CSE 311L (Database Management System)
LAB-Week 04 (Part A)

Instructor: ***Asif*** **Ahmed Neloy**

Displaying Data from Multiple Tables

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 employees, departments;
```

Retrieving Records with Equijoins

```
SELECT e.employee_id, e.last_name, e.department_id,
d.department_id, d.location_id
FROM employees e , departments d
WHERE e.department_id = d.department_id;
```

Joining a Table to Itself

```
SELECT worker.last_name || ' works for '
                    || manager.last_name
FROM employees worker, employees 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 employees e JOIN departments 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

CSE 311L(Database Management System)
LAB-Week 04 (Part B)

***Instructor: Asif* Ahmed Neloy**

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 employees e
JOIN departments d
ON d.department_id = e.department_id
JOIN locations 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 employees e
LEFT OUTER JOIN departments 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 employees e
RIGHT OUTER JOIN departments 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 employees e
FULL OUTER JOIN departments 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 employees e JOIN departments 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

CSE 311L(Database Management System)
LAB-Week 05 (Part A)

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 employees
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 employees;
```

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

Using the COUNT Function

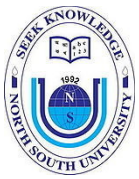
```
SELECT COUNT(DISTINCT department_id)
FROM employees;
```

COUNT(DISTINCTDEPARTMENT_ID)
7

Using the GROUP BY Clause

```
SELECT department_id, AVG(salary)
FROM employees
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



CSE 311L(Database Management System)
LAB-Week 05 (Part B)

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 employees
```

```
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 employees;

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

Excluding Group Results: The HAVING Clause

```
SELECT job_id, SUM(salary) PAYROLL
FROM employees
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 employees
GROUP BY department_id;
```

MAX(AVG(SALARY))
19333.3333