

1. Name of the employee work in deptno with the job_id use last name coloum

Limit to 1000 rows

```
1 select department_id, job_id, Last_name from employees
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

| department_id | job_id | Last_name |
|---------------|------------|-----------|
| 90 | AD_PRES | King |
| 90 | AD_VP | Kochhar |
| 90 | AD_VP | De Haan |
| 60 | IT_PROG | Hunold |
| 60 | IT_PROG | Ernst |
| 60 | IT_PROG | Austin |
| 60 | IT_PROG | Pataballa |
| 60 | IT_PROG | Lorentz |
| 100 | FI_MGR | Greenberg |
| 100 | FI_ACCOUNT | Faviet |
| 100 | FI_ACCOUNT | Chen |
| 100 | FI_ACCOUNT | Sciarra |
| 100 | FI_ACCOUNT | Herman |

2. Employees who joined in the year 2000

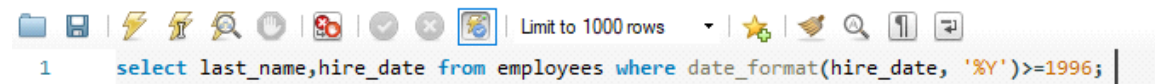
Limit to 1000 rows

```
1 select last_name, hire_date from employees where date_format(hire_date, '%Y')=2000
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

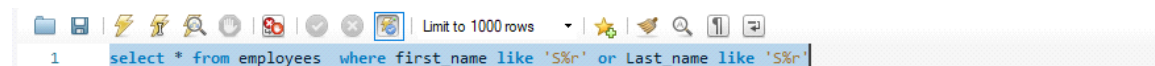
| last_name | hire_date |
|------------|------------|
| Markle | 2000-03-08 |
| Philtanker | 2000-02-06 |
| Zlotkey | 2000-01-29 |
| Marvins | 2000-01-24 |
| Lee | 2000-02-23 |
| Ande | 2000-03-24 |
| Banda | 2000-04-21 |
| Kumar | 2000-04-21 |
| Johnson | 2000-01-04 |
| Geoni | 2000-02-03 |

Employees who joined in the after jan_1996



| Result Grid | | | Filter Rows: | | Export: | | Wrap Cell Content: | |
|-------------|-------------|------------|--------------|--|---------|--|--------------------|--|
| | last_name | hire_date | | | | | | |
| ▶ | Austin | 1997-06-25 | | | | | | |
| | Pataballa | 1998-02-05 | | | | | | |
| | Lorentz | 1999-02-07 | | | | | | |
| | Chen | 1997-09-28 | | | | | | |
| | Sciarra | 1997-09-30 | | | | | | |
| | Urman | 1998-03-07 | | | | | | |
| | Popp | 1999-12-07 | | | | | | |
| | Baida | 1997-12-24 | | | | | | |
| | Tobias | 1997-07-24 | | | | | | |
| | Himuro | 1998-11-15 | | | | | | |
| | Colmenares | 1999-08-10 | | | | | | |
| | Weiss | 1996-07-18 | | | | | | |
| | Fripp | 1997-04-10 | | | | | | |
| | Vollman | 1997-10-10 | | | | | | |
| | Mourgos | 1999-11-16 | | | | | | |
| | Nayer | 1997-07-16 | | | | | | |
| | Mikkilineni | 1998-09-28 | | | | | | |

4. Employees whose name starts between "S" to "R"

[illegible]

5. Employees who works under manager_id (200,201)

```
1 select * from employees
2 select employee_id,First_name,last_name, manager_id from employees where last_name like manager_id=200 or manager_id=201
```

Result Grid

| employee_id | First_name | Last_name | manager_id |
|-------------|------------|-----------|------------|
| 202 | Pat | Fay | 201 |

6. Employees who are "REP" or "MAN" and who are paid more than 6000

```
1 select * from employees
2 select employee_id,First_name,last_name, job_id, salary from employees where (salary>6000) and (job_id='SA_REP' or job_id='SA_MAN')
```

Result Grid

| employee_id | First_name | Last_name | job_id | salary |
|-------------|-------------|-----------|--------|----------|
| 145 | John | Russell | SA_MAN | 14000.00 |
| 146 | Karen | Partners | SA_MAN | 13500.00 |
| 147 | Alberto | Errazuriz | SA_MAN | 12000.00 |
| 148 | Gerald | Cambrault | SA_MAN | 11000.00 |
| 149 | Eleni | Zlotkey | SA_MAN | 10500.00 |
| 150 | Peter | Tucker | SA_REP | 10000.00 |
| 151 | David | Bernstein | SA_REP | 9500.00 |
| 152 | Peter | Hall | SA_REP | 9000.00 |
| 153 | Christopher | Olsen | SA_REP | 8000.00 |
| 154 | Nanette | Cambrault | SA_REP | 7500.00 |
| 155 | Oliver | Tuvault | SA_REP | 7000.00 |
| 156 | Janette | King | SA_REP | 10500.00 |
| 157 | Patrick | Sully | SA_REP | 6000.00 |

7. Calculate anual salary of each employee and print them in decending order

```
1 select * from employees
2
3 select employee_id,First_name,last_name, job_id, salary,(salary*12) Annual_salary from employees;
4 order by Annual_salary desc;
```

Result Grid

| employee_id | First_name | Last_name | job_id | salary | Annual_salary |
|-------------|------------|-----------|---------|----------|---------------|
| 100 | Steven | King | AD_PRES | 24000.00 | 288000.00 |
| 101 | Neena | Kochhar | AD_VP | 17000.00 | 204000.00 |
| 102 | Lex | De Haan | AD_VP | 17000.00 | 204000.00 |
| 103 | Alexander | Hunold | IT_PROG | 9000.00 | 108000.00 |
| 104 | Bruce | Ernst | IT_PROG | 6000.00 | 72000.00 |
| 105 | David | Austin | IT_PROG | 4800.00 | 57600.00 |
| 106 | Valli | Pataballa | IT_PROG | 4800.00 | 57600.00 |
| 107 | Diana | Lorentz | IT_PROG | 4200.00 | 50400.00 |

8. Replace the lastname of "Landry" to "JOE"

The screenshot shows the SQL Developer interface. The query editor contains the following SQL statement:

```
1 • select * from employees
2
3 select Last_name, replace(Last_name,'Landry','JOE') from employees where Last_name='Landry'
4
```

Below the query editor, the "Result Grid" tab is active, displaying the results of the query:

| Last_name | replace(Last_name,'Landry','JOE') |
|-----------|-----------------------------------|
| Landry | JOE |

9. find the position of the 1st occurrence of the char. o in the lastname of all employees who have o in their Lastname

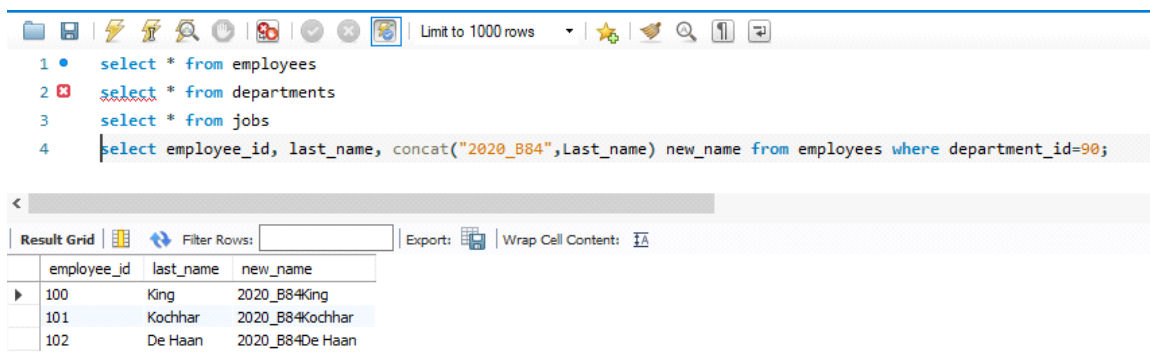
The screenshot shows the SQL Developer interface. The query editor contains the following SQL statement:

```
1 • select * from employees
2 select Last_name, position('o' in Last_name) from employees
3
```

Below the query editor, the "Result Grid" tab is active, displaying the results of the query:

| Last_name | position('o' in Last_name) |
|-----------|----------------------------|
| King | 0 |
| Kochhar | 2 |
| De Haan | 0 |
| Hunold | 4 |
| Ernst | 0 |
| Austin | 0 |
| Pataballa | 0 |
| Lorentz | 2 |
| Greenberg | 0 |
| Faviet | 0 |
| Chen | 0 |
| Sciarra | 0 |

10. Prefix '2020_B84" for employee last_names who works in department 90



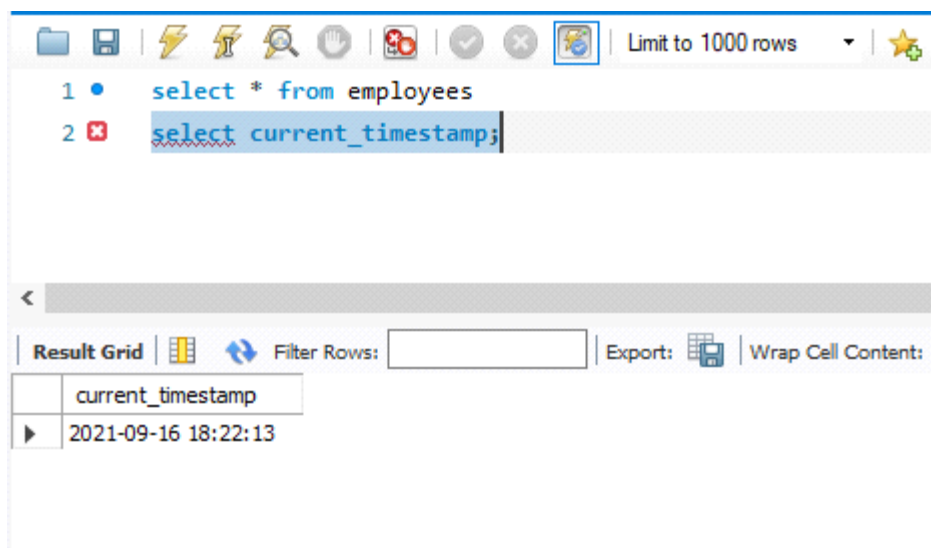
The screenshot shows the SQL Developer interface. The query editor contains four lines of SQL code:

```
1 • select * from employees
2 ✖ select * from departments
3 select * from jobs
4 select employee_id, last_name, concat("2020_B84",last_name) new_name from employees where department_id=90;
```

The results grid shows the output of the query:

| employee_id | last_name | new_name |
|-------------|-----------|-----------------|
| 100 | King | 2020_B84King |
| 101 | Kochhar | 2020_B84Kochhar |
| 102 | De Haan | 2020_B84De Haan |

11. find the current date with local date and time



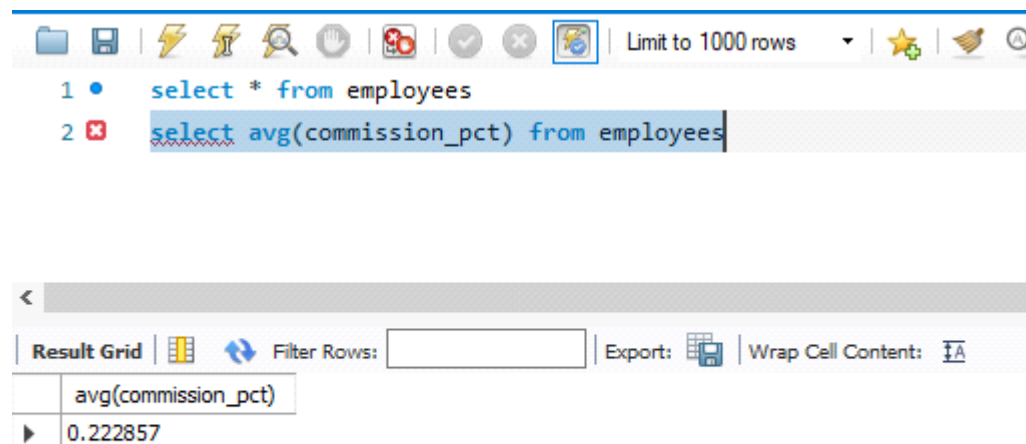
The screenshot shows the SQL Developer interface. The query editor contains two lines of SQL code:

```
1 • select * from employees
2 ✖ select current_timestamp;
```

The results grid shows the output of the query:

| current_timestamp |
|---------------------|
| 2021-09-16 18:22:13 |

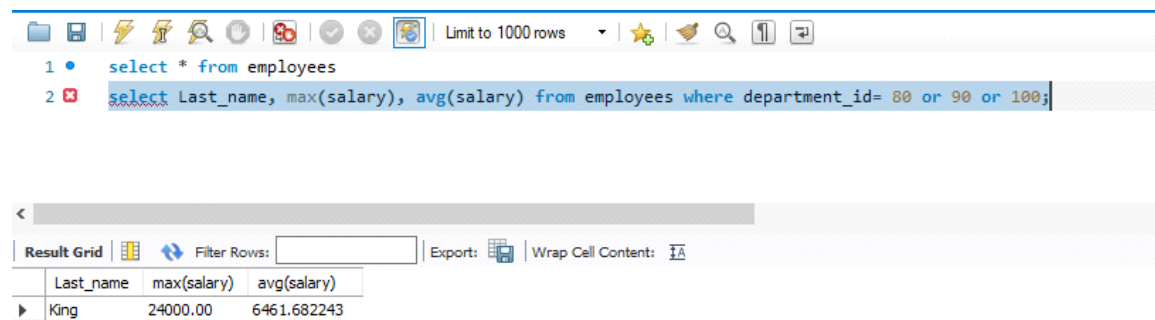
12. find the average commission paid for all the employees



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The SQL editor contains two lines of code: `1 • select * from employees` and `2 ✖ select avg(commission_pct) from employees`. The second line is highlighted. Below the editor, the 'Result Grid' tab is active, displaying a single row with the value `0.222857` under the column `avg(commission_pct)`. The interface also includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox.

| avg(commission_pct) |
|---------------------|
| 0.222857 |

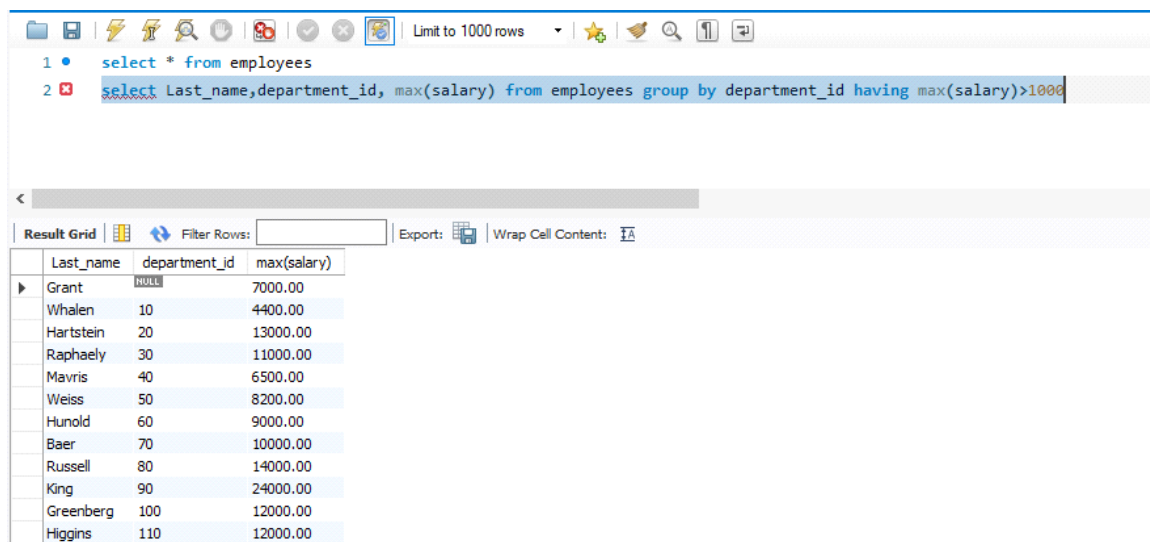
13. find the avg and highest salary paid for department 80,90,100



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The SQL editor contains two lines of code: `1 • select * from employees` and `2 ✖ select Last_name, max(salary), avg(salary) from employees where department_id= 80 or 90 or 100;`. The second line is highlighted. Below the editor, the 'Result Grid' tab is active, displaying a single row with the values `King`, `24000.00`, and `6461.682243` under the columns `Last_name`, `max(salary)`, and `avg(salary)` respectively. The interface also includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox.

| Last_name | max(salary) | avg(salary) |
|-----------|-------------|-------------|
| King | 24000.00 | 6461.682243 |

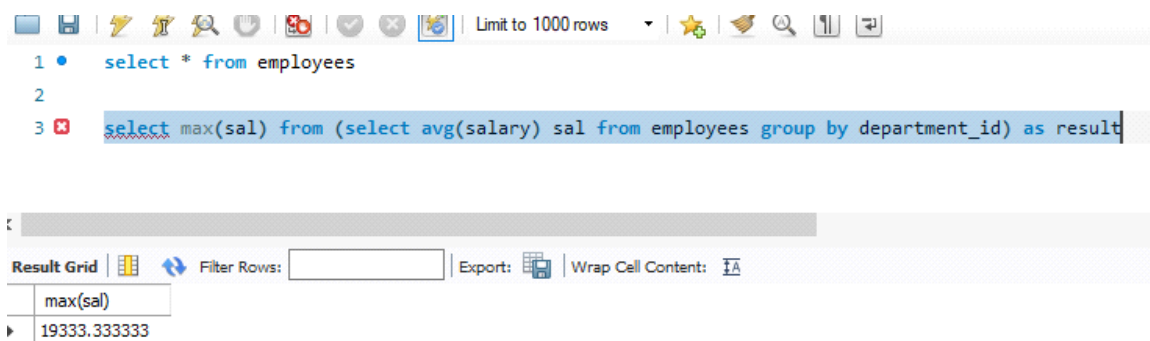
14. find the department id where the highest paid employee salary is more than 1000



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The SQL editor contains two lines of code: `1 • select * from employees` and `2 ✖ select Last_name, department_id, max(salary) from employees group by department_id having max(salary)>1000`. Below the editor, the 'Result Grid' tab is active, displaying a table with three columns: 'Last_name', 'department_id', and 'max(salary)'. The table lists 12 employees with their respective department IDs and maximum salaries.

| Last_name | department_id | max(salary) |
|-----------|---------------|-------------|
| Grant | NULL | 7000.00 |
| Whalen | 10 | 4400.00 |
| Hartstein | 20 | 13000.00 |
| Raphaely | 30 | 11000.00 |
| Mavris | 40 | 6500.00 |
| Weiss | 50 | 8200.00 |
| Hunold | 60 | 9000.00 |
| Baer | 70 | 10000.00 |
| Russell | 80 | 14000.00 |
| King | 90 | 24000.00 |
| Greenberg | 100 | 12000.00 |
| Higgins | 110 | 12000.00 |

15.



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The SQL editor contains three lines of code: `1 • select * from employees`, `2` (blank), and `3 ✖ select max(sal) from (select avg(salary) sal from employees group by department_id) as result`. Below the editor, the 'Result Grid' tab is active, displaying a table with one column: 'max(sal)'. The table contains a single row with the value 19333.33333.

| max(sal) |
|-------------|
| 19333.33333 |

16. find the department name and location id in which 'Ernst' work

The screenshot shows a SQL query editor with the following query:

```
1 • select * from employees
2 ✖ select * from departments
3   select department_name, location_id from departments d
4   inner join employees e on Last_name='Ernst' and e.department_id=d.department_id
```

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

| department_name | location_id |
|-----------------|-------------|
| IT | 1400 |

17.

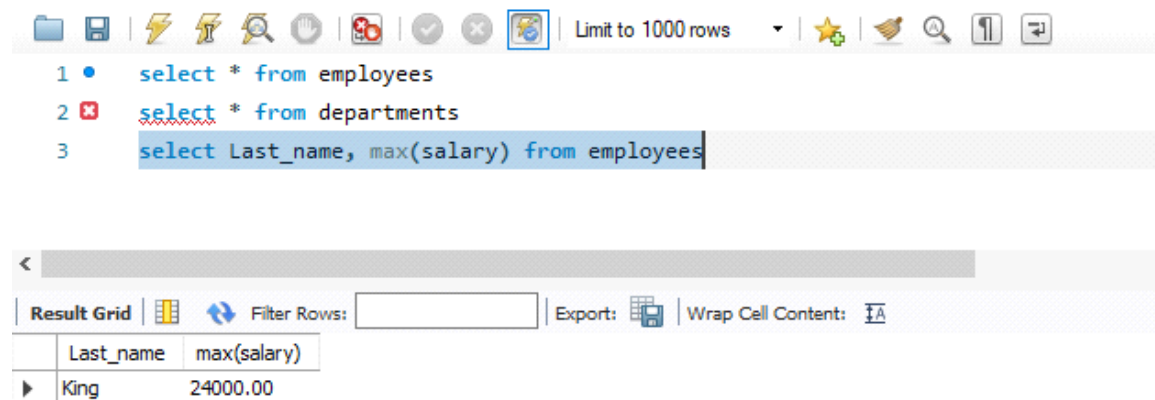
The screenshot shows a SQL query editor with the following query:

```
1 • select * from employees
2
3 ✖ select First_name, Last_name from employees
4   where department_id=(select department_id from departments
5   where location_id=(select location_id from locations where city='Tokyo'));
```

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

| First_name | Last_name |
|------------|-----------|
|------------|-----------|

18. find the employees who are the max paid salary in the organization



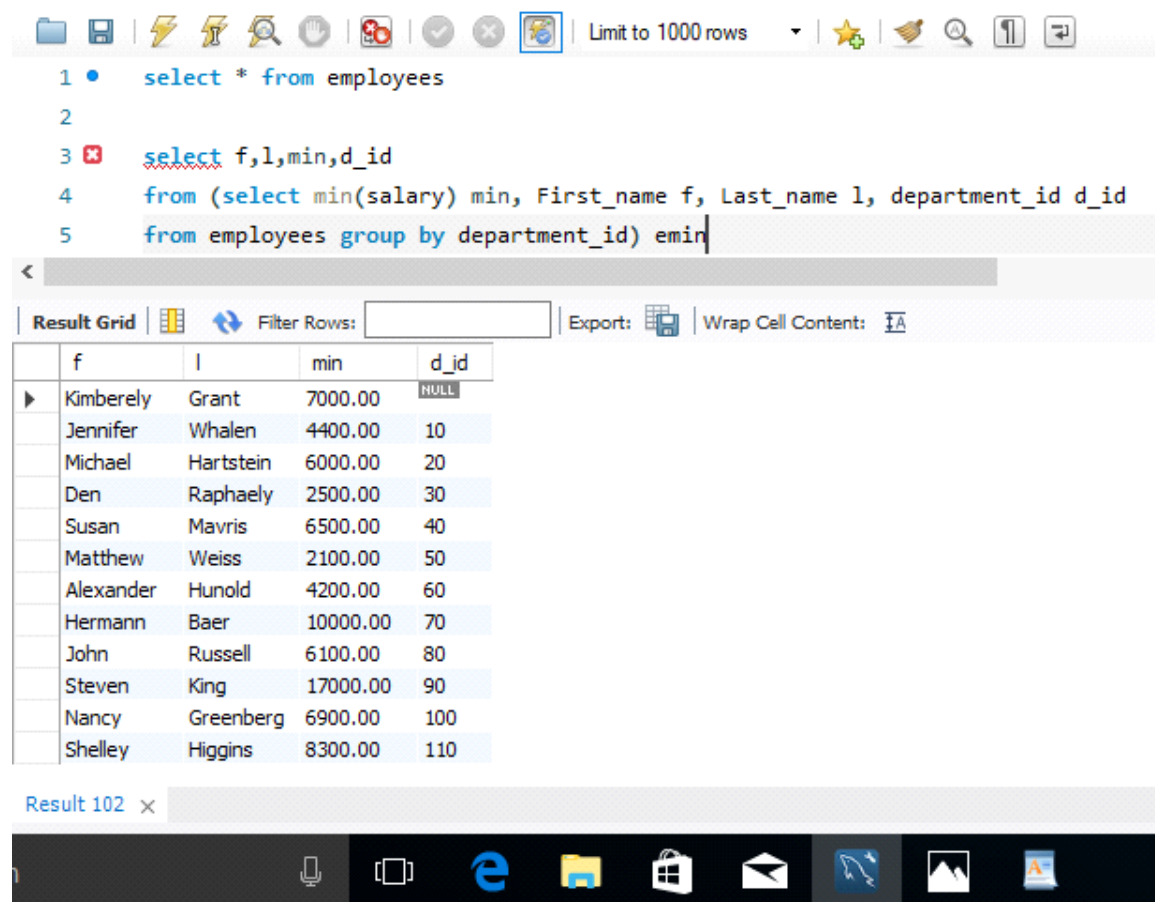
The screenshot shows the SQL Developer interface. The query editor contains three lines of SQL code:

```
1 • select * from employees
2 ✖ select * from departments
3 select Last_name, max(salary) from employees
```

The result grid below the query shows the following data:

| Last_name | max(salary) |
|-----------|-------------|
| King | 24000.00 |

19.



The screenshot shows the SQL Developer interface. The query editor contains five lines of SQL code:

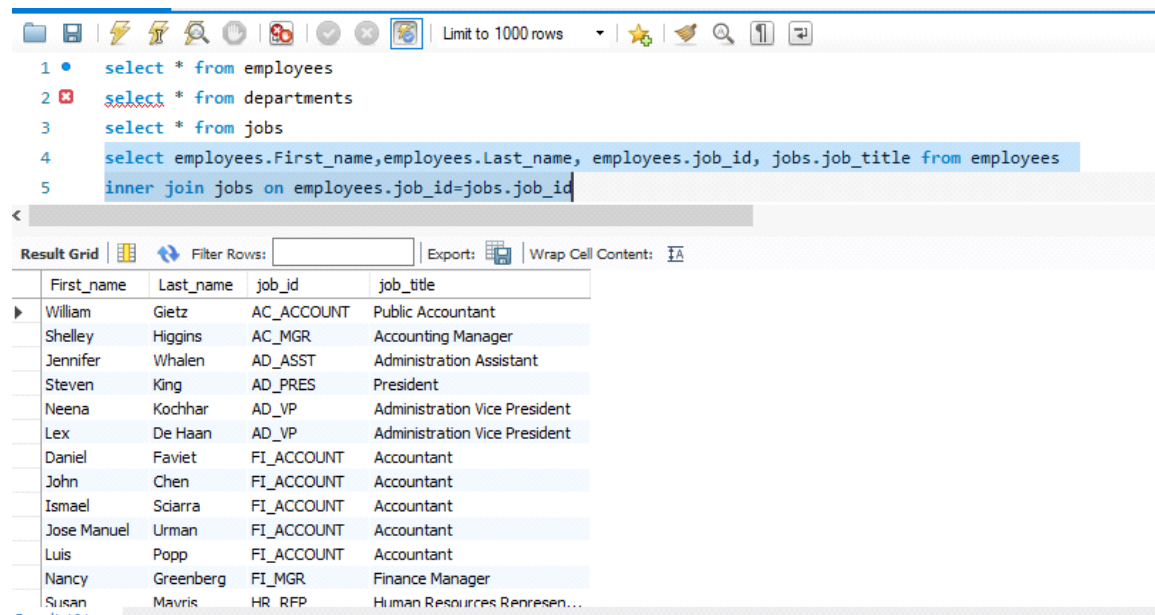
```
1 • select * from employees
2
3 ✖ select f,l,min,d_id
4 from (select min(salary) min, First_name f, Last_name l, department_id d_id
5 from employees group by department_id) emin
```

The result grid below the query shows the following data:

| f | l | min | d_id |
|-----------|-----------|----------|------|
| Kimberely | Grant | 7000.00 | NULL |
| Jennifer | Whalen | 4400.00 | 10 |
| Michael | Hartstein | 6000.00 | 20 |
| Den | Raphaely | 2500.00 | 30 |
| Susan | Mavris | 6500.00 | 40 |
| Matthew | Weiss | 2100.00 | 50 |
| Alexander | Hunold | 4200.00 | 60 |
| Hermann | Baer | 10000.00 | 70 |
| John | Russell | 6100.00 | 80 |
| Steven | King | 17000.00 | 90 |
| Nancy | Greenberg | 6900.00 | 100 |
| Shelley | Higgins | 8300.00 | 110 |

Result 102 x

20. find the employee names and their job id and job title by joining employee and jobs table



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and search. Below the toolbar, a list of five SQL statements is shown. The fifth statement is selected and highlighted in blue. Below the statements, a 'Result Grid' section displays the query results. The grid has four columns: 'First_name', 'Last_name', 'job_id', and 'job_title'. It contains 15 rows of data, with the first row highlighted in blue. The interface also includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox.

```
1 • select * from employees
2 ✖ select * from departments
3 select * from jobs
4 select employees.First_name,employees.Last_name, employees.job_id, jobs.job_title from employees
5 inner join jobs on employees.job_id=jobs.job_id
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: ☐

| First_name | Last_name | job_id | job_title |
|-------------|-----------|------------|-------------------------------|
| William | Gietz | AC_ACCOUNT | Public Accountant |
| Shelley | Higgins | AC_MGR | Accounting Manager |
| Jennifer | Whalen | AD_ASST | Administration Assistant |
| Steven | King | AD PRES | President |
| Neena | Kochhar | AD_VP | Administration Vice President |
| Lex | De Haan | AD_VP | Administration Vice President |
| Daniel | Faviet | FI_ACCOUNT | Accountant |
| John | Chen | FI_ACCOUNT | Accountant |
| Ismael | Sciarra | FI_ACCOUNT | Accountant |
| Jose Manuel | Urman | FI_ACCOUNT | Accountant |
| Luis | Popp | FI_ACCOUNT | Accountant |
| Nancy | Greenberg | FI_MGR | Finance Manager |
| Susan | Mavris | HR_RFP | Human Resources Represen... |