Web Application Development : MySQL and Manipulating MySQL with PHP

Week 4

SWINBURNE UNIVERSITY OF TECHNOLOGY

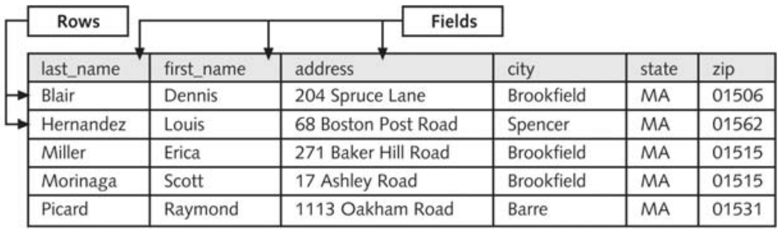
Content of this Lecture

- Basics of databases
- Working with MySQL databases
- Managing user accounts, databases and tables
- Working with data records
- Accessing MySQL database with PHP
- Creating and deleting databases and tables
- Creating, updating, selecting and deleting records

2

Introduction to Databases

- A **database** is an ordered collection of information from which a computer program can quickly access information
- Each **row** in a database **table** is called a **record** (**tuple**)
- A **record** in a database is a single complete set of related information
- Each column in a database table is called a **field** (**attribute**)
- **Fields** are the individual categories of information stored in a record


Rows
Fields

last_name	first_name	address	city	state	zip
Blair	Dennis	204 Spruce Lane	Brookfield	MA	01506
Hernandez	Louis	68 Boston Post Road	Spencer	MA	01562
Miller	Erica	271 Baker Hill Road	Brookfield	MA	01515
Morinaga	Scott	17 Ashley Road	Brookfield	MA	01515
Picard	Raymond	1113 Oakham Road	Barre	MA	01531

Employee directory database

3

Relational Databases

- A **relational database** consists of one or more related tables, where relationships between tables are implemented by primary/foreign keys
- A **primary key** is a field that contains a unique identifier for each record in a primary table
- A **primary key** is a type of index, which identifies records in a database to make retrievals and sorting faster
- A **foreign key** is a field in a related table that refers to the primary key in a primary table

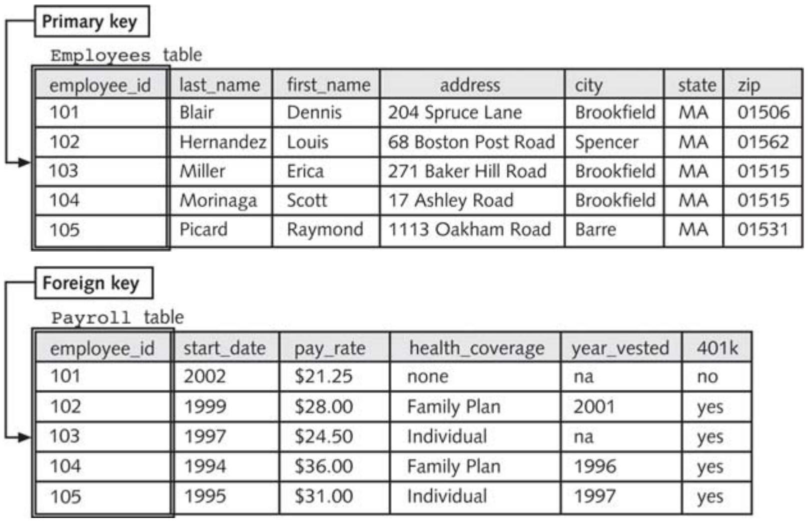
4

One-to-One Relationships

- A **one-to-one relationship** exists between two tables when a related table contains exactly one record for each record in the primary table
- Create one-to-one relationships to break information into multiple, logical sets
- Information in the tables in a one-to-one relationship can be placed within a single table
- Make the information in one of the tables confidential and accessible only by certain individuals
- Alternatively, create a base table including all information about an object, and then create views for different people to access

5

One-to-One Relationships (continued)



Primary key

Employees table

employee_id	last_name	first_name	address	city	state	zip
101	Blair	Dennis	204 Spruce Lane	Brookfield	MA	01506
102	Hernandez	Louis	68 Boston Post Road	Spencer	MA	01562
103	Miller	Erica	271 Baker Hill Road	Brookfield	MA	01515
104	Morinaga	Scott	17 Ashley Road	Brookfield	MA	01515
105	Picard	Raymond	1113 Oakham Road	Barre	MA	01531

Foreign key

Payroll table

employee_id	start_date	pay_rate	health_coverage	year_vested	401k
101	2002	$21.25	none	na	no
102	1999	$28.00	Family Plan	2001	yes
103	1997	$24.50	Individual	na	yes
104	1994	$36.00	Family Plan	1996	yes
105	1995	$31.00	Individual	1997	yes

One-to-one relationship

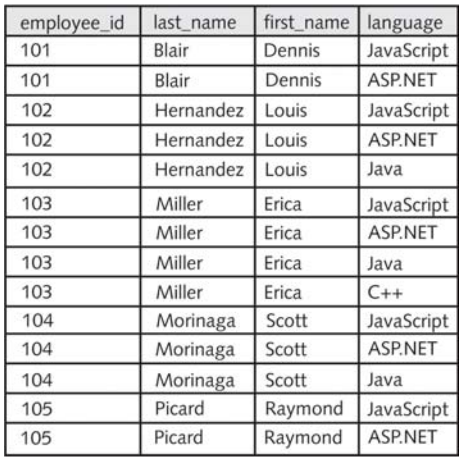
6

One-to-Many Relationship

- A **one-to-many relationship** exists in a relational database when one record in a primary table has many related records in a related table
- Breaking tables into multiple related tables to reduce redundant information is called **normalization**
- Provides a more efficient and less redundant method of storing this information in a database

7

One-to-Many Relationship (continued)

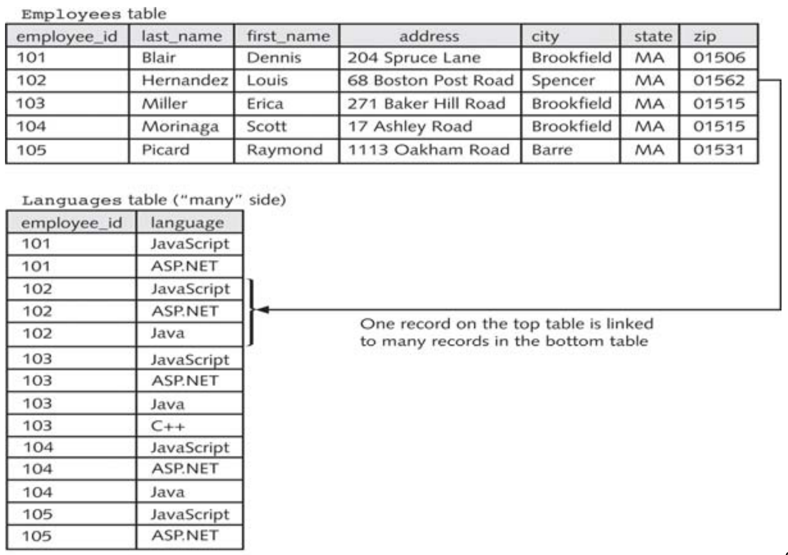


employee_id	last_name	first_name	language
101	Blair	Dennis	JavaScript
101	Blair	Dennis	ASP.NET
102	Hernandez	Louis	JavaScript
102	Hernandez	Louis	ASP.NET
102	Hernandez	Louis	Java
103	Miller	Erica	JavaScript
103	Miller	Erica	ASP.NET
103	Miller	Erica	Java
103	Miller	Erica	C++
104	Morinaga	Scott	JavaScript
104	Morinaga	Scott	ASP.NET
104	Morinaga	Scott	Java
105	Picard	Raymond	JavaScript
105	Picard	Raymond	ASP.NET

Table with redundant information

8

One-to-Many Relationship (continued)



Employees table

employee_id	last_name	first_name	address	city	state	zip
101	Blair	Dennis	204 Spruce Lane	Brookfield	MA	01506
102	Hernandez	Louis	68 Boston Post Road	Spencer	MA	01562
103	Miller	Erica	271 Baker Hill Road	Brookfield	MA	01515
104	Morinaga	Scott	17 Ashley Road	Brookfield	MA	01515
105	Picard	Raymond	1113 Oakham Road	Barre	MA	01531

Languages table ("many" side)

employee_id	language
101	JavaScript
101	ASP.NET
102	JavaScript
102	ASP.NET
102	Java
103	JavaScript
103	ASP.NET
103	Java
103	C++
104	JavaScript
104	ASP.NET
104	Java
105	JavaScript
105	ASP.NET

One record on the top table is linked
to many records in the bottom table

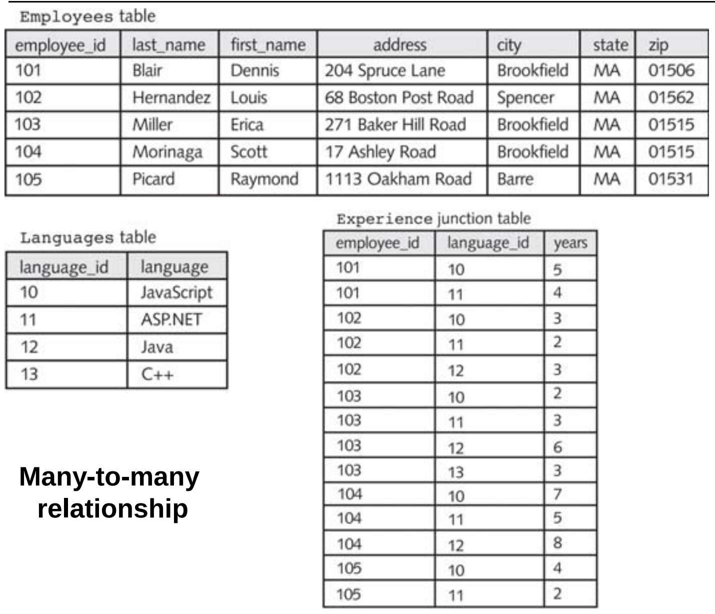
9

Many-to-Many Relationship

- A **many-to-many relationship** exists in a relational database when many records in one table are related to many records in another table
  e.g. relationship between programmers and languages
- Must use a **junction table** which creates a one-to-many relationship for each of the two tables in a many-to-many relationship
- A junction table contains foreign keys from the two tables

10

Many-to-Many Relationship (continued)



Employees table

employee_id	last_name	first_name	address	city	state	zip
101	Blair	Dennis	204 Spruce Lane	Brookfield	MA	01506
102	Hernandez	Louis	68 Boston Post Road	Spencer	MA	01562
103	Miller	Erica	271 Baker Hill Road	Brookfield	MA	01515
104	Morinaga	Scott	17 Ashley Road	Brookfield	MA	01515
105	Picard	Raymond	1113 Oakham Road	Barre	MA	01531

Languages table

language_id	language
10	JavaScript
11	ASP.NET
12	Java
13	C++

Experience junction table

employee_id	language_id	years
101	10	5
101	11	4
102	10	3
102	11	2
102	12	3
103	10	2
103	11	3
103	12	6
103	13	3
104	10	7
104	11	5
104	12	8
105	10	4
105	11	2

Many-to-many relationship

11

Working with Database Management Systems

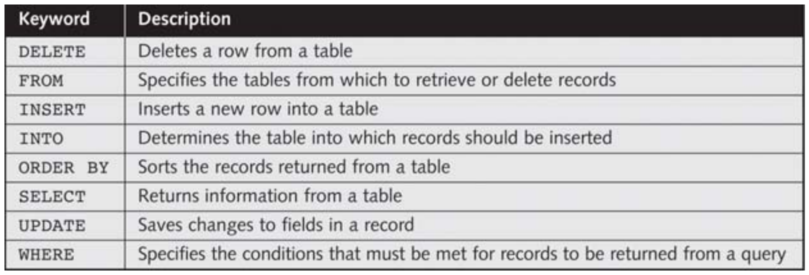
- A **relational database management system** (or RDBMS) is a system that stores and manages data in a relational format
- A **schema** is the structure of a database including its tables, fields, and relationships
- A **query** is a structured set of instructions and criteria for retrieving, adding, modifying, and deleting database information
- **Structured query language** (or SQL – pronounced as sequel) is a standard data manipulation language used among many database management systems
- **Open database connectivity** (or ODBC) allows ODBC-compliant applications to access any data source for which there is an ODBC driver

Examples of RDBMS

12

Querying Databases with SQL

Common SQL keywords



Keyword	Description
`DELETE`	Deletes a row from a table
`FROM`	Specifies the tables from which to retrieve or delete records
`INSERT`	Inserts a new row into a table
`INTO`	Determines the table into which records should be inserted
`ORDER BY`	Sorts the records returned from a table
`SELECT`	Returns information from a table
`UPDATE`	Saves changes to fields in a record
`WHERE`	Specifies the conditions that must be met for records to be returned from a query

Note: In a SELECT statement, GROUP BY can be used in conjunction with aggregate functions to group the result set by one or more fields. HAVING can further be used together with GROUP BY to specify conditions on groups constructed by GROUP BY.

13

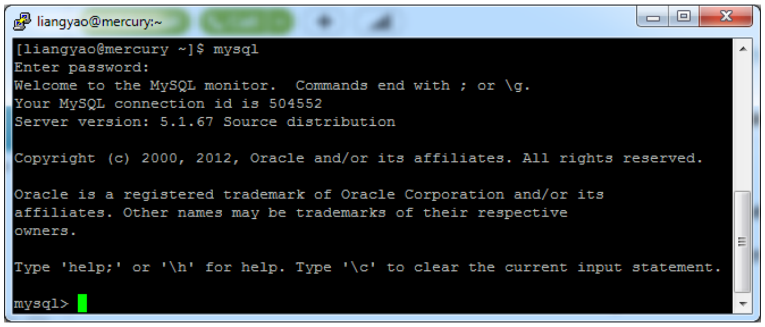
Getting Started with MySQL

- MySQL is an open source database server, and it is fast and reliable.
- There are several ways to interface with a MySQL database server:
  - Using MySQL Monitor, a command-line program
  - Using phpMyAdmin, a web interface program (http://mercury.ict.swin.edu.au/mysql/)
  - Using PHP database functions within PHP scripts
- Our MySQL database server is: **`mysql.ict.swin.edu.au`**
- See: https://csg.ict.swin.edu.au/livecsg/help/Doc_databases_student_mysql

14

Logging in to MySQL Monitor

- *We will be accessing the MySQL database server at* **`mysql.ict.swin.edu.au`** *Your account and database have been created.*



```
[liangyao@mercury ~]$ mysql
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 504552
Server version: 5.1.67 Source distribution

Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

15

Working with the MySQL Monitor

- At the `mysql>` command prompt terminate the command with a semicolon

```
mysql> SELECT * FROM inventory;
```

- Without a semicolon, the MySQL Monitor enters a multiple-line command and changes the prompt to ->

```
mysql> SELECT * FROM inventory
    -> WHERE make = "Holden";
```

- The SQL keywords entered in the MySQL Monitor are *not* case sensitive

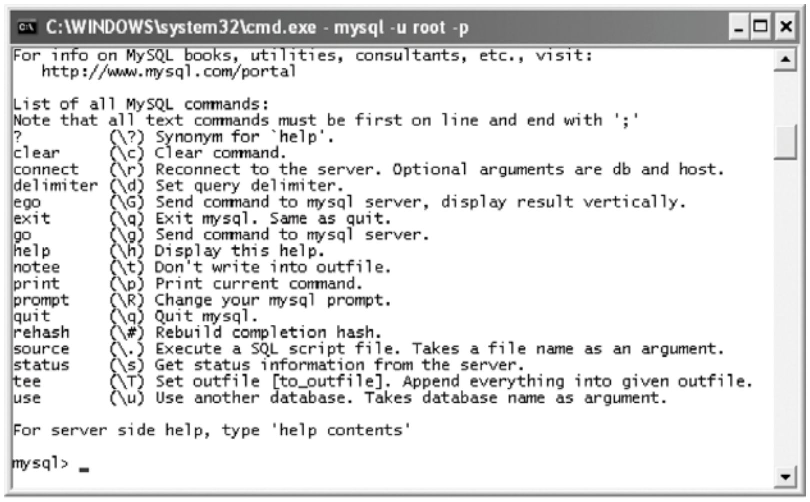
16

Understanding MySQL Identifiers

- Identifiers for databases, tables, fields, indexes, and aliases
- An **alias** is an alternate name used to refer to a table or field in SQL statements
- The case sensitivity of database and table identifiers depends on the operating system
  - Not case sensitive on Windows platforms
  - Case sensitive on UNIX/Linux systems
- MySQL stores each database in a directory of the same name as the database identifier
- Field and index identifiers are case insensitive on all platforms

17

Getting Help with MySQL Commands



```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
For info on MySQL books, utilities, consultants, etc., visit:
   http://www.mysql.com/portal

List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
?         (\?) Synonym for `help'.
clear     (\c) Clear command.
connect   (\r) Reconnect to the server. Optional arguments are db and host.
delimiter (\d) Set query delimiter.
ego       (\G) Send command to mysql server, display result vertically.
exit      (\q) Exit mysql. Same as quit.
go        (\g) Send command to mysql server.
help      (\h) Display this help.
notee     (\t) Don't write into outfile.
print     (\p) Print current command.
prompt    (\R) Change your mysql prompt.
quit      (\q) Quit mysql.
rehash    (\#) Rebuild completion hash.
source    (\.) Execute a SQL script file. Takes a file name as an argument.
status    (\s) Get status information from the server.
tee       (\T) Set outfile [to_outfile]. Append everything into given outfile.
use       (\u) Use another database. Takes database name as argument.

For server side help, type 'help contents'

mysql> _
```

MySQL command help

18

Securing the Initial MySQL Accounts

- Deleting the Anonymous User Account

```
mysql> DELETE FROM mysql.user WHERE User = '';
mysql> FLUSH PRIVILEGES;
```

- Assigning a Password to the Root Account

```
mysql> UPDATE mysql.user SET Password =
PASSWORD('newpwd')
  -> WHERE User = 'root';
mysql> FLUSH PRIVILEGES;
```

- The password assigned to the root account and other user accounts is case sensitive

19

Creating Users and Grant Privileges

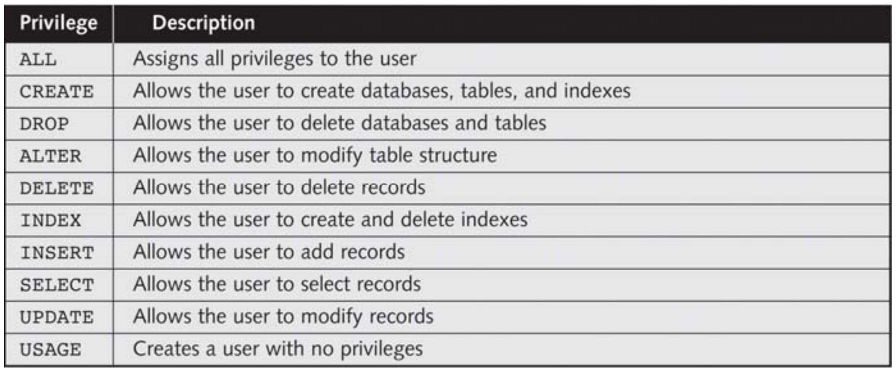
- Use a GRANT statement to create user accounts and assign privileges
- **Privileges** are the operations that a user can perform with a database
- The GRANT statement creates the user account if it does not exist and assigns the specified privileges

```
GRANT privilege [(column)][, privilege [(columns)]]...
     ON {table | * | *.* | database.*}
     TO user [IDENTIFIED BY 'password'];
```

- If the user account already exists, the GRANT statement just updates the privileges

20

Common MySQL Database Privileges



Privilege	Description
ALL	Assigns all privileges to the user
CREATE	Allows the user to create databases, tables, and indexes
DROP	Allows the user to delete databases and tables
ALTER	Allows the user to modify table structure
DELETE	Allows the user to delete records
INDEX	Allows the user to create and delete indexes
INSERT	Allows the user to add records
SELECT	Allows the user to select records
UPDATE	Allows the user to modify records
USAGE	Creates a user with no privileges

21

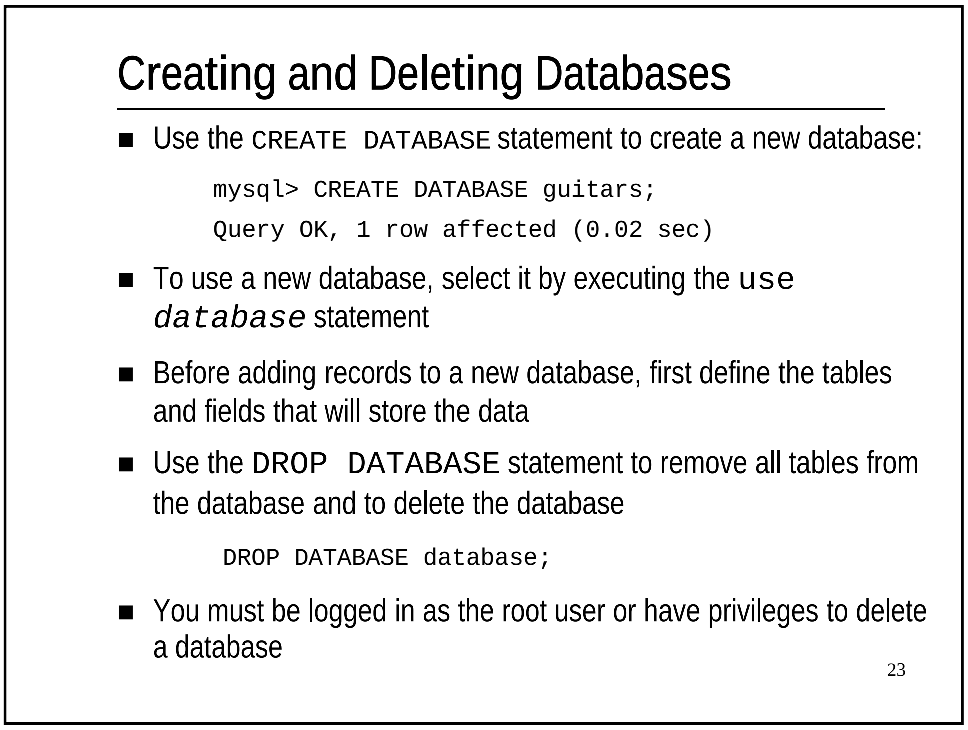
Revoking Privileges and Deleting Users

- You must be logged in with the root account or have sufficient privileges to revoke privileges from another user account

```
REVOKE privilege [(column)][, privilege [(columns)]]...
     ON {table | * | *.* | database.*}
     FROM user;
```

- The REVOKE ALL PRIVILEGES statement removes all privileges from a user account for a specified table or database
- Before deleting a user, you must first revoke all privileges assigned to the user account for all databases
  - Use the `REVOKE ALL PRIVILEGES` statement
  - View the privileges assigned to a user account with the `SHOW GRANTS FOR` *`user`* statement
- To delete an existing user, use the `DROP USER` *`user`* statement to delete the account from the user table in the `mysql` database

22



Creating and Deleting Databases

- Use the `CREATE DATABASE` statement to create a new database:

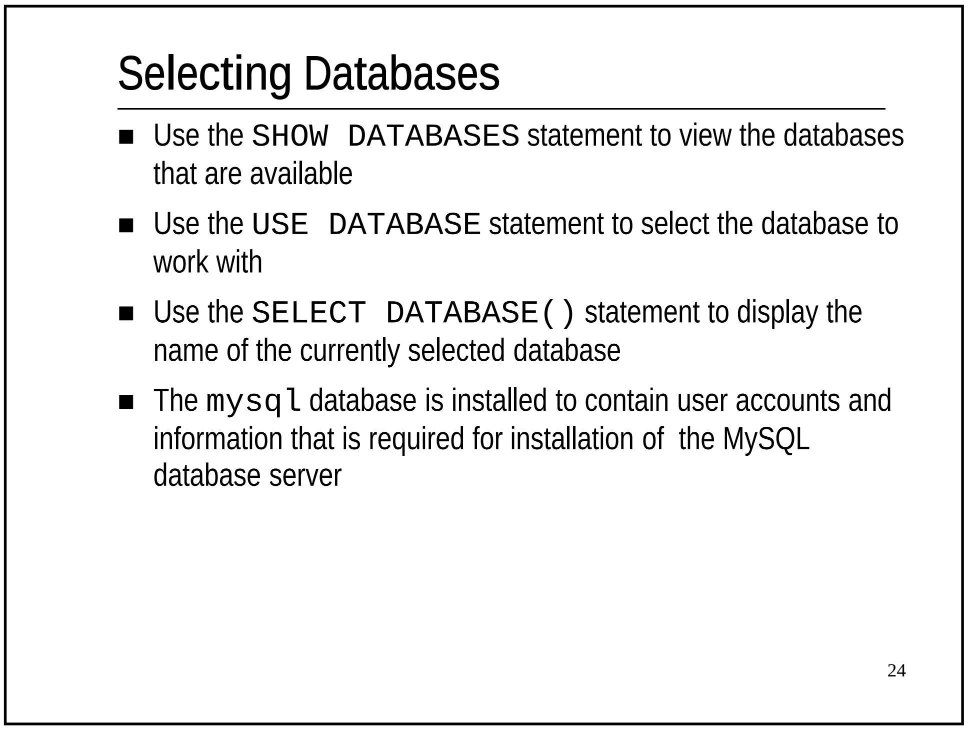
```
mysql> CREATE DATABASE guitars;
Query OK, 1 row affected (0.02 sec)
```

- To use a new database, select it by executing the `use` *`database`* statement
- Before adding records to a new database, first define the tables and fields that will store the data
- Use the `DROP DATABASE` statement to remove all tables from the database and to delete the database

```
DROP DATABASE database;
```

- You must be logged in as the root user or have privileges to delete a database

23



Selecting Databases

- Use the `SHOW DATABASES` statement to view the databases that are available
- Use the `USE DATABASE` statement to select the database to work with
- Use the `SELECT DATABASE()` statement to display the name of the currently selected database
- The `mysql` database is installed to contain user accounts and information that is required for installation of the MySQL database server

24

Creating and Deleting Tables

- The `CREATE TABLE` statement specifies the table and column names and the data type for each column

```
CREATE TABLE table_name (column_name TYPE, ...);
```

- Execute the `USE` statement to select a database before executing the `CREATE TABLE` statement
- The `DROP TABLE` statement removes all data and the table definition

```
DROP TABLE table;
```

- You must be logged in as the `root` user or have `DROP` privileges to delete a table

25

Example - Creating Table

```
mysql> CREATE TABLE inventory (
       item_number int NOT NULL AUTO_INCREMENT,
       make varchar(30) NOT NULL,
       model varchar(30) NOT NULL,
       price double NOT NULL,
       quantity int NOT NULL,
       PRIMARY KEY (item_number)
       )
```

26

Adding Records

- Use the `INSERT` statement to add individual records to a table

```
INSERT INTO table_name VALUES(value1, value2, ...);
```

- The values entered in the `VALUES` list must be in the same order in which you defined the table fields
- Specify `NULL` in any fields for which you do not have a value
- Add multiple records, use the `LOAD DATA` statement

```
LOAD DATA LOCAL INFILE ‘file_path_name’
INTO TABLE table_name;
```

27

Updating Records

- To update records in a table, use the `UPDATE` statement
- The syntax for the `UPDATE` statement is:

```
UPDATE table_name
SET column_name=value
WHERE condition;
```

  - The `UPDATE` keyword specifies the name of the table to update
  - The `SET` keyword specifies the value to assign to the fields in the records that match the condition in the `WHERE` keyword

28

Deleting Records

- Use the `DELETE` statement to delete records in a table
- The syntax for the `DELETE` statement is:

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

- The `DELETE` statement deletes all records that match the condition
- To delete all the records in a table, leave off the `WHERE` keyword

29

Retrieving Records

- Use the `SELECT` statement to retrieve records from a table:

```
mysql> SELECT model, quantity FROM inventory;
```

- Use the asterisk (*) wildcard with the `SELECT` statement to retrieve all fields from a table
- You can also specify which records to return by using the `WHERE` keyword

```
mysql> SELECT * FROM inventory WHERE make='Martin';
```

- Use the keywords `AND` and `OR` to specify more detailed conditions about the records you want to return

```
mysql> SELECT * FROM inventory WHERE make='Washburn'
    -> AND price<400;
```

30

Retrieving Records – Sorting

- Use the `ORDER BY` keyword with the `SELECT` statement to perform an alphanumeric sort of the results returned from a query

```
mysql> SELECT make, model FROM inventory
    -> ORDER BY make, model;
```

- To perform a reverse sort, add the `DESC` keyword after the name of the field by which you want to perform the sort

```
mysql> SELECT make, model FROM inventory
    -> ORDER BY make DESC, model;
```

31

Accessing Databases from PHP

- PHP has the ability to access and manipulate any database that is ODBC compliant
- PHP includes functionality that allows you to work *directly* with different types of databases, without going through ODBC
- PHP also supports SQLite, database abstraction layer functions, and PEAR DB
  - PEAR (PHP Extension and Application Repository) is a library of open source PHP code.
  - PEAR DB, one of the most popular PEAR code modules, performs similar functions as ODBC, but designed specifically to work with PHP.

http://www.php.net/manual/en/refs.database.php

32

Accessing Databases from PHP

- There are three main options when considering connecting to a MySQL database server using PHP:
  - ☐ PHP's MySQL Extension
  - ☐ PHP's mysqli Extension — We will use mysqli
  - ☐ PHP Data Objects (PDO)
- The mysqli extension features a dual interface, supporting both procedural (functions) and object-oriented interfaces.
- These notes and examples use the *procedural interface*.

http://www.php.net/manual/en/book.mysqli.php

33

Connecting to MySQL

- Open a connection to a MySQL database server with the `mysqli_connect()` function
- The `mysqli_connect()` function returns a *positive integer* if it connects to the database successfully or `false` if it does not
- Assign the return value from the `mysqli_connect()` function to a variable that you can use to access the database in your script

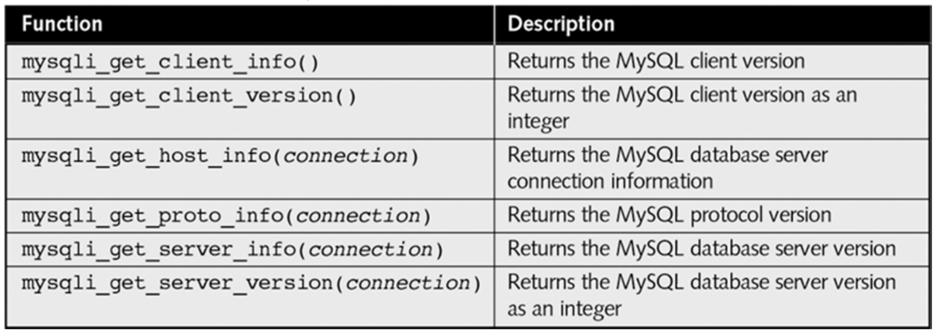
```
$connection = mysqli_connect("host"[, "user ",
"password", "database"])
```

  - ☐ The *host* argument specifies the host name where your MySQL database server is installed e.g. `mysql.ict.swin.edu.au`
  - ☐ The *user* and *password* arguments specify a MySQL account name and password e.g. `s1234567 yourMySQLpassword`
  - ☐ The *database* argument specifies a database e.g. `s1234567_db`

34

MySQL Server Information

MySQL server information functions



Function	Description
`mysqli_get_client_info()`	Returns the MySQL client version
`mysqli_get_client_version()`	Returns the MySQL client version as an integer
`mysqli_get_host_info(`*`connection`*`)`	Returns the MySQL database server connection information
`mysqli_get_proto_info(`*`connection`*`)`	Returns the MySQL protocol version
`mysqli_get_server_info(`*`connection`*`)`	Returns the MySQL database server version
`mysqli_get_server_version(`*`connection`*`)`	Returns the MySQL database server version as an integer

35

Selecting a Database

- The function for selecting a database is

  `mysqli_select_db(`*`connection`*`, `*`database`*`)`

- The function returns a value of `true` if it successfully selects a database or `false` if it does not
- Note, the statement for selecting a database with the MySQL Monitor, is the `use` *`database`* statement
- The mysqli_connect also allows one to connect and select the database at once.

```
$connection = mysqli_connect( "mysql.ict.swin.edu.au",
"s123456", "ddmmyy", "s1234567_db")
```

36

Handling MySQL Errors

- Reasons for not connecting to a database server include:
  - The database server is not running
  - Insufficient privileges to access the data source
  - Invalid username and/or password
- Writing code that anticipates and handles potential problems is often called **bulletproofing.** Bulletproofing techniques include:
  - Validating submitted form data, e.g.

    ```
    if (isset($_GET['height']) ...
    ```
  - Using the **error control operator** (@) to suppress error messages, e.g.

    ```
    $dbConnect = @mysqli_connect(...);
    if (!$dbConnect) ...
    ```

37

Terminating Script Execution

- The `die()` and `exit()` functions ***terminate*** script execution
- The `die()` version is usually used when attempting to access a data source
- Both functions accept a single string argument
- Call the `die()` and `exit()` functions as separate statements or by appending either function to an expression with the `or` operator

Note: When script is terminated, an *incomplete* html page is sent to the client. This is useful for error diagnostics, but poor in a production application.

38

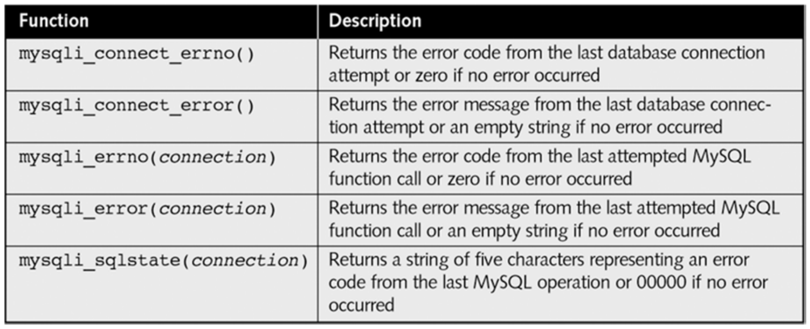
Handling MySQL Errors (continued)

```
$dbConnect = @mysqli_connect("localhost", "root", "paris");
if (!$dbConnect)
      die("<p>The database server is not available.</p>");
echo "<p>Successfully connected to the database server.</p>";
$dbSelect = @mysqli_select_db($dbConnect, "flightlog");
if (!$dbSelect)
      die("<p>The database is not available.</p>");
echo "<p>Successfully opened the database.</p>";
// additional statements that access the database
mysqli_close($dbConnect);
```

39

Handling MySQL Errors(continued)

MySQL error reporting functions



Function	Description
`mysqli_connect_errno()`	Returns the error code from the last database connection attempt or zero if no error occurred
`mysqli_connect_error()`	Returns the error message from the last database connection attempt or an empty string if no error occurred
`mysqli_errno(`*`connection`*`)`	Returns the error code from the last attempted MySQL function call or zero if no error occurred
`mysqli_error(`*`connection`*`)`	Returns the error message from the last attempted MySQL function call or an empty string if no error occurred
`mysqli_sqlstate(`*`connection`*`)`	Returns a string of five characters representing an error code from the last MySQL operation or 00000 if no error occurred

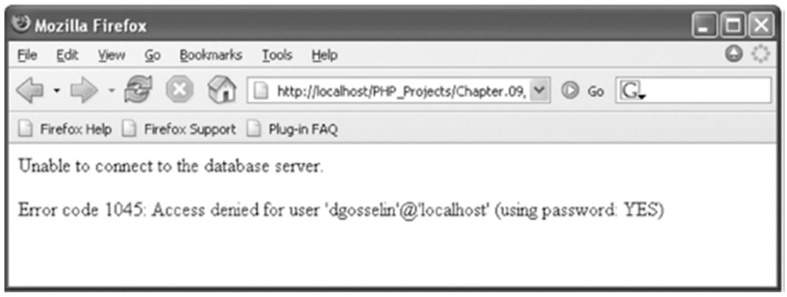
40

Handling MySQL Errors(continued)

```
$user = $_GET['username'];
$password = $_GET['password'];
$dbConnect = @mysqli_connect("localhost", $user, $password)
   or die("<p>Unable to connect to the database server.</p>"
   . "<p>Error code " . mysqli_connect_errno()
   . ": " . mysqli_connect_error() . "</p>");
echo "<p>Successfully connected to the database server.</p>";
@mysqli_select_db($dbConnect, "flightlog")
   or die("<p>The database is not available.</p>");
echo "<p>Successfully opened the database.</p>";
// additional statements that access the database
mysqli_close($dbConnect);
```

41

Handling MySQL Errors(continued)


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/PHP_Projects/Chapter.09,
Go
Firefox Help Firefox Support Plug-in FAQ

Unable to connect to the database server.

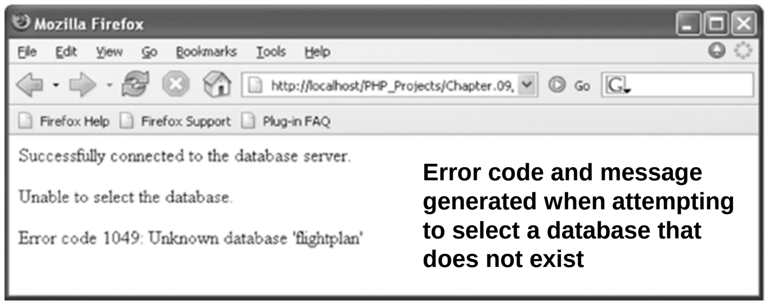
Error code 1045: Access denied for user 'dgosselin'@'localhost' (using password: YES)

Error number and message generated by an invalid username and password

42

Handling MySQL Errors(continued)

```
... ...
@mysqli_select_db($dbConnect, "flightplan")
     or die("<p>Unable to select the database.</p>"
     . "<p>Error code " . mysqli_errno($dbConnect)
     . ": " . mysqli_error($dbConnect) . "</p>");
echo "<p>Successfully opened the database.</p>";
// additional statements that access the database
mysqli_close($dbConnect);
```


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/PHP_Projects/Chapter.09.
Go
Firefox Help Firefox Support Plug-in FAQ

```
Successfully connected to the database server.

Unable to select the database.

Error code 1049: Unknown database 'flightplan'
```

Error code and message generated when attempting to select a database that does not exist

43

Executing SQL Statements

The `mysqli_query()` function returns one of three values:

- For SQL statements that do not return results (`CREATE DATABASE` and `CREATE TABLE` statements) they return a value of `true` if the statement executes successfully
- For SQL statements that return results (`SELECT` and `SHOW` statements) they return a *result pointer* that represents the query results
  - A **result pointer** is a special type of variable that refers to the currently selected row in a resultset
- For SQL statements that fail, `mysqli_query()` function returns a value of `false`, regardless of whether they return results

44

Cleaning Up and Closing Connection

- When you are finished working with query results retrieved with the `mysqli_query()` function, use the `mysqli_free_result()` function to close the result set
- To close the resultset, pass to the `mysqli_free_result()` function the variable containing the result pointer from the `mysqli_query()` function, e.g.

```
mysqli_free_result($QueryResult);
```

- Close a connection to a MySQL database server with the `mysqli_close()` function, e.g.

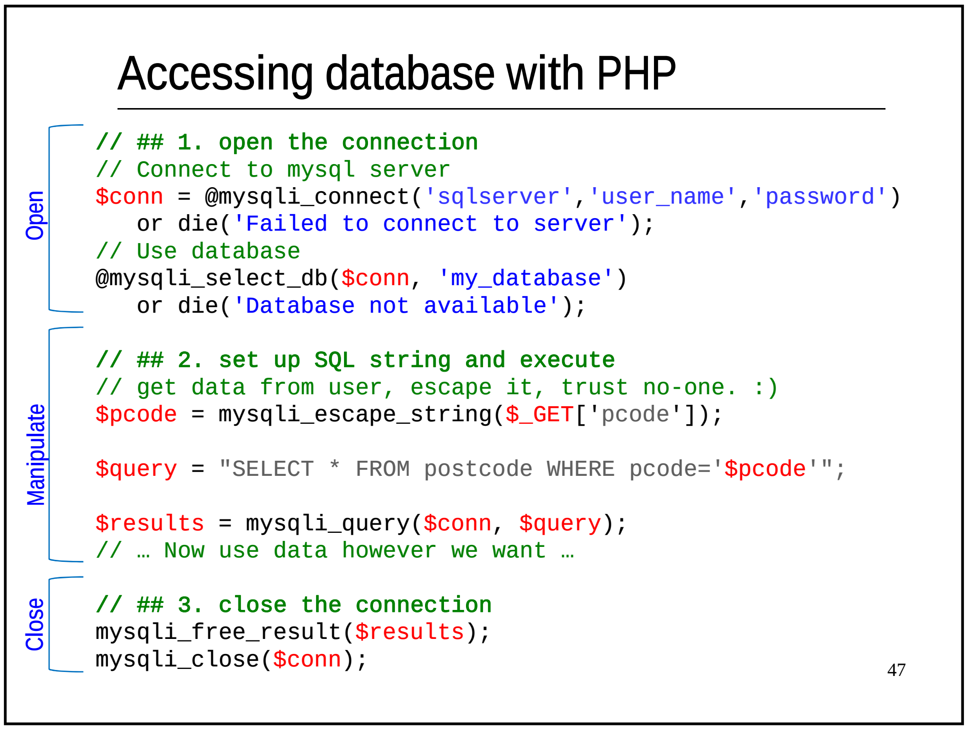
```
mysqli_close($connection);
```

45

Accessing database with PHP

- Step 1 – Open a connection:
  - Connect to the Database Server, and select the Database,
- Step 2 – Manipulate the database:
  - Prepare SQL strings
  - Talk to the Database and executes SQL string
- Step 3 – Close the connection:
  - Clean-up, discard the “query” result objects, or other related objects (if any)
  - Close the connection to the Database and the Database Server

46



Accessing database with PHP

Open

```
// ## 1. open the connection
// Connect to mysql server
$conn = @mysqli_connect('sqlserver','user_name','password')
   or die('Failed to connect to server');
// Use database
@mysqli_select_db($conn, 'my_database')
   or die('Database not available');
```

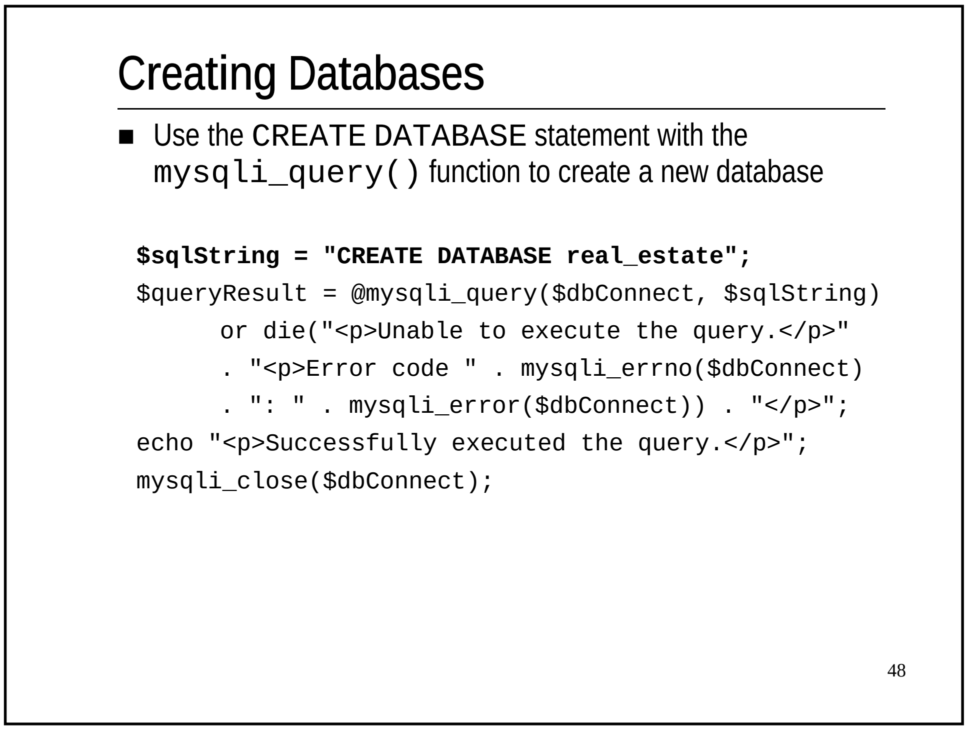
Manipulate

```
// ## 2. set up SQL string and execute
// get data from user, escape it, trust no-one. :)
$pcode = mysqli_escape_string($_GET['pcode']);

$query = "SELECT * FROM postcode WHERE pcode='$pcode'";

$results = mysqli_query($conn, $query);
// … Now use data however we want …
```

Close

```
// ## 3. close the connection
mysqli_free_result($results);
mysqli_close($conn);
```

47



Creating Databases

- Use the CREATE DATABASE statement with the `mysqli_query()` function to create a new database

```
$sqlString = "CREATE DATABASE real_estate";
$queryResult = @mysqli_query($dbConnect, $sqlString)
      or die("<p>Unable to execute the query.</p>"
      . "<p>Error code " . mysqli_errno($dbConnect)
      . ": " . mysqli_error($dbConnect)) . "</p>";
echo "<p>Successfully executed the query.</p>";
mysqli_close($dbConnect);
```

48

Deleting Databases

- Use the DROP DATABASE statement with the `mysqli_query()` function
- Make sure the database does exist by executing the `mysqli_select_db()` function

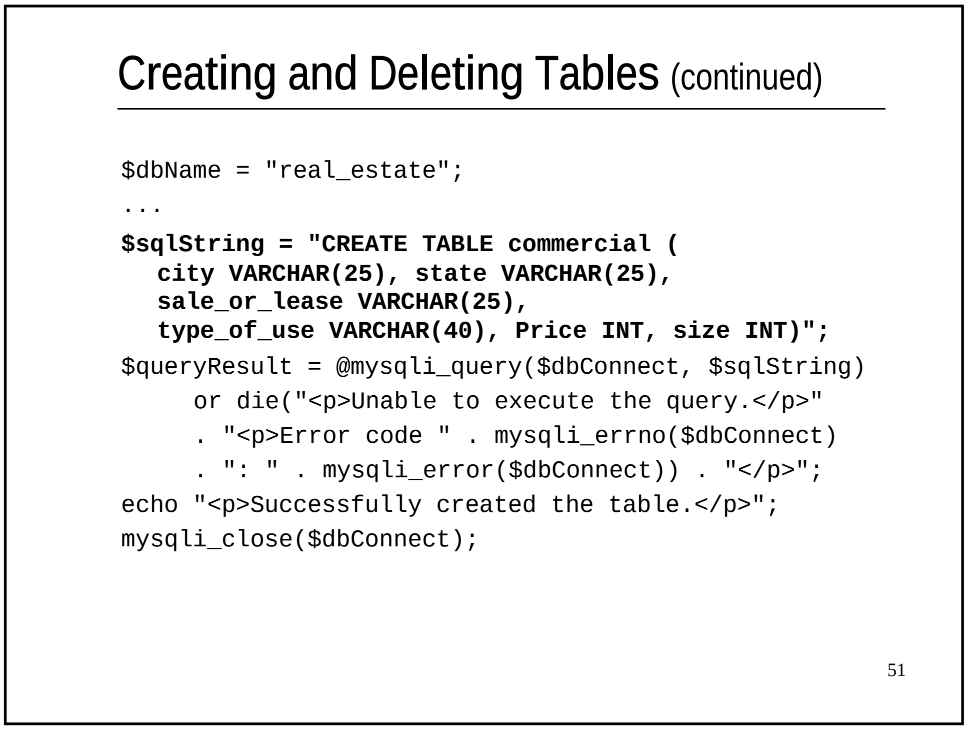
```
$dbName = "real_estate";
...
if (@!mysqli_select_db($dbConnect, $dbName))
     echo "<p>The $dbName database does not exist!</p>";
else {
     $sqlString = "DROP DATABASE $dbName";
     $queryResult = @mysqli_query($dbConnect, $sqlString)
           or die("<p>Unable to execute the query.</p>"
           . "<p>Error code " . mysqli_errno($dbConnect)
           . ": " . mysqli_error($dbonnect)) . "</p>";
     echo "<p>Successfully deleted the database.</p>";
}
mysqli_close($dbConnect);
```

49

Creating and Deleting Tables

- To create a table, use the `CREATE TABLE` statement with the `mysqli_query()` function
- Execute the `mysqli_select_db()` function before executing the `CREATE TABLE` statement or the new table might be created in the wrong database
- To delete a table, use the `DROP TABLE` statement with the `mysqli_query()` function
- To prevent code from attempting to create a table that already exists or to delete a table that does not exist, use a `mysqli_query()` function that either attempts to `SELECT` records from the table, or attempts to '`SHOW TABLES LIKE`'

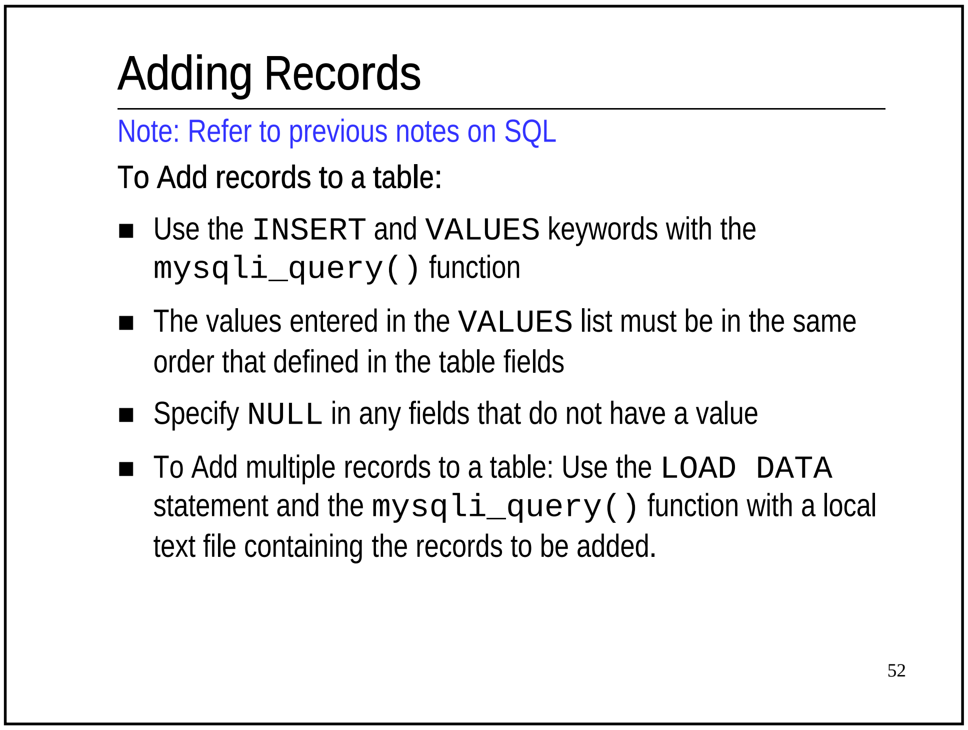
50



Creating and Deleting Tables (continued)

```
$dbName = "real_estate";
...
$sqlString = "CREATE TABLE commercial (
   city VARCHAR(25), state VARCHAR(25),
   sale_or_lease VARCHAR(25),
   type_of_use VARCHAR(40), Price INT, size INT)";
$queryResult = @mysqli_query($dbConnect, $sqlString)
      or die("<p>Unable to execute the query.</p>"
      . "<p>Error code " . mysqli_errno($dbConnect)
      . ": " . mysqli_error($dbConnect)) . "</p>";
echo "<p>Successfully created the table.</p>";
mysqli_close($dbConnect);
```

51



Adding Records

Note: Refer to previous notes on SQL

To Add records to a table:

- Use the `INSERT` and `VALUES` keywords with the `mysqli_query()` function
- The values entered in the `VALUES` list must be in the same order that defined in the table fields
- Specify `NULL` in any fields that do not have a value
- To Add multiple records to a table: Use the `LOAD DATA` statement and the `mysqli_query()` function with a local text file containing the records to be added.

52

Updating and Deleting Records

To Update records in a table:

- Use the `UPDATE`, `SET`, and `WHERE` keywords with the `mysqli_query()` function
- The `UPDATE` keyword specifies the name of the table to update
- The `SET` keyword specifies the value to assign to the fields in the records that match the condition in the `WHERE` keyword

To Delete records from a table:

- Use the `DELETE` and `WHERE` keywords with the `mysqli_query()` function
- The `WHERE` keyword determines which records to delete in the table
- Be careful, if no `WHERE` keyword, all records are deleted !!

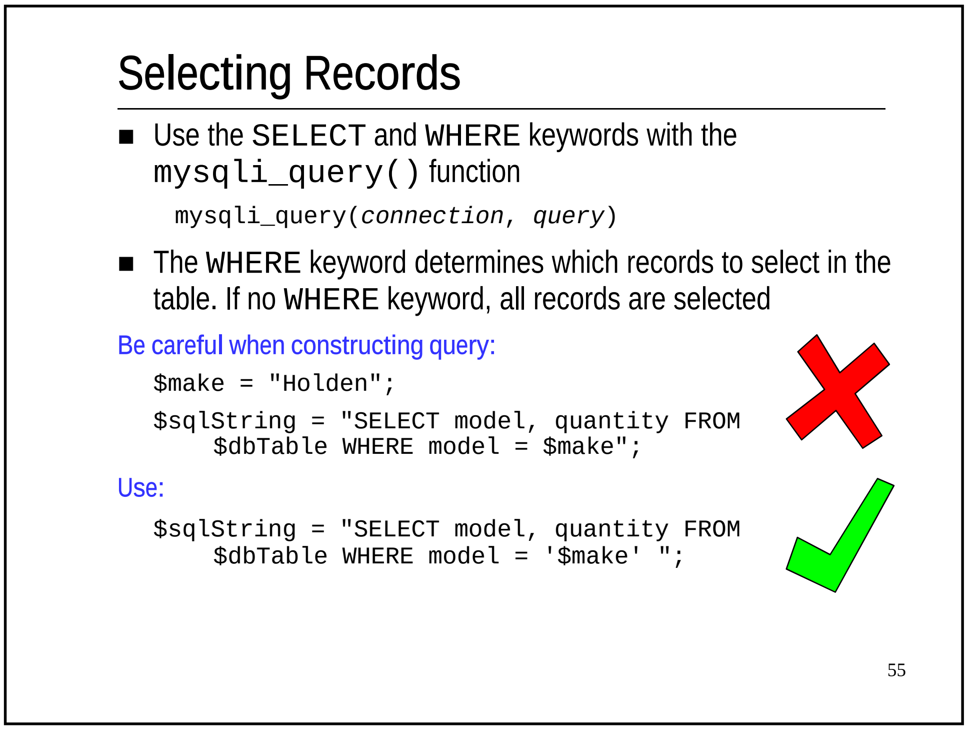
53

mysqli_affected_rows()

With queries that modify tables but do not return results (`INSERT`, `UPDATE`, and `DELETE` queries), use the `mysqli_affected_rows()` function to determine the number of affected rows by the query

```
$sqlString = "UPDATE inventory SET price=368.20
      WHERE make='Fender' AND model='DG7'";
$queryResult = @mysqli_query($dbConnect, $sqlString)
      or die("<p>Unable to execute the query.</p>"
      . "<p>Error code " . mysqli_errno($dbConnect)
      . ": " . mysqli_error($dbConnect) . "</p>");
echo "<p>Successfully updated "
      . mysqli_affected_rows($dbConnect) . " record(s).</p>";
// print "Successfully updated 1 record(s)" if 1 record satisfies the condition.
```

54



Selecting Records

- Use the `SELECT` and `WHERE` keywords with the `mysqli_query()` function

  `mysqli_query(`*`connection`*`, `*`query`*`)`

- The `WHERE` keyword determines which records to select in the table. If no `WHERE` keyword, all records are selected

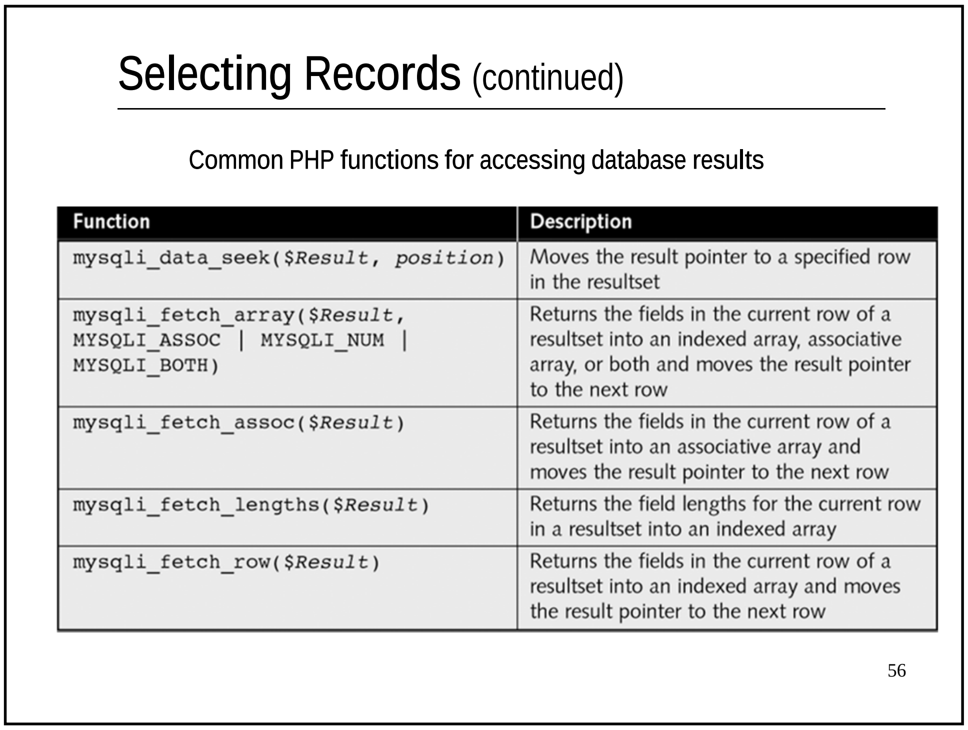
Be careful when constructing query:

```
$make = "Holden";
$sqlString = "SELECT model, quantity FROM
    $dbTable WHERE model = $make";
```

Use:

```
$sqlString = "SELECT model, quantity FROM
    $dbTable WHERE model = '$make' ";
```

55



Selecting Records (continued)

Common PHP functions for accessing database results

Function	Description
`mysqli_data_seek(`*`$Result`*`, `*`position`*`)`	Moves the result pointer to a specified row in the resultset
`mysqli_fetch_array(`*`$Result`*`, MYSQLI_ASSOC | MYSQLI_NUM | MYSQLI_BOTH)`	Returns the fields in the current row of a resultset into an indexed array, associative array, or both and moves the result pointer to the next row
`mysqli_fetch_assoc(`*`$Result`*`)`	Returns the fields in the current row of a resultset into an associative array and moves the result pointer to the next row
`mysqli_fetch_lengths(`*`$Result`*`)`	Returns the field lengths for the current row in a resultset into an indexed array
`mysqli_fetch_row(`*`$Result`*`)`	Returns the fields in the current row of a resultset into an indexed array and moves the result pointer to the next row

56

Selecting Records (continued)

Retrieving Records into an Indexed Array

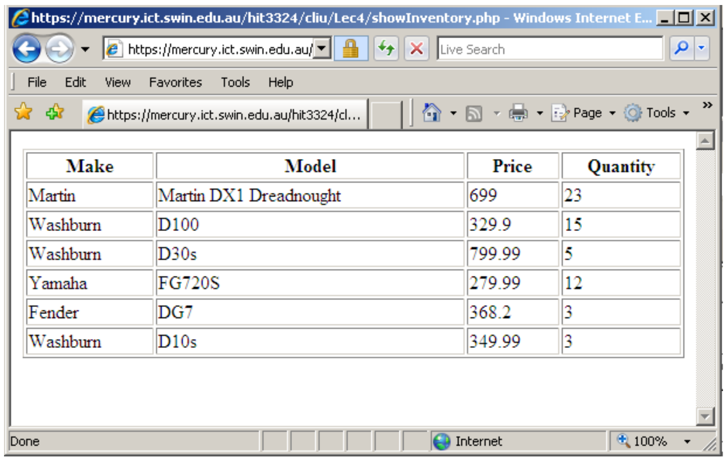
■ The `mysqli_fetch_row()` function returns the fields in the current row of a result set into an indexed array and moves the result pointer to the next row

```
echo "<table width='100%' border='1'>";
echo "<tr><th>Make</th><th>Model</th>
        <th>Price</th><th>Quantity</th></tr>";
$row = mysqli_fetch_row($queryResult);
while ($row) {
        echo "<tr><td>{$row[1]}</td>";
        echo "<td>{$row[2]}</td>";
        echo "<td>{$row[3]}</td>";
        echo "<td>{$row[4]}</td></tr>";
        $row = mysqli_fetch_row($queryResult);
}
echo "</table>";
```

Note: `item_number` in `$row[0]` is not displayed

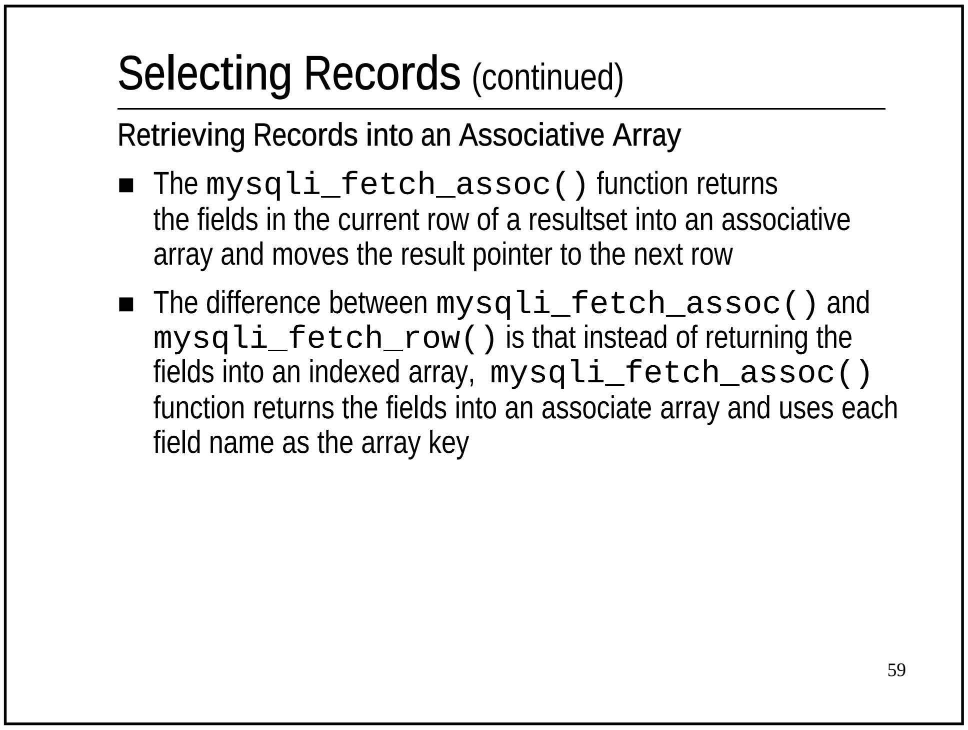
57

Output of the Inventory Table



Make	Model	Price	Quantity
Martin	Martin DX1 Dreadnought	699	23
Washburn	D100	329.9	15
Washburn	D30s	799.99	5
Yamaha	FG720S	279.99	12
Fender	DG7	368.2	3
Washburn	D10s	349.99	3

58

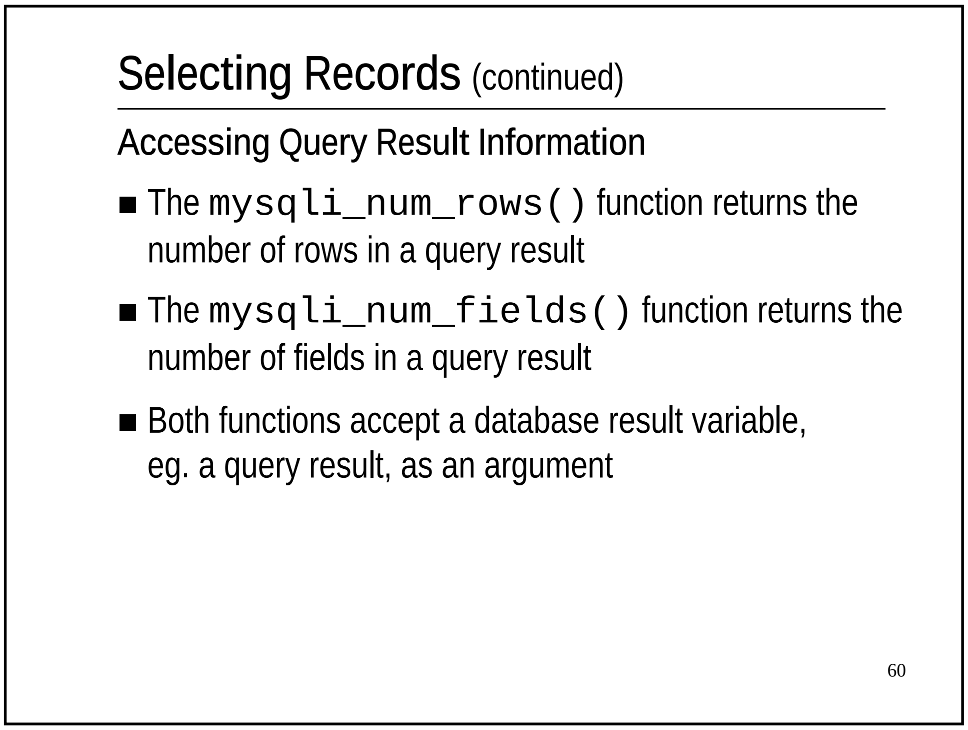


Selecting Records (continued)

Retrieving Records into an Associative Array

- The `mysqli_fetch_assoc()` function returns the fields in the current row of a resultset into an associative array and moves the result pointer to the next row
- The difference between `mysqli_fetch_assoc()` and `mysqli_fetch_row()` is that instead of returning the fields into an indexed array, `mysqli_fetch_assoc()` function returns the fields into an associate array and uses each field name as the array key

59



Selecting Records (continued)

Accessing Query Result Information

- The `mysqli_num_rows()` function returns the number of rows in a query result
- The `mysqli_num_fields()` function returns the number of fields in a query result
- Both functions accept a database result variable, eg. a query result, as an argument

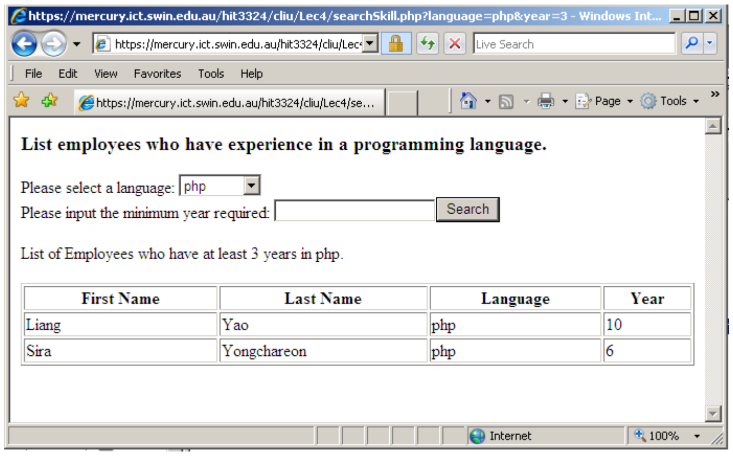
60

Selecting Records (continued)

```
$sqlString = "SELECT * FROM inventory";
$queryResult = @mysqli_query($dbConnect, $sqlString)
     or die("<p>Unable to execute the query.</p>"
     . "<p>Error code " . mysqli_errno($dbConnect)
     . ": " . mysqli_error($dbConnect) . "</p>");
echo "<p>Successfully executed the query.</p>";
$numRows = mysqli_num_rows($queryResult);
$numFields = mysqli_num_fields($queryResult);
if ($numRows != 0 && $numFields != 0) {
     echo "<p>Your query returned " , $numRows ,
          " rows and “, $numFields , " fields.</p>";
} else {
     echo "<p>Your query returned no results.</p>";
}
mysqli_close($dbConnect);
// print “Your query returned 6 rows and 5 fields”
```

61

Example – Searching Skillful People


https://mercury.ict.swin.edu.au/hit3324/cliu/Lec4/searchSkill.php?language=php&year=3 - Windows Int...

List employees who have experience in a programming language.

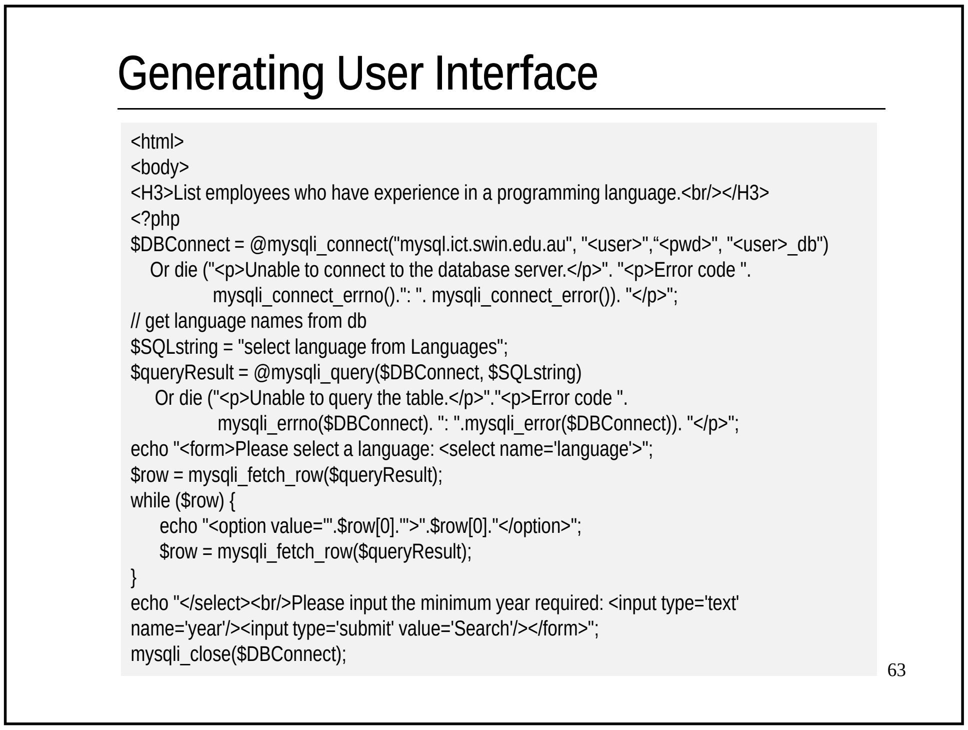
Please select a language: php

Please input the minimum year required: [ ] Search

List of Employees who have at least 3 years in php.

First Name	Last Name	Language	Year
Liang	Yao	php	10
Sira	Yongchareon	php	6

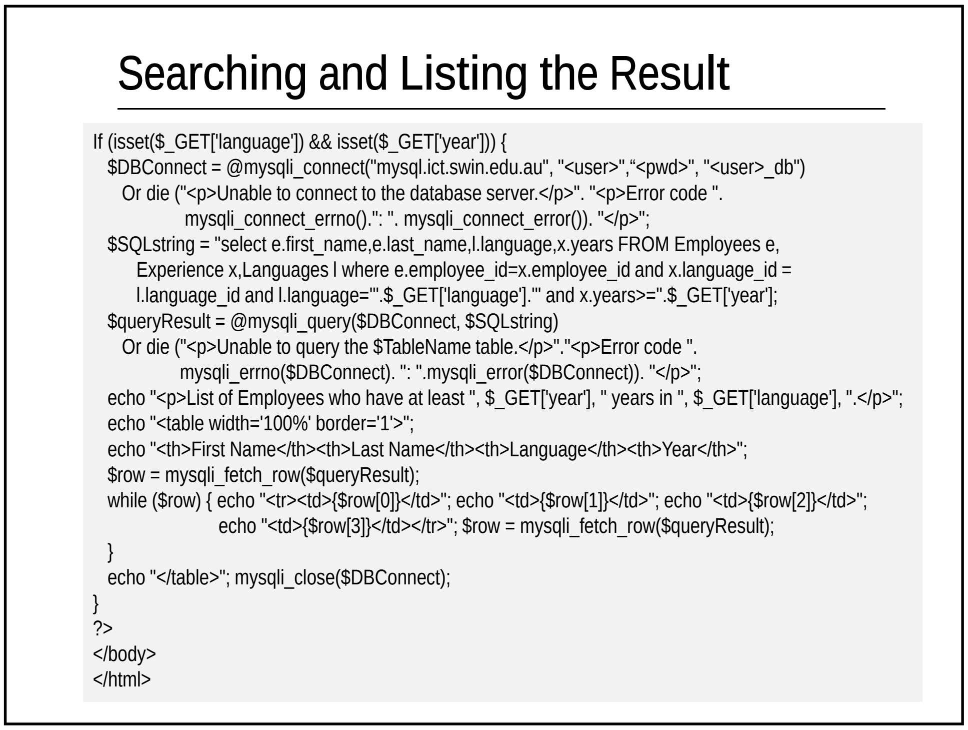
62



Generating User Interface

```
<html>
<body>
<H3>List employees who have experience in a programming language.<br/></H3>
<?php
$DBConnect = @mysqli_connect("mysql.ict.swin.edu.au", "<user>",“<pwd>", "<user>_db")
    Or die ("<p>Unable to connect to the database server.</p>". "<p>Error code ".
                mysqli_connect_errno().": ". mysqli_connect_error()). "</p>";
// get language names from db
$SQLstring = "select language from Languages";
$queryResult = @mysqli_query($DBConnect, $SQLstring)
     Or die ("<p>Unable to query the table.</p>"."<p>Error code ".
                 mysqli_errno($DBConnect). ": ".mysqli_error($DBConnect)). "</p>";
echo "<form>Please select a language: <select name='language'>";
$row = mysqli_fetch_row($queryResult);
while ($row) {
      echo "<option value='".$row[0]."'>".$row[0]."</option>";
      $row = mysqli_fetch_row($queryResult);
}
echo "</select><br/>Please input the minimum year required: <input type='text'
name='year'/><input type='submit' value='Search'/></form>";
mysqli_close($DBConnect);
```

63



Searching and Listing the Result

```
If (isset($_GET['language']) && isset($_GET['year'])) {
   $DBConnect = @mysqli_connect("mysql.ict.swin.edu.au", "<user>",“<pwd>", "<user>_db")
      Or die ("<p>Unable to connect to the database server.</p>". "<p>Error code ".
                mysqli_connect_errno().": ". mysqli_connect_error()). "</p>";
   $SQLstring = "select e.first_name,e.last_name,l.language,x.years FROM Employees e,
        Experience x,Languages l where e.employee_id=x.employee_id and x.language_id =
        l.language_id and l.language='".$_GET['language']."' and x.years>=".$_GET['year'];
   $queryResult = @mysqli_query($DBConnect, $SQLstring)
      Or die ("<p>Unable to query the $TableName table.</p>"."<p>Error code ".
              mysqli_errno($DBConnect). ": ".mysqli_error($DBConnect)). "</p>";
   echo "<p>List of Employees who have at least ", $_GET['year'], " years in ", $_GET['language'], ".</p>";
   echo "<table width='100%' border='1'>";
   echo "<th>First Name</th><th>Last Name</th><th>Language</th><th>Year</th>";
   $row = mysqli_fetch_row($queryResult);
   while ($row) { echo "<tr><td>{$row[0]}</td>"; echo "<td>{$row[1]}</td>"; echo "<td>{$row[2]}</td>";
                    echo "<td>{$row[3]}</td></tr>"; $row = mysqli_fetch_row($queryResult);
   }
   echo "</table>"; mysqli_close($DBConnect);
}
?>
</body>
</html>
```