Course	:	BIC 21404 Database
Session	:	II 2024/2025
Lab sheet	:	3
Objective	:	At the end of the session, students are able to: i. Import database schema ii. Queries using SQL a. SELECT statement b. WHERE statement c. Rename the column heading d. Concatenation e. Distinct iii. Backup the database and table

UNIT 1: Import Database or table on XAMPP phpMyAdmin

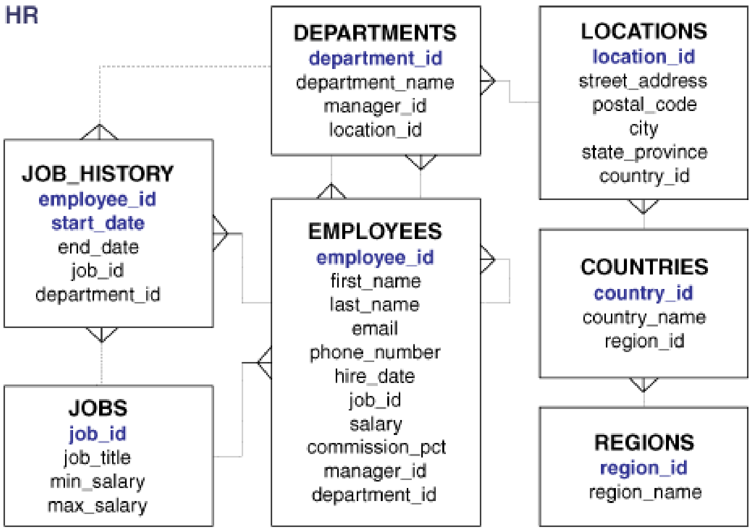


Figure 3-1: HR Schema

1. Open the Database in phpMyAdmin.
2. Click on the Databases from the top menu.
3. Create new database, **hr_main**.
4. Click on the Import tab.
5. Browse **hr-schema-mysql.sql** file by clicking on the 'Choose File' option that you wish to import.
   - hr-schema-mysql.sql is provided in the Author
6. And then click on the 'Go' button at the bottom.

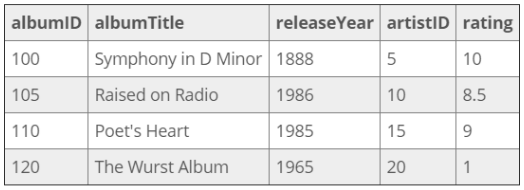
Unit 2: Writing SQL Statements

- SQL statements are not case sensitive
- 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
- SQL statement is terminated by a semicolon (;)
- Keyword typically are entered in uppercase; all other words, such as table names and columns names are entered in lowercase.

Queries

- A query is **a question or request for data**.
- When we make queries to the database, we need to use common language to get the information.
- Structured Query Language (SQL) is a fairly universal language for queries

- A **database** stores information in tables, consisting of rows and columns of information
- In order to ask a well-defined question, you're going to need to know the following:
  - **Where** are you getting the data from? **Which** table?
  - **What** are you selecting? Which fields? And,
  - Are there any **conditions** to the selection?



albumID	albumTitle	releaseYear	artistID	rating
100	Symphony in D Minor	1888	5	10
105	Raised on Radio	1986	10	8.5
110	Poet's Heart	1985	15	9
120	The Wurst Album	1965	20	1

How to Build a Query

- Let's build a query based on the sample data.
- Example:
  - We want to know the list of company's employees
- Therefore, our SQL query will be close to the plain statement
  - **SELECT**, (go get from the database),
    FROM (which table are you looking at?), and
    WHERE (what criteria?).

SELECT Statement

The `SELECT` statement is used to query data from tables. The retrieved rows are selected from one or more table. Such a result table can be used as the basis of a report.

The basic syntax of the `SELECT` statement is:

```
SELECT select_expr [, select_expr ...] FROM table_name
```

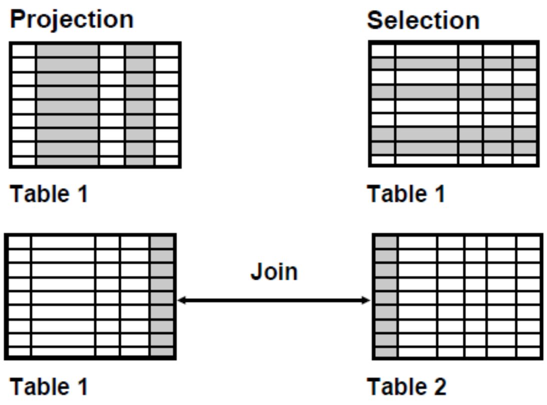
Each ***select_expr*** indicates a column that you want to retrieve.
* is used instead of ***select_expr*** as a wildcard if you want to retrieve all columns from a table.

- **SELECT** identifies the columns to be displayed.
- **FROM** identifies the table containing those columns.

Capabilities of SELECT statement

A SELECT statement retrieves information from the database. With a SELECT statement, you can do the following:

- **Projection**: Select the columns in a table that are returned by a query. Select as few or as many of the columns as required.
- **Selection**: Select the rows in a table that are returned by a query. Various criteria can be used to restrict the rows that are retrieved.
- **Joins**: Bring together data that is stored in different tables by specifying the link between them.



Try this:

(i) Selecting All Columns

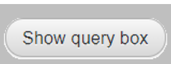
```
SELECT *
FROM departments;

SELECT *
FROM employees;
```

(ii) Selecting Specific Columns

```
SELECT department_id, location_id
FROM departments;
```

You might want to practice other queries yourself.
For example, write a query to retrieve all employees first and last names.

Tips



- Note that the Show query box usually at the top of the page which you click to keep the recent query you wrote.
- Two single quotations are used for the text and date.

WHERE Clause

- In the `WHERE` clause, a **condition** is used to select rows from a table.
- These selected rows form the intermediate result of the `WHERE` clause.
- The `WHERE` clause acts as a kind of filter.
- Basic syntax:

```
SELECT select_expr [, select_expr ...]

FROM table [WHERE where_condition]
```

Try this:	(i) Retrieve all employees in department 100 SELECT * FROM `employees` WHERE department_id='100'
	(ii) Retrieve all employees with IT_PROG job SELECT * FROM `employees` where job_id='IT_PROG'
	(iii)What is department 100? SELECT * FROM departments where department_id=100
	(iv) What is job name of IT_PROG? SELECT * FROM jobs WHERE job_id='IT_PROG'
	(v) Find all employees hired on the date '1987-06-17' SELECT * FROM employees WHERE hire_date='1987-06-17'
	(vi) get the information of the employees whose salary is $17000 SELECT * FROM employees WHERE salary=17000

Rename a column heading: Column Alias

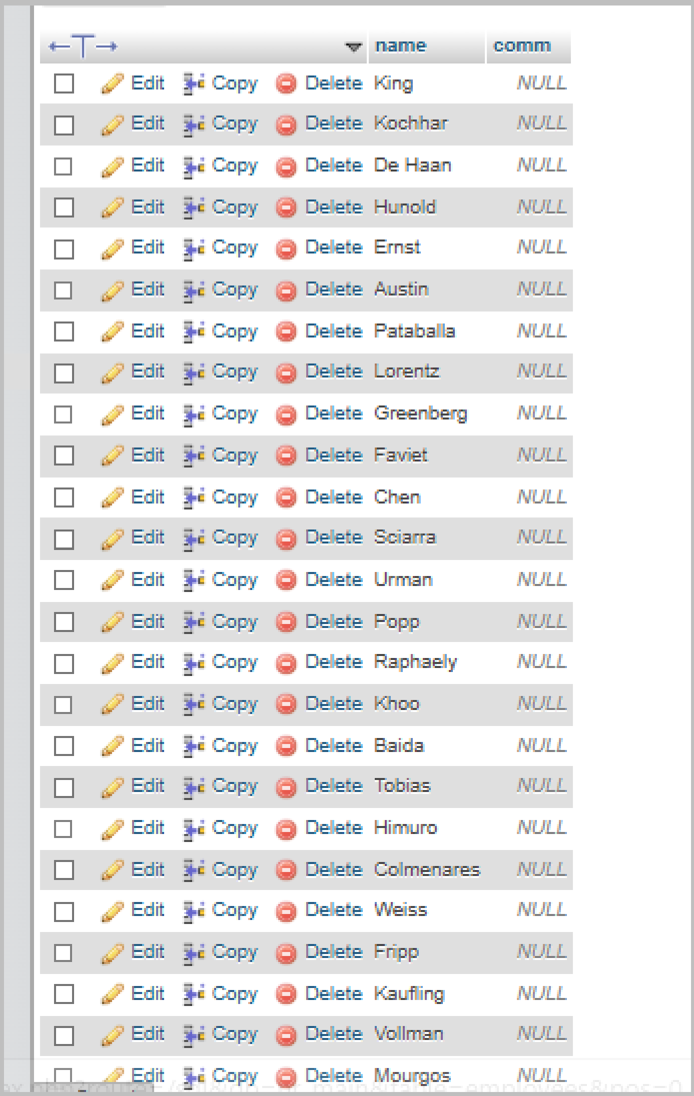
A column alias:

- Rename a column heading
- Is useful with calculations
- Immediately follows the column name
- Required double quotation marks if it contains spaces or special characters, or if it is case-sensitive

Try this:

```
1.    SELECT last_name AS name, commission_pct comm
              FROM employees;
```

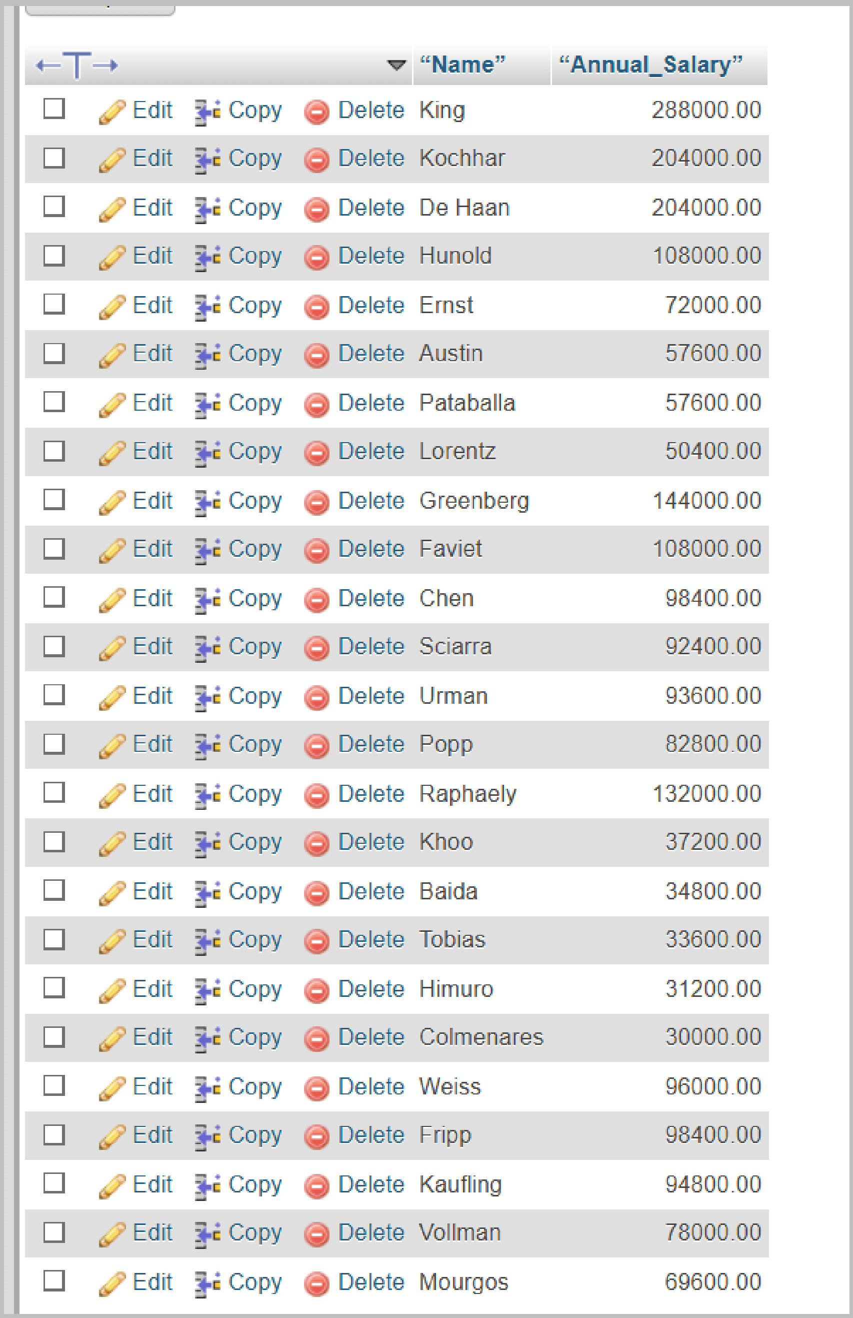
Show part of the output:



←T→				name	comm
□	Edit	Copy	Delete	King	NULL
□	Edit	Copy	Delete	Kochhar	NULL
□	Edit	Copy	Delete	De Haan	NULL
□	Edit	Copy	Delete	Hunold	NULL
□	Edit	Copy	Delete	Ernst	NULL
□	Edit	Copy	Delete	Austin	NULL
□	Edit	Copy	Delete	Pataballa	NULL
□	Edit	Copy	Delete	Lorentz	NULL
□	Edit	Copy	Delete	Greenberg	NULL
□	Edit	Copy	Delete	Faviet	NULL
□	Edit	Copy	Delete	Chen	NULL
□	Edit	Copy	Delete	Sciarra	NULL
□	Edit	Copy	Delete	Urman	NULL
□	Edit	Copy	Delete	Popp	NULL
□	Edit	Copy	Delete	Raphaely	NULL
□	Edit	Copy	Delete	Khoo	NULL
□	Edit	Copy	Delete	Baida	NULL
□	Edit	Copy	Delete	Tobias	NULL
□	Edit	Copy	Delete	Himuro	NULL
□	Edit	Copy	Delete	Colmenares	NULL
□	Edit	Copy	Delete	Weiss	NULL
□	Edit	Copy	Delete	Fripp	NULL
□	Edit	Copy	Delete	Kaufling	NULL
□	Edit	Copy	Delete	Vollman	NULL
□	Edit	Copy	Delete	Mourgos	NULL

2. SELECT last_name “Name”, salary*12 “Annual Salary”
   FROM employees;

Show part of the output:



←T→			▼	“Name”	“Annual_Salary”
☐	Edit	Copy	Delete	King	288000.00
☐	Edit	Copy	Delete	Kochhar	204000.00
☐	Edit	Copy	Delete	De Haan	204000.00
☐	Edit	Copy	Delete	Hunold	108000.00
☐	Edit	Copy	Delete	Ernst	72000.00
☐	Edit	Copy	Delete	Austin	57600.00
☐	Edit	Copy	Delete	Pataballa	57600.00
☐	Edit	Copy	Delete	Lorentz	50400.00
☐	Edit	Copy	Delete	Greenberg	144000.00
☐	Edit	Copy	Delete	Faviet	108000.00
☐	Edit	Copy	Delete	Chen	98400.00
☐	Edit	Copy	Delete	Sciarra	92400.00
☐	Edit	Copy	Delete	Urman	93600.00
☐	Edit	Copy	Delete	Popp	82800.00
☐	Edit	Copy	Delete	Raphaely	132000.00
☐	Edit	Copy	Delete	Khoo	37200.00
☐	Edit	Copy	Delete	Baida	34800.00
☐	Edit	Copy	Delete	Tobias	33600.00
☐	Edit	Copy	Delete	Himuro	31200.00
☐	Edit	Copy	Delete	Colmenares	30000.00
☐	Edit	Copy	Delete	Weiss	96000.00
☐	Edit	Copy	Delete	Fripp	98400.00
☐	Edit	Copy	Delete	Kaufling	94800.00
☐	Edit	Copy	Delete	Vollman	78000.00
☐	Edit	Copy	Delete	Mourgos	69600.00

Concatenation Operator

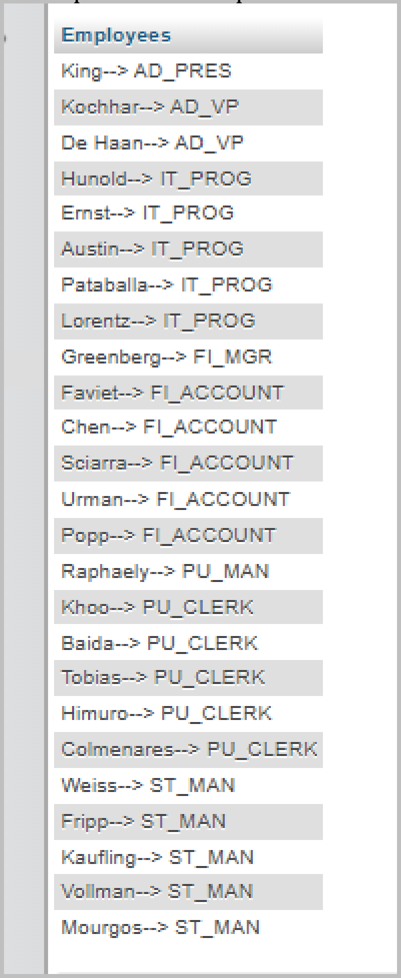
A concatenation operator:

- Links columns or character strings to other columns
- **CONCAT()** function is used to add two or more strings.
- Creates a resultant column that is a character expression

Try this:

```
1.   SELECT CONCAT(last_name,'--> ', job_id) AS  "Employees"
          FROM employees;
```

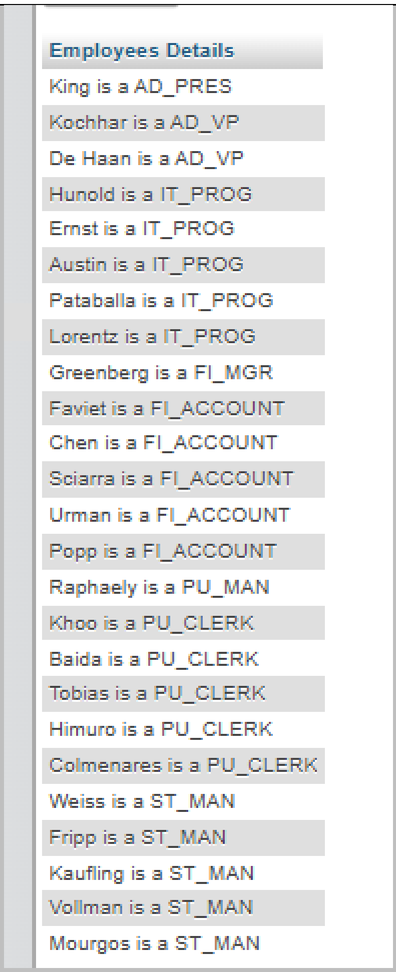
Show part of the output:



```
Employees
King--> AD_PRES
Kochhar--> AD_VP
De Haan--> AD_VP
Hunold--> IT_PROG
Ernst--> IT_PROG
Austin--> IT_PROG
Pataballa--> IT_PROG
Lorentz--> IT_PROG
Greenberg--> FI_MGR
Faviet--> FI_ACCOUNT
Chen--> FI_ACCOUNT
Sciarra--> FI_ACCOUNT
Urman--> FI_ACCOUNT
Popp--> FI_ACCOUNT
Raphaely--> PU_MAN
Khoo--> PU_CLERK
Baida--> PU_CLERK
Tobias--> PU_CLERK
Himuro--> PU_CLERK
Colmenares--> PU_CLERK
Weiss--> ST_MAN
Fripp--> ST_MAN
Kaufling--> ST_MAN
Vollman--> ST_MAN
Mourgos--> ST_MAN
```

```
2.   SELECT CONCAT(last_name, ' is a ', job_id) AS "Employees Details"
          FROM employees;
```

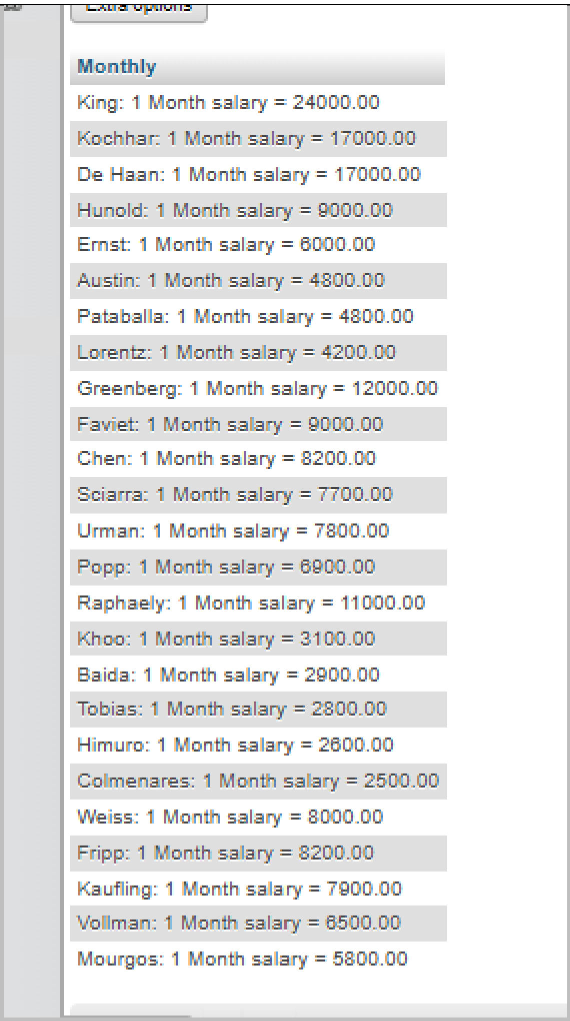
Show part of the output:



Employees Details
King is a AD_PRES
Kochhar is a AD_VP
De Haan is a AD_VP
Hunold is a IT_PROG
Ernst is a IT_PROG
Austin is a IT_PROG
Pataballa is a IT_PROG
Lorentz is a IT_PROG
Greenberg is a FI_MGR
Faviet is a FI_ACCOUNT
Chen is a FI_ACCOUNT
Sciarra is a FI_ACCOUNT
Urman is a FI_ACCOUNT
Popp is a FI_ACCOUNT
Raphaely is a PU_MAN
Khoo is a PU_CLERK
Baida is a PU_CLERK
Tobias is a PU_CLERK
Himuro is a PU_CLERK
Colmenares is a PU_CLERK
Weiss is a ST_MAN
Fripp is a ST_MAN
Kaufling is a ST_MAN
Vollman is a ST_MAN
Mourgos is a ST_MAN

3.
```
SELECT CONCAT(last_name, ': 1 Month salary = ', salary) AS Monthly
       FROM employees;
```

Show part of the output:



```
Monthly
King: 1 Month salary = 24000.00
Kochhar: 1 Month salary = 17000.00
De Haan: 1 Month salary = 17000.00
Hunold: 1 Month salary = 9000.00
Ernst: 1 Month salary = 6000.00
Austin: 1 Month salary = 4800.00
Pataballa: 1 Month salary = 4800.00
Lorentz: 1 Month salary = 4200.00
Greenberg: 1 Month salary = 12000.00
Faviet: 1 Month salary = 9000.00
Chen: 1 Month salary = 8200.00
Sciarra: 1 Month salary = 7700.00
Urman: 1 Month salary = 7800.00
Popp: 1 Month salary = 6900.00
Raphaely: 1 Month salary = 11000.00
Khoo: 1 Month salary = 3100.00
Baida: 1 Month salary = 2900.00
Tobias: 1 Month salary = 2800.00
Himuro: 1 Month salary = 2600.00
Colmenares: 1 Month salary = 2500.00
Weiss: 1 Month salary = 8000.00
Fripp: 1 Month salary = 8200.00
Kaufling: 1 Month salary = 7900.00
Vollman: 1 Month salary = 6500.00
Mourgos: 1 Month salary = 5800.00
```

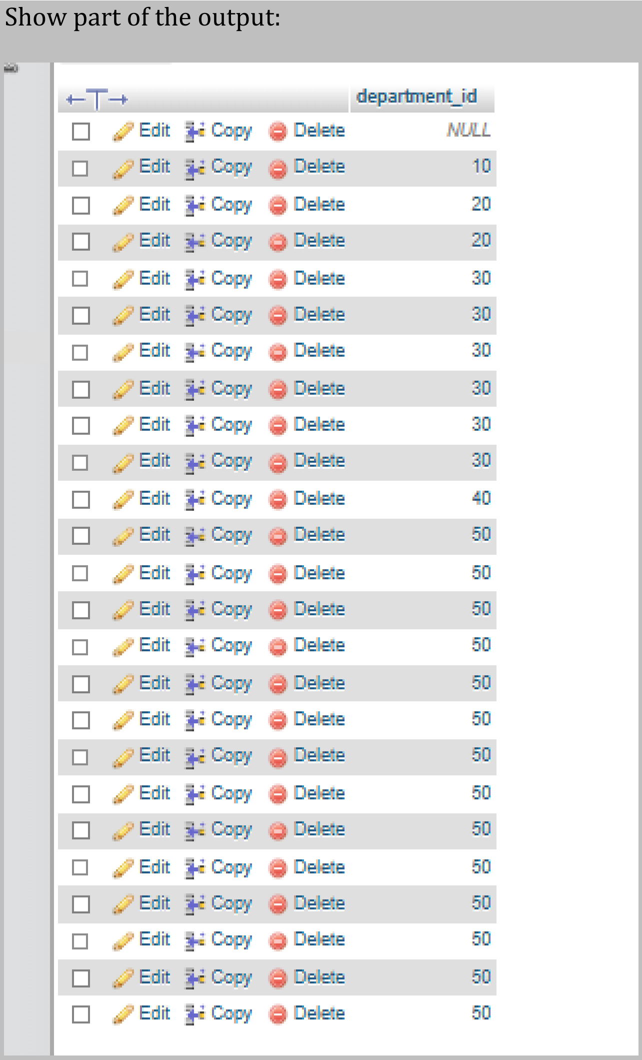
DISTINCT

- The SELECT DISTINCT statement is used to return only distinct (different) values.
- Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.
- Use the DISTINCT keyword to eliminate duplicate rows in the result.

Try this:

1.
```
SELECT department_id
   FROM employees;
```

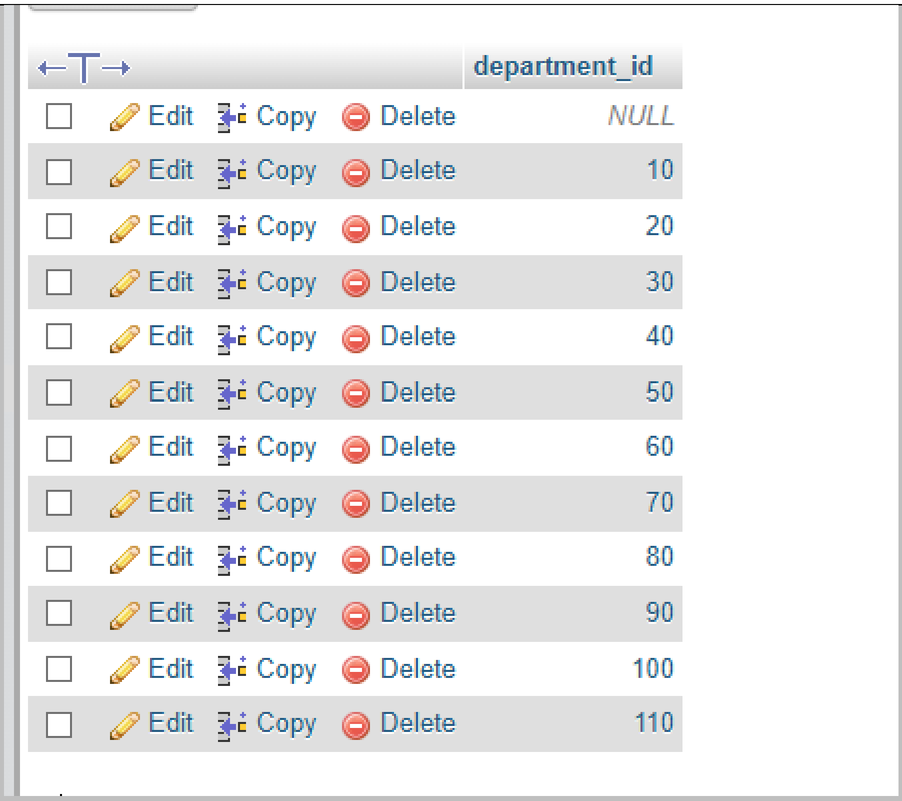
Show part of the output:



←T→				department_id
□	Edit	Copy	Delete	NULL
□	Edit	Copy	Delete	10
□	Edit	Copy	Delete	20
□	Edit	Copy	Delete	20
□	Edit	Copy	Delete	30
□	Edit	Copy	Delete	30
□	Edit	Copy	Delete	30
□	Edit	Copy	Delete	30
□	Edit	Copy	Delete	30
□	Edit	Copy	Delete	30
□	Edit	Copy	Delete	40
□	Edit	Copy	Delete	50
□	Edit	Copy	Delete	50
□	Edit	Copy	Delete	50
□	Edit	Copy	Delete	50
□	Edit	Copy	Delete	50
□	Edit	Copy	Delete	50
□	Edit	Copy	Delete	50
□	Edit	Copy	Delete	50
□	Edit	Copy	Delete	50
□	Edit	Copy	Delete	50
□	Edit	Copy	Delete	50
□	Edit	Copy	Delete	50
□	Edit	Copy	Delete	50
□	Edit	Copy	Delete	50

2.
```
SELECT DISTINCT department_id
   FROM employees;
```

Show part of the output:

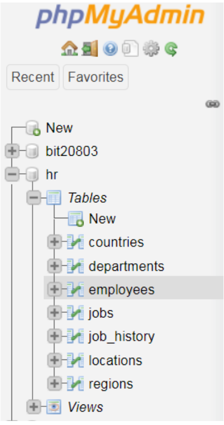


←T→				department_id
□	Edit	Copy	Delete	NULL
□	Edit	Copy	Delete	10
□	Edit	Copy	Delete	20
□	Edit	Copy	Delete	30
□	Edit	Copy	Delete	40
□	Edit	Copy	Delete	50
□	Edit	Copy	Delete	60
□	Edit	Copy	Delete	70
□	Edit	Copy	Delete	80
□	Edit	Copy	Delete	90
□	Edit	Copy	Delete	100
□	Edit	Copy	Delete	110

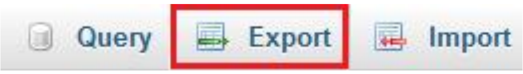
Do you notice the difference?

Yes the difference is the list for distinct less than select

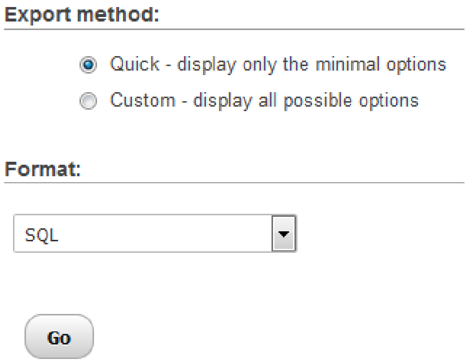
UNIT 3: Backup or Database and Table in PHPmyAdmin - MySQL



- Select the source database on the left pane.



- Click on the Export tab in the top center pane.


Export method:
● Quick - display only the minimal options
○ Custom - display all possible options
Format:
SQL
Go

- On the next page you must select a Quick or Custom export method.
    - 'Quick' method: to download the .sql file immediately.
    - 'Custom' method: to get more control over the data.
- From the dropdown menu, choose the format you'd like to save the file as. SQL is most common.
- Click the Go button to continue.
- Check the new created file.