

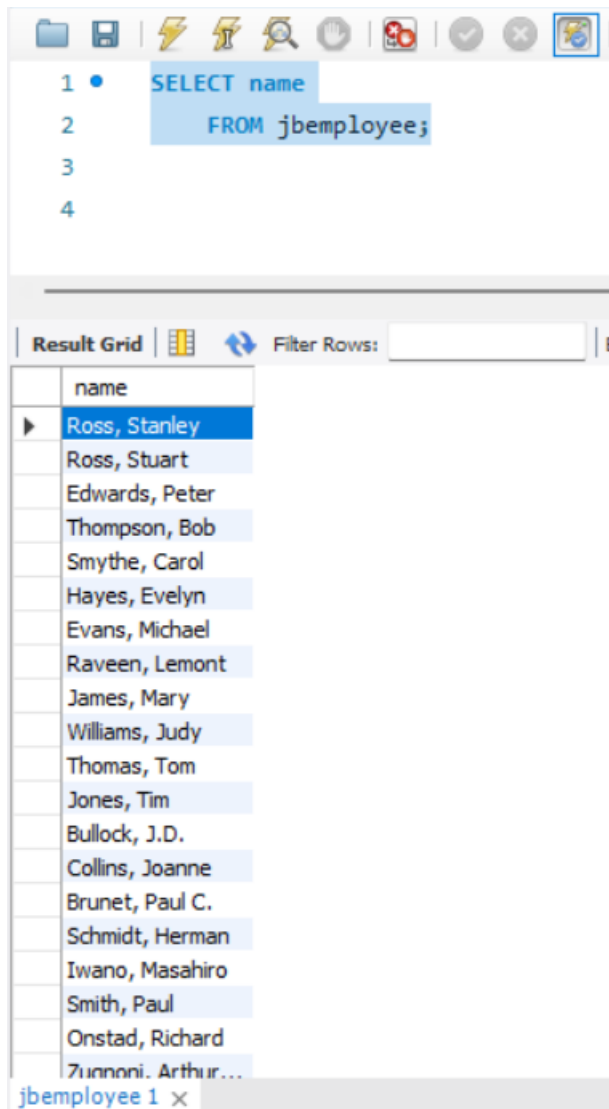
## Relational Database Lab1

1) List all employees, i.e. all tuples in the jbemployee relation

Selecting all the employees name from name column.

### Query:

**SELECT name FROM jbemployee;**



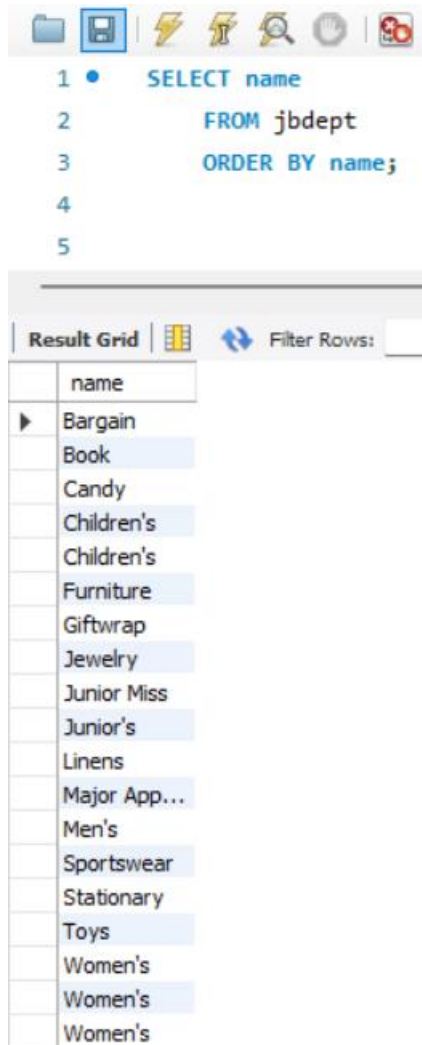
2)List the name of all departments in alphabetical order. Note: by “name” we mean the name attribute for all tuples in the jbdept relation.

The names are listed in ascending order.

### Query:

**SELECT name FROM jbdept ORDER BY name;**

## Relational Database Lab1



The screenshot shows a database query editor with a toolbar at the top containing icons for file operations, execution, and search. Below the toolbar, a SQL query is entered in a text area, with line numbers 1 through 5 on the left. The query is: `SELECT name` (line 1), `FROM jbdept` (line 2), `ORDER BY name;` (line 3), followed by blank lines 4 and 5. Below the query editor is a 'Result Grid' tab. The grid has a single column header 'name'. The first row is expanded, showing a list of department names: Bargain, Book, Candy, Children's, Children's, Furniture, Giftwrap, Jewelry, Junior Miss, Junior's, Linens, Major App..., Men's, Sportswear, Stationary, Toys, Women's, Women's, and Women's.

```
1 • SELECT name
2     FROM jbdept
3     ORDER BY name;
4
5
```

name
Bargain
Book
Candy
Children's
Children's
Furniture
Giftwrap
Jewelry
Junior Miss
Junior's
Linens
Major App...
Men's
Sportswear
Stationary
Toys
Women's
Women's
Women's

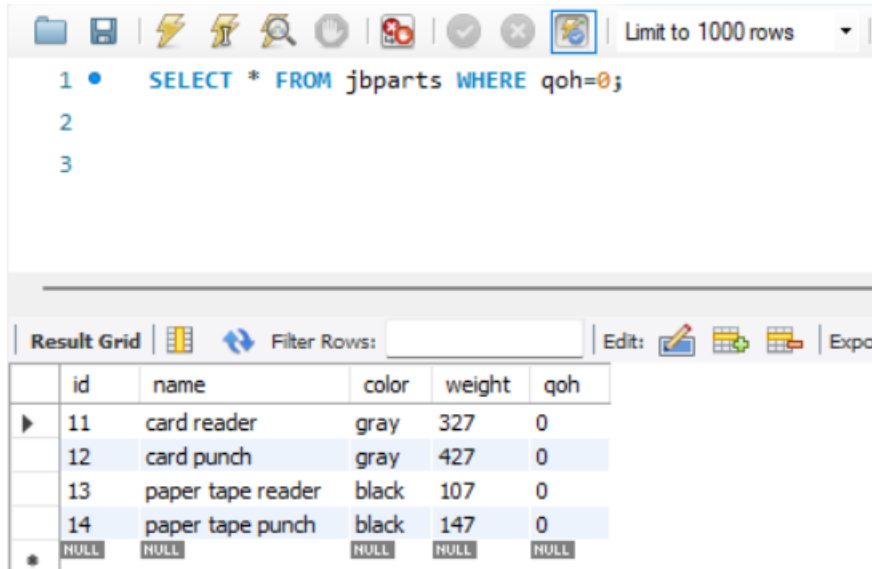
3)What parts are not in store, i.e. qoh = 0? (qoh = Quantity On Hand)

Listing any parts from the jbparts table that do not have a qoh in store.

**Query:**

```
SELECT * FROM jbparts WHERE qoh=0;
```

## Relational Database Lab1



The screenshot shows a database query tool interface. At the top, there is a toolbar with various icons and a dropdown menu set to "Limit to 1000 rows". Below the toolbar, a SQL query is entered in a text area:

```
1 • SELECT * FROM jbparts WHERE qoh=0;  
2  
3
```

Below the query editor, there is a "Result Grid" section. It includes a "Filter Rows:" field and an "Edit:" button. The result grid displays the following data:

	id	name	color	weight	qoh
▶	11	card reader	gray	327	0
	12	card punch	gray	427	0
	13	paper tape reader	black	107	0
	14	paper tape punch	black	147	0
*	NULL	NULL	NULL	NULL	NULL

4) Which employees have a salary between 9000 (included) and 10000 (included)?

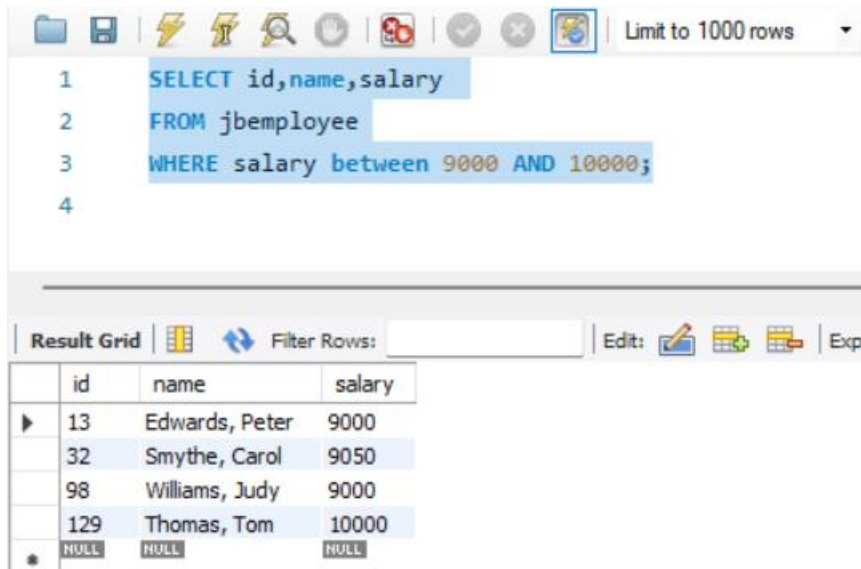
Selecting the employee name whose salary between 9000 & 10000.

### Query:

SELECT id,name,salary

FROM jbemployee

WHERE salary between 9000 AND 10000;



The screenshot shows a database query tool interface. At the top, there is a toolbar with various icons and a dropdown menu set to "Limit to 1000 rows". Below the toolbar, a SQL query is entered in a text area:

```
1 SELECT id,name,salary  
2 FROM jbemployee  
3 WHERE salary between 9000 AND 10000;  
4
```

Below the query editor, there is a "Result Grid" section. It includes a "Filter Rows:" field and an "Edit:" button. The result grid displays the following data:

	id	name	salary
▶	13	Edwards, Peter	9000
	32	Smythe, Carol	9050
	98	Williams, Judy	9000
	129	Thomas, Tom	10000
*	NULL	NULL	NULL

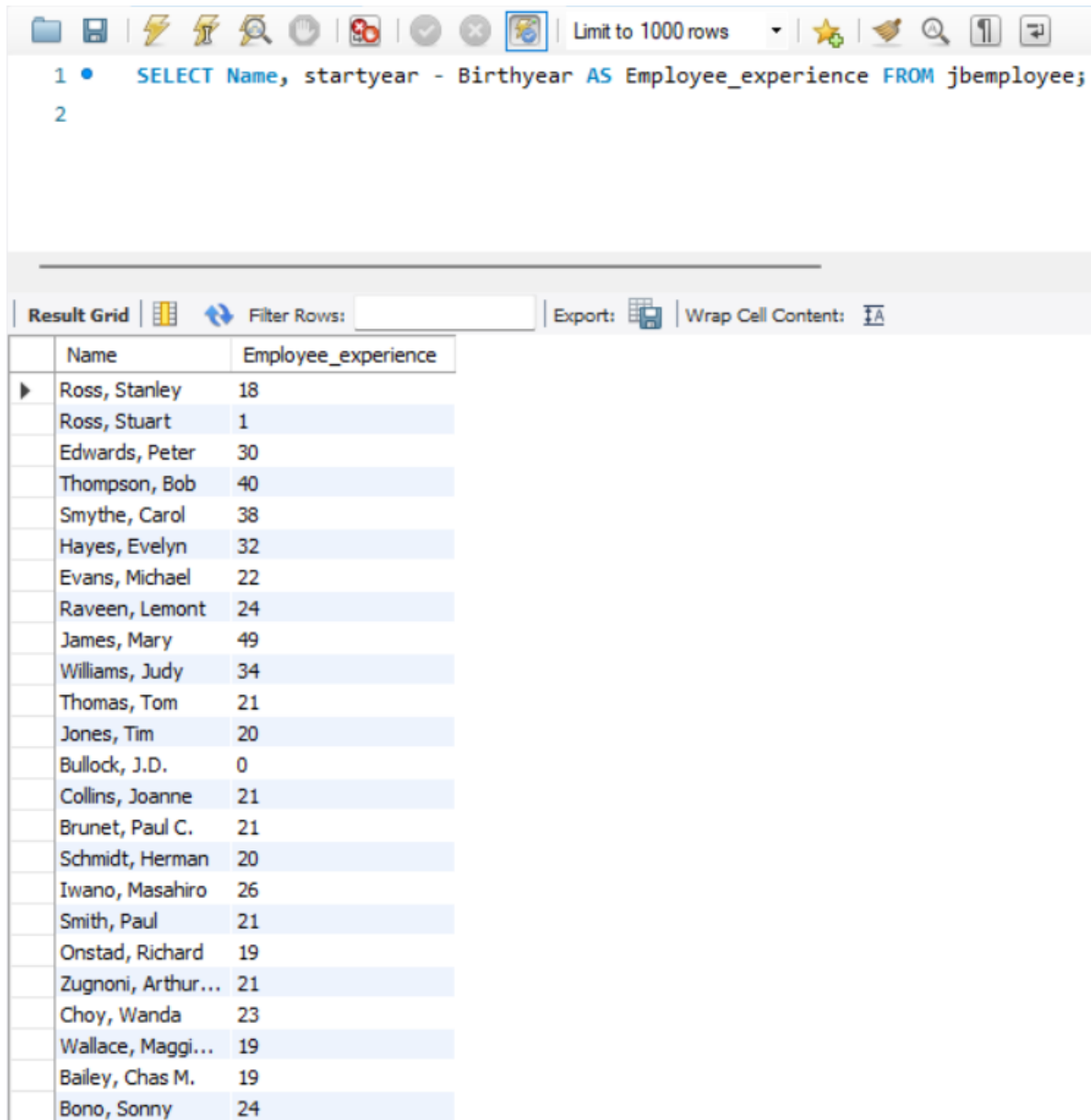
5) What was the age of each employee when they started working (startyear)?  
Finding the age of an employee when they started working by using 2 columns (startyear, and birthyear)

Finding the age of the employee by using started year of the company.

## Relational Database Lab1

### Query:

**SELECT Name, startyear - Birthyear AS Employee\_experience FROM jbemployee;**



The screenshot shows a database query tool interface. At the top, there is a toolbar with various icons and a dropdown menu set to 'Limit to 1000 rows'. Below the toolbar, the SQL query is entered in a text area: `1 • SELECT Name, startyear - Birthyear AS Employee_experience FROM jbemployee;`. Below the query area, the results are displayed in a table with two columns: 'Name' and 'Employee\_experience'. The table contains 24 rows of data.

Name	Employee_experience
Ross, Stanley	18
Ross, Stuart	1
Edwards, Peter	30
Thompson, Bob	40
Smythe, Carol	38
Hayes, Evelyn	32
Evans, Michael	22
Raveen, Lemont	24
James, Mary	49
Williams, Judy	34
Thomas, Tom	21
Jones, Tim	20
Bullock, J.D.	0
Collins, Joanne	21
Brunet, Paul C.	21
Schmidt, Herman	20
Iwano, Masahiro	26
Smith, Paul	21
Onstad, Richard	19
Zugnoni, Arthur...	21
Choy, Wanda	23
Wallace, Maggi...	19
Bailey, Chas M.	19
Bono, Sonny	24

6) Which employees have a last name ending with “son”?

Finding the employee whose last name ending with “son”.

### Query:

**SELECT name**

**FROM jbemployee**

**WHERE name LIKE '%son, %';**

## Relational Database Lab1

The screenshot shows a database query interface. The query editor at the top contains the following SQL query:

```
1 • SELECT name from jbemployee where name like '%son, %';
```

Below the query editor, the results are displayed in a table with the following data:

name
Thompson, Bob

7) Which items (note items, not parts) have been delivered by a supplier called Fisher-Price? Formulate this query using a subquery in the where-clause.

Using subquery to get the items have been delivered by the supplier.

### Query:

**SELECT id,name,supplier FROM jbitem**

**WHERE supplier in (SELECT id FROM jbsupplier WHERE name='fisher-price')**

**ORDER BY name ASC;**

The screenshot shows a database query interface. The query editor at the top contains the following SQL query:

```
1 • SELECT id,name,supplier FROM jbitem
2 WHERE supplier in (SELECT id FROM jbsupplier WHERE name='fisher-price')
3 ORDER BY name ASC;
4
```

Below the query editor, the results are displayed in a table with the following data:

id	name	supplier
43	Maze	89
119	Squeeze Ball	89
107	The 'Feel' Book	89
NULL	NULL	NULL

8)Formulate the same query as above, but without a subquery.

Inner join is used here to get the supplier.

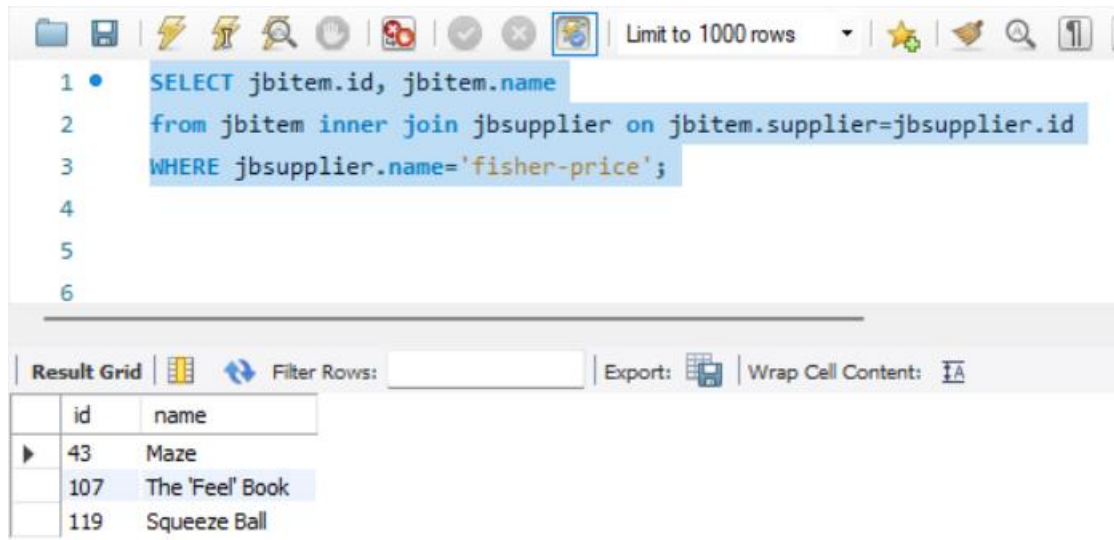
### Query:

## Relational Database Lab1

**SELECT jbitem.id, jbitem.name**

**from jbitem inner join jbsupplier on jbitem.supplier=jbsupplier.id**

**WHERE jbsupplier.name='fisher-price';**



9) Show all cities that have suppliers located in them. Formulate this query using a subquery in the where-clause.

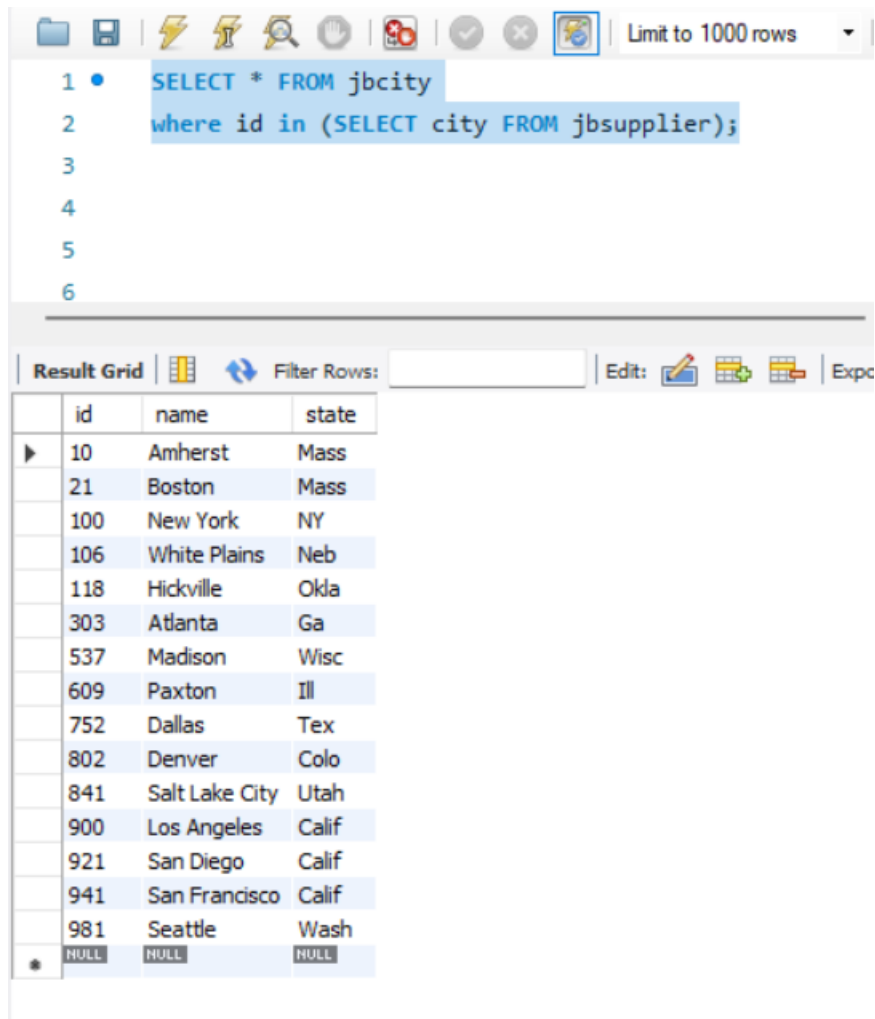
Selecting cities using subquery where the id of a city is in supplier table.

**Query:**

**SELECT \* FROM jbcity**

**where id in (SELECT city FROM jbsupplier);**

## Relational Database Lab1



The screenshot shows a database query tool interface. At the top, there is a toolbar with various icons and a dropdown menu set to 'Limit to 1000 rows'. Below the toolbar, a SQL query is entered in a text area:

```
1 • SELECT * FROM jbcity
2 where id in (SELECT city FROM jbsupplier);
3
4
5
6
```

Below the query editor, there is a 'Result Grid' section. It includes a 'Filter Rows:' input field and an 'Edit:' button. The result grid displays a table with the following data:

	id	name	state
▶	10	Amherst	Mass
	21	Boston	Mass
	100	New York	NY
	106	White Plains	Neb
	118	Hickville	Okla
	303	Atlanta	Ga
	537	Madison	Wisc
	609	Paxton	Ill
	752	Dallas	Tex
	802	Denver	Colo
	841	Salt Lake City	Utah
	900	Los Angeles	Calif
	921	San Diego	Calif
	941	San Francisco	Calif
	981	Seattle	Wash
*	NULL	NULL	NULL

10) What is the name and color of the parts that are heavier than a card reader?

In order to obtain the name and color of the specific parts, a subquery is utilized to retrieve their weight.

### Query:

SELECT name, color

FROM jbparts

WHERE weight > (SELECT weight FROM jbparts WHERE name='card reader');

## Relational Database Lab1

The screenshot shows a database query editor with a toolbar at the top. The SQL query is as follows:

```
1 • SELECT name, color
2 FROM jbparts
3 WHERE weight> (SELECT weight FROM jbparts WHERE name='card reader');
4
5
6
```

Below the query editor is the 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The result grid displays the following data:

	name	color
▶	disk drive	black
	tape drive	black
	line printer	yellow
	card punch	gray

11) Formulate the same query as above, but without a subquery. (The query must not contain the weight as a constant.)

Obtain the name and color of the specific parts without the subquery

### Query:

SELECT tab1.name, tab1.color

FROM jbparts tab1 JOIN jbparts tab2

WHERE tab2.name='card reader' AND tab1.weight>tab2.weight;

The screenshot shows a database query editor with a toolbar at the top. The SQL query is as follows:

```
1 • SELECT tab1.name, tab1.color
2 FROM jbparts tab1 JOIN jbparts tab2
3 WHERE tab2.name='card reader' AND tab1.weight>tab2.weight;
4
5
6
```

Below the query editor is the 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The result grid displays the following data:

	name	color
▶	disk drive	black
	tape drive	black
	line printer	yellow
	card punch	gray



## Relational Database Lab1

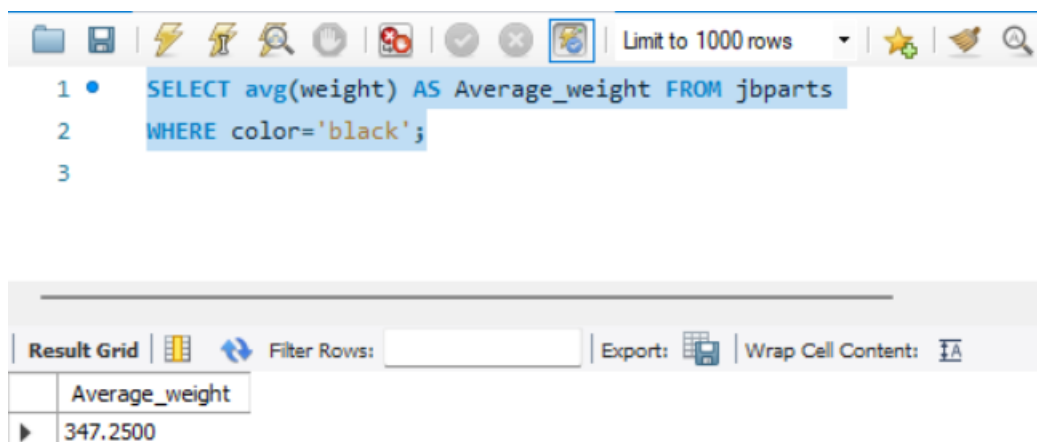
12) What is the average weight of black parts?

The average is obtained by utilizing the avg function and applying a where-clause for a particular color.

### Query:

```
SELECT avg(weight) AS Average_weight FROM jbparts
```

```
WHERE color='black';
```



13) What is the total weight of all parts that each supplier in Massachusetts ("Mass") has delivered? Retrieve the name and the total weight for each of these suppliers. Do not forget to take the quantity of delivered parts into account. Note that one row should be returned for each supplier.

Using inner join to merge the tables to find the total weight of each supplier in Massachusetts (mass).

### Query:

```
SELECT jbsupplier.name as Supplier_name, SUM(jbsupply.quan * jbparts.weight) as TotalWeight
```

```
FROM jbsupplier
```

```
INNER JOIN jbcity ON jbsupplier.city = jbcity.id
```

```
INNER JOIN jbsupply ON jbsupplier.id = jbsupply.supplier
```

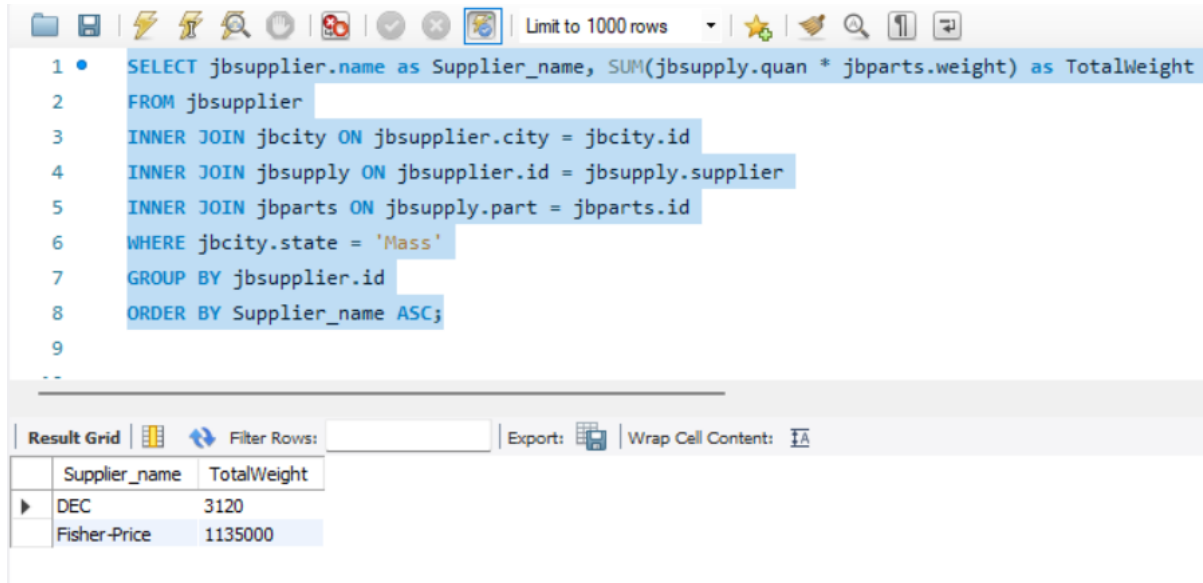
```
INNER JOIN jbparts ON jbsupply.part = jbparts.id
```

```
WHERE jbcity.state = 'Mass'
```

```
GROUP BY jbsupplier.id
```

```
ORDER BY Supplier_name ASC;
```

## Relational Database Lab1



```
1 • SELECT jbsupplier.name as Supplier_name, SUM(jbsupply.quan * jbparts.weight) as TotalWeight
2 FROM jbsupplier
3 INNER JOIN jbcity ON jbsupplier.city = jbcity.id
4 INNER JOIN jbsupply ON jbsupplier.id = jbsupply.supplier
5 INNER JOIN jbparts ON jbsupply.part = jbparts.id
6 WHERE jbcity.state = 'Mass'
7 GROUP BY jbsupplier.id
8 ORDER BY Supplier_name ASC;
9
--
```

Supplier_name	TotalWeight
DEC	3120
Fisher-Price	1135000

14) Create a new relation (a table), with the same attributes as the table items using the CREATE TABLE syntax where you define every attribute explicitly (i.e. not as a copy of another table). Then fill the table with all items that cost less than the average price for items. Remember to define primary and foreign keys in your table!

Creating the table with the same attribute as jbitem and added the values in the table where the cost is less than the average price by defining the primary and foreign key.

### Query:

```
CREATE TABLE jbitems_cost (
id int,
Name varchar(255),
Dept int not null,
Price int not null,
qoh int,
Supplier int not null,
primary key(id), foreign key(dept) references jbdept(id),
foreign key(supplier) references jbsupplier(id)
);
```

## Relational Database Lab1

INSERT into jbitems\_cost (id, name, dept, price, qoh, supplier)

SELECT id, name, dept, price, qoh, supplier from jbitem

WHERE price < (SELECT AVG(price) FROM jbitem);

The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, execution, and search. The SQL editor displays the following code:

```
1 CREATE TABLE jbitems_cost (  
2     id int,  
3     Name varchar(255),  
4     Dept int not null, Price int not null, qoh int, Supplier int not null,  
5     primary key(id), foreign key(dept) references jbdept(id),  
6     foreign key(supplier) references jbsupplier(id));  
7 INSERT into jbitems_cost (id, name, dept, price, qoh, supplier)  
8 SELECT id, name, dept, price, qoh, supplier from jbitem  
9 WHERE price < (SELECT AVG(price) FROM jbitem);  
10  
11 SELECT * FROM jbitems_cost;
```

Below the editor, the 'Result Grid' tab is active, showing the data inserted into the table. The table has columns: id, Name, Dept, Price, qoh, and Supplier. The results are as follows:

	id	Name	Dept	Price	qoh	Supplier
▶	11	Wash Cloth	1	75	575	213
	19	Bellbottoms	43	450	600	33
	21	ABC Blocks	1	198	405	125
	23	1 lb Box	10	215	100	42
	25	2 lb Box, Mix	10	450	75	42
	26	Earrings	14	1000	20	199
	43	Maze	49	325	200	89
	106	Clock Book	49	198	150	125
	107	The 'Feel' Book	35	225	225	89
	118	Towels, Bath	26	250	1000	213
	119	Squeeze Ball	49	250	400	89
	120	Twin Sheet	26	800	750	213
	165	Jean	65	825	500	33
	258	Shirt	58	650	1200	33
*	NULL	NULL	NULL	NULL	NULL	NULL