


Module 7: MySQL

<p>CREATE Table</p> <pre>CREATE TABLE <table name> (<column name> <column type> [<attribute constraint>] {, <column name> <column type> [<attribute constraint>] } [<table constraint> {, <table constraint>}])</pre>	<p>CREATE TABLE Example</p> <p>DEPARTMENT (DNUMBER, DNAME, MGRSSN, MGRSTARTDATE]</p>  <p>EMPLOYEE [SSN, FNAME, MINIT, LNAME, BDATE, ADDRESS, SEX, SALARY, SUPERSSN, DNO]</p> <pre>CREATE TABLE EMPLOYEE (FNAME VARCHAR (15) NOT NULL, MINIT CHAR NOT NULL, LNAME VARCHAR (15) NOT NULL, SSN CHAR (9) NOT NULL, BDATE DATE, ADDRESS VARCHAR (30), SEX CHAR, SALARY DECIMAL (10, 2), SUPERSSN CHAR (9), DNO INT NOT NULL,</pre> <p>Either of these:</p> <ul style="list-style-type: none"> PRIMARY KEY (SSN), FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE (SSN), FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNUMBER)); <p>CONSTRAINT EMPCHK PRIMARY KEY (SSN),</p> <p>CONSTRAINT EMPSUPERFK FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE (SSN),</p> <p>CONSTRAINT EMPDNOFK FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNUMBER)</p>
<p>ALTER Table</p> <pre>ALTER TABLE <table name> ADD <column name> <column type> [<attribute constraint>] {, <column name> <column type> [<attribute constraint>] } DROP <column name> [CASCADE] ALTER <column name> <column-options> ADD <constraint name> <constraint-options> DROP <constraint name> [CASCADE];</pre>	<p>To add an attribute (Value in all tuples will be initially NULL, so NOT NULL cannot be specified)</p> <ul style="list-style-type: none"> – ALTER TABLE EMPLOYEE ADD JOB VARCHAR(12); <p>To drop an attribute</p> <ul style="list-style-type: none"> – ALTER TABLE EMPLOYEE DROP ADDRESS; <p>To drop a constraint (constraint must have been given a name when it was specified)</p> <ul style="list-style-type: none"> – ALTER TABLE EMPLOYEE DROP CONSTRAINT EMPSUPERFK CASCADE;
<p>DROP Table</p> <pre>DROP TABLE <table name> [CASCADE];</pre>	<p>DROP TABLE</p> <ul style="list-style-type: none"> – Drops all constraints defined on the table including constraints in other tables which reference this table – Deletes all tuples within the table – Removes the table definition from the system catalog
<p>INSERT statement</p> <ul style="list-style-type: none"> • Add tuples to an existing relation <pre>INSERT INTO <table name> [(<column name> {, <column name> })] (VALUES (<constant value>, {,<constant value> }) <select statement>);</pre>	<p>INSERT INTO EMPLOYEE</p> <p>VALUES</p> <p>('Richard', 'K.', 'Marini', '653298653',</p> <p>'30-DEC-52', '98 Oak Forest, Katy, TX', 'M', 37000, '987654321', 4);</p> <p>Given:</p> <p>EMPLOYEE (FNAME, MINIT, LNAME, SSN, BDATE, ADDRESS, SEX, SALARY, SUPERSSN, DNO)</p>

	<div>INSERT INTO DEPTS-INFO (DNAME, NO-OF-EMPLOYEES, TOTAL-SALARY) SELECT DNAME, COUNT (*), SUM (SALARY) FROM DEPARTMENT, EMPLOYEE WHERE DNUMBER = DNO GROUP BY DNAME ;</div> <div>Given: EMPLOYEE [FNAME, MINIT, LNAME, SSN, BDATE, ADDRESS, SEX, SALARY, SUPERSSN, DNO] DEPARTMENT [DNAME, DNUMBER, MGRSSN, MGRSTARTDATE]</div>											
<div>DELETE statement</div> <div><ul style="list-style-type: none">Remove existing tuples from a relation</div> <div>DELETE FROM <table name> [WHERE <select condition>;</div>	<div>DELETE FROM EMPLOYEE WHERE DNO = 5;</div> <div>Given: EMPLOYEE [FNAME, MINIT, LNAME, SSN, BDATE, ADDRESS, SEX, SALARY, SUPERSSN, DNO]</div>											
<div>UPDATE statement</div> <div><ul style="list-style-type: none">Modify attribute values of one/more selected tuples in a relation</div> <div>UPDATE <table name> SET <column name> = <value expression> {, <column name> = <value expression>} [WHERE <select condition>;</div>	<div>UPDATE EMPLOYEE SET SALARY = SALARY * 1.1 WHERE LNAME = 'McGowen';</div> <div>Given: EMPLOYEE [FNAME, MINIT, LNAME, SSN, BDATE, ADDRESS, SEX, SALARY, SUPERSSN, DNO]</div>											
<div>SELECT statement</div> <div>SELECT <attribute list> FROM <table list> [WHERE <condition>] ; ORDER BY</div>	<div>SELECT EMP_ADDRESS FROM EMPLOYEE WHERE EMP_NAME = "Joe Bates"</div> <div>EMPLOYEE</div> <table><tr><th>EMP_NAME</th><th>EMP_ADDRESS</th><th>DEPARTMENT</th></tr><tr><td>Nicole Smith</td><td>1 Pine Road</td><td>Info. Systems</td></tr><tr><td>Joe Bates</td><td>32 Chandler Rd</td><td>Manufacturing</td></tr></table> <div>Results:</div> <table><tr><th>EMP_ADDRESS</th></tr><tr><td>32 Chandler Rd</td></tr></table>	EMP_NAME	EMP_ADDRESS	DEPARTMENT	Nicole Smith	1 Pine Road	Info. Systems	Joe Bates	32 Chandler Rd	Manufacturing	EMP_ADDRESS	32 Chandler Rd
EMP_NAME	EMP_ADDRESS	DEPARTMENT										
Nicole Smith	1 Pine Road	Info. Systems										
Joe Bates	32 Chandler Rd	Manufacturing										
EMP_ADDRESS												
32 Chandler Rd												

Complex WHERE conditions

Substring Comparisons	Examples														
LIKE	<ul style="list-style-type: none"> ... WHERE Address LIKE '%StLucia%' ... WHERE StrDate LIKE '__/05/__' 														
IN	<ul style="list-style-type: none"> ... WHERE LName IN ('Jones', 'Wong', 'Harrison') 														
IS (usually used in conjunction with NULL and NOT NULL)	<ul style="list-style-type: none"> ... WHERE DNo is NULL 														
Arithmetic Operators & Functions															
+, -, *, /, date and time functions	<ul style="list-style-type: none"> ... WHERE Salary * 2 > 5000 ... WHERE Year(Sys_Date - Bdate) > 55 														
BETWEEN	<ul style="list-style-type: none"> ... WHERE Salary BETWEEN 10000 AND 30000 														
Other functions															
DISTINCT <ul style="list-style-type: none"> Removes duplicates 	<p>9. List the <i>distinct</i> salaries paid to employees in each department</p> <p>SELECT DISTINCT Salary, DNo FROM EMPLOYEE;</p> <table border="1"> <thead> <tr> <th>Salary</th><th>Dno</th></tr> </thead> <tbody> <tr><td>30000</td><td>5</td></tr> <tr><td>40000</td><td>5</td></tr> <tr><td>25000</td><td>4</td></tr> <tr><td>43000</td><td>4</td></tr> <tr><td>25000</td><td>5</td></tr> <tr><td>55000</td><td>1</td></tr> </tbody> </table>	Salary	Dno	30000	5	40000	5	25000	4	43000	4	25000	5	55000	1
Salary	Dno														
30000	5														
40000	5														
25000	4														
43000	4														
25000	5														
55000	1														

Sorting Example - By Heading

14b. List the last names of all employees working in department 6, and their salaries given a 10% increase.

```
SELECT Lname, 1.1 * Salary AS Inc-Sal
FROM EMPLOYEE
WHERE Dno = 6
ORDER BY Inc-Sal;
```

Given:

EMPLOYEE [FNAME, MINIT, LNAME, SSN, BDATE, ADDRESS, SEX, SALARY, SUPERSN, DNO]

14a. List the last names of all employees working in department 6, and their salaries given a 10% increase.

```
SELECT Lname, 1.1 * Salary
FROM EMPLOYEE
WHERE Dno = 6
ORDER BY 2;
```

Given:

EMPLOYEE [FNAME, MINIT, LNAME, SSN, BDATE, ADDRESS, SEX, SALARY, SUPERSN, DNO]

Aggregation

- Functions that produce summary values, which can be applied to a selected set of tuples, to all tuples, or to multiple groups of tuples, specified by the GROUP BY clause

COUNT	Counts the number of tuples query returns
SUM	Calculates the sum of a set of numeric values
AVG	Calculates the average of a set of numeric values
MAX	Returns the maximum value from a set of values, which have a total ordering. (Note that domain of values can be non-numeric)
MIN	Returns the minimum value from a set of values, which have a total ordering. (Note that domain of values can be non-numeric)

GROUPING in SQL

SELECT [DISTINCT] (attribute / expression / aggregation-function list | *)

FROM <table list>

[WHERE [join condition and
search_condition]

GROUP BY grouping attributes]

[ORDER BY column_name
[ASC|DESC] {, column-name [ASC|DESC]}];

Attributes **must** also appear
in the GROUP BY clause or
in Aggregation functions

- When GROUP BY is used in an SQL statement, any attribute appeared in SELECT Clause must also appeared in an aggregation function or in GROUP BY clause.

Conditions on Groups

HAVING clause (following the GROUP BY clause) is used to specify the conditions (similar to WHERE clause), but can also include aggregates

SELECT [DISTINCT] (attribute / expression / aggregation-function list | *)

FROM <table list>

[WHERE [join condition and
search_condition]

[**GROUP BY** grouping attributes]

HAVING <group condition>]

[ORDER BY column_name
[ASC|DESC] {, column-name [ASC|DESC]}];

Set Operators

UNION	<ul style="list-style-type: none">Produces a relation that includes all tuples that appear only in R1, or only in R2, or in both R1 and R2.Duplicate entries are eliminated. Two relations are union compatible if: <ul style="list-style-type: none">they have the same no. of columns Their columns have corresponding domains (i.e. $\text{dom}(A_i) = \text{dom}(B_i)$)
--------------	---

25. List the ESSN's of employees who have dependents or work on projects

SELECT ESSN FROM WORKS_ON

UNION

SELECT ESSN FROM DEPENDENTS

Given:

WORKS_ON [ESSN, PNo, Hours]

DEPENDENT [ESSN, Dep_Name, Sex, DOB, Relationship]

25a. List the ESSN's of employees who have dependents or work on projects

SELECT ESSN FROM WORKS_ON

UNION ALL

SELECT ESSN FROM DEPENDENTS

Result is a multi-set (containing duplicate tuples).

Can also be applied to Intersection and Difference

Given:

WORKS_ON [ESSN, PNo, Hours]

DEPENDENT [ESSN, Dep_Name, Sex, DOB, Relationship]

Intersection	<ul style="list-style-type: none"> Produces a relation that includes the tuples that appear in both R1 and R2 R1 & R2 must be union compatible.
---------------------	---

27. List the ESSN's of employees who have dependents and work on projects

```
SELECT ESSN FROM WORKS_ON
INTERSECT
SELECT ESSN FROM DEPENDENTS
```

Given:

WORKS_ON [ESSN, PNo, Hours]
DEPENDENT [ESSN, Dep_Name, Sex, DOB, Relationship]

Difference	<ul style="list-style-type: none"> Produces a relation that includes all the tuples that appear in R1, but do not appear in R2. R1 and R2 must be union compatible.
-------------------	---

28. List the ESSN's of employees who have dependents but do not work on projects

```
SELECT ESSN FROM DEPENDENTS
MINUS
SELECT ESSN FROM WORKS_ON
```

Given:

WORKS_ON [ESSN, PNo, Hours]
DEPENDENT [ESSN, Dep_Name, Sex, DOB, Relationship]

$A \cup B$	commutative	$A \cup B = B \cup A$
	associative	$(A \cup B) \cup C = A \cup (B \cup C)$
$A \cap B$	commutative	$A \cap B = B \cap A$
	associative	$(A \cap B) \cap C = A \cap (B \cap C)$
$A - B$	not commutative	$A - B \neq B - A$
	not associative	$(A - B) - C \neq A - (B - C)$

EXAMPLE (commutative):

```
SELECT * FROM WORKS_ON UNION SELECT * FROM WORKED_ON
= SELECT * FROM WORKED_ON UNION SELECT * FROM WORKS_ON
```

Renaming in SQL

- Qualifying attributes names or declaring an alias

Declaring an Alias

29. Get employee names and the corresponding department names

```
SELECT EMPLOYEE.Name, DEPARTMENT.Name
FROM EMPLOYEE, DEPARTMENT
WHERE EMPLOYEE.Dno = DEPARTMENT.Dnumber
```

Given:

EMPLOYEE [NAME, SSN, BDATE, ADDRESS, SEX, SALARY, SUPERSSN, DNO]
DEPARTMENT [NAME, DNUMBER, MGRSSN, MGRSTARTDATE]

Foreign Keys:

EMPLOYEE.SUPERSSN → EMPLOYEE.SSN
EMPLOYEE.DNO → DEPARTMENT.DNUMBER
DEPARTMENT.MGRSSN → EMPLOYEE.SSN

30. Get employee names and the corresponding supervisor names

```
SELECT SUBORDINATE.Name, SUPERVISOR.Name
FROM EMPLOYEE AS SUBORDINATE, EMPLOYEE AS SUPERVISOR
WHERE SUBORDINATE.SuperSSN = SUPERVISOR.SSN
```

Attributes can also be renamed using AS

Given:

EMPLOYEE [NAME, SSN, BDATE, ADDRESS, SEX, SALARY, SUPERSSN, DNUMBER]

Foreign Key:

EMPLOYEE.SUPERSSN → EMPLOYEE.SSN