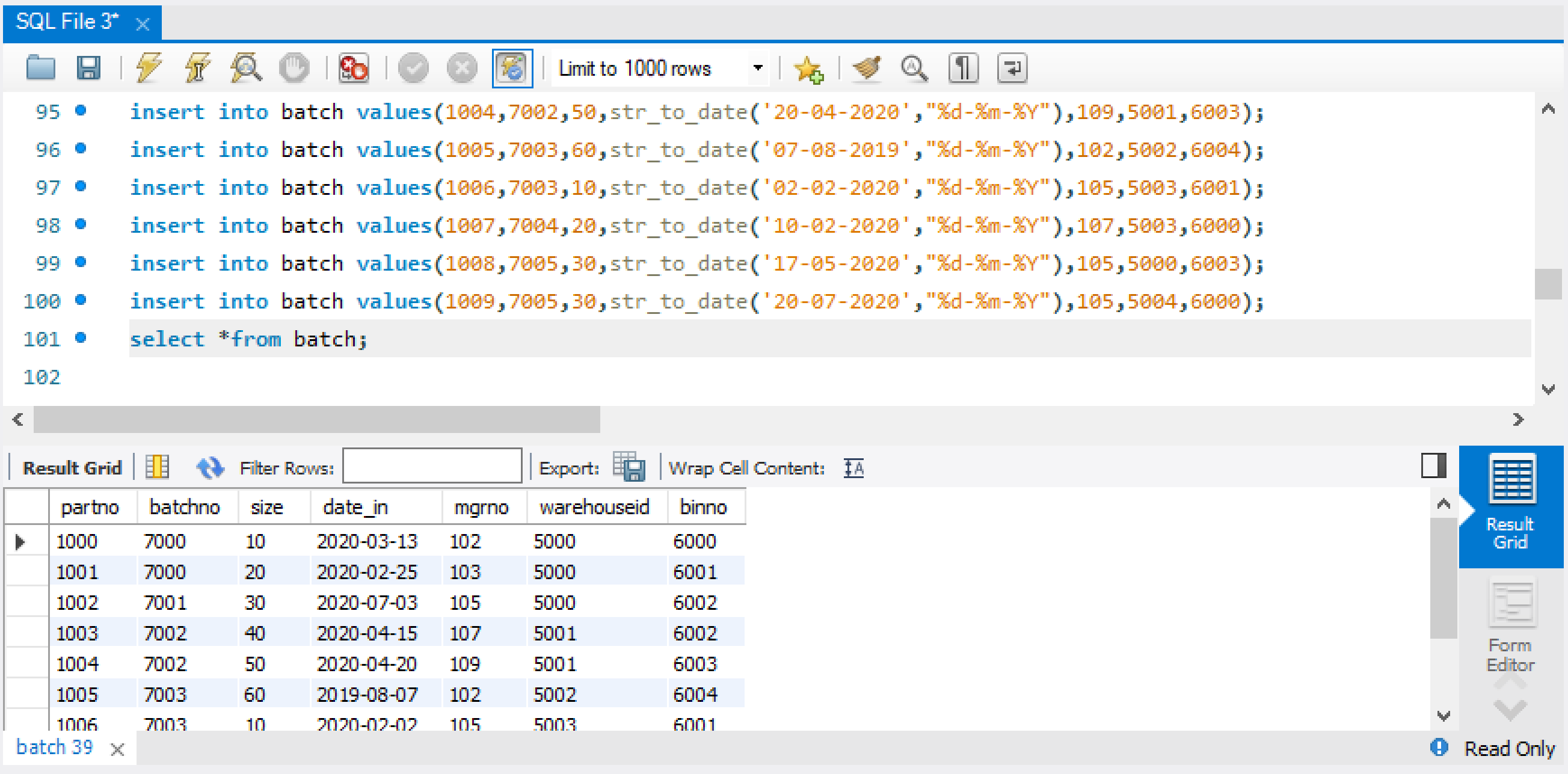


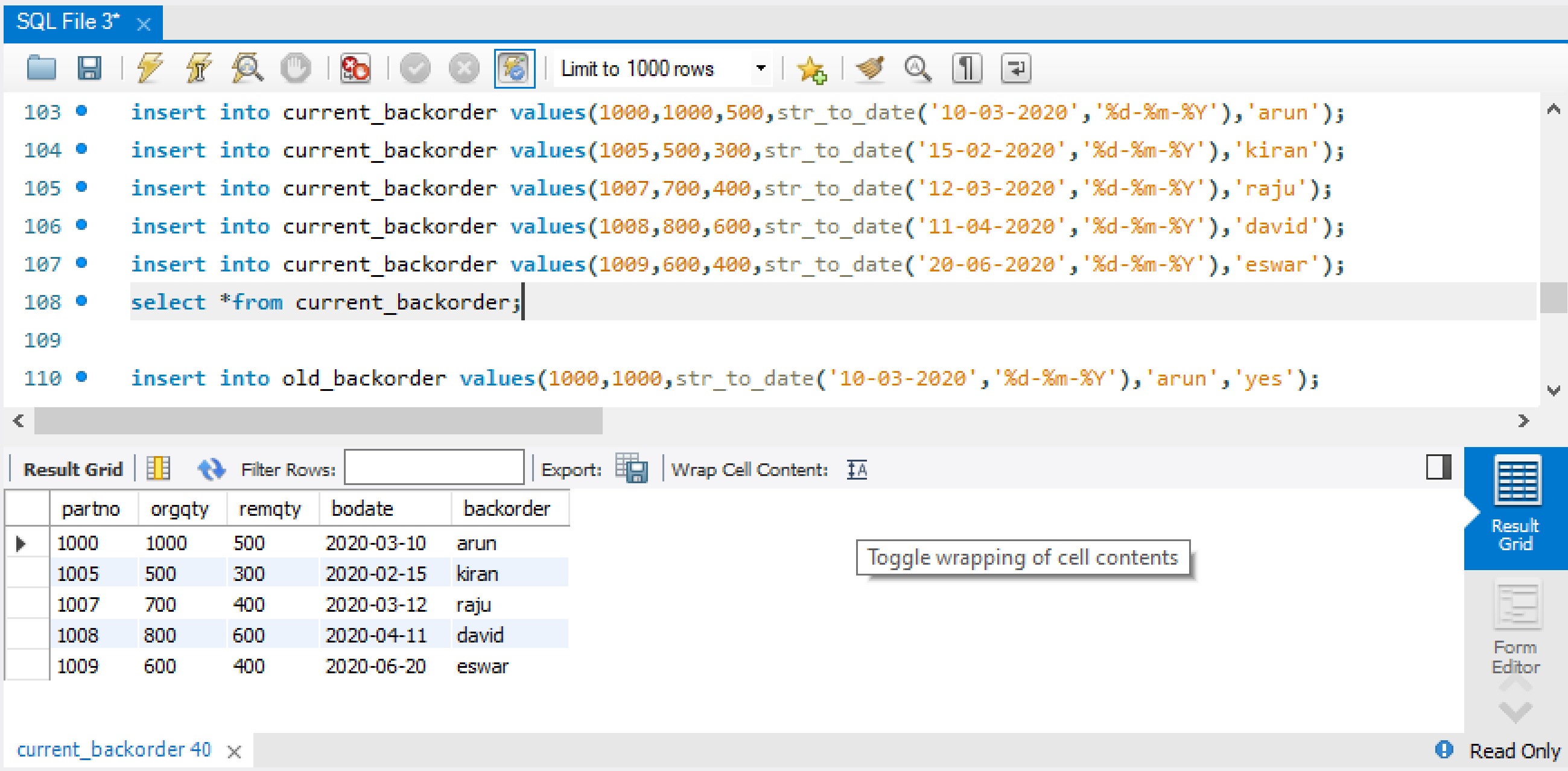


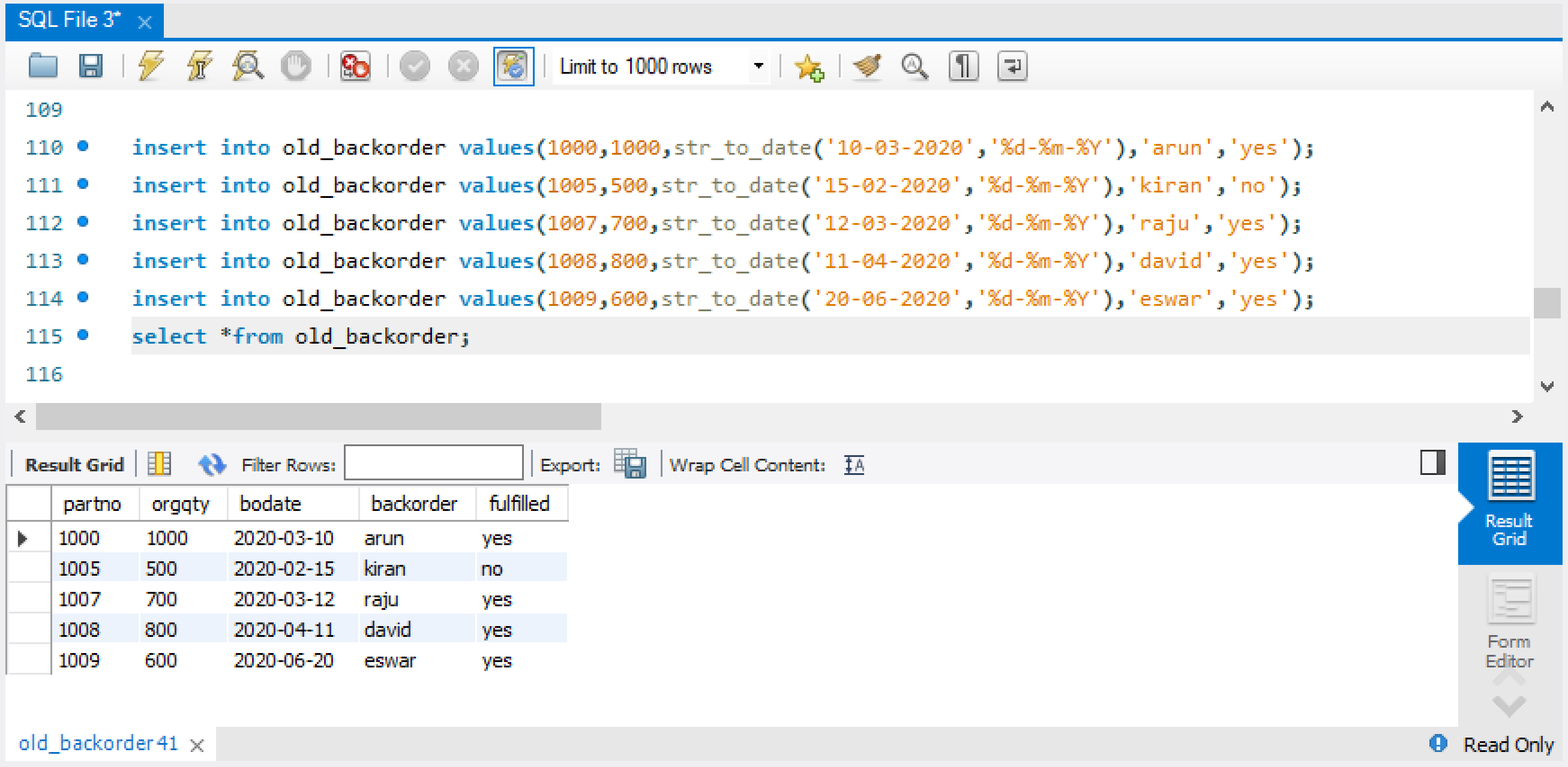


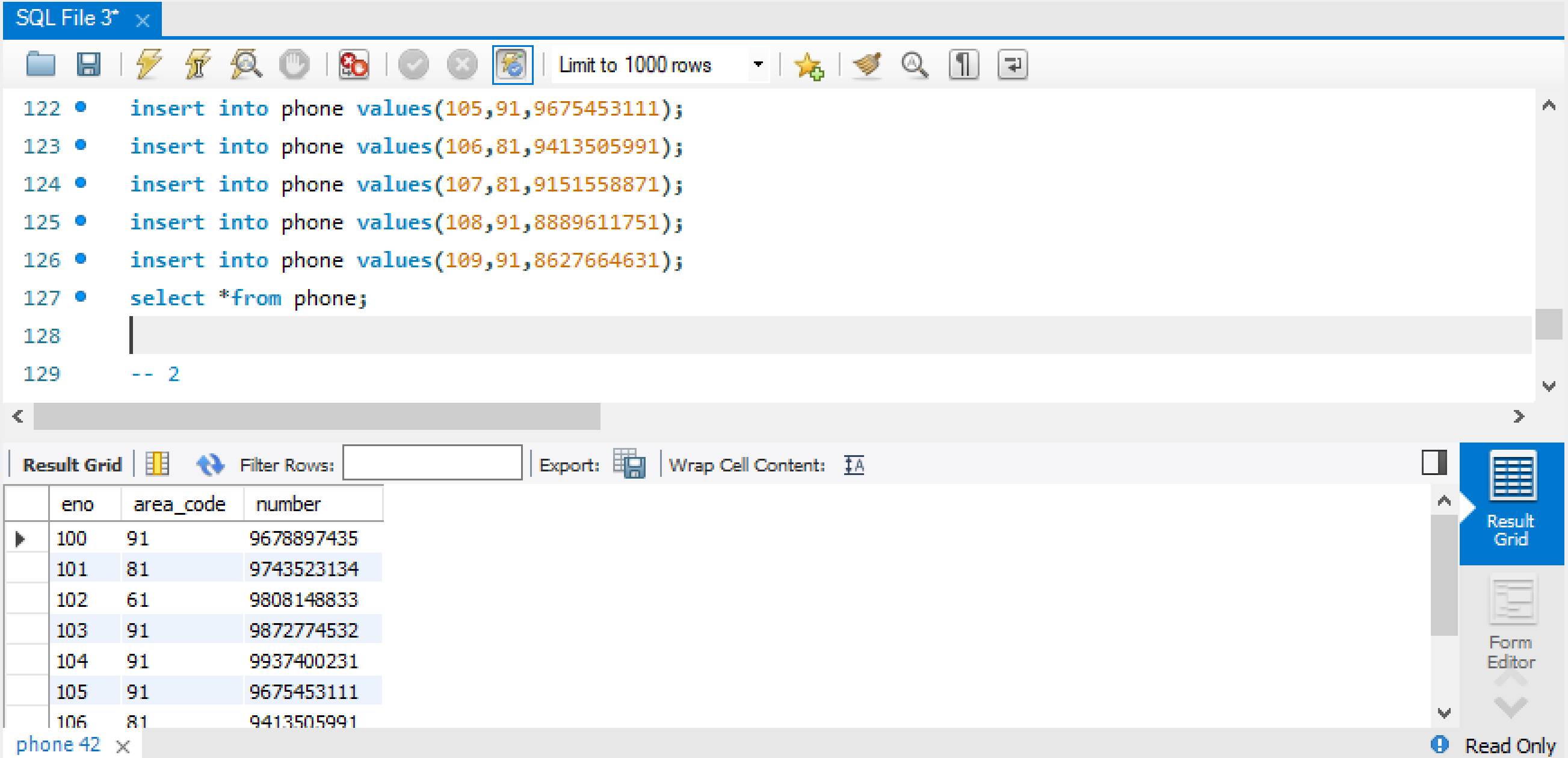






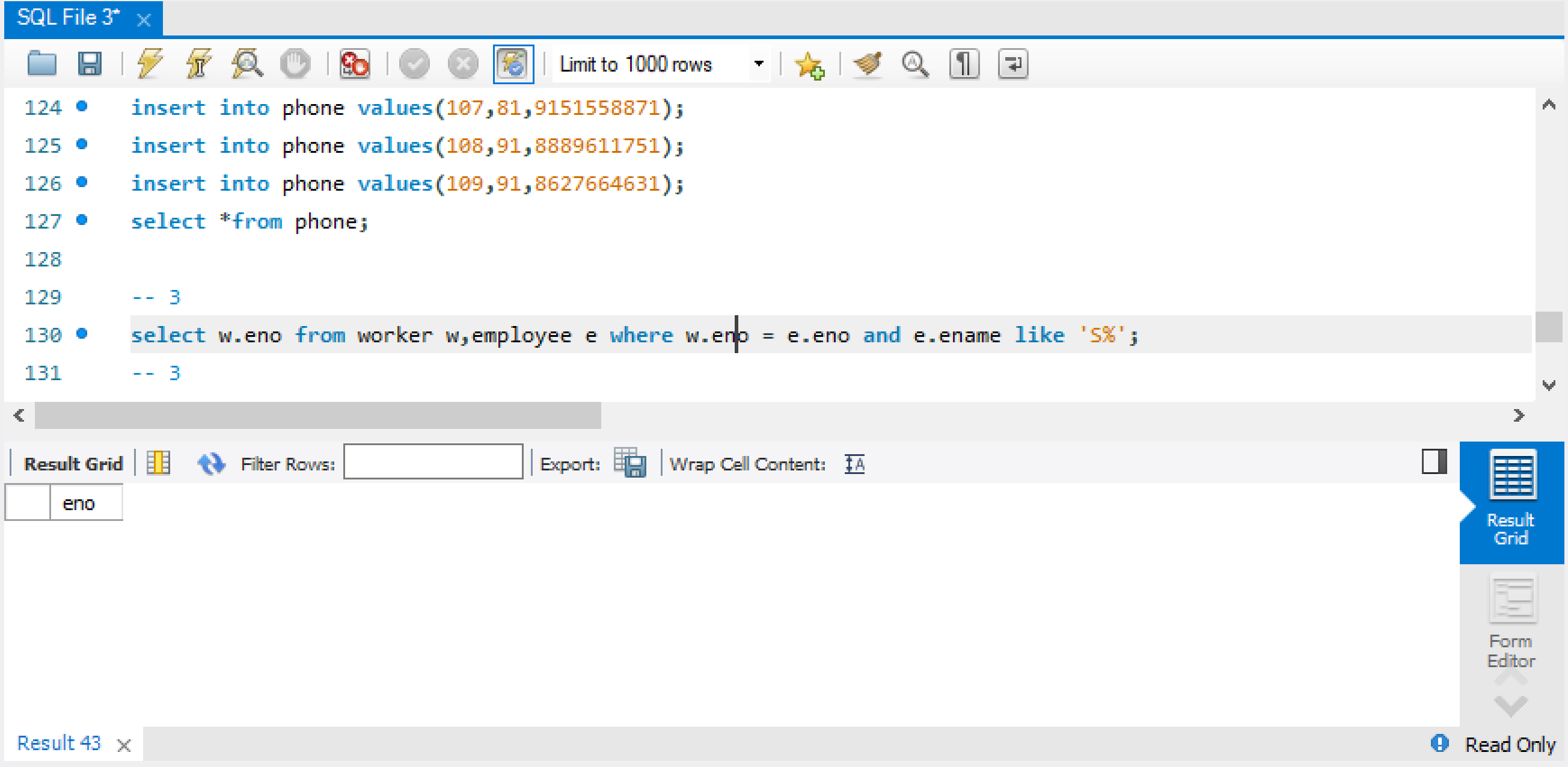






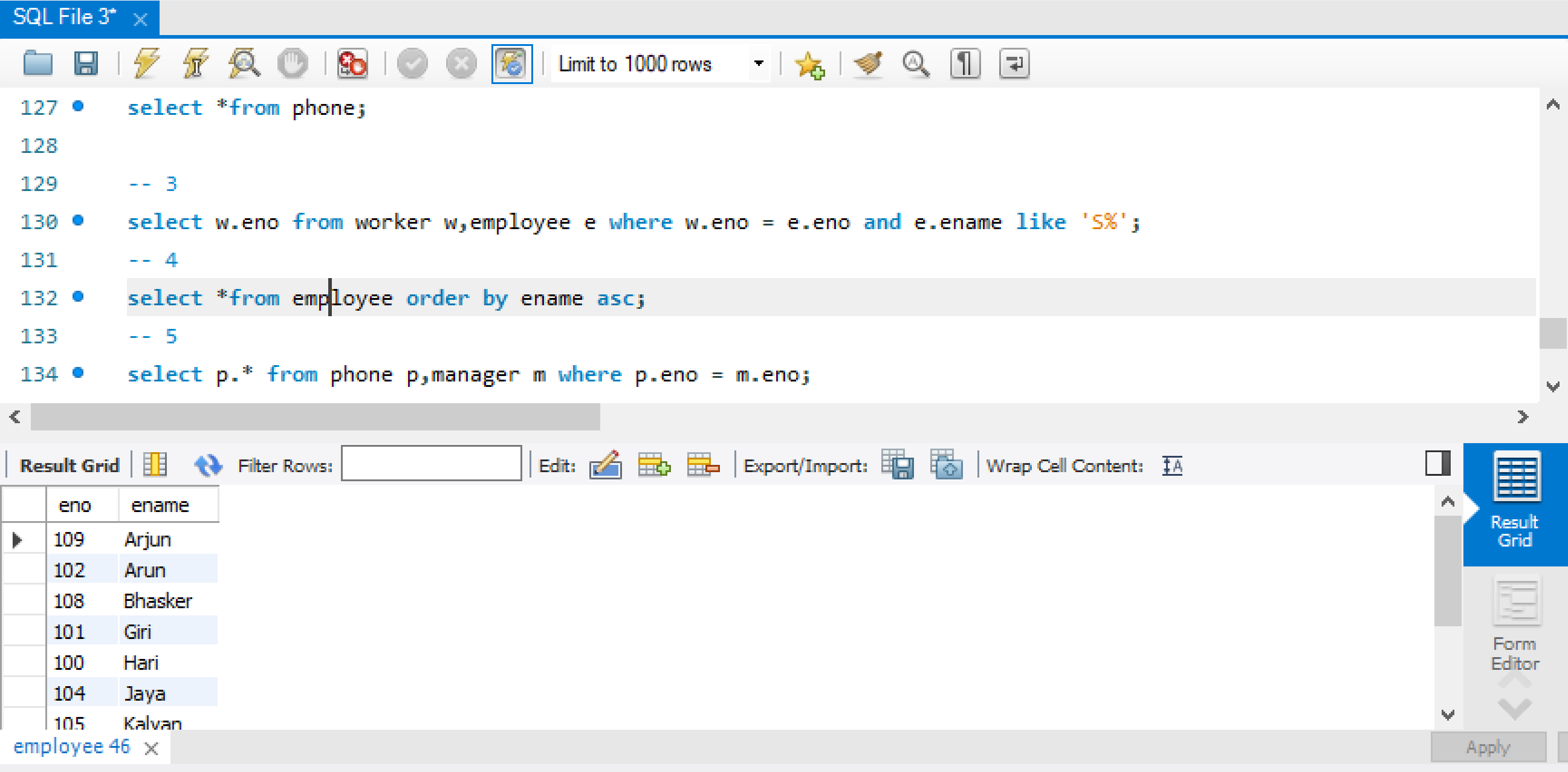
#### Display all employee numbers for all the workers that work under manager with the name starting with “S”

select w.eno from worker w,employee e where w.eno = e.eno and e.ename like 'S%';

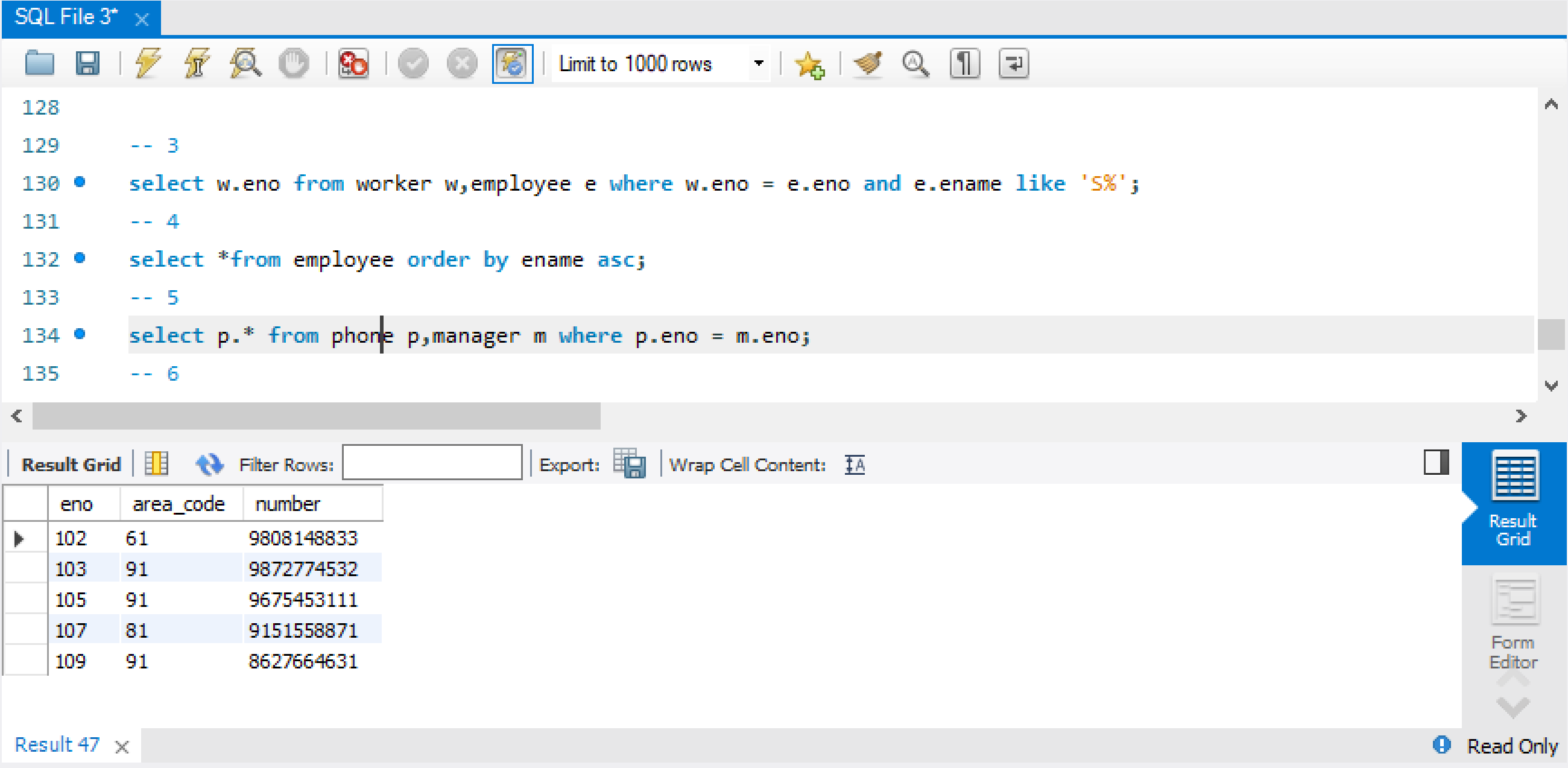


#### Display name and employee number of all employees in the ascending order on their name.The names should be listed in an alphabetic order

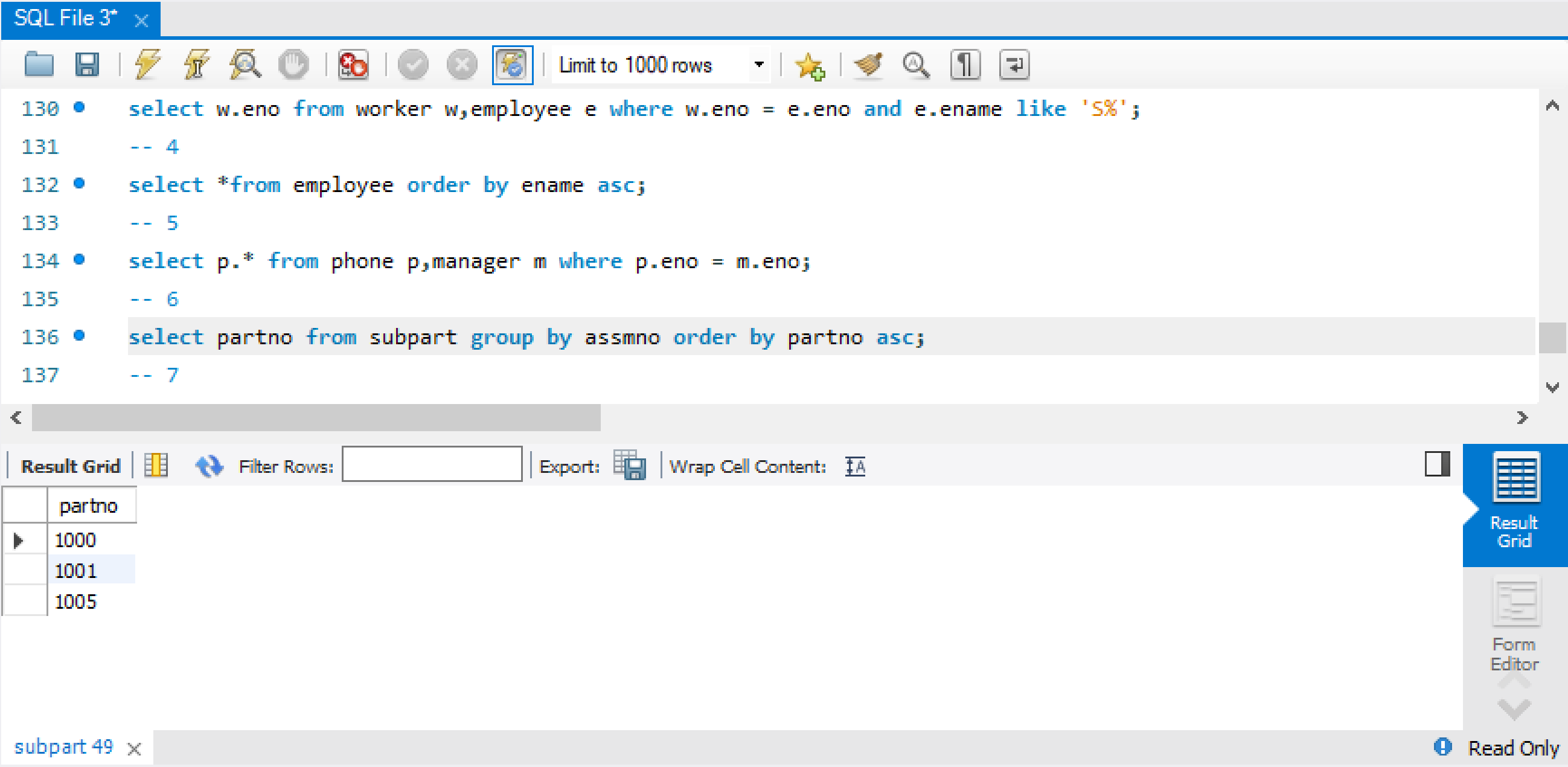
select \*from employee order by ename asc;



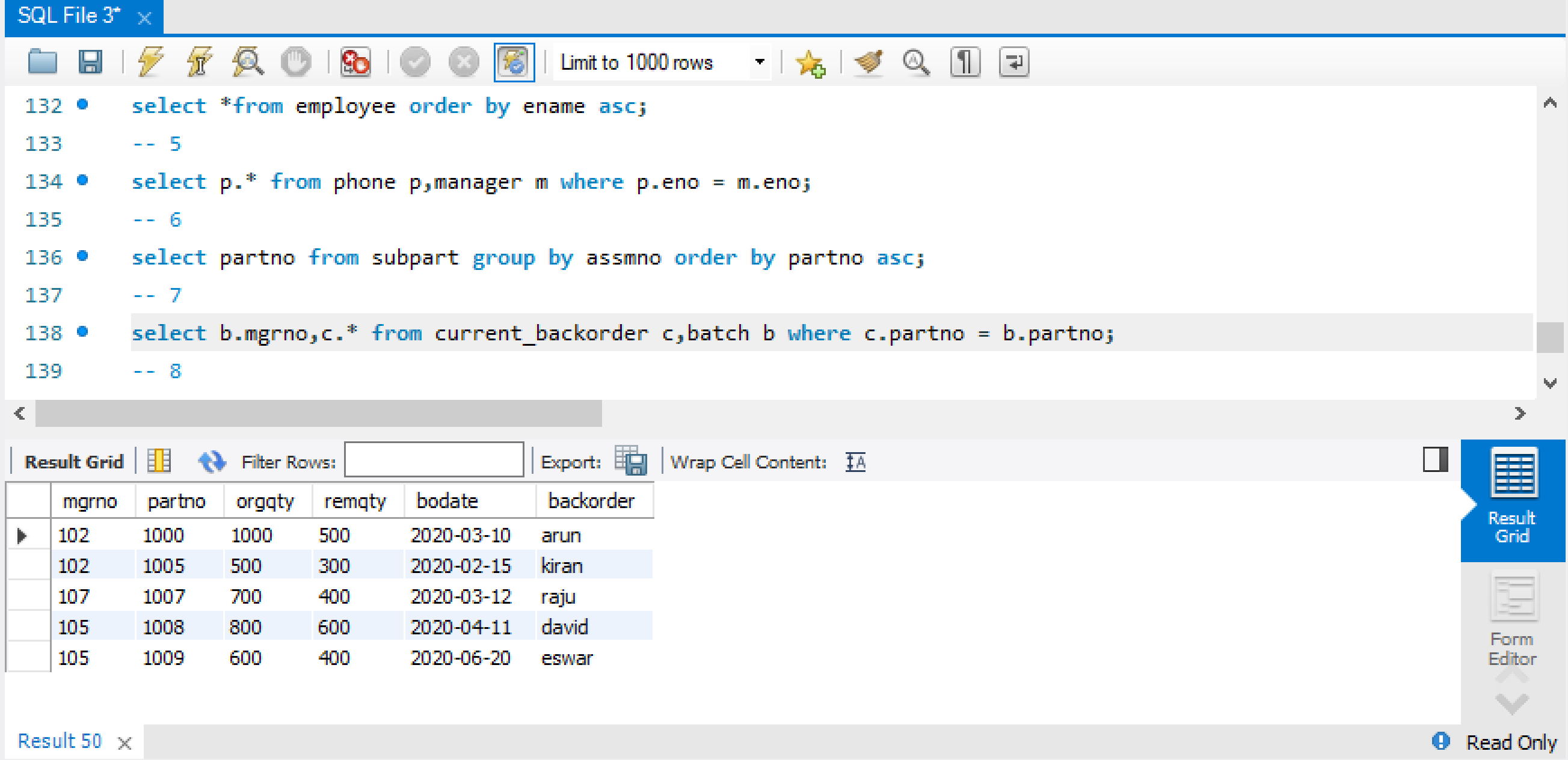
#### Display all the phones and employee number for all the managers.



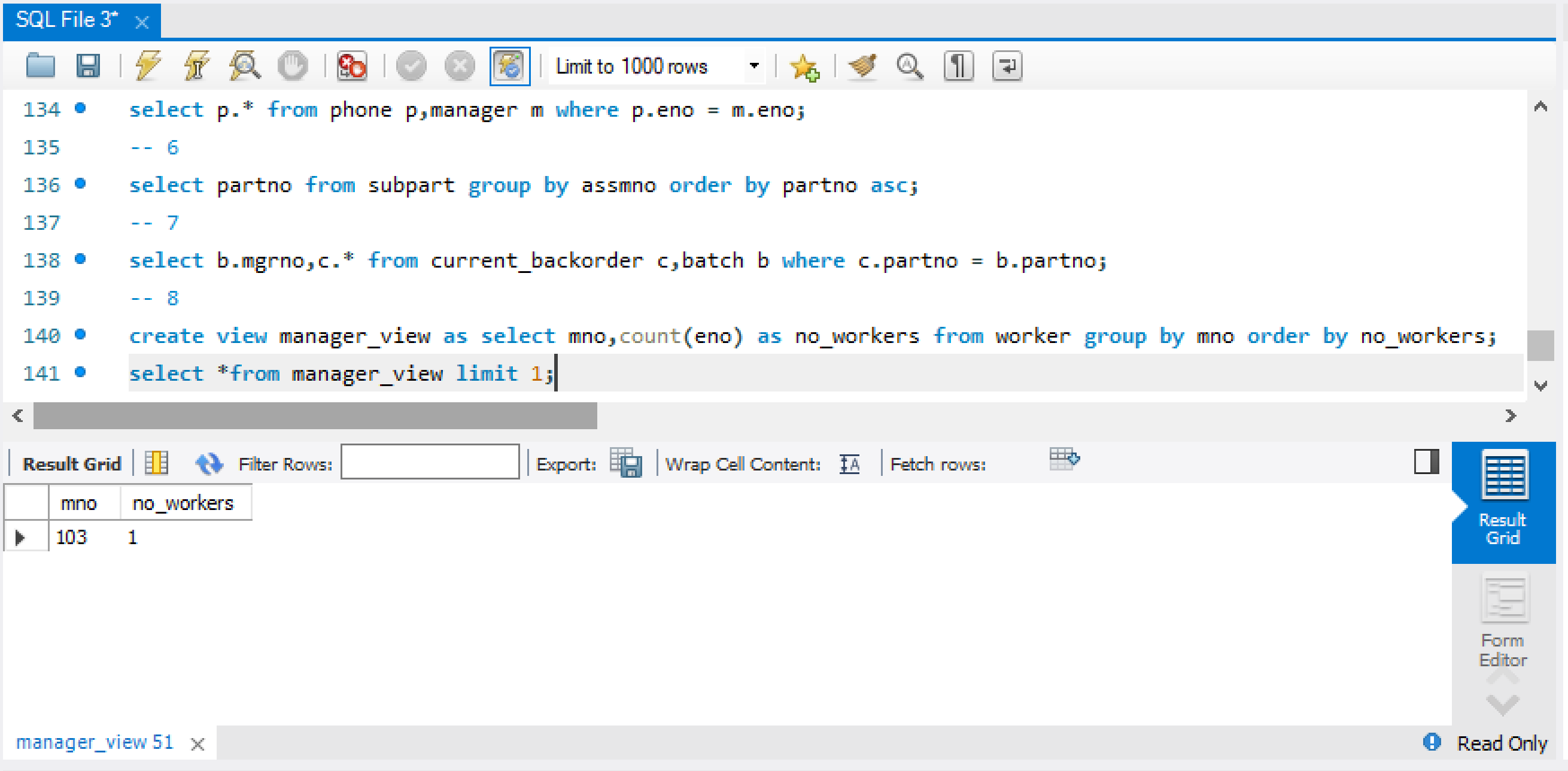
#### Display the all parts that are assemblies they should be listed in ascending order.



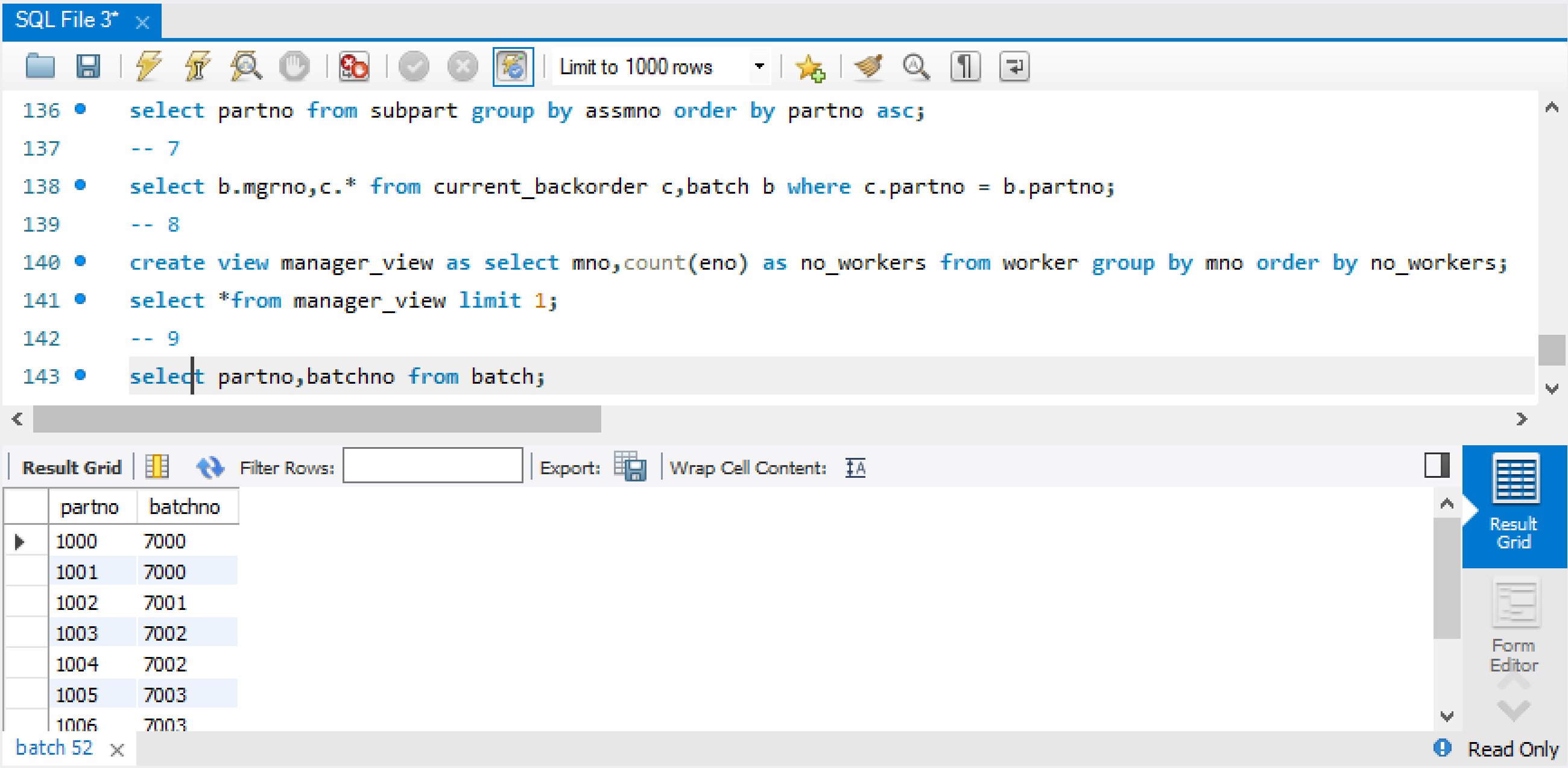
1. Display all current backorders done by each manager.



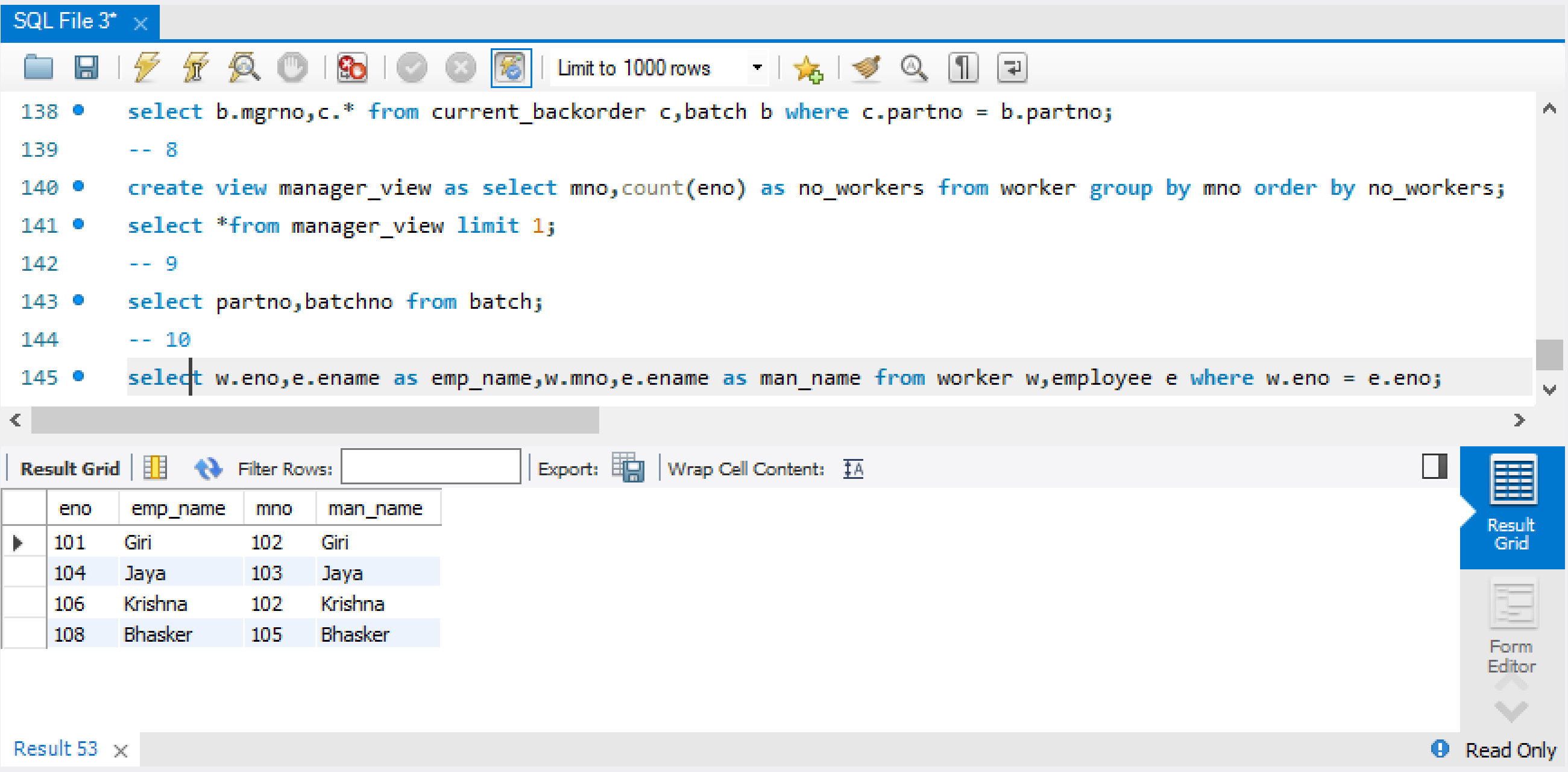
1. Display employee id and number of workers managed for all the managers with the smallest number of workers managed.



1. Display the list of all parts with their batchno



1. Display the employee details with their manager details



**Case Study 6 : PAINITING HIRE BUSINESS**

|  |  |  |  |
| --- | --- | --- | --- |
| **aid** | **name** | **address** | **phone** |
| 200 | John | Delhi | 7786549803 |
| 201 | Samuel | Mumbai | 7123458790 |
| 202 | Samson | Lucknow | 9460367777 |
| 203 | David | Hyderabad | 9797276764 |
| 204 | Raghu | Hyderabad | 8134185751 |
| 205 | Ravi | Mumbai | 7471094738 |
| 206 | Kiran | Delhi | 9808003725 |

**CUSTOMER:** **ARTIST:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **cid** | **cname** | **address** | **phone** | **category** |
| 100 | Raju | Hyderabad | 9876045789 | Bronze |
| 101 | Hari | Vijayawada | 8877678956 | Gold |
| 102 | Devi | Guntur | 7879312123 | Silver |
| 103 | Rani | Delhi | 8780945290 | Platinum |
| 104 | Jaya | Mumbai | 9612578457 | Gold |
| 105 | Haritha | Kolkata | 9611665513 | Silver |
| 106 | Kalyan | Vijayawada | 9610752569 | Bronze |
| 107 | Roja | Hyderabad | 9609839625 | Platinum |
| 108 | Amar | Vijayawada | 9608926681 | Gold |
| 109 | Padma | Vijayawada | 9608013737 | Bronze |

**PAINTING**: **RENT**:

|  |  |  |  |
| --- | --- | --- | --- |
| **pid** | **aid** | **rental\_cost** | **type** |
| 300 | 201 | 4500 | Hired |
| 301 | 202 | 3500 | Not hired |
| 302 | 203 | 7500 | Hired |
| 303 | 205 | 2500 | Not hired |
| 304 | 202 | 10000 | Not hired |
| 305 | 201 | 8000 | Not hired |
| 306 | 203 | 6500 | Not hired |

|  |  |  |  |
| --- | --- | --- | --- |
| **cid** | **pid** | **renatl\_date** | **rental\_period** |
| 104 | 300 | 10-06-2020 | 6 |
| 105 | 302 | 05-07-2020 | 10 |
| 108 | 304 | 15-07-2020 | 3 |
| 109 | 305 | 25-06-2020 | 6 |

**OWNER**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **pid** | **oid** | **name** | **address** | **phone** |
| 300 | 500 | Raju | Hyderabad | 9460367777 |
| 301 | 500 | Hari | Vijayawada | 8134185751 |
| 302 | 501 | Giri | Hyderabad | 7808003725 |
| 303 | 501 | Gopi | Delhi | 9481821699 |
| 304 | 503 | Krishna | Mumbai | 7155639673 |
| 305 | 502 | Verma | Delhi | 8829457647 |
| 306 | 502 | Guna | Delhi | 7503275621 |

1. Create the tables identifying the constraints and relationships

create table Customer(cid int primary key,cname varchar(10),address varchar(10),phone bigint,category varchar(10));

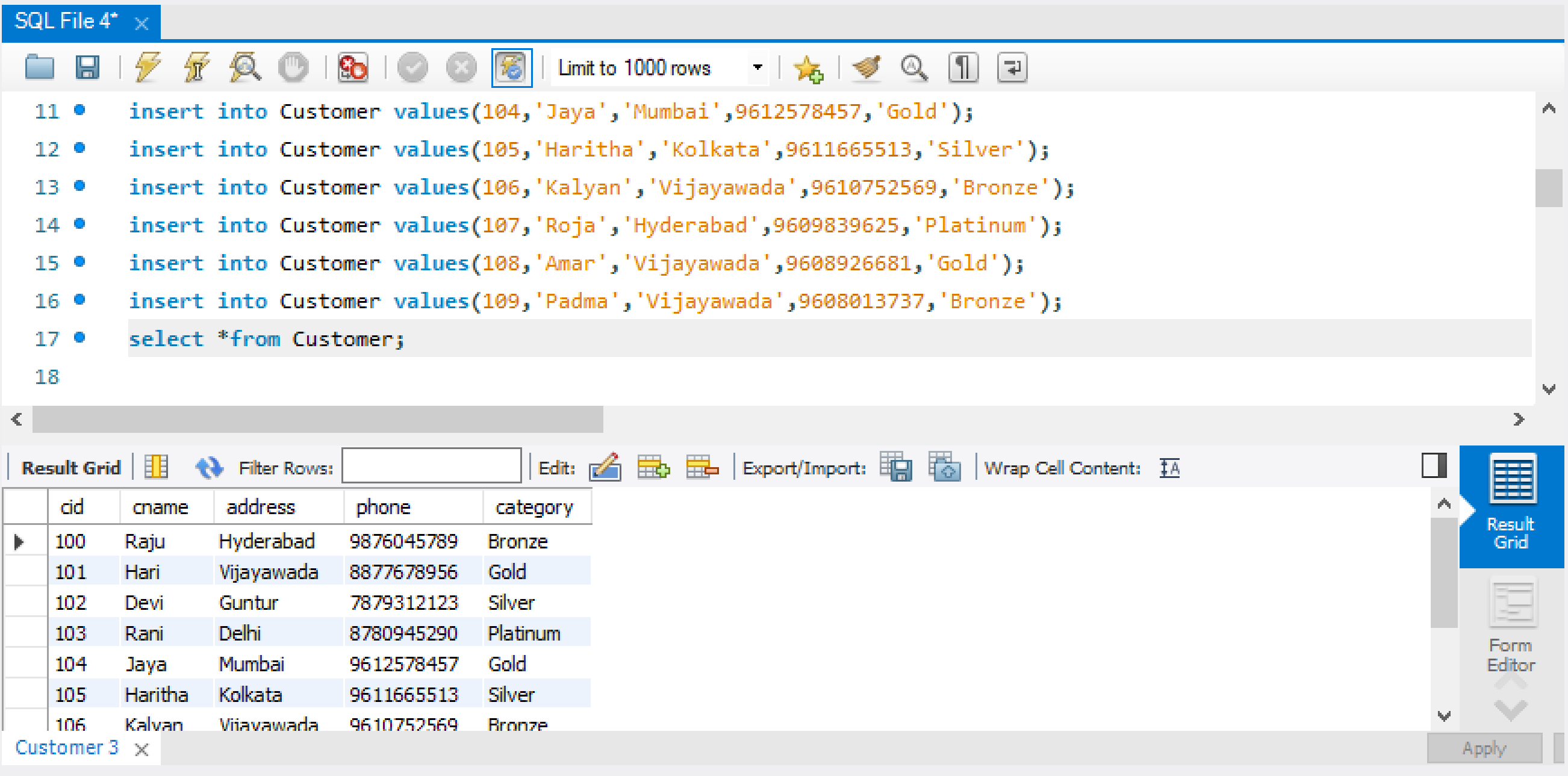
create table Artist(aid int primary key,name varchar(10),address varchar(10),phone bigint);

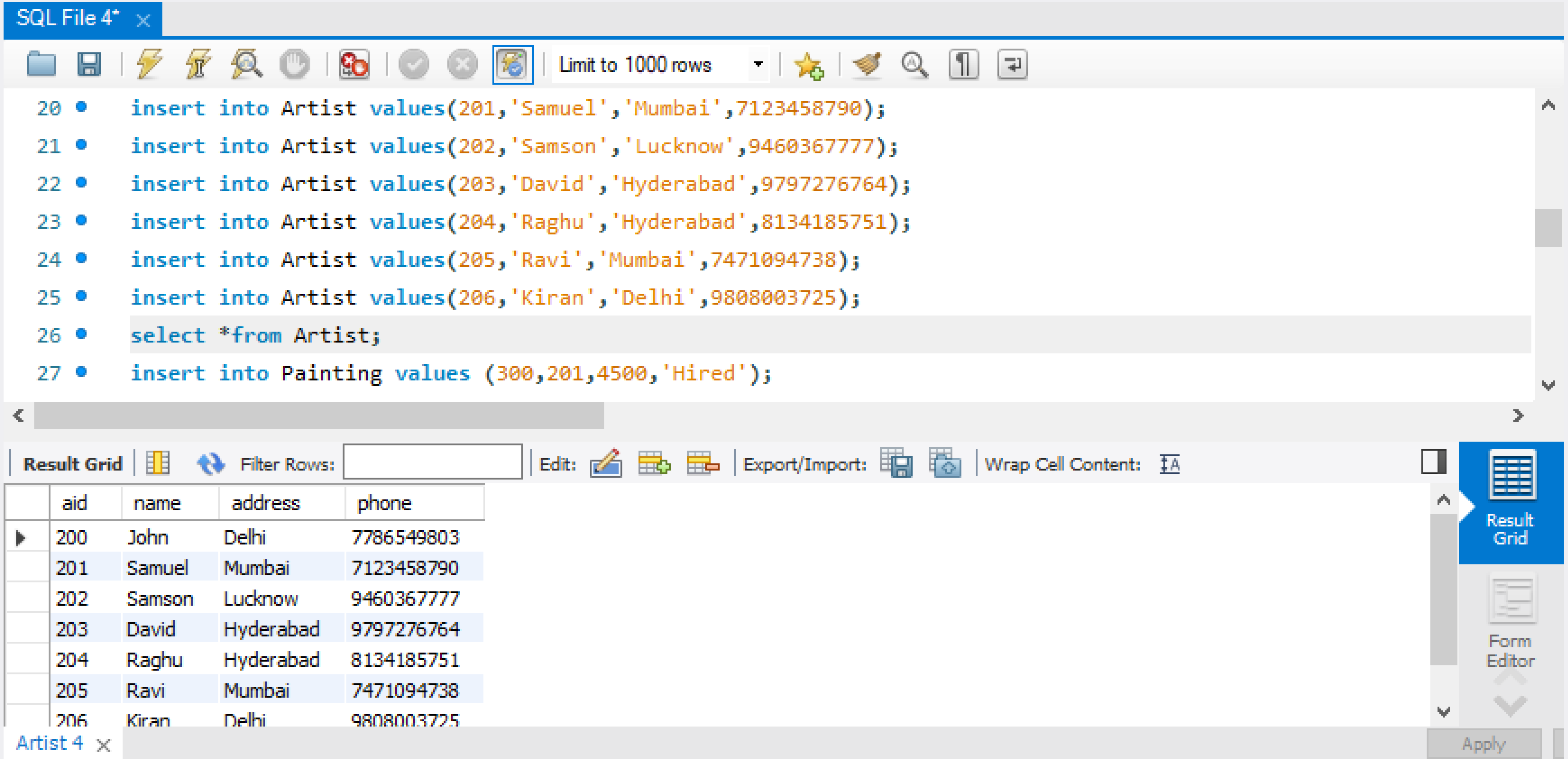
create table Painting(pid int primary key,aid int,rental\_cost int,type varchar(10),foreign key(aid) references Artist(aid));

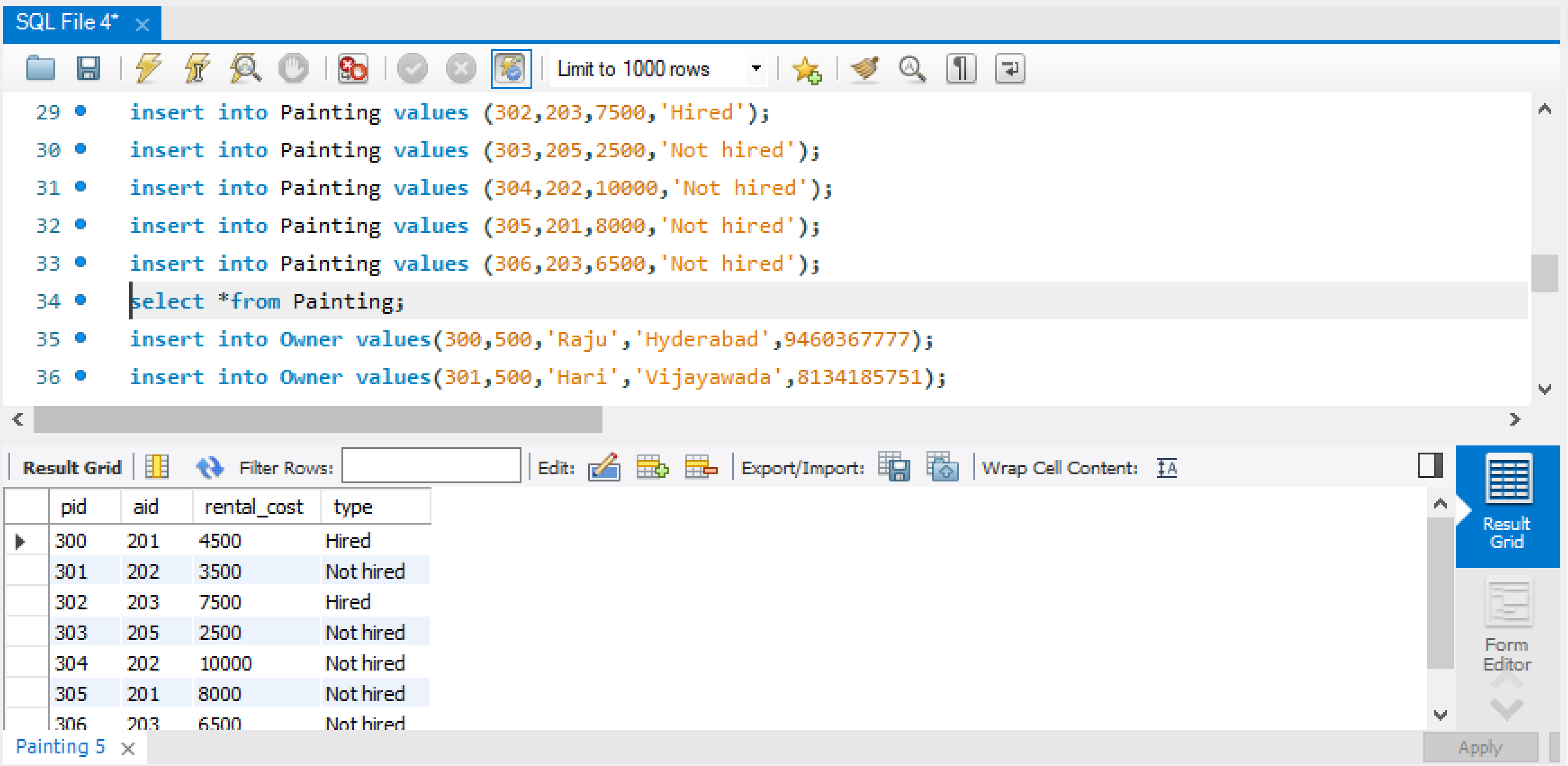
create table Owner(pid int,oid int,name varchar(10),address varchar(10),phone bigint,foreign key(pid) references Painting(pid));

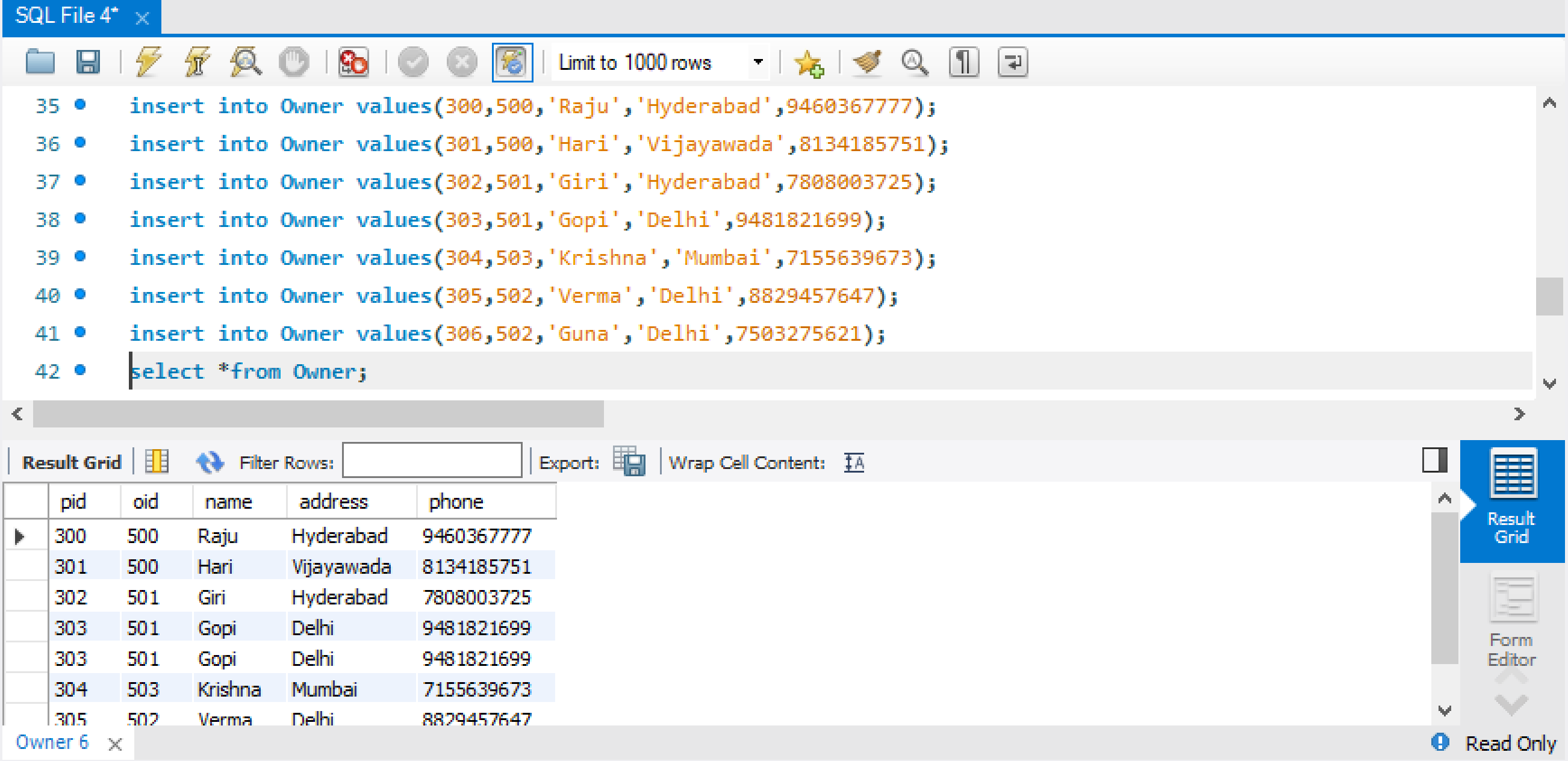
create table Rent(cid int,pid int,rental\_date date,rental\_period int,foreign key(cid) references Customer(cid),foreign key(pid) references Painting(pid));

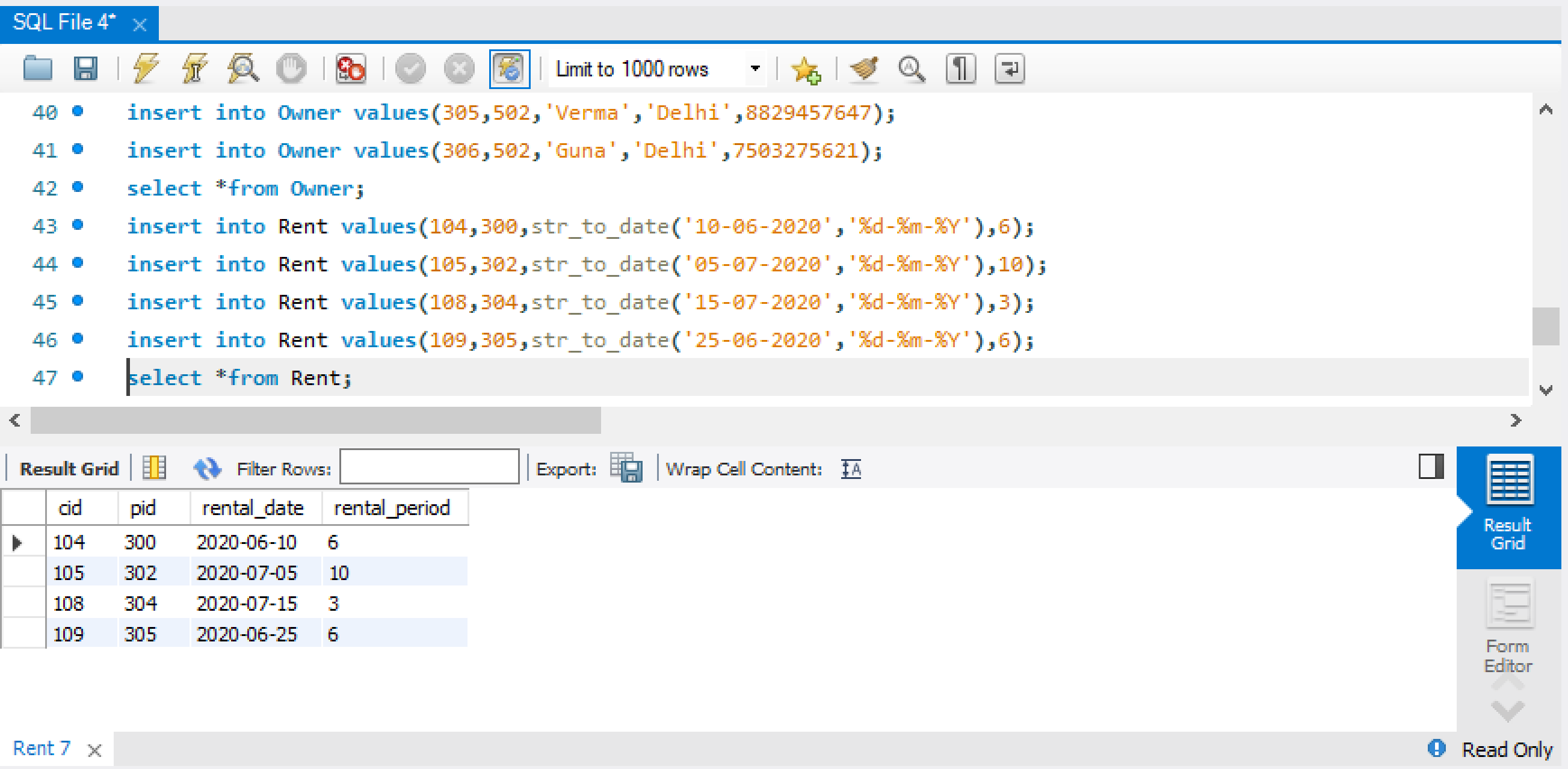
1. Insert atleast 10 records into these tables



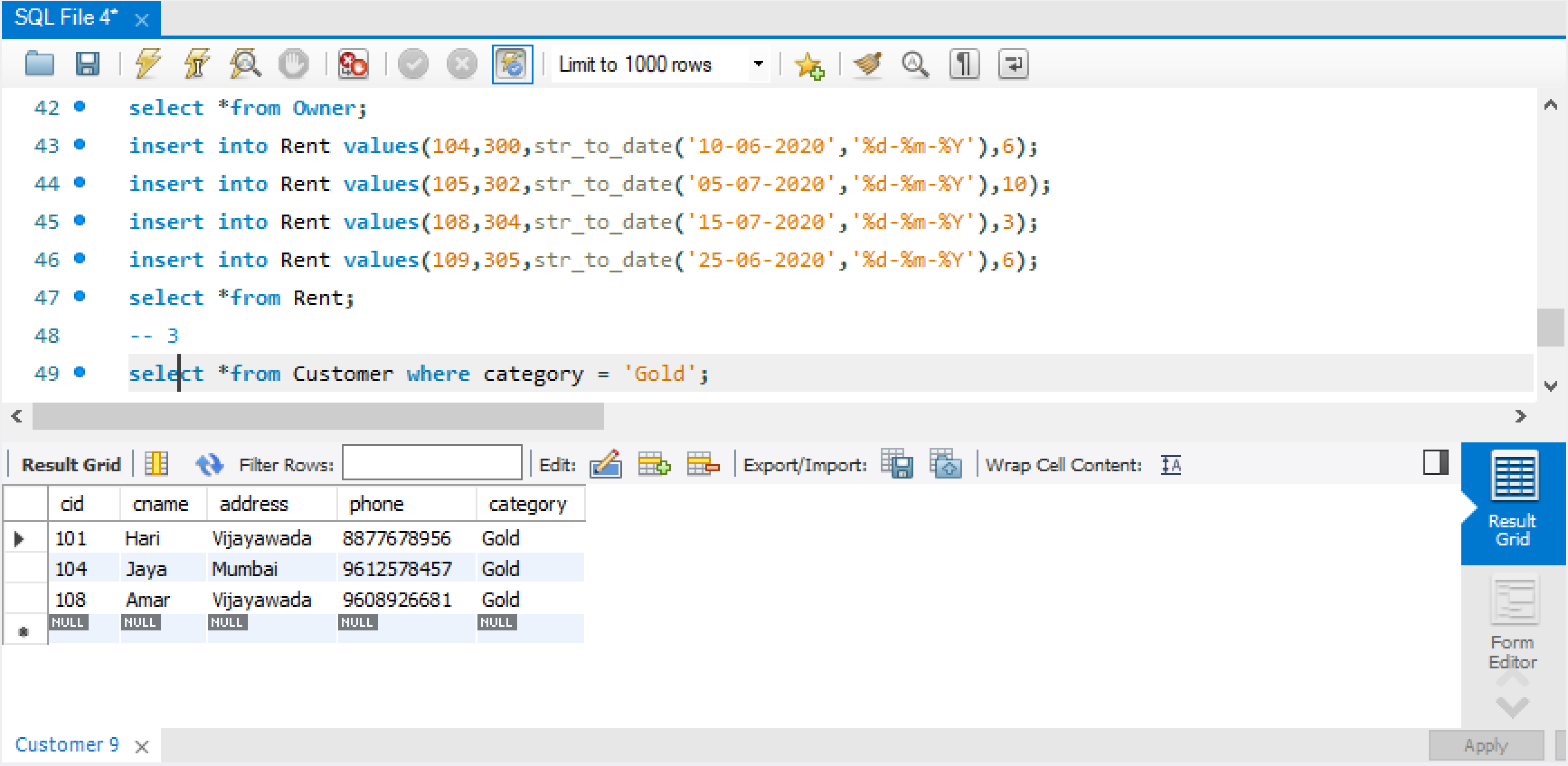




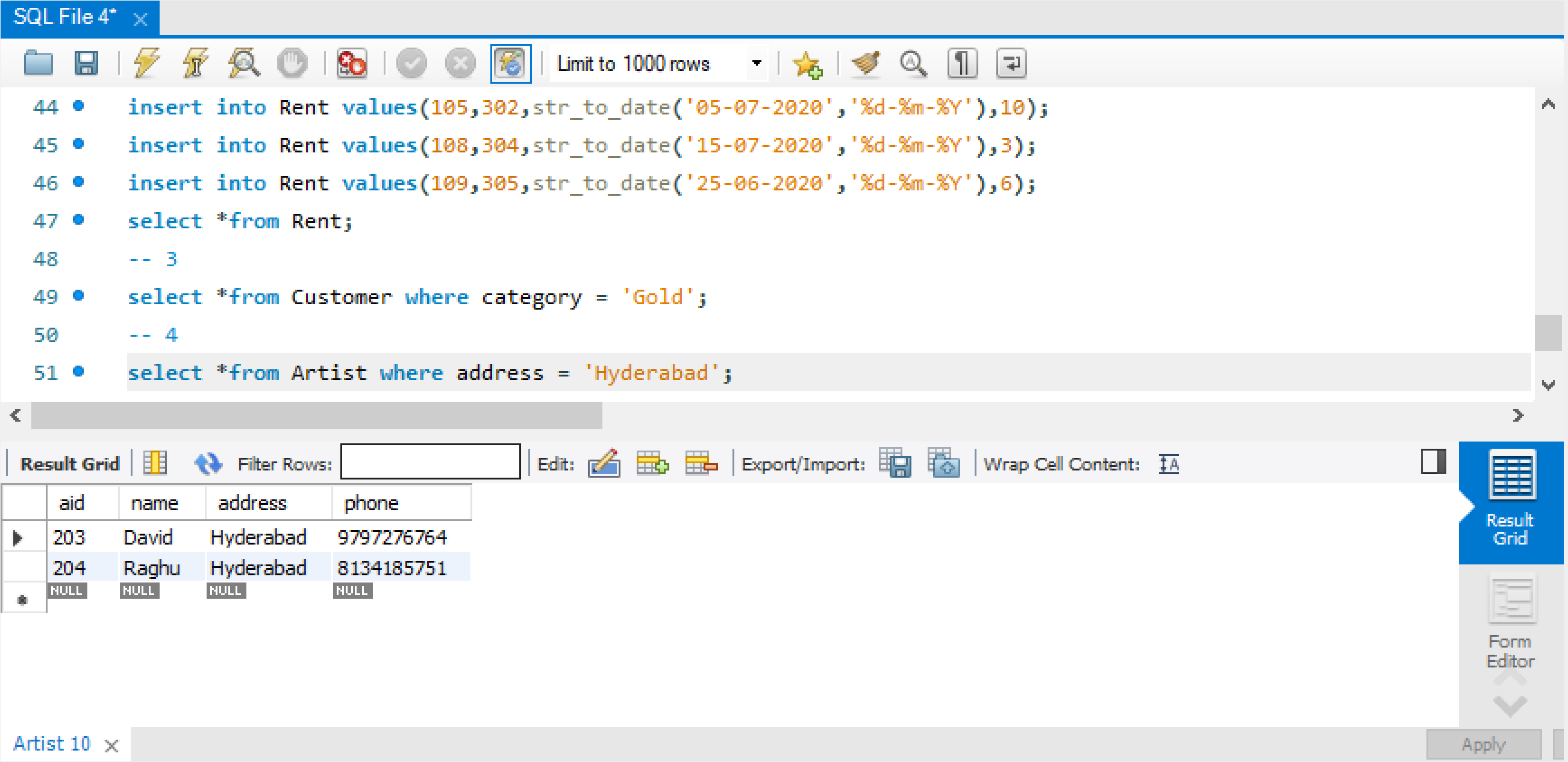




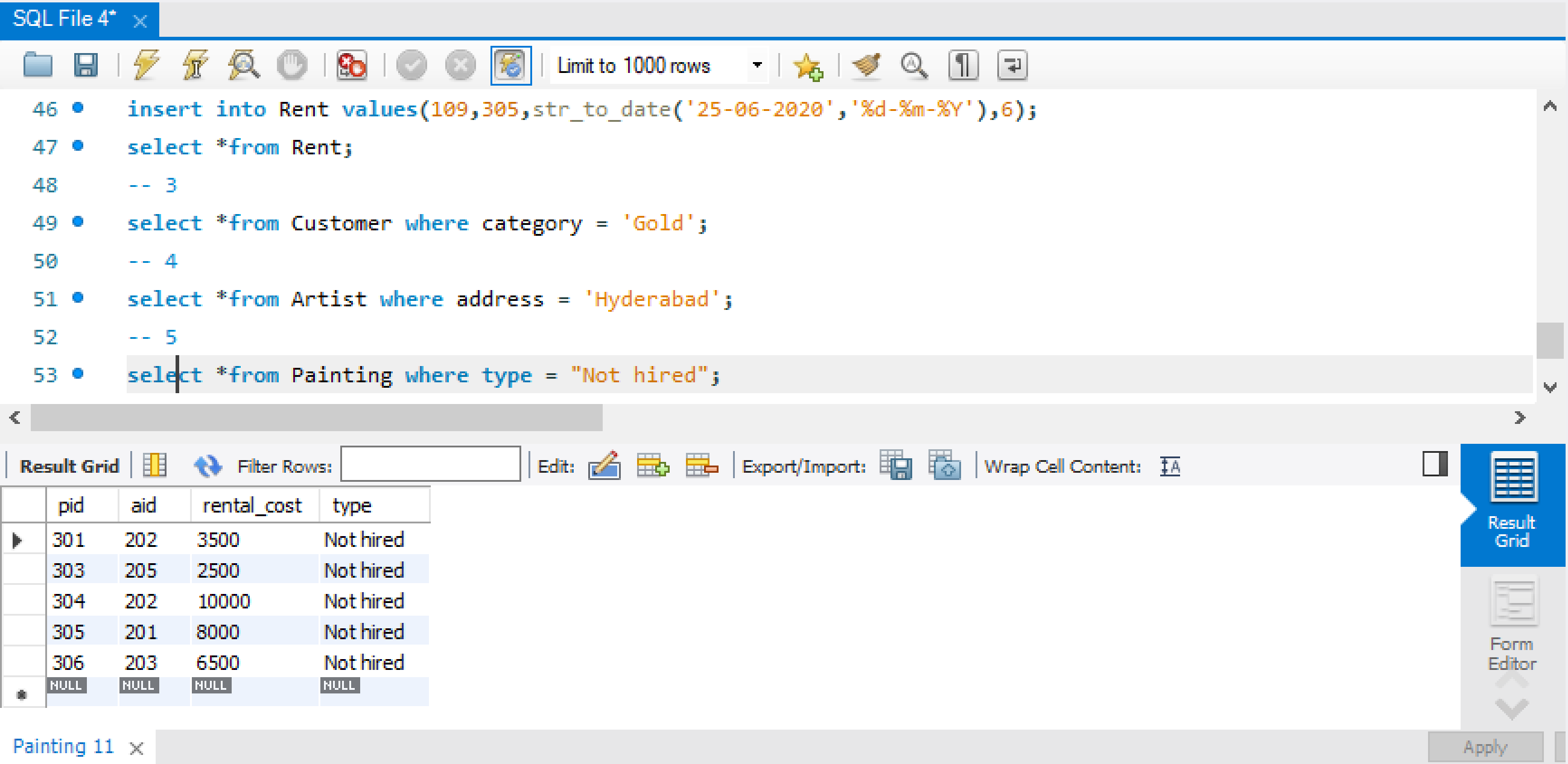
1. Display the customer details who got the category as “Gold”



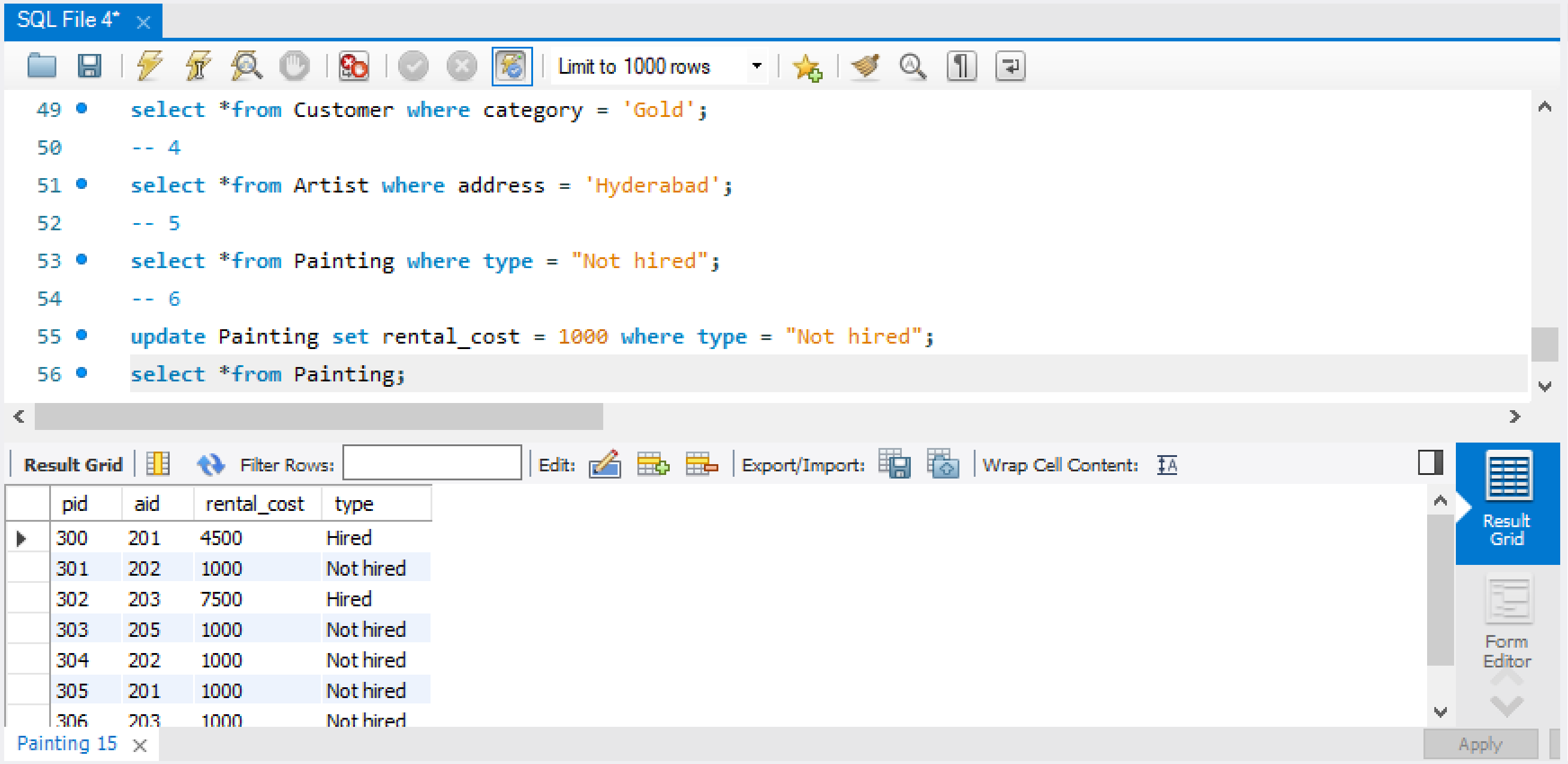
1. Display the list of artists who belong to “Hyderabad”



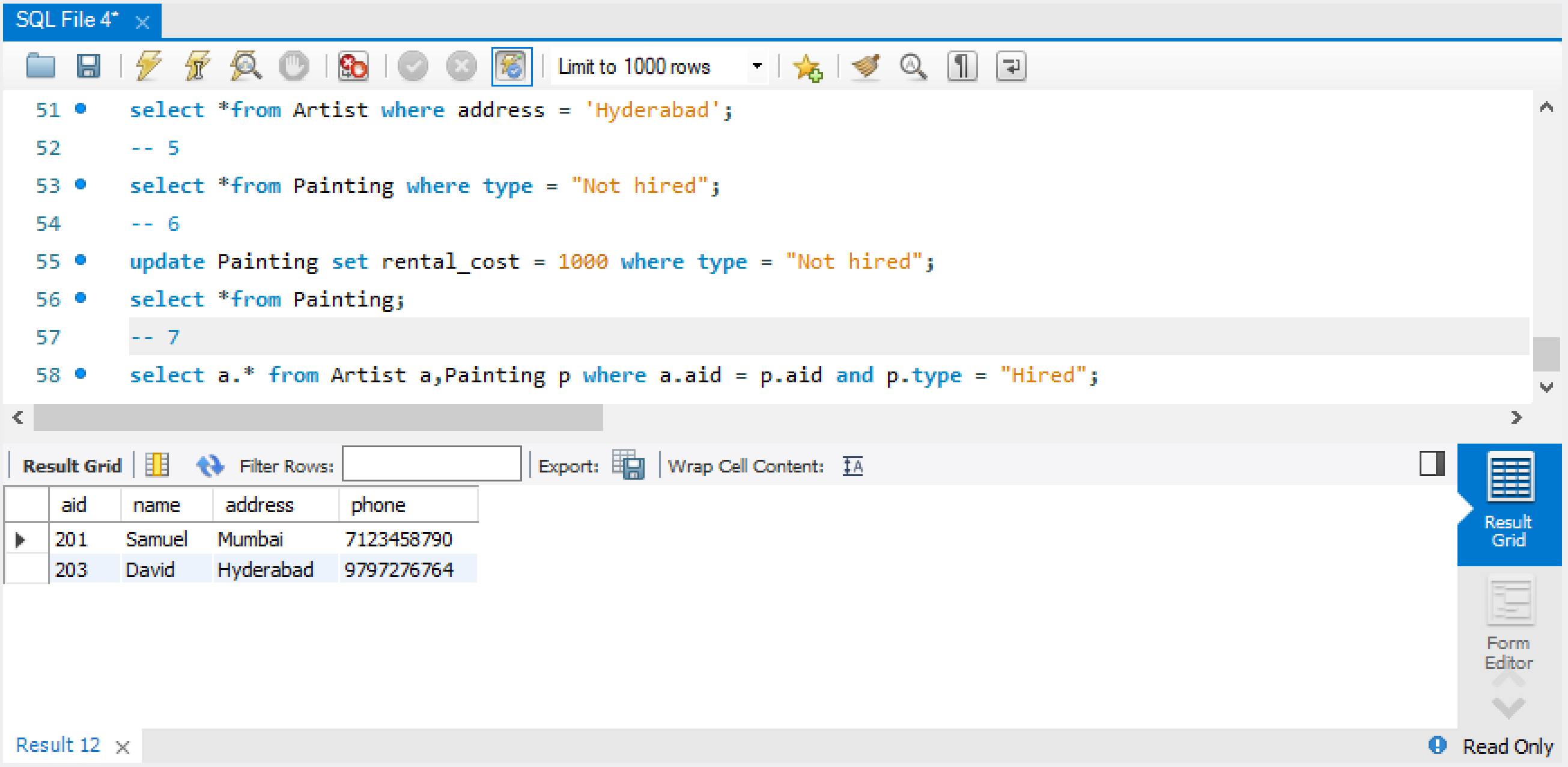
1. Create a query display the painting details which are not hired



1. Create a query to update the rental cost of the painting with 1000 for the paintings which are not hired



1. Display the details of the artist whose paintings are hired



**POST-LAB**

1. Company 'ABC' has decided to get the details of every employee. The company wants to know *id, name, manager id, manager name and computer brand* of every employee.Your job is to generate the report in the above mentioned format. Output must be in ascending order of employee name.

|  |  |
| --- | --- |
| Field | Type |
| id | int |
| name | text |
| manager | int |
| comp\_id | int |

Table: **computer** Table: **employee Table: Computer**

|  |  |
| --- | --- |
| Field | Type |
| id | int |
| brand | varchar |

|  |  |
| --- | --- |
| id | brand |
| 1001 | Dell |
| 1002 | HP |
| 1003 | Lenovo |
| 1004 | Asus |
| 1005 | Compac |
| 1006 | Apple |
| 1007 | Samsung |
| 1008 | Acer |
| 1009 | Sony |

**Table:Employee**

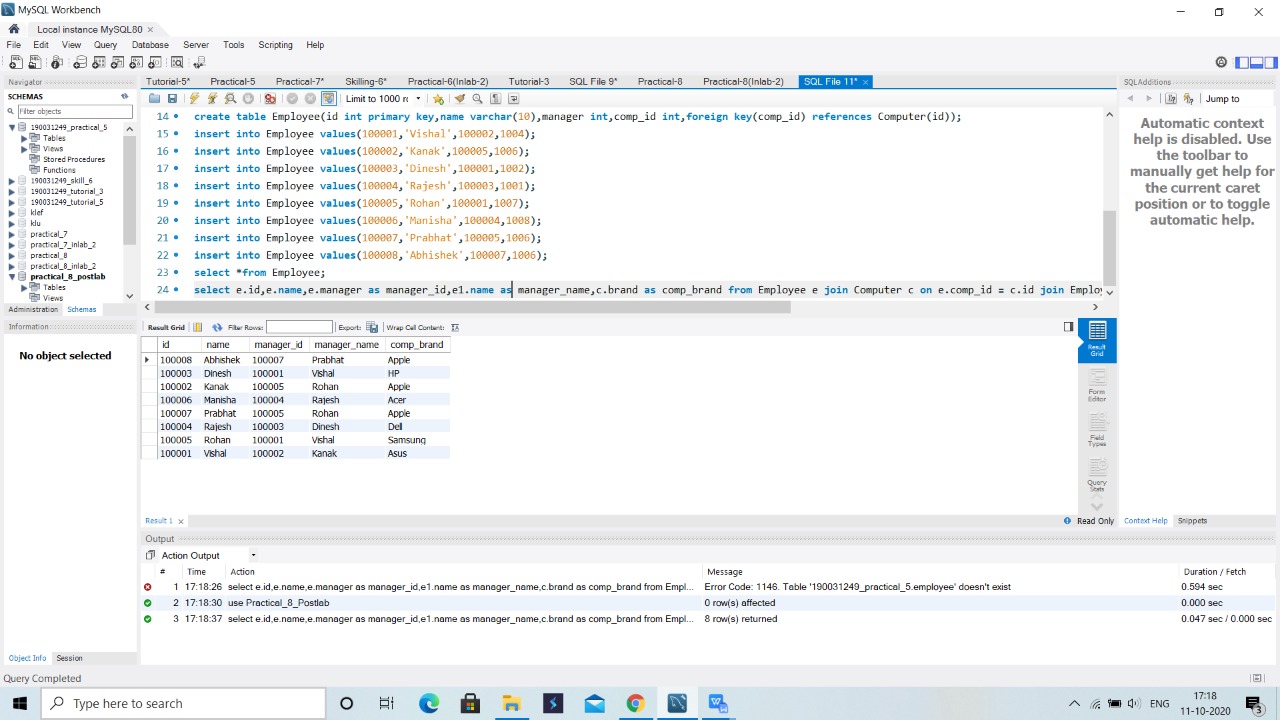
|  |  |  |  |
| --- | --- | --- | --- |
| id | name | manager | comp\_id |
| 100001 | Vishal | 100002 | 1004 |
| 100002 | Kanak | 100005 | 1006 |
| 100003 | Dinesh | 100001 | 1002 |
| 100004 | Rajesh | 100003 | 1001 |
| 100005 | Rohan | 100001 | 1007 |
| 100006 | Manisha | 100004 | 1008 |
| 100007 | Prabhat | 100005 | 1006 |
| 100008 | Abhishek | 100007 | 1006 |

**Table: Output**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | name | manager\_id | manager\_name | comp\_brand |
| 100008 | Abhishek | 100007 | Prabhat | Apple |
| 100003 | Dinesh | 100001 | Vishal | HP |
| 100002 | Kanak | 100005 | Rohan | Apple |
| 100006 | Manisha | 100004 | Rajesh | Acer |
| 100007 | Prabhat | 100005 | Rohan | Apple |
| 100004 | Rajesh | 100003 | Dinesh | Dell |
| 100005 | Rohan | 100001 | Vishal | Samsung |
| 100001 | Vishal | 100002 | Kanak | Asus |

**ANS)**

**select e.id,e.name,e.manager as manager\_id,e1.name as manager\_name,c.brand as comp\_brand from Employee e join Computer c on e.comp\_id = c.id join Employee e1 on e1.id = e.manager order by e.name asc;**



2) Micro has made a new search algorithm. What it does is check whether the query string is a subsequence of the search string or not. A string *A* is considered a subsequence of string *B* if some characters from *B* can be deleted so that it becomes equal to *A*.Micro ran his algorithm on a list of messages he has and got some output but now he needs to verify it. Help Micro verify the output by doing a search on the same list of messages. String he queried for is "hack" (without quotes).

### Input Format: Sample: Output:

|  |  |
| --- | --- |
| Content | id |
| hacker | 1 |
| hacak | 2 |

|  |  |
| --- | --- |
| Content | id |
| hacker | 1 |
| hacak | 2 |
| happy | 3 |

Table : **Messages**

|  |  |
| --- | --- |
| Field | Type |
| Content | text |
| id | integer |

**Explanation**

String "hack" appears as a subsequence in strings "hacker" and "hacak".

ANS) **select \*from Messages where Content like '%h%a%c%k%' order by id asc;**

3) You are given a table Employee.Employee: It will consist of 2 column, first is Employee Id, and second is the Joining Date of that employee.Date format will be: YYYY-MM-DD. You need to find the number of joinings happened on the even days, for all the special years in the given data.Special years are the ones, in which joining happened only in the even months in the given data.

Example:

i) Even months of an year are : February (2nd month), April (4th month) and so on.

ii) Number of even days of a normal year are:Given dates: 1999-12-01 (odd day), 1999-02-02 (even day), 1999-06-08 (even day). Here number of joinings happened on even days for year 1999 is 2.

Note: Output the data in the ascending order of the Special Year.

### Input Format: Output Format:

Table : Employee

|  |  |
| --- | --- |
| Field | Type |
| EmployeeID | int |
| JoiningDate | Date |

|  |  |
| --- | --- |
| Field | Type |
| Special\_Year | int |
| EvenDays | int |

### 

### Sample Input:

### **Employee** Table: Sample Output:

|  |  |
| --- | --- |
| Special\_Year | EvenDays |
| 1998 | 2 |
| 1999 | 2 |
| 2000 | 3 |

|  |  |
| --- | --- |
| EmployeeID | JoiningDate |
| 1 | 1998-12-01 |
| 2 | 1998-06-02 |
| 3 | 1998-02-28 |
| 4 | 2000-04-16 |
| 5 | 2000-04-18 |
| 6 | 2000-06-18 |
| 7 | 1999-08-12 |
| 8 | 1999-10-14 |
| 9 | 1999-10-15 |
| 10 | 2002-01-12 |
| 11 | 2002-02-14 |

**ANS)**

**SELECT SPECIAL\_YEAR.ONLYYEAR, COUNT(\*) FROM (SELECT ONLYYEAR FROM ( SELECT YEAR(JOININGDATE) ONLYYEAR , COUNT(\*) ALL\_DAY, COUNT(IF(XX = 'TRUE', 1, NULL)) EVEN\_MONTH\_COUNT FROM**

**(SELECT JOININGDATE, MONTH(JOININGDATE) , CASE WHEN MONTH(JOININGDATE)%2 = 0 THEN 'TRUE' ELSE 'FALSE' END AS XX FROM Employee) X GROUP BY YEAR(JOININGDATE)) X WHERE X.ALL\_DAY=X.EVEN\_MONTH\_COUNT) SPECIAL\_YEAR JOIN Employee**

**ON SPECIAL\_YEAR.ONLYYEAR = YEAR(Employee.JOININGDATE) WHERE DAY(Employee.JOININGDATE)%2 = 0 GROUP BY SPECIAL\_YEAR.ONLYYEAR ;**

4) Fredo and his friends regularly visit their college canteen. As with any group, on a day, one of the friends pays the canteen bill of all the friends. You are given a table of logs which shows the entries of transactions between friends. The table consists of three fields as described below:

1. P1: Name of the person who pays the bill.

2. P2: Name of the person whose bill is paid by P1.

3. amount: Amount paid by P1 for P2.

You have to summarise the transaction between all pairs of friends. See the sample input and output for explanation.

### Input Format:

|  |  |
| --- | --- |
| Field | Type |
| P1 | text |
| P2 | text |
| NetAmount | int |

### Table : logs Output Format:

|  |  |
| --- | --- |
| Field | Type |
| P1 | text |
| P2 | text |
| amount | int |

### 

### Sample Input: **logs** Table:

|  |  |  |
| --- | --- | --- |
| P1 | P2 | amount |
| Fredo | Zeus | 81 |
| Fredo | John | 59 |
| Zeus | Fredo | 81 |
| Zeus | John | 16 |
| John | Fredo | 27 |
| John | Zeus | 83 |
| Fredo | Zeus | 27 |
| Fredo | John | 17 |

### Sample Output:

|  |  |  |
| --- | --- | --- |
| P1 | P2 | NetAmount |
| Fredo | John | 49 |
| Fredo | Zeus | 27 |
| John | Zeus | 67 |

Explanation:

Here Fredo lends John 59+17=76 and John lends Fredo 27. So, in all John owes Fredo 49 units.Similarly, Fredo lends Zeus 81+27=108 and Zeus lends Fredo 81. So, in all Zeus owes Fredo 17 units.Similarly, Zeus owes John 67 units.

Note:

Only direct transactions are to be covered in the output table.

Only those entries should come in the output table which have NetAmount greater than 0.

The output table should be ordered by P1 in ascending order and then by P2 in ascending order.

It is guaranteed that the input table will contain all ordered pairs of friends atleast once.

ANS)

select e.P1 as P1,e.P2 as P2,(e.total-f.total) as net from (select P1,P2,sum(amount) as total from logs group by P1,P2) as e, (select P1,P2,sum(amount) as total from logs group by P1,P2) as f where e.total-f.total > 0 and (e.P1=f.P2 and e.P2=f.P1) order by P1 asc,P2 asc;