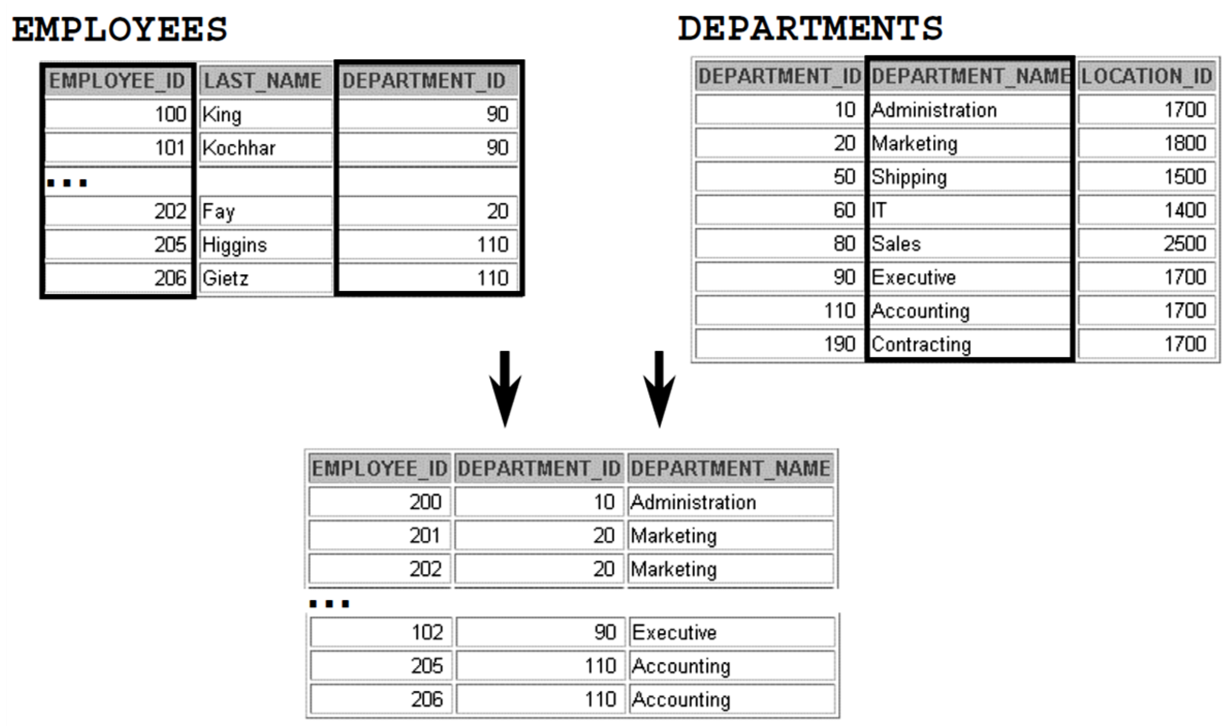
CSE 311L (Database Management System)

LAB-Week 04 (Part A)

# Displaying Data from Multiple Tables (Based on Company2.sql)

**Topics:**

- Obtaining Data from Multiple Tables
- Generating a Cartesian Product
- Retrieving Records with Equijoins
- Joining a Table to Itself
- Creating Joins with the ON Clause



EMPLOYEES

| EMPLOYEE_ID | LAST_NAME | DEPARTMENT_ID |
|---|---|---|
| 100 | King | 90 |
| 101 | Kochhar | 90 |
| ... | | |
| 202 | Fay | 20 |
| 205 | Higgins | 110 |
| 206 | Gietz | 110 |

DEPARTMENTS

| DEPARTMENT_ID | DEPARTMENT_NAME | LOCATION_ID |
|---|---|---|
| 10 | Administration | 1700 |
| 20 | Marketing | 1800 |
| 50 | Shipping | 1500 |
| 60 | IT | 1400 |
| 80 | Sales | 2500 |
| 90 | Executive | 1700 |
| 110 | Accounting | 1700 |
| 190 | Contracting | 1700 |

| EMPLOYEE_ID | DEPARTMENT_ID | DEPARTMENT_NAME |
|---|---|---|
| 200 | 10 | Administration |
| 201 | 20 | Marketing |
| 202 | 20 | Marketing |
| ... | | |
| 102 | 90 | Executive |
| 205 | 110 | Accounting |
| 206 | 110 | Accounting |

## Generating a Cartesian Product

```
SELECT last_name, department_name dept_name
FROM emps, depts;
```

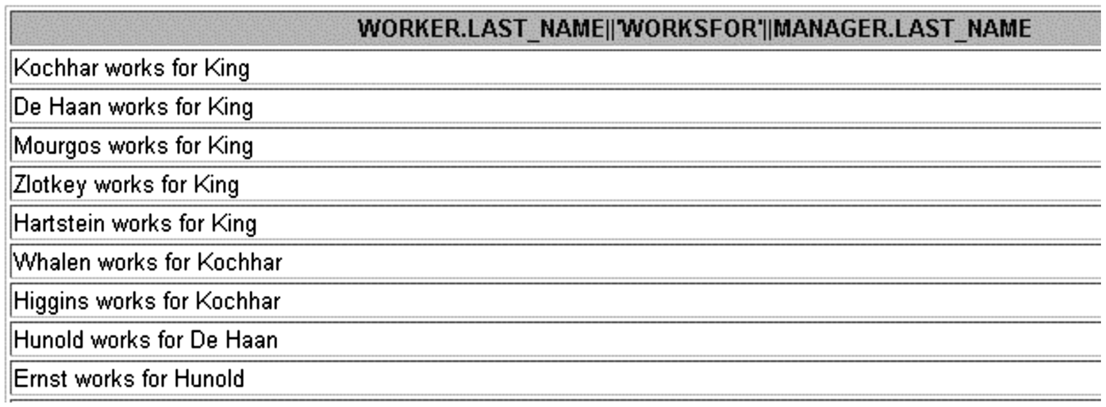
## Retrieving Records with Equijoins

```
SELECT e.employee_id, e.last_name, e.department_id,
d.department_id, d.location_id
FROM emps e , depts d
```

```
     WHERE e.department_id = d.department_id;
```

## Joining a Table to Itself

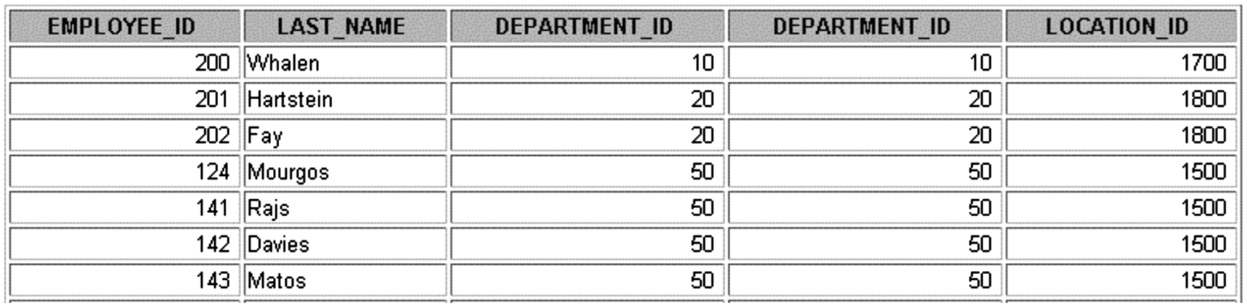
```
     SELECT worker.last_name || ' works for '
                        || manager.last_name
     FROM emps worker, emps manager
     WHERE worker.manager_id = manager.employee_id ;
```



| WORKER.LAST_NAME\|\|'WORKSFOR'\|\|MANAGER.LAST_NAME |
|---|
| Kochhar works for King |
| De Haan works for King |
| Mourgos works for King |
| Zlotkey works for King |
| Hartstein works for King |
| Whalen works for Kochhar |
| Higgins works for Kochhar |
| Hunold works for De Haan |
| Ernst works for Hunold |

## Creating Joins with the ON Clause

```
     SELECT e.employee_id, e.last_name, e.department_id,
     d.department_id, d.location_id
     FROM emps e JOIN depts d
     ON (e.department_id = d.department_id);
```



| EMPLOYEE_ID | LAST_NAME | DEPARTMENT_ID | DEPARTMENT_ID | LOCATION_ID |
|---|---|---|---|---|
| 200 | Whalen | 10 | 10 | 1700 |
| 201 | Hartstein | 20 | 20 | 1800 |
| 202 | Fay | 20 | 20 | 1800 |
| 124 | Mourgos | 50 | 50 | 1500 |
| 141 | Rajs | 50 | 50 | 1500 |
| 142 | Davies | 50 | 50 | 1500 |
| 143 | Matos | 50 | 50 | 1500 |

**Activity 01:**

Write a query to display the last name, department number, and department name for all employees.

**Activity 02:**

Write a query to display the employee last name, department name, location ID, and city of all employees who earn a commission.

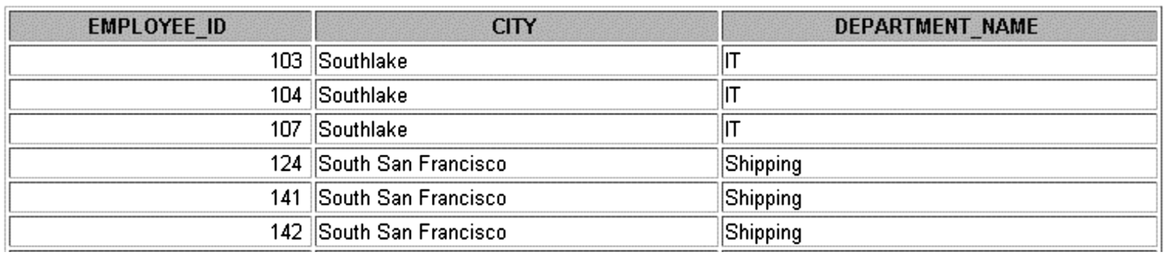
**Topics:**

After completing this lesson, you should be able to do:

- Creating Three-Way Joins with the ON Clause
- `LEFT OUTER JOIN`
- `RIGHT OUTER JOIN`
- `FULL OUTER JOIN`
- Additional Conditions

## Creating Three-Way Joins with the ON Clause

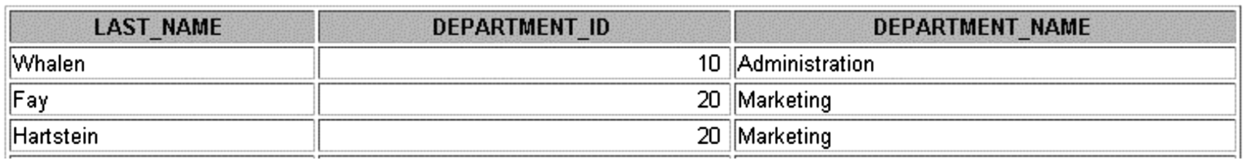
```
SELECT employee_id, city, department_name
FROM emps e
JOIN depts d
ON d.department_id = e.department_id
JOIN locs l
ON d.location_id = l.location_id;
```



| EMPLOYEE_ID | CITY | DEPARTMENT_NAME |
|---|---|---|
| 103 | Southlake | IT |
| 104 | Southlake | IT |
| 107 | Southlake | IT |
| 124 | South San Francisco | Shipping |
| 141 | South San Francisco | Shipping |
| 142 | South San Francisco | Shipping |

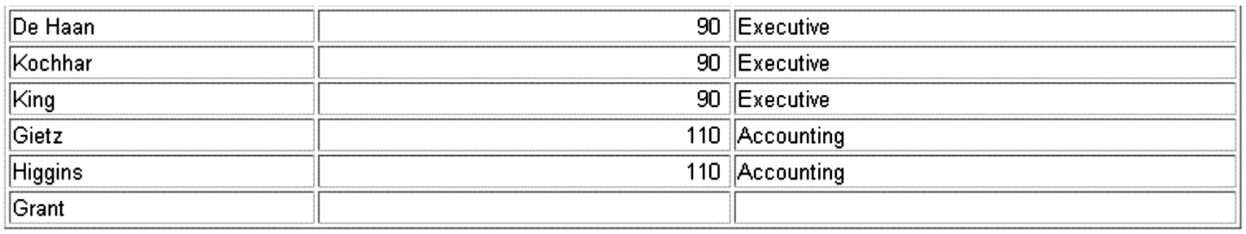
## LEFT OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM emps e
LEFT OUTER JOIN depts d
ON (e.department_id = d.department_id) ;
```



| LAST_NAME | DEPARTMENT_ID | DEPARTMENT_NAME |
|---|---|---|
| Whalen | 10 | Administration |
| Fay | 20 | Marketing |
| Hartstein | 20 | Marketing |

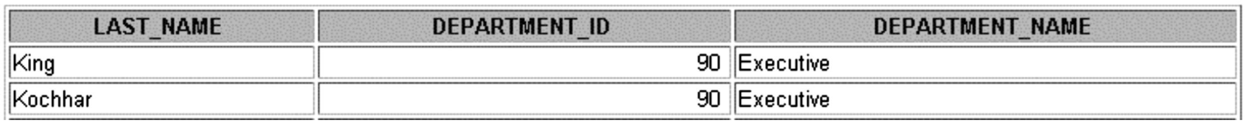
**..........**



| | | |
|---|---|---|
| De Haan | 90 | Executive |
| Kochhar | 90 | Executive |
| King | 90 | Executive |
| Gietz | 110 | Accounting |
| Higgins | 110 | Accounting |
| Grant | | |

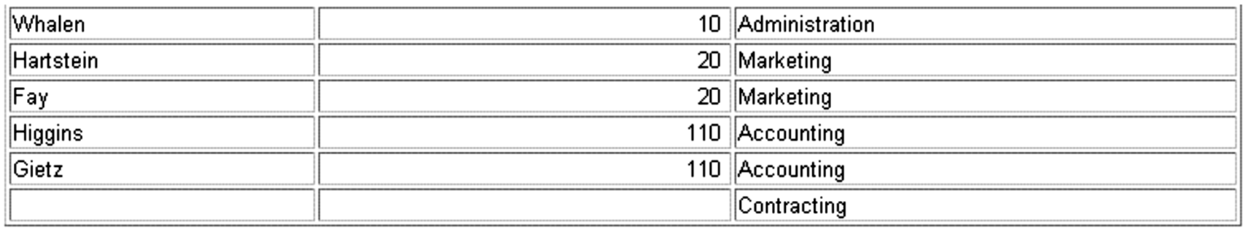
**RIGHT OUTER JOIN**

```
SELECT e.last_name, e.department_id, d.department_name
FROM emps e
RIGHT OUTER JOIN depts d
ON (e.department_id = d.department_id) ;
```



| LAST_NAME | DEPARTMENT_ID | DEPARTMENT_NAME |
|---|---|---|
| King | 90 | Executive |
| Kochhar | 90 | Executive |

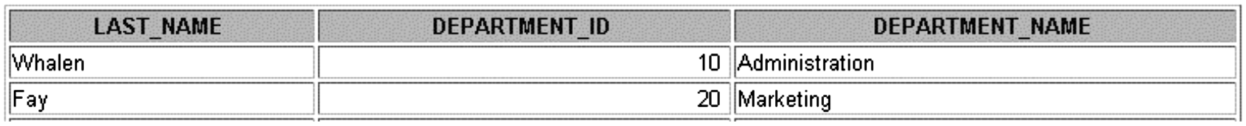
……….



| | | |
|---|---|---|
| Whalen | 10 | Administration |
| Hartstein | 20 | Marketing |
| Fay | 20 | Marketing |
| Higgins | 110 | Accounting |
| Gietz | 110 | Accounting |
| | | Contracting |

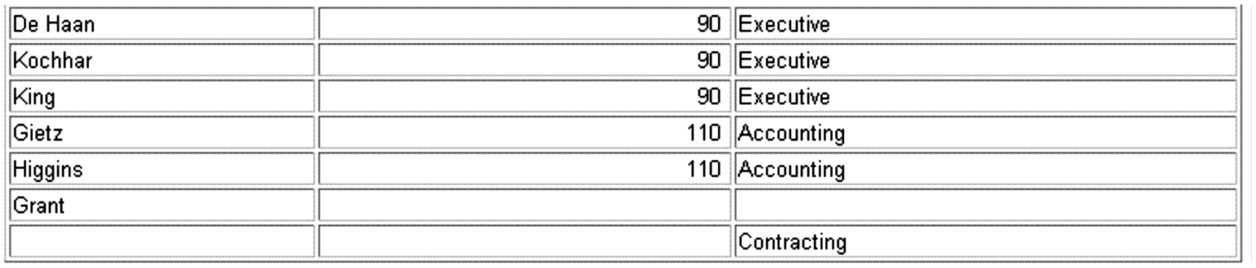
**FULL OUTER JOIN**

```
SELECT e.last_name, e.department_id, d.department_name
FROM emps e
FULL OUTER JOIN depts d
ON (e.department_id = d.department_id) ;
```



| LAST_NAME | DEPARTMENT_ID | DEPARTMENT_NAME |
|---|---|---|
| Whalen | 10 | Administration |
| Fay | 20 | Marketing |

…

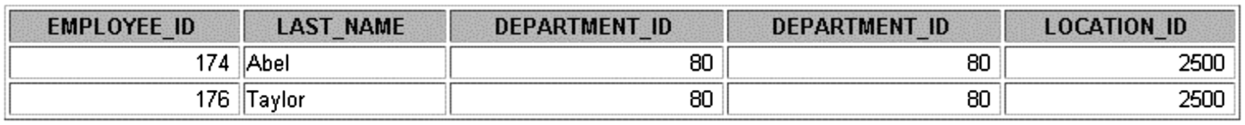


| | | |
|---|---|---|
| De Haan | 90 | Executive |
| Kochhar | 90 | Executive |
| King | 90 | Executive |
| Gietz | 110 | Accounting |
| Higgins | 110 | Accounting |
| Grant | | |
| | | Contracting |

**Additional Conditions**

```
SELECT e.employee_id, e.last_name, e.department_id,
d.department_id, d.location_id
FROM emps e JOIN depts d
```

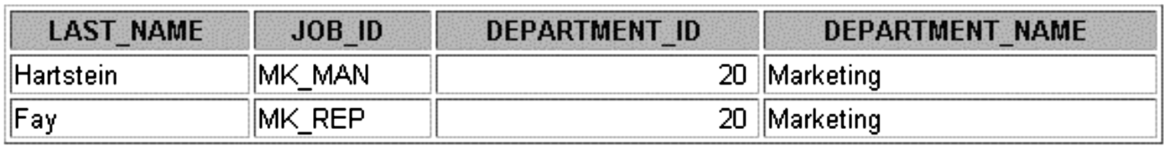
```
    ON (e.department_id = d.department_id)
    AND e.manager_id = 149 ;
```



| EMPLOYEE_ID | LAST_NAME | DEPARTMENT_ID | DEPARTMENT_ID | LOCATION_ID |
|---|---|---|---|---|
| 174 | Abel | 80 | 80 | 2500 |
| 176 | Taylor | 80 | 80 | 2500 |

**Activity 01:**

Write a query to display the last name, job, department number, and department name for all employees who work in Toronto.



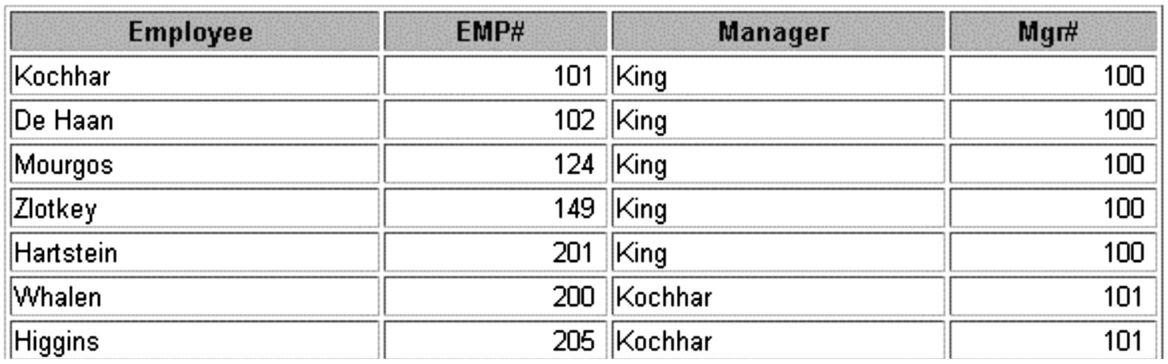
| LAST_NAME | JOB_ID | DEPARTMENT_ID | DEPARTMENT_NAME |
|---|---|---|---|
| Hartstein | MK_MAN | 20 | Marketing |
| Fay | MK_REP | 20 | Marketing |

**Activity 02:**

Display the last name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions.Title.

**Activity 03:**

Display the employee last name and employee number along with their manager's last name and manager number. Label the columns Employee, Emp#, Manager, and Mgr#, respectively.



| Employee | EMP# | Manager | Mgr# |
|---|---|---|---|
| Kochhar | 101 | King | 100 |
| De Haan | 102 | King | 100 |
| Mourgos | 124 | King | 100 |
| Zlotkey | 149 | King | 100 |
| Hartstein | 201 | King | 100 |
| Whalen | 200 | Kochhar | 101 |
| Higgins | 205 | Kochhar | 101 |

CSE 311L(Database Management System)

LAB-Week 04 (Part B)

# Aggregating Data Using Group Functions

**Topics:**

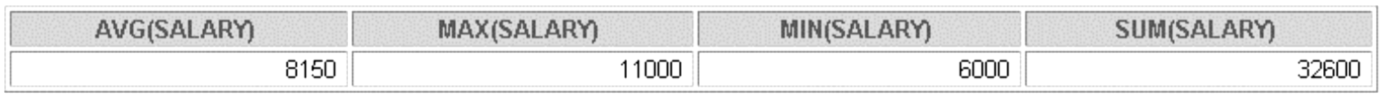
- Types of Group Functions
- Using the AVG and SUM Functions
- Using the MIN and MAX Functions
- Using the COUNT Function
- Using the GROUP BY Clause

## Types of Group Functions

- **AVG**
- **COUNT**
- **MAX**
- **STDDEV**
- **MIN**
- **SUM**
- **VARIANCE**

## Using the AVG and SUM Functions

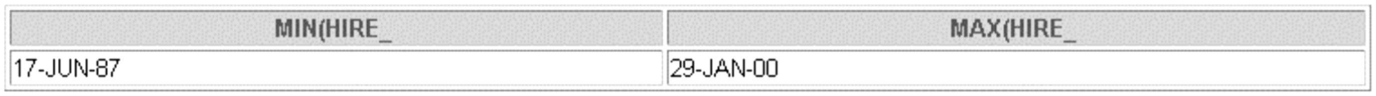
```
SELECT AVG(salary), MAX(salary),
MIN(salary), SUM(salary)
FROM emps
WHERE job_id LIKE '%REP%';
```



| AVG(SALARY) | MAX(SALARY) | MIN(SALARY) | SUM(SALARY) |
|---|---|---|---|
| 8150 | 11000 | 6000 | 32600 |

## Using the MIN and MAX Functions

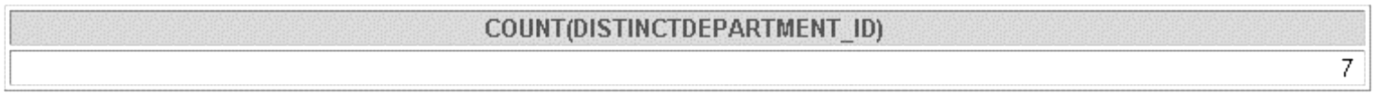
```
SELECT MIN(hire_date), MAX(hire_date)
FROM emps;
```



| MIN(HIRE_ | MAX(HIRE_ |
|---|---|
| 17-JUN-87 | 29-JAN-00 |

## Using the COUNT Function

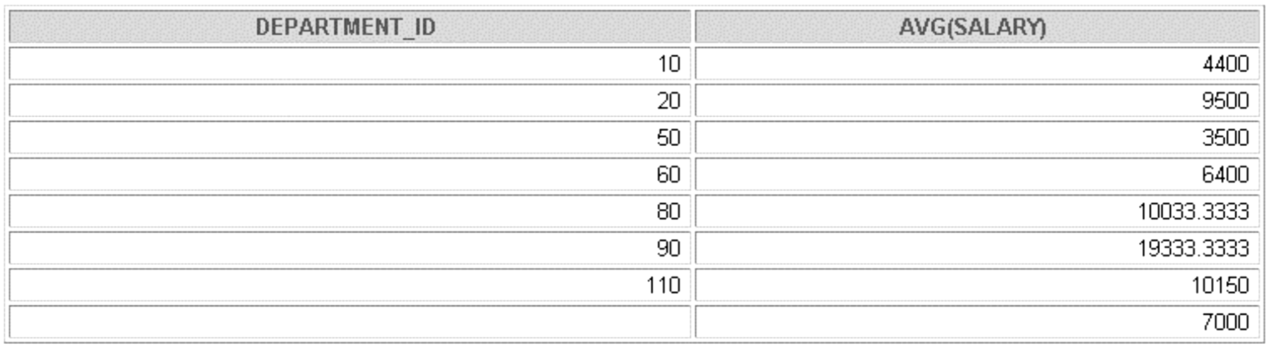
```
SELECT COUNT(DISTINCT department_id)
FROM emps;
```



| COUNT(DISTINCTDEPARTMENT_ID) |
|---:|
| 7 |

## Using the GROUP BY Clause

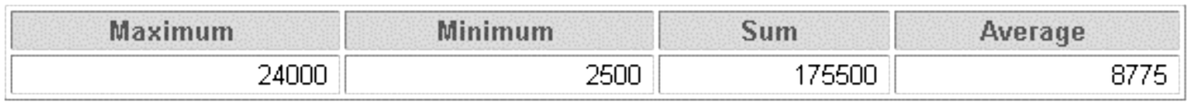
```
SELECT department_id, AVG(salary)
FROM emps
GROUP BY department_id ;
```



| DEPARTMENT_ID | AVG(SALARY) |
|---:|---:|
| 10 | 4400 |
| 20 | 9500 |
| 50 | 3500 |
| 60 | 6400 |
| 80 | 10033.3333 |
| 90 | 19333.3333 |
| 110 | 10150 |
|  | 7000 |

**Activity 01:**

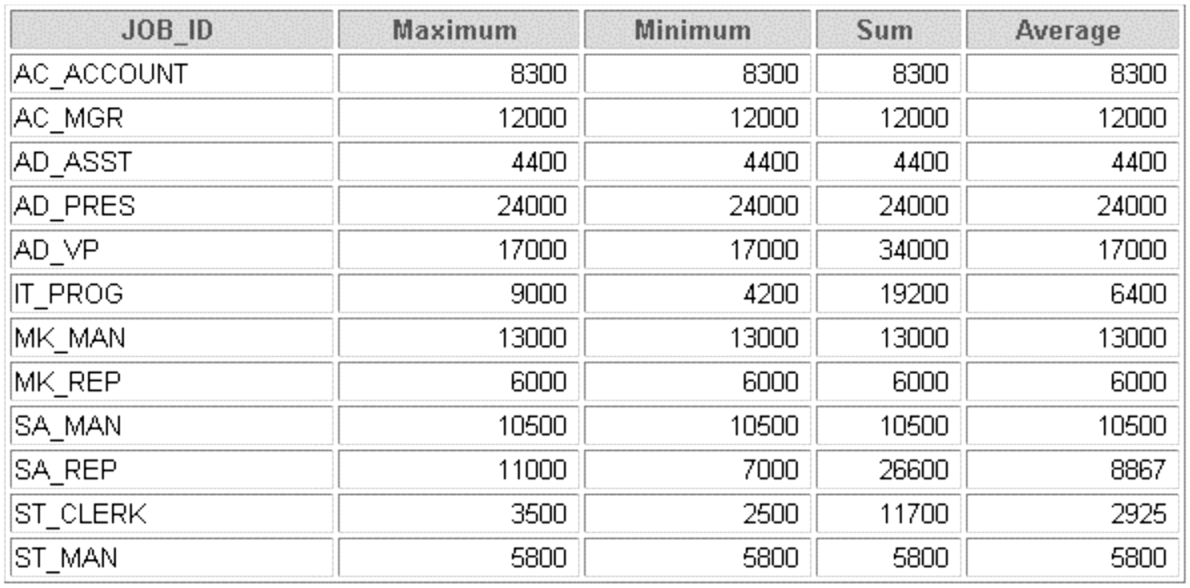
Display the highest, lowest, sum, and average salary of all employees. Label the columns
Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole
number.



| Maximum | Minimum | Sum | Average |
|---:|---:|---:|---:|
| 24000 | 2500 | 175500 | 8775 |

**Activity 02:**

display the minimum, maximum, sum, and average salary
for each job type



| JOB_ID | Maximum | Minimum | Sum | Average |
|---|---|---|---|---|
| AC_ACCOUNT | 8300 | 8300 | 8300 | 8300 |
| AC_MGR | 12000 | 12000 | 12000 | 12000 |
| AD_ASST | 4400 | 4400 | 4400 | 4400 |
| AD_PRES | 24000 | 24000 | 24000 | 24000 |
| AD_VP | 17000 | 17000 | 34000 | 17000 |
| IT_PROG | 9000 | 4200 | 19200 | 6400 |
| MK_MAN | 13000 | 13000 | 13000 | 13000 |
| MK_REP | 6000 | 6000 | 6000 | 6000 |
| SA_MAN | 10500 | 10500 | 10500 | 10500 |
| SA_REP | 11000 | 7000 | 26600 | 8867 |
| ST_CLERK | 3500 | 2500 | 11700 | 2925 |
| ST_MAN | 5800 | 5800 | 5800 | 5800 |

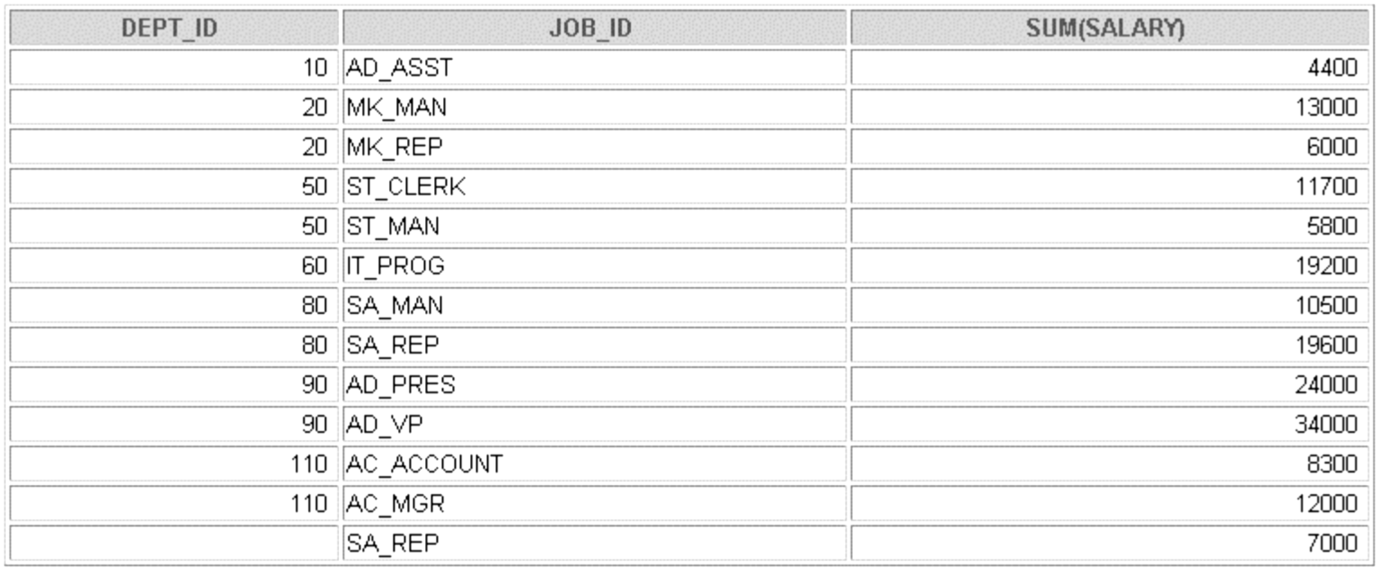
## Aggregating Data Using Group Functions

**Topics:**

- Using the GROUP BY Clause on Multiple Columns
- Illegal Queries Using Group Functions
- Excluding Group Results: The HAVING Clause
- Nesting Group Functions

### Using the GROUP BY Clause on Multiple Columns

```
SELECT department_id dept_id, job_id, SUM(salary)
FROM emps
GROUP BY department_id, job_id ;
```



| DEPT_ID | JOB_ID | SUM(SALARY) |
|---|---|---|
| 10 | AD_ASST | 4400 |
| 20 | MK_MAN | 13000 |
| 20 | MK_REP | 6000 |
| 50 | ST_CLERK | 11700 |
| 50 | ST_MAN | 5800 |
| 60 | IT_PROG | 19200 |
| 80 | SA_MAN | 10500 |
| 80 | SA_REP | 19600 |
| 90 | AD_PRES | 24000 |
| 90 | AD_VP | 34000 |
| 110 | AC_ACCOUNT | 8300 |
| 110 | AC_MGR | 12000 |
|  | SA_REP | 7000 |

**What is wrong with them?!!**

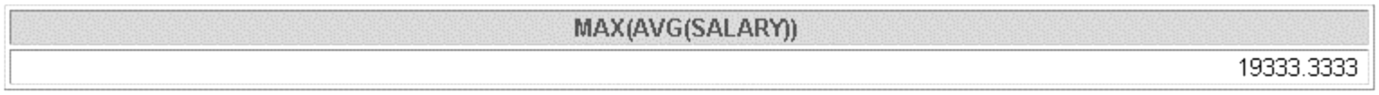
```
>SELECT department_id, COUNT(last_name)
 FROM emps;

>SELECT department_id, AVG(salary)
 FROM emps
 WHERE AVG(salary) > 8000
 GROUP BY department_id;
```

**Excluding Group Results: The HAVING Clause**

```
SELECT job_id, SUM(salary) PAYROLL
FROM emps
WHERE job_id NOT LIKE '%REP%'
GROUP BY job_id
HAVING SUM(salary) > 13000
ORDER BY SUM(salary);
```

**Nesting Group Functions**

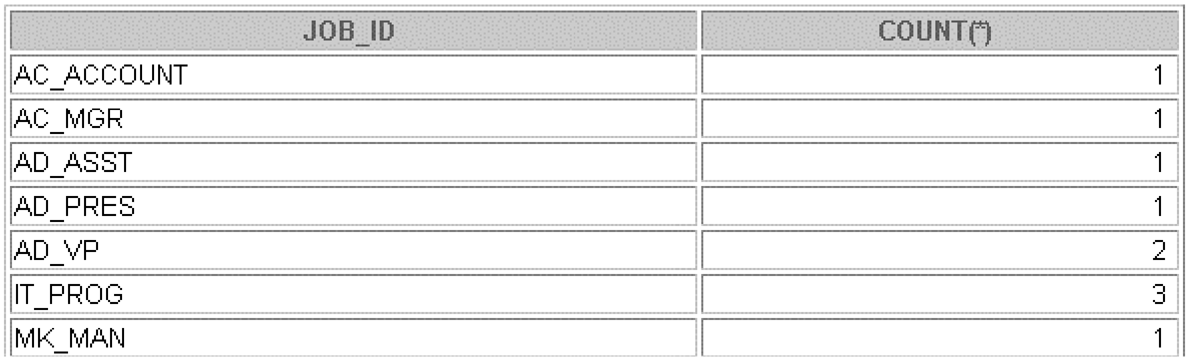
```
SELECT MAX(AVG(salary))
FROM emps
GROUP BY department_id;
```



| MAX(AVG(SALARY)) |
|---|
| 19333.3333 |

**Activity 01:**

Write a query to display the number of people with the same job.

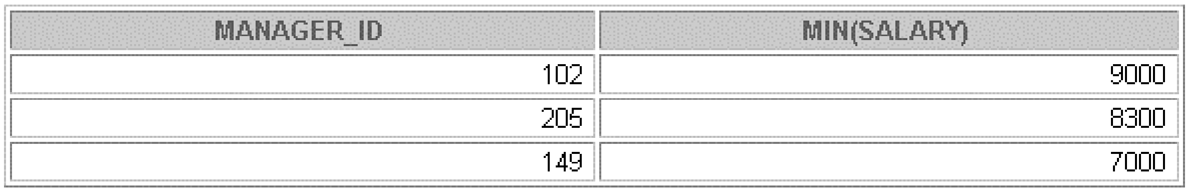


| JOB_ID | COUNT(*) |
|---|---|
| AC_ACCOUNT | 1 |
| AC_MGR | 1 |
| AD_ASST | 1 |
| AD_PRES | 1 |
| AD_VP | 2 |
| IT_PROG | 3 |
| MK_MAN | 1 |

**Activity 02:**

Display the manager number and the salary of the lowest paid employee for that manager.

Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is $6,000 or less. Sort the output in descending order of salary.



| MANAGER_ID | MIN(SALARY) |
|---|---|
| 102 | 9000 |
| 205 | 8300 |
| 149 | 7000 |

**Home Work:**

Write a query to display each department's name, location, number of employees, and the

average salary for all employees in that department. Label the columns Name, Location, Number of People, and Salary, respectively. Round the average salary to two decimal places.