

# Database I

(60012301-1)

## Lecture 8: Data Manipulation Language

Dr. Obead Alhadreti



# Data Manipulation Language

---

- ▶ A DML statement is executed when you:
  - Add new rows to a table
  - Modify existing rows in a table
  - Remove existing rows from a table
- ▶ A *transaction* consists of a collection of DML statements that form a logical unit of work.

# Insert a New Row to a Table

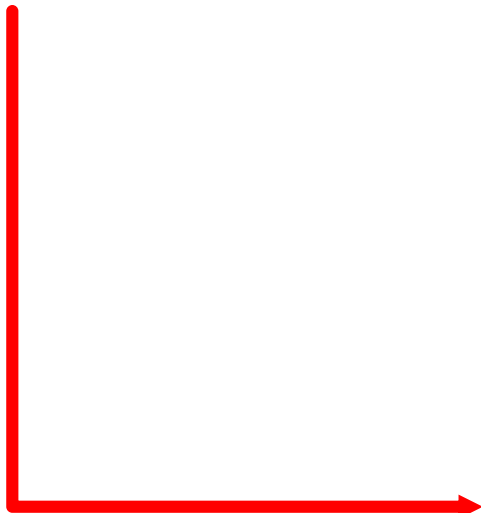
70	Public Relations	100	1700
----	------------------	-----	------

**New  
row**

## DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

**Insert new row  
into the  
DEPARTMENTS table**



DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700
70	Public Relations	100	1700

# INSERT INTO Statement

---

- ▶ To add a **new row** to a table by using the **INSERT INTO** statement:

```
INSERT INTO table_Name  
VALUES      ( 'value1', 'value2', 'value3', ..... );
```

- ▶ With this syntax, only one row is inserted at a time.

# INSERT INTO Statement

---

- ▶ Insert a new row containing **values** for each column.
- ▶ List values in the default order of the columns in the table.

```
INSERT INTO EMPLOYEE
VALUES
('John', 'B', 'Smith', '123456789', '1965-01-09',
'731 Fondren, Houston, TX', 'M', 30000, '333445555', 5);
```

- ▶ Enclose character and date values in single quotation marks.

# Inserting Special Values

---

- ▶ The **SYSDATE** function records the current date and time.

```
INSERT INTO employees
VALUES (113, 'Louis', 'Popp', 'LPOPP', '515.124.4567',
       SYSDATE, 'AC_ACCOUNT', 6900,
       NULL, 205, 100);
```

1 row created.

# Inserting Specific Date Values

---

- ▶ Add a new employee.

```
INSERT INTO employees
VALUES      (114,
            'Den', 'Raphealy',
            'DRAPHEAL', '515.127.4561',
            TO_DATE('FEB 3, 1999', 'MON DD, YYYY'),
            'AC_ACCOUNT', 11000, NULL, 100, 30);
```

1 row created.

- ▶ Verify your addition


EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_P
114	Den	Raphealy	DRAPHEAL	515.127.4561	03-FEB-99	AC_ACCOUNT	11000	

# Update Data in a Table

## EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSION_P
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	60	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	60	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	60	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

Update rows in the **EMPLOYEES** table:



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSIO
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	30	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	30	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	30	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	



# UPDATE Statement

---

- ▶ Modify existing rows with the **UPDATE** statement:

```
UPDATE      table_Name
SET         column1 = value1, column2 = value2, ...
WHERE      condition;
```

- ▶ Update more than one row at a time (if required).

# Update Statement

---

- ▶ **Specific row is modified if you specify the WHERE clause:**

```
UPDATE employees
SET    department_id = 70
WHERE  employee_id = 113;
1 row updated.
```

- ▶ **All rows in the table are modified if you omit the WHERE clause:**

```
UPDATE    copy_emp
SET       department_id = 110;
22 rows updated.
```

- ▶ **Updates the first customer (CustomerID=1) with a new contact person and a new city**

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

# DELETE Statement

---

- ▶ **Syntax**

```
DELETE FROM table_name WHERE condition;
```

- ▶ **Note:** Be careful when deleting records in a table!  
Notice the **WHERE** clause in the **DELETE** statement.
- ▶ The **WHERE** clause specifies which record(s) should be deleted.
- ▶ If you omit the **WHERE** clause, all records in the table will be deleted!

# DELETE Examples

---

- ▶ To delete the customer "**Alfreds Futterkiste**" from the "Customers" table

```
DELETE FROM Customers  
WHERE CustomerName='Alfreds Futterkiste';
```

- ▶ To delete all rows in the "**Customers**" table, without deleting the table

```
DELETE FROM Customers;
```

# Select One Attribute

---

- ▶ Syntax for selecting only **one attribute** for all the records.

```
SELECT attribute_name  
FROM table_name;
```

- ▶ Example to retrieve all EMPLOYEE Ssns.

```
SELECT SSn  
FROM EMPLOYEE;
```

(e)

<u>E.Fname</u>
123456789
333445555
999887777
987654321
666884444
453453453
987987987
888665555

# Select Statement

---

- ▶ Syntax for selecting more than one **attribute** for all the records

```
SELECT  column1, column2, ...  
FROM    table_name;
```

- ▶ **Example**

```
SELECT  Bdate, Address  
FROM    employee
```

- ▶ Retrieve **all** attribute values in EMPLOYEE table

```
SELECT  *  
FROM    EMPLOYEE
```

# Distinct to Remove the Duplicates

---

- ▶ Retrieve the salary of every employee.

```
SELECT ALL Salary
```

```
FROM EMPLOYEE;
```

- ▶ Retrieve all distinct salary values.

```
SELECT DISTINCT Salary
```

```
FROM EMPLOYEE;
```

(a)	Salary	(b)	Salary
	30000		30000
	40000		40000
	25000		25000
	43000		43000
	38000		38000
	25000		55000
	25000		
	55000		

# WHERE Clause

---

- ▶ **WHERE** clause is used to specify a condition while fetching the data from a single table or by joining with multiple tables.
- ▶ If the given condition is satisfied, then only it returns a specific value from the table.

- ▶ **Syntax**

```
SELECT column1, column2, columnN  
FROM table_name  
WHERE [condition]
```

- ▶ You can specify a condition using the comparison or logical operators like  $>$ ,  $<$ ,  $=$ , LIKE, NOT, etc.



# Where Clause

---

- ▶ To retrieve **all attribute** values of any employee who works in DEPARTMENT number 5

```
SELECT *  
FROM EMPLOYEE  
WHERE Dno = 5;
```

<u>Fname</u>	<u>Minit</u>	<u>Lname</u>	<u>Ssn</u>	<u>Rdate</u>	<u>Address</u>	<u>Sex</u>	<u>Salary</u>	<u>Super_ssn</u>	<u>Dno</u>
John	B	Smith	123456789	1965-09-01	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

# Where Clause

---

- ▶ Retrieve the **birth date** and **address** of the employees whose name is 'John B. Smith'

```
SELECT Bdate, Address
FROM employee
WHERE Fname = 'John' and
Minit = 'B' and Lname = 'Smith';
```

(a)

<u>Bdate</u>	<u>Address</u>
1965-01-09	731Fondren, Houston, TX

# Where Clause

---

- ▶ Retrieve the **name** and **address** of all employees who work for the 'Research' department

```
SELECT Fname, Lname, Address
from employee, department
where Dname = 'Research' and Dnumber = Dno;
```

(b)

<u>Fname</u>	<u>Lname</u>	<u>Address</u>
John	Smith	731 Fondren, Houston, TX
Franklin	Wong	638 Voss, Houston, TX
Ramesh	Narayan	975 Fire Oak, Humble, TX
Joyce	English	5631 Rice, Houston, TX

# Where Clause

---

- ▶ To select all fields from "Customers" where country is "Germany" OR "Spain"

```
SELECT *  
FROM Customers  
WHERE Country='Germany' OR Country='Spain';
```

- ▶ Selects all fields from "Customers" where country is NOT "Germany"

```
SELECT *  
FROM Customers  
WHERE NOT Country='Germany';
```

## Example Using 3 Tables

---

- ▶ For every project located in 'Stafford', list the project number, department number, manager's last name, address, and birth date.

```
SELECT Pnumber, Dnum, Lname, Address, Bdate
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE Dnum = Dnumber AND Mgr_ssn = SSn AND
Plocation = 'Stafford';
```

# IN Operator

---

## ▶ IN Syntax

```
ELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

### **OR**

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (SELECT STATEMENT);
```

- ▶ To selects all customers that are located in "Germany", "France" or "UK"

```
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');
```

# IN Operator Examples

---

- ▶ To select all customers that are **NOT** located in "Germany", "France" or "UK"

```
SELECT *  
FROM Customers  
WHERE Country NOT IN ('Germany', 'France', 'UK');
```

- ▶ To select all customers that are from the same countries as the suppliers

```
SELECT *  
FROM Customers  
WHERE Country IN (SELECT Country FROM Suppliers);
```

# IN Operator Example

---

- ▶ Retrieve the name of each employee who has a dependent with the same first name and is the same sex as the employee

```
SELECT E.Fname, E.Lname
FROM   EMPLOYEE E
WHERE  E.Ssn IN
(
SELECT Essn
FROM   DEPENDENT D
WHERE  E.FName = D.Dependent_name
AND   E.Sex = D.Sex
);
```

(c)

Fname	Lname
-------	-------

**Result: No Row match**



# COMPANY Database State

(a)

<u>Edate</u>	<u>Address</u>
1965-01-09	731Fondren, Houston, TX

(b)

<u>Fname</u>	<u>Lname</u>	<u>Address</u>
John	Smith	731 Fondren, Houston, TX
Franklin	Wong	638 Voss, Houston, TX
Ramesh	Narayan	975 Fire Oak, Humble, TX
Joyce	English	5631 Rice, Houston, TX

(c)

<u>Pnumber</u>	<u>Dnum</u>	<u>Lname</u>	<u>Address</u>	<u>Edate</u>
10	4	Wallace	291Berry, Bellaire, TX	1941-06-20
30	4	Wallace	291Berry, Bellaire, TX	1941-06-20

(f)

<u>Ssn</u>	<u>Dname</u>
123456789	Research
333445555	Research
999887777	Research
987654321	Research
666884444	Research
453453453	Research
987987987	Research
888665555	Research
123456789	Administration
333445555	Administration
999887777	Administration
987654321	Administration
666884444	Administration
453453453	Administration
987987987	Administration
888665555	Administration
123456789	Headquarters
333445555	Headquarters
999887777	Headquarters
987654321	Headquarters
666884444	Headquarters
453453453	Headquarters
987987987	Headquarters
888665555	Headquarters

(d)

<u>E.Fname</u>	<u>E.Lname</u>	<u>S.Fname</u>	<u>S.Lname</u>
John	Smith	Franklin	Wong
Franklin	Wong	James	Borg
Alicia	Zelaya	Jennifer	Wallace
Jennifer	Wallace	James	Borg
Ramesh	Narayan	Franklin	Wong
Joyce	English	Franklin	Wong
Ahmad	Jabbar	Jennifer	Wallace

(e)

<u>E.Fname</u>
123456789
333445555
999887777
987654321
666884444
453453453
987987987
888665555

(g)

<u>Fname</u>	<u>Minit</u>	<u>Lname</u>	<u>Ssn</u>	<u>Edate</u>	<u>Address</u>	<u>Sex</u>	<u>Salary</u>	<u>Super_ssn</u>	<u>Dno</u>
John	B	Smith	123456789	1965-09-01	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

# Select the Cartesian Product Between Two Relations

- Suppose  $A = \{a, b, c\}$  and  $B = \{1, 2, 3\}$   
 $A \times B = \{(a, 1), (a, 2), (a, 3), (b, 1), (b, 2), (b, 3), (c, 1), (c, 2), (c, 3)\}$

- Retrieve all combinations of EMPLOYEE Ssn and DEPARTMENT Dname

```
SELECT Ssn, Dname  
FROM EMPLOYEE, DEPARTMENT;
```

(f)

Ssn	Dname
123456789	Research
333445555	Research
999887777	Research
987654321	Research
666884444	Research
453453453	Research
987987987	Research
888665555	Research
123456789	Administration
333445555	Administration
999887777	Administration
987654321	Administration
666884444	Administration
453453453	Administration
987987987	Administration
888665555	Administration
123456789	Headquarters
333445555	Headquarters
999887777	Headquarters
987654321	Headquarters
666884444	Headquarters
453453453	Headquarters
987987987	Headquarters
888665555	Headquarters

# IS NULL Value

---

- ▶ Syntax to test for **Null** value:

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;
```

- ▶ Retrieve the names of all employees who do not have supervisors

```
SELECT Fname, Lname
FROM EMPLOYEE
WHERE Super_ssn IS NULL;
```

(d)

Fname	Lname
James	Borg

# Matching Substring

---

- ▶ Partial strings are specified using two reserved characters:  
**%** replaces an arbitrary number of zero or more characters.
- ▶ **%???%** wildcard matching, anything with **???** as a substring.

```
SELECT Fname, Lname  
FROM EMPLOYEE  
WHERE Address LIKE '%Houston, TX%';
```

- ▶ The result is 5 rows.

# BETWEEN, AND

---

## ▶ Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

## ▶ Example

```
SELECT *
FROM EMPLOYEE
WHERE (Salary BETWEEN 30000 AND 40000)
AND (Dno = 5);
```

**Result is 3 rows**

# Order by

---

- ▶ The **ORDER BY** keyword sorts the records in ascending order by default.
- ▶ To sort the records in descending order, use the DESC keyword.

- ▶ **Syntax**

```
SELECT column1, column2, ...  
FROM table_name  
ORDER BY column1, column2, ... ASC|DESC;
```

- ▶ **Example**

```
SELECT *  
FROM Customers  
ORDER BY Country DESC;
```

To select all customers from the "Customers" table, sorted **DESCENDING** by the "Country" column

## Example Order by Using two table

---

```
SELECT D.Dname, E.Lname, E.Fname, P.Pname
FROM   DEPARTMENT D, EMPLOYEE E,
WORKS_ON W, PROJECT P
WHERE  D.Dnumber = E.Dno AND E.Ssn = W.Essn
AND W.Pno = P.Pnumber
ORDER BY D.Dname, E.Lname, E.Fname;
ORDER BY D.Dname DESC, E.Lname ASC, E.Fname ASC
```

- ▶ The result is 16 rows

# SUM, MAX, MIN, AVG

---

## ▶ SUM Syntax

```
SELECT SUM(column_name)  
FROM table_name  
WHERE condition;
```

## ▶ MAX Syntax

```
SELECT MAX(column_name)  
FROM table_name  
WHERE condition;
```

## ▶ MIN Syntax

```
SELECT MIN(column_name)  
FROM table_name  
WHERE condition;
```



# SUM, MAX, MIN, AVG

---

## ▶ **AVG** Syntax

```
SELECT AVG(column_name)  
FROM table_name  
WHERE condition;
```

## ▶ **Example:**

```
SELECT SUM(Salary), MAX(Salary),  
MIN(Salary), AVG(Salary)  
FROM EMPLOYEE;
```

## ▶ To limit the digits after period, use round (AVG(salary), 2)

# COUNT

---

## ▶ COUNT Syntax

```
SELECT COUNT(column_name)  
FROM table_name  
WHERE condition;
```

- ▶ The **COUNT()** function returns the number of rows that matches a specified criteria.

# GROUP by

---

- ▶ The **GROUP BY** statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

- ▶ Syntax

```
SELECT column_name (s)
FROM table_name
WHERE condition
GROUP BY column_name (s)
ORDER BY column_name (s) ;
```

- ▶ The **GROUP BY** statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

# GROUP by Examples

---

- ▶ To lists the number of customers in each country

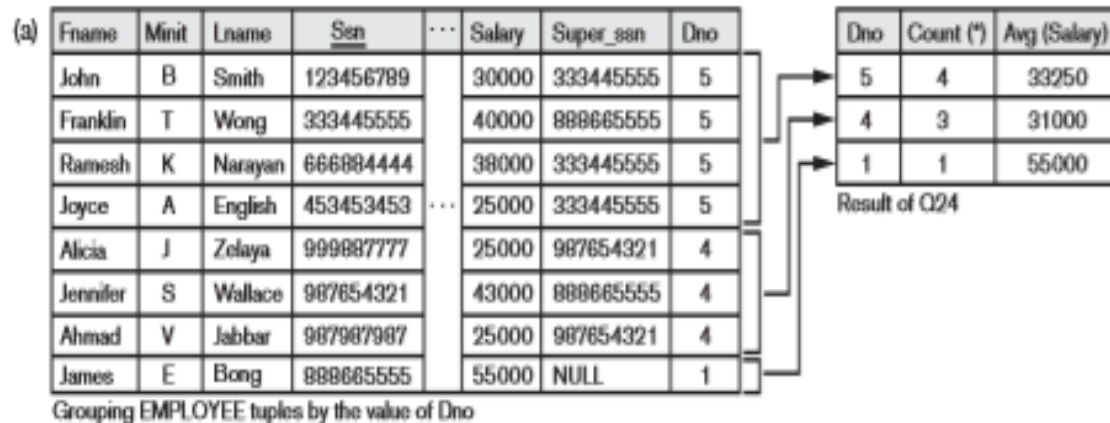
```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

- ▶ To lists the number of customers in each country, sorted high to low

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
ORDER BY COUNT(CustomerID) DESC;
```

# COUNT, GROUP by Example

```
SELECT Dno, COUNT (*),  
AVG (Salary)  
FROM EMPLOYEE  
GROUP BY Dno;
```



# HAVING

---

- ▶ **Syntax**

```
SELECT column_name (s)  
FROM table_name  
WHERE condition  
GROUP BY column_name (s)  
HAVING condition  
ORDER BY column_name (s);
```

- ▶ The **HAVING** clause was added to SQL because the **WHERE** keyword could not be used with aggregate functions(COUNT, MAX, MIN, SUM, AVG).

# HAVING Examples

---

- ▶ To lists the number of customers in each country. Only include countries with more than 5 customers:

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5;
```

- ▶ lists the number of customers in each country, sorted high to low (Only include countries with more than 5 customers)

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5
ORDER BY COUNT(CustomerID) DESC;
```

# HAVING Example using two tables

```

SELECT Pnumber, Pname, COUNT (*)
FROM PROJECT, WORKS_ON
WHERE Pnumber = Pno
GROUP BY Pnumber, Pname
HAVING COUNT (*) > 2;

```

