

SECD2523 DATABASE

Structured Query Language (SQL) 2 |
Data Manipulation Language (DML) 1

Content adapted from Connolly, T., Begg, C., 2015. Database Systems: A Practical Approach to Design, Implementation, and Management, Global Edition. Pearson Education.

Innovating Solutions

LECTURE LEARNING OUTCOME

By the end of this lecture, students should be able to:

01 Write DML statements to add data into table, update data, and delete data using the following commands:

- INSERT INTO ... VALUES ...
- INSERT INTO ... SELECT ...
- UPDATE ... SET ... [WHERE] ...
- DELETE FROM ... [WHERE] ...

02 Write DML statement to display records in tables

- SELECT ... FROM ...
- SELECT ... FROM ... WHERE
- SELECT ... FROM ... WHERE ... BETWEEN ... AND
- SELECT ... FROM ... WHERE ... IN
- SELECT ... FROM ... WHERE ... LIKE

- 01** INSERT INTO ... VALUES
- 02** INSERT INTO ... SELECT
- 03** UPDATE ... SET ... [WHERE]
- 04** DELETE ... FROM [WHERE]
- 05** SELECT ... FROM ... [WHERE]
- 06** SELECT ... FROM ... WHERE ... [BETWEEN ...
AND] / [IN] / [LIKE]
- 07** RULES OF PRECEDENCE

SQL Statements

STATEMENTS	TYPE
SELECT INSERT UPDATE DELETE MERGE	DATA MANIPULATION LANGUAGE (DML) Retrieves data from database, enters new rows, changes existing rows, and removes unwanted rows from tables in the database, respectively
CREATE ALTER DROP RENAME TRUNCATE COMMENT	DATA DEFINITION LANGUAGE (DDL) Sets up, changes, and removes data structures from tables
GRANT REVOKE	DATA CONTROL LANGUAGE (DCL) Provides or removes access rights to both the Database and the structures within it
COMMIT ROLLBACK SAVEPOINT	TRANSACTION CONTROL Manages the changes made by DML statements. Changes to the data can be grouped together into logical transactions

Writing SQL Statements

- SQL statements are **not case sensitive** (unless indicated).
- SQL statements can be entered on **one or more lines**.
- Keywords cannot be abbreviated or split across lines
- Clauses are usually placed on separate lines
- **Indents** are used to enhance readability

Human Resource (HR) schema

- In the HR records, each employee has an identification number, email address, job identification code, salary, and manager. Some employees earn commissions in addition to their salary
- The company also tracks information about jobs within the organization. Each job has an identification code, job title, and a minimum and maximum salary range for the job. Some employees have been with the company for a long time and have held different positions within the company. When an employee resigns, the duration the employee was working for, the job identification number, and the department are recorded.

Human Resource (HR) schema

- The sample company is regionally diverse, so it tracks the locations of its warehouses and departments. Each employee is assigned to a department, and each department is identified by a unique department number or short name. Each department is associated with one location, and each location has a full address that includes the street name, postal code, city, state or province, and the country code.
- In places where the departments and warehouses are located, the company records details such as the country name, currency symbol, currency name, and the region where the country is located geographically.

INSERT INTO ... VALUES

- Purpose: To **insert a row of data** into a table.

- Syntax:

```
INSERT INTO tableName (column1, column2, column3)
VALUES (value1, value2, value3);
```

- Example: Insert a new row into table Department:

```
INSERT INTO departments
VALUES (70, 'Public Relations', 100, 70);
```

OR

```
INSERT INTO departments (department_id, department_name,
manager_id, location_id)
VALUES (70, 'Public Relations', 100, 70);
```

NOTE: Use single quote (') for string / character data

Inserting rows with **NULL** values

- Implicit method: Omit the column from the column list

```
INSERT INTO departments (department_id, department_name)
VALUES (30, 'Purchasing');
```

- Explicit method: Specify the NULL keyword in the VALUES clause

```
INSERT INTO departments (department_id, department_name)
VALUES (30, 'Purchasing', NULL, NULL);
```

department_id	department_name	manager_id	location_id
30	Purchasing		

Attention: Inserting **NULL** values

- Be sure that you can use the **NULL** value in the targeted column by verifying the NULL status with the **DESCRIBE** command

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
department_id	int unsigned	NO	PRI	NULL	
department_name	varchar(30)	NO		NULL	
manager_id	int unsigned	YES	MUL	NULL	
location_id	int unsigned	YES	MUL	NULL	

Output of DESCRIBE in
MySQL

Script Output x Task completed in 0.055 seconds

Name	Null?	Type
DEPT_ID	NOT NULL	CHAR(1)
DEPT_NAME		VARCHAR2(20)
MANAGER_ID		NUMBER(38)
LOCATION	NOT NULL	VARCHAR2(100)

Output of DESCRIBE in
Oracle

Attention: Inserting **NULL** values

- Common errors that can occur during the user input are checked in the following order:
 - Mandatory value missing for a **NOT NULL** column
 - Duplicate value violating any **unique** or **primary key** constraint
 - Any value violating a **CHECK** constraint
 - **Referential integrity maintained** for a foreign key constraint
 - Data **type mismatch** or values too wide to fit in column

Attention: Inserting **NULL** values

- Any value violating a CHECK constraint
 - CHECK is a constraint that set a specific condition for the input data to follow.
E.g.: DDL statement below:

```
CREATE TABLE Dept (  
  Dept_ID char(1),  
  Dept_Name varchar(20),  
  Manager_ID INTEGER CHECK (Manager_ID > 0),  
  Location varchar(30) NOT NULL,  
  CONSTRAINT Dept_PK PRIMARY KEY (Dept_ID)  
);
```

- Example above: the column “Manager_ID” has a **CHECK** constraint that checks if the input is larger than zero,
- Therefore, only input that is larger than zero is allowed to enter the column.

Error example in mySQL:

```
INSERT INTO Dept (Dept_ID, Dept_Name, Manager_ID, Location)  
VALUES ('A', 'Accounting', 0, 'JB');
```

Error Code: 3819. Check constraint 'dept_chk_1' is violated.

Attention: Inserting **NULL** values

- **Referential integrity maintained** for a foreign key constraint
 - Whenever the foreign key exists in the table, according to **Referential Integrity constraint**, the value can either be:
 - **Values exist** in the parent table foreign key refers to. OR
 - Wholly **NULL**
 - Therefore, non-NULL values that does not exist at the table where the foreign key refers to are not allowed to enter in the column. Error example in Oracle:

```
INSERT INTO employees (employee_id, first_name, last_name, email, hire_date, job_id, salary)
VALUES (300, 'Muhammad', 'Ali', 'MALI', '2024-09-01', 'BOXER', 8500);
```

```
Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails
(`hr`.`employees`, CONSTRAINT `employees_ibfk_1` FOREIGN KEY (`job_id`) REFERENCES
`jobs` (`job_id`))
```

Recommendation: Inserting **NULL** values

- Use of the column list is **recommended** because it makes the INSERT statement more readable and reliable, or less prone to mistakes.

Example

```
INSERT INTO tableName (column1, column2, column3)  
VALUES (value1, value2, value3);
```

Inserting Special Values

- The **SYSDATE ()** function records the current date and time

```
INSERT INTO employees  
(employee_id, first_name, last_name, email, hire_date, job_id, salary)  
VALUES  
(300, 'Mikail', 'Hafiz', 'MHAFIZ', SYSDATE (), 'IT_PROG', 9000);
```

Inserting specific Date and Time

- The DD-MON-RR format is generally used to insert a date value.
- You may also supply the date value in DD-MON-YYYY format.
- This is recommended because it clearly specifies the century and does not depend on the internal RR format logic specifying the correct century

```
INSERT INTO employees  
(employee_id, first_name, last_name, email, hire_date, job_id, salary)  
VALUES  
(400, 'Yusuf', 'Syarin', 'YSYARIN', '2024-12-12', 'SA_REP', 8000);
```

For MySQL, use “YYYY-MM-DD” format for DATE datatype

INSERT INTO ... SELECT

- To insert data **from an existing table**
- Write the INSERT statement with a **subquery**
- DO NOT use the VALUES clause
- Match the number of columns in the INSERT clause to those in the subquery

```
INSERT INTO sales_reps (id, name, salary, commission_pct)
SELECT employee_id, last_name, salary, commission_pct
FROM employees
WHERE job_id LIKE '%REP%';
```

UPDATE ... SET

- Purpose: To modify existing values in a table.
- Syntax:

[] = OPTIONAL

```
UPDATE table-name  
SET col1 = update-value[,col2 = update-value]  
[WHERE search-condition]
```

- Example: Update the department ID of employee with the ID of 113 to 50

```
UPDATE employees  
SET department_id = 50  
WHERE employee_id = 113;
```

Update Two Columns with a Subquery

- Can also be done for multiple subqueries

[] = OPTIONAL

```
UPDATE      table
SET         column = (SELECT column
                      FROM table
                      WHERE condition)
           [,column2 = (SELECT column
                      FROM table
                      WHERE condition) ]
[WHERE condition]
```

Update Two Columns with Subquery

- Example: Update employee 113's job and salary to match those of employee 205

```
UPDATE employees
SET (job_id, salary) =
    (SELECT job_id, salary
     FROM employees
     WHERE employee_id = 205)
WHERE employee_id = 113;
```

DELETE FROM

- Purpose: To delete existing rows from table
- Syntax:

[] = OPTIONAL

```
DELETE FROM table-name  
[WHERE search-condition]
```

- Example: Delete the record of department Finance

```
DELETE FROM departments  
WHERE department_name = 'Finance';
```

- All rows in the table are deleted if you omit the WHERE clause.

```
DELETE FROM departments
```

SELECT ... FROM

- To retrieve and display data from one or more tables.

[] = OPTIONAL

```
SELECT col1, col2, ... coln  
FROM TableName [,TableName]  
[WHERE condition]  
[GROUP BY columnList]  
[HAVING condition]  
[ORDER BY columnList]
```

SELECT ... FROM

- Retrieve all columns and all rows
- List the full details of the DEPARTMENTS table

```
SELECT *
FROM departments;
```

ina — mysql -u root -p — mysql — mysql -u root -p — 69x37

```
mysql> SELECT *
      -> FROM departments;
```

department_id	department_name	manager_id	location_id
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700
120	Treasury	NULL	1700
130	Corporate Tax	NULL	1700
140	Control And Credit	NULL	1700
150	Shareholder Services	NULL	1700
160	Benefits	NULL	1700
170	Manufacturing	NULL	1700
180	Construction	NULL	1700
190	Contracting	NULL	1700
200	Operations	NULL	1700
210	IT Support	NULL	1700
220	NOC	NULL	1700
230	IT Helpdesk	NULL	1700
240	Government Sales	NULL	1700
250	Retail Sales	NULL	1700
260	Recruiting	NULL	1700
270	Payroll	NULL	1700

27 rows in set (0.01 sec)

```
mysql>
```

Result Grid

Filter Rows: Search

department_id	department_name	manager_id	location_id
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700
120	Treasury	NULL	1700
130	Corporate Tax	NULL	1700
140	Control And Credit	NULL	1700
150	Shareholder Serv...	NULL	1700
160	Benefits	NULL	1700
170	Manufacturing	NULL	1700
180	Construction	NULL	1700
190	Contracting	NULL	1700
200	Operations	NULL	1700
210	IT Support	NULL	1700
220	NOC	NULL	1700
230	IT Helpdesk	NULL	1700
240	Government Sales	NULL	1700
250	Retail Sales	NULL	1700
260	Recruiting	NULL	1700
270	Payroll	NULL	1700

SELECT ... FROM

- Retrieve specific columns, all rows
- Example:
- Produce a list of salaries for all staff, displaying only the employee ID, first name and salary

```
SELECT employee_id, first_name, salary
FROM employees;
```

```
ina — mysql -u root -p — mysql — mysql -u root
mysql> SELECT employee_id, first_name, salary
-> FROM employees;
```

employee_id	first_name	salary
100	Steven	24000.00
101	Neena	17000.00
102	Lex	17000.00
103	Alexander	9000.00
104	Bruce	6000.00
105	David	4800.00
106	Valli	4800.00
107	Diana	4200.00
108	Nancy	12000.00
109	Daniel	9000.00
110	John	8200.00
111	Ismael	7700.00
...		
200	Jennifer	4400.00
201	Michael	13000.00
202	Pat	6000.00
203	Susan	6500.00
204	Hermann	10000.00
205	Shelley	12000.00
206	William	8300.00
300	Mikail	9000.00

```
108 rows in set (0.00 sec)

mysql>
```


INSERT INTO ... SELECT

- Copy multiple row of records into a different table

- Example

- Get the command for creating existing table **departments**

```
SHOW CREATE TABLE departments;
```

- Create a new table name **my_dept** with the same structure as table **departments**

- Copy all records from **departments** table to **my_dept**

```
INSERT INTO my_dept (  
    SELECT *  
    FROM departments);
```

```
mysql> CREATE TABLE my_dept (  
-> department_id int unsigned NOT NULL,  
-> department_name varchar(30) NOT NULL,  
-> manager_id int unsigned DEFAULT NULL,  
-> location_id int unsigned DEFAULT NULL,  
-> PRIMARY KEY (department_id),  
-> KEY location_id (location_id),  
-> KEY manager_id (manager_id),  
-> CONSTRAINT my_dept_ibfk_1 FOREIGN KEY (location_id) REFERENCES locations (location_id),  
-> CONSTRAINT my_dept_ibfk_2 FOREIGN KEY (manager_id) REFERENCES employees (employee_id)  
-> );
```

Query OK, 0 rows affected (0.02 sec)

```
mysql> INSERT INTO my_dept(  
-> SELECT *  
-> FROM departments);
```

```
Query OK, 27 rows affected (0.00 sec)  
Records: 27 Duplicates: 0 Warnings: 0
```

Records: 27 Duplicates: 0 Warnings: 0

```
mysql> SELECT *  
-> FROM my_dept;
```

department_id	department_name	manager_id	location_id
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700
120	Treasury	NULL	1700
130	Corporate Tax	NULL	1700
140	Control And Credit	NULL	1700
150	Shareholder Services	NULL	1700
160	Benefits	NULL	1700
170	Manufacturing	NULL	1700
180	Construction	NULL	1700
190	Contracting	NULL	1700
200	Operations	NULL	1700
210	IT Support	NULL	1700
220	NOC	NULL	1700
230	IT Helpdesk	NULL	1700
240	Government Sales	NULL	1700
250	Retail Sales	NULL	1700
260	Recruiting	NULL	1700
270	Payroll	NULL	1700

27 rows in set (0.01 sec)

```
mysql>
```

Constraint name must be unique

Create new table by copying the structure of an existing table

- Create a new table name **dept_baharu** with the same structure as table **departments**.

```
CREATE TABLE dept_baharu
AS (SELECT *
    FROM departments
    WHERE 1=2);
```

- This would create a new table called **dept_baharu** that includes all columns from the **departments** table **WITHOUT** the data from **departments**

```
ina — mysql -u root -p — mysql — mysql -u root -p — 67x35

mysql> CREATE TABLE dept_baharu
-> AS (SELECT *
->     FROM departments
->     WHERE 1=2);
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESCRIBE dept_baharu;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| department_id  | int unsigned  | NO   |     | NULL    |       |
| department_name | varchar(30)   | NO   |     | NULL    |       |
| manager_id     | int unsigned  | YES  |     | NULL    |       |
| location_id    | int unsigned  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql> DESCRIBE departments;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| department_id  | int unsigned  | NO   | PRI | NULL    |       |
| department_name | varchar(30)   | NO   |     | NULL    |       |
| manager_id     | int unsigned  | YES  | MUL | NULL    |       |
| location_id    | int unsigned  | YES  | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql> SELECT *
-> FROM dept_baharu;
Empty set (0.00 sec)

mysql>
```

Create new table by copying the structure of an existing table (and data)

- Create a new table name **new_dept** with the same structure as table **departments**.

```
CREATE TABLE new_dept
AS (SELECT * FROM
departments);
```

- This would create a new table called **new_dept** that includes all columns from the **departments** table **INCLUDING** all the data from **departments**

```
mysql> CREATE TABLE new_dept
-> AS (SELECT * FROM departments);
Query OK, 27 rows affected (0.00 sec)
Records: 27 Duplicates: 0 Warnings: 0

mysql> SELECT * FROM new_dept;
```

department_id	department_name	manager_id	location_id
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700
120	Treasury	NULL	1700
130	Corporate Tax	NULL	1700
140	Control And Credit	NULL	1700
150	Shareholder Services	NULL	1700
160	Benefits	NULL	1700
170	Manufacturing	NULL	1700
180	Construction	NULL	1700
190	Contracting	NULL	1700
200	Operations	NULL	1700
210	IT Support	NULL	1700
220	NOC	NULL	1700
230	IT Helpdesk	NULL	1700
240	Government Sales	NULL	1700
250	Retail Sales	NULL	1700
260	Recruiting	NULL	1700
270	Payroll	NULL	1700

```
27 rows in set (0.00 sec)

mysql>
```

SELECT...FROM...WHERE (1)

- To list only selected row of records based on conditions
- Conditions involve comparison operators and/or logical operators
 - Comparison operators:
 - = <> < > <= >= !=
 - BETWEEN..AND LIKE IN(set)
 - Logical operators:
 - AND OR NOT

```
SELECT list-of-columns  
FROM list of tables  
WHERE search-condition;
```

SELECT...FROM...WHERE...BETWEEN...AND

- Example:
 - List all employee ID, last_name, salary where salary is between 3000 and 4000

```
SELECT employee_id, last_name, salary
FROM employees
WHERE salary BETWEEN 3000 AND 4000;
```

```
ina — mysql -u root -p — mysql — mysql -u
mysql> SELECT employee_id, last_name, salary
-> FROM employees
-> WHERE salary BETWEEN 3000 AND 4000;
+-----+-----+-----+
| employee_id | last_name | salary |
+-----+-----+-----+
|          115 | Khoo      | 3100.00 |
|          125 | Nayer     | 3200.00 |
|          129 | Bissot    | 3300.00 |
|          133 | Mallin    | 3300.00 |
|          137 | Ladwig    | 3600.00 |
|          138 | Stiles    | 3200.00 |
|          141 | Rajs      | 3500.00 |
|          142 | Davies    | 3100.00 |
|          180 | Taylor    | 3200.00 |
|          181 | Fleaur    | 3100.00 |
|          186 | Dellinger | 3400.00 |
|          187 | Cabrio    | 3000.00 |
|          188 | Chung     | 3800.00 |
|          189 | Dilly     | 3600.00 |
|          192 | Bell      | 4000.00 |
|          193 | Everett   | 3900.00 |
|          194 | McCain    | 3200.00 |
|          196 | Walsh     | 3100.00 |
|          197 | Feeney    | 3000.00 |
+-----+-----+-----+
19 rows in set (0.00 sec)

mysql>
```


SELECT...FROM...WHERE...IN

- To search values in a list set
- Example:
 - List all employees whose salaries are 17000, 2500, 8600, 1000

```
SELECT employee_id, last_name, salary
FROM employees
WHERE salary IN (17000, 2500, 8600, 1000);
```

```
ina — mysql -u root -p — mysql — mysql -u root -p — 55x20

[mysql> SELECT employee_id, last_name, salary
-> FROM employees
-> WHERE salary IN (17000, 2500, 8600, 1000);

+-----+-----+-----+
| employee_id | last_name | salary |
+-----+-----+-----+
|          101 | Kochhar   | 17000.00 |
|          102 | De Haan   | 17000.00 |
|          119 | Colmenares | 2500.00 |
|          131 | Marlow    | 2500.00 |
|          140 | Patel     | 2500.00 |
|          144 | Vargas    | 2500.00 |
|          176 | Taylor    | 8600.00 |
|          182 | Sullivan  | 2500.00 |
|          191 | Perkins   | 2500.00 |
+-----+-----+-----+
9 rows in set (0.00 sec)

mysql>
```

There is no employee with a salary of 1000.

Therefore, it does not return a value.

SELECT...FROM...WHERE...LIKE (1)

- To perform wild card searchers of valid search string values
 - `_` (underscore symbol) denotes one character
 - `%` denotes zero or many character
- Example:
 - List all employee whose first name starts with a J

```
SELECT employee_id, first_name, salary
FROM employees
WHERE first_name LIKE 'J%';
```

```
ina — mysql -u root -p — mysql — mysql -u root -p — 49x27
mysql> SELECT employee_id, first_name, salary
-> FROM employees
-> WHERE first_name LIKE 'J%';
```

employee_id	first_name	salary
110	John	8200.00
112	Jose Manuel	7800.00
125	Julia	3200.00
127	James	2400.00
131	James	2500.00
133	Jason	3300.00
139	John	2700.00
140	Joshua	2500.00
145	John	14000.00
156	Janette	10000.00
176	Jonathon	8600.00
177	Jack	8400.00
181	Jean	3100.00
186	Julia	3400.00
189	Jennifer	3600.00
200	Jennifer	4400.00

```
16 rows in set (0.00 sec)

mysql>
```

SELECT...FROM...WHERE...LIKE (2)

- Example:
- List all employees whose first name's third letter is an E

```
SELECT employee_id, first_name, salary
FROM employees
WHERE first_name LIKE '__e%';
```

(2 underscore symbols)

```
ina — mysql -u root -p — mysql — mysql -u root -p —...

mysql> SELECT employee_id, first_name, salary
-> FROM employees
-> WHERE first_name LIKE '__e%';

+-----+-----+-----+
| employee_id | first_name | salary |
+-----+-----+-----+
|          100 | Steven    | 24000.00 |
|          101 | Neena     | 17000.00 |
|          103 | Alexander | 9000.00  |
|          115 | Alexander | 3100.00  |
|          116 | Shelli    | 2900.00  |
|          126 | Irene     | 2700.00  |
|          128 | Steven    | 2200.00  |
|          138 | Stephen   | 3200.00  |
|          141 | Tenna     | 3500.00  |
|          149 | Eleni     | 10500.00 |
|          185 | Alexis    | 4100.00  |
|          205 | Shelley   | 12000.00 |
+-----+-----+-----+

12 rows in set (0.00 sec)

mysql>
```


SELECT...FROM...WHERE...IS NULL

- To search for columns with **null value**
- Example:
 - List all employees who has not receive any commission.

```
SELECT employee_id, first_name,
commission_pct
FROM employees
WHERE commission_pct IS NULL;
```

```
ina — mysql -u root -p — mysql — mysql -u root -p — 56x42
mysql> SELECT employee_id, first_name, commission_pct
-> FROM employees
-> WHERE commission_pct IS NULL;
```

employee_id	first_name	commission_pct
100	Steven	NULL
101	Neena	NULL
102	Lex	NULL
103	Alexander	NULL

...

201	Michael	NULL
202	Pat	NULL
203	Susan	NULL
204	Hermann	NULL
205	Shelley	NULL
206	William	NULL
300	Mikail	NULL

```
73 rows in set (0.00 sec)

mysql>
```

SELECT...FROM...WHERE...IS NOT NULL

- To search for columns with value is not NULL
- Example:
 - List all employees who have received any commissions

```
SELECT employee_id, first_name,
commission_pct
FROM employees
WHERE commission_pct IS NOT NULL;
```

ina — mysql -u root -p — mysql — mysql -u root -p — 59x45

```
mysql> SELECT employee_id, first_name, commission_pct
-> FROM employees
-> WHERE commission_pct IS NOT NULL;
```

employee_id	first_name	commission_pct
145	John	0.40
146	Karen	0.30
147	Alberto	0.30
148	Gerald	0.30
149	Eleni	0.20
150	Peter	0.30
151	David	0.25
152	Peter	0.25
153	Christopher	0.20

...

174	Ellen	0.30
175	Alyssa	0.25
176	Jonathon	0.20
177	Jack	0.20
178	Kimberely	0.15
179	Charles	0.10

35 rows in set (0.00 sec)

```
mysql>
```

SELECT...FROM...WHERE...AND

- **AND** requires both component conditions to be true

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary >= 10000 CONDITION 1
AND job_id LIKE '%MAN%' ; CONDITION 2
```

```
ina — mysql -u root -p — mysql — mysql -u root -p — 59x45
mysql> SELECT employee_id, last_name, job_id, salary
-> FROM employees
-> WHERE salary >= 10000
-> AND job_id LIKE '%MAN%';
```

employee_id	last_name	job_id	salary
114	Raphaely	PU_MAN	11000.00
145	Russell	SA_MAN	14000.00
146	Partners	SA_MAN	13500.00
147	Errazuriz	SA_MAN	12000.00
148	Cambrault	SA_MAN	11000.00
149	Zlotkey	SA_MAN	10500.00
201	Hartstein	MK_MAN	13000.00

```
7 rows in set (0.00 sec)

mysql>
```

SELECT...FROM...WHERE...OR

- OR requires either component conditions to be true

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary >= 10000 CONDITION 1
OR job_id LIKE '%MAN%' ; CONDITION 2
```

```
ina — mysql -u root -p — mysql — mysql -u root -p — 59x36

mysql> SELECT employee_id, last_name, job_id, salary
-> FROM employees
-> WHERE salary >= 10000
-> OR job_id LIKE '%MAN%';
```

employee_id	last_name	job_id	salary
100	King	AD_PRES	24000.00
101	Kochhar	AD_VP	17000.00
102	De Haan	AD_VP	17000.00
108	Greenberg	FI_MGR	12000.00
114	Raphaely	PU_MAN	11000.00
120	Weiss	ST_MAN	8000.00
121	Fripp	ST_MAN	8200.00
122	Kaufling	ST_MAN	7900.00
123	Vollman	ST_MAN	6500.00
124	Mourgos	ST_MAN	5800.00
145	Russell	SA_MAN	14000.00
146	Partners	SA_MAN	13500.00
147	Errazuriz	SA_MAN	12000.00
148	Cambrault	SA_MAN	11000.00
149	Zlotkey	SA_MAN	10500.00
150	Tucker	SA_REP	10000.00
156	King	SA_REP	10000.00
162	Vishney	SA_REP	10500.00
168	Ozer	SA_REP	11500.00
169	Bloom	SA_REP	10000.00
174	Abel	SA_REP	11000.00
201	Hartstein	MK_MAN	13000.00
204	Baer	PR_REP	10000.00
205	Higgins	AC_MGR	12000.00

```
24 rows in set (0.00 sec)

mysql>
```

NOT operator

- Example:
 - Display the last name and job ID of employees whose job ID is not IT_PROG, ST_CLERK, SA_REP, SH_CLERK, or PU_CLERK

```
SELECT last_name, job_id
FROM employees
WHERE job_id
NOT IN ('IT_PROG', 'ST_CLERK',
'SA_REP', 'SH_CLERK', 'PU_CLERK');
```

```
ina — mysql -u root -p — mysql — mysql -u root -p — 59x39
mysql> SELECT last_name, job_id
-> FROM employees
-> WHERE job_id
-> NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP', 'SH_CLERK',
'PU_CLERK');
```

last_name	job_id
Gietz	AC_ACCOUNT
Higgins	AC_MGR
Whalen	AD_ASST
King	AD PRES
Kochhar	AD_VP
De Haan	AD_VP
Faviet	FI_ACCOUNT
Chen	FI_ACCOUNT
Sciarra	FI_ACCOUNT
Urman	FI_ACCOUNT
Popp	FI_ACCOUNT
Greenberg	FI_MGR
Mavris	HR_REP
Hartstein	MK_MAN
Fay	MK_REP
Baer	PR_REP
Raphaely	PU_MAN
Russell	SA_MAN
Partners	SA_MAN
Errazuriz	SA_MAN
Cambrault	SA_MAN
Zlotkey	SA_MAN
Weiss	ST_MAN
Fripp	ST_MAN
Kaufling	ST_MAN
Vollman	ST_MAN
Mourgos	ST_MAN

```
27 rows in set (0.00 sec)

mysql>
```

Rules of precedence

- You can use parenthesis to override rules of precedence

Operator	Meaning
1	Arithmetic operators
2	Concatenation operators
3	Comparison operators
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Not equal to
7	NOT logical condition
8	AND logical condition
9	OR logical condition

Rules of Precedence

1

```
SELECT last_name, job_id, salary
FROM employees
WHERE job_id = 'SA_REP'
OR job_id = 'AD_PRES'
AND salary > 15000;
```

Check this condition first

```
SQL> select last_name, job_id, salary
2 from employees
3 where job_id = 'SA_REP'
4 OR job_id = 'AD_PRES'
5 AND salary > 15000;
```

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000
Abel	SA_REP	11000
Taylor	SA_REP	8600
Grant	SA_REP	7000

2

```
SELECT last_name, job_id, salary
FROM employees
WHERE (job_id = 'SA_REP'
OR job_id = 'AD_PRES')
AND salary > 15000;
```

Check the one with parenthesis first

```
SQL> select last_name, job_id, salary
2 from employees
3 where (job_id = 'SA_REP'
4 or job_id = 'AD_PRES')
5 and salary > 15000;
```

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000

DISTINCT (1)

- The default display of queries is all rows, including duplicate rows
- Use the DISTINCT keyword immediately after the SELECT keyword to eliminate duplicate rows

```
SELECT DISTINCT department_id  
FROM employees;
```

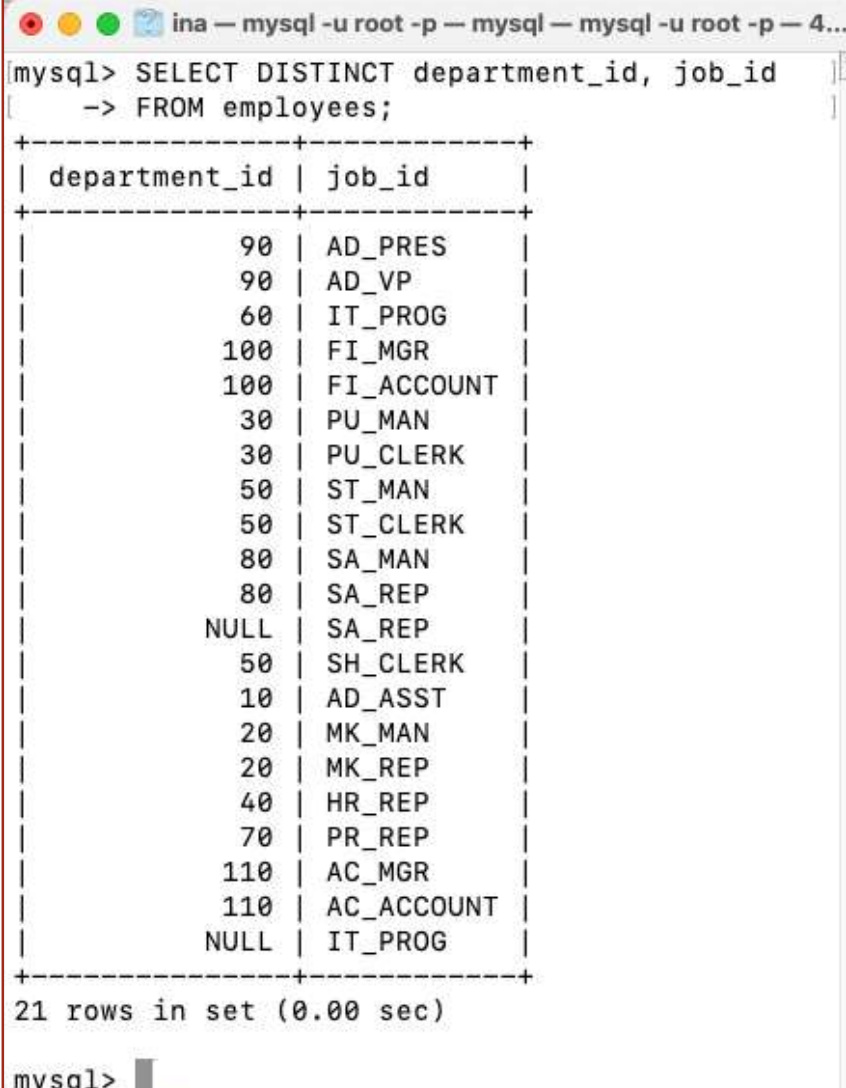
```
SQL> select distinct department_id  
2 from employees;  
  
DEPARTMENT_ID  
-----  
20  
90  
110  
50  
80  
10  
60  
  
8 rows selected.
```

```
SQL> select department_id  
2 from employees;  
  
DEPARTMENT_ID  
-----  
10  
20  
20  
110  
110  
90  
90  
90  
60  
60  
60  
  
DEPARTMENT_ID  
-----  
50  
50  
50  
50  
50  
80  
80  
80  
  
20 rows selected.
```


DISTINCT (2)

- Multiply columns can be specified after the DISTINCT qualifier
- This affects all the selected columns, and the result is every distinct combination of the columns

```
SELECT DISTINCT department_id, job_id  
FROM employees;
```



```
ina — mysql -u root -p — mysql — mysql -u root -p — 4...  
mysql> SELECT DISTINCT department_id, job_id  
-> FROM employees;  
+-----+-----+  
| department_id | job_id |  
+-----+-----+  
| 90 | AD_PRES |  
| 90 | AD_VP |  
| 60 | IT_PROG |  
| 100 | FI_MGR |  
| 100 | FI_ACCOUNT |  
| 30 | PU_MAN |  
| 30 | PU_CLERK |  
| 50 | ST_MAN |  
| 50 | ST_CLERK |  
| 80 | SA_MAN |  
| 80 | SA_REP |  
| NULL | SA_REP |  
| 50 | SH_CLERK |  
| 10 | AD_ASST |  
| 20 | MK_MAN |  
| 20 | MK_REP |  
| 40 | HR_REP |  
| 70 | PR_REP |  
| 110 | AC_MGR |  
| 110 | AC_ACCOUNT |  
| NULL | IT_PROG |  
+-----+-----+  
21 rows in set (0.00 sec)  
  
mysql>
```

DESCRIBE

- Use the DESCRIBE command to **display** the **structure** of a table

```

ina — mysql -u root -p — mysql — mysql -u root -p — 68x19
mysql> DESCRIBE employees;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| employee_id    | int unsigned  | NO   | PRI | NULL    |       |
| first_name     | varchar(20)   | YES  |     | NULL    |       |
| last_name      | varchar(25)   | NO   |     | NULL    |       |
| email          | varchar(25)   | NO   |     | NULL    |       |
| phone_number   | varchar(20)   | YES  |     | NULL    |       |
| hire_date      | date          | NO   |     | NULL    |       |
| job_id         | varchar(10)   | NO   | MUL | NULL    |       |
| salary         | decimal(8,2)  | NO   |     | NULL    |       |
| commission_pct | decimal(2,2)  | YES  |     | NULL    |       |
| manager_id     | int unsigned  | YES  | MUL | NULL    |       |
| department_id  | int unsigned  | YES  | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
11 rows in set (0.00 sec)

mysql>
  
```

SUMMARY

- **INSERT INTO** : To add new values (data) into a table
- **UPDATE ... SET** : To update existing value of a column with a new value
- **SELECT ... FROM** : To retrieve records in table(s)
- **SELECT ... FROM ... WHERE** : To retrieve record that match certain condition

Simple Handwritten Exercise

- Given the following relation schemas, construct the SQL statement for following tasks:

```
CUSTOMER (customer_id, store_id, first_name, last_name, email, address, active)
RENTAL (rental_id, rental_date, inventory_id, customer_id, return_date, staff_id)
INVENTORY (inventory_id, film_id, store_id)
STORE (store_id, manager_staff_id, location)
STAFF (staff_id, first_name, last_name, address, email, store_id, salary)
PAYMENT (payment_id, customer_id, staff_id, rental_id, amount, payment_date)
FILM (film_id, title, description, rental_duration, rental_rate)
```

- 1) List all films title, rental duration and rental rate.
- 2) List all customers' details who registered to store with ID of ST_002.
- 3) Insert a new film in database which has the film ID 'F_0099', title 'My Neighbour Totoro', description 'Is a 1998 Japanese animated film by Hayao Miyazaki', rental duration 14 days, and rental rate of RM8.

Simple Handwritten Exercise (Answer)

1. List all films title, rental duration and rental rate.

```
SELECT title, rental_duration, rental_rate FROM FILM;
```

2. List all customers' details who registered to store with ID of ST_002.

```
SELECT * FROM CUSTOMER WHERE store_id='ST_002';
```

3. Insert a new film in database which has the film ID 'F_0099', title 'My Neighbour Totoro', description 'Is a 1998 Japanese animated film by Hayao Miyazaki', rental duration 14 days, and rental rate of RM8.

```
INSERT INTO FILM (film_id, title, description,  
rental_duration, rental_rate)  
VALUES ('F_0099', 'My Neighbour Totoro', 'Is a 1998 Japanese  
animated film by Hayao Miyazaki', 14, 8)
```



UTM
UNIVERSITI TEKNOLOGI MALAYSIA

***Innovating Solutions
Menginovasi Penyelesaian***