**SQL**

**#1**

**Part 1**

**Intro**

**What is a Database?**

* A collection of data
* Made up of “tables”
* Tables store similar information

**Tables**

* Tables are made up of “columns” and “rows”
* Columns are like categories
* Each entry goes on a separate row

E.g.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Age** | **Profession** |
| 1 | Ali | 26 | ERI |
| 2 | Lauren | 27 | Student nurse |

*ID = primary key*

Primary Key – A column which is 100% unique. No 2 rows can have the same primary key.

**What is SQL?**

* SQL is the language that databases use

**What is MySQL?**

* A programme that understands SQL
* There is other software, BUT MySQL is the standard

**How to create a database**

1. With XAMPP open and the server running type in *localhost/phpmyadmin* in web browser
2. Click the *database* tab
3. Create database by naming it something. The database is created and will now appear on the left-hand side of the page
4. Use the import tab if you wish to import data from an outside source

**SQL language**

SQL language is type in capital letters as standard. Although SQL is case insensitive it is the standard to write the keywords in capital letters for legibility.

Click the *SQL* tab and type: “SHOW DATABASES” > go, to view all available databases

**Commands**

|  |  |
| --- | --- |
| **Keywords** | **Description** |
| SHOW TABLES | Shows tables for currently selected database |
| SHOW COLUMNS FROM tablename | Shows columns from the table selected |
| SELECT columnname FROM tablename | Shows the specified column from the specified table |
| SELECT columnname, columnname FROM tablename | Select multiple columns from the same table using a comma |
| SELECT \* FROM tablename | Select all columns from specified table |
| SELECT DISTINCT columnname FROM tablename | Returns a unique list with no duplicates. E.g. if you want a list of all cities, 5 entries from the same city would display 5 times if DISTINCT was not inserted |
| LIMIT #,#  e.g. SELECT id FROM customers LIMIT 5,10 | Limits search results – the first # is the starting point and the second # displays how many rows you want to retrieve |
| ORDER BY columnname | Orders alphabetically or numerically depending on data retrieving. You can order by a column name even if the column name was not specified when SELECTED |
| ORDER BY columnname1, columnname2 | This would first order the data by columnname1, and then order columnname2 out within columnname1 (see example below) |
| DESC  E.g. SELECT name FROM customers ORDER BY zip DESC | This will descend the order in which data is displayed |
| WHERE | Condition. See example below |
| AND | Similar to && |
| OR | Similar to || |
| IN | Use this instead of using loads of OR statements. See example below |
| LIKE ‘word’  LIKE ‘%word%’  LIKE ‘\_word’ | Returns data with the word specified. You can use different wildcards to retrieve different data: %, \_  See example below |
| REGEXP | Regular expressions allow to search for more complex patterns than what wildcards can achieve.  See example below |
| AS | Name a custom column |
| GROUP BY | Group data by column name. This is used instead of completing a load of WHERE clauses |
| UNION | Used to combine the result-set of 2 or more SELECT statements |
| UNION ALL | Same as UNION but does not remove duplicate records |
| GROUP BY | The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns |
| HAVING | Filters records that work on summarised GROUP BY results. HAVING applies to summarised group records, whereas WHERE applies to individual records |
| INSERT INTO | Used to insert new records in a table |
| UPDATE | Used when wanting to edit a record. Ensure to use a WHERE clause – without one you will edit the full table. See example below (p2) |
| DELETE | Delete a record. See example below (p2) |
| CREATE TABLE | Creates a table. See example below (p2) |
| NOT NULL | When creating a table, NOT NULL refers to a row needing the data where NOT NULL is specified |
| AUTO\_INCREMENT | Used for auto-generating values for a particular column whenever a new row is inserted |
| ALTER TABLE | Edit the table – you can ADD a new column, DROP (remove) a column… |
| DROP TABLE tablename | Delete full table including data |
| RENAME TABLE tablename TO newname | Rename table |
| CREATE VIEW | Create a view – temporary dynamic table. See example below (p2) |

You can run numerous commands at the same time. You must ensure to include a semi-colon at the end of each query if you do this.

Fully qualified names

Fully qualified names = tablename.columnname

This is useful if you have multiple tables with the same column name.

E.g SELECT buyers.address FROM buyers

ORDER BY columnname1, columnname2

Example:

SELECT state, name FROM customers ORDER BY state, name

|  |  |
| --- | --- |
| State | Name |
| Lincolnshire | Aaa |
| Lincolnshire | Bbb |
| Yorkshire | Ali |
| Yorkshire | Lauren |
| Yorkshire | Shahin |

How to retrieve the last id number example

SELECT name, id FROM customers ORDER BY id DESC LIMIT 1

WHERE example

SELECT id, name FROM customers WHERE id=54

You can also use different operators: < <= > >= != = BETWEEN

e.g. WHERE id BETWEEN 25 AND 30

If you are filtering text instead of numbers, surround with single quotations:

SELECT name, id FROM customers WHERE state=’CA’

AND OR

Use parenthesis when using multiple AND OR keywords:

SELECT id, name, city FROM customers WHERE(id=1 OR id=2) AND city = ‘Hull’

IN & NOT IN

SELECT name, state FROM customers WHERE state IN(‘CA’,’NC’,’NY’)

You can then use ORDER BY to clean the results:

ORDER BY state

NOT IN – does the opposite to IN

LIKE

SELECT columnname FROM tablename WHERE columnname LIKE ‘word%’

% = any characters can come after or before the word specified, depending on where the wildcard was placed. E.g. %word% (before and after), word% (just after), ‘a%s’(starts with ‘a’ and ends with ‘s’)

\_ = single character after or before depending on where the wildcard was placed. E.g. LIKE ‘\_ boxes of frogs’

Regular expressions

SELECT name FROM items WHERE name REGEXP ‘expression’

| = or

REGEXP ‘gold|car’ => returns any word with gold or car in the name

[] = set. E.g:

REGEXP ‘[12345] boxes of frogs’ => reads like 1 or 2 or 3 or 4 or 5. You can also use REGEXP ‘[1-5] boxes of frogs’

REGEXP ‘[^12345] boxes of frogs’ => Not including 1,2,3,4 or 5

Creating custom columns/functions

SELECT CONCAT(city,’, ‘,state) FROM customers

CONCAT() = concatenate string.

In order to name a new column, you need to specify using the keyword AS:

SELECT CONTCAT(city,‘, ’,state) AS postal\_address FROM customers

You can also use maths: +, -, /, \* to create custom columns:

SELECT name, cost, cost-1 AS sale\_price FROM items

|  |  |
| --- | --- |
| **Functions** | **Description** |
| CONCAT() | Concatenate strings |
| UPPER() | Upper case all letters |
| SQRT() | Sq |
| *\*view MySQL website for all functions\** | |

Aggregate functions

Aggregate functions return one value:

|  |  |
| --- | --- |
| **Aggregate functions** | **Descriptions** |
| AVG() | Returns average value of data passed in |
| SUM() | Adds total |
| COUNT() | Counts total of data added in |
| MAX() | Max value of data passed in |
| MIN() | Min value of data passed in |
| *\*view MySQL website for all aggregate functions\** | |

You can run several aggregate functions at once, e.g.:

SELECT COUNT(\*) AS item\_count,

MAX(cost) AS max,

AVG(cost) AS avg

FROM items WHERE seller\_id = 12

This function creates 3 custom columns, 1 showing the total number of items that seller with id 12 is selling, the maximum cost of an item id=12 is selling and the average cost of all items together.

GROUP BY

SELECT seller\_id, COUNT(\*) AS item\_count FROM items GROUP BY seller\_id

This can be filtered using the HAVING keyword. Similar to the WHERE keyword but used instead when grouping:

GROUP BY seller\_id HAVING COUNT(\*) >=3

You can ORDER BY for custom columns by the typing the name you gave it

…ORDER BY item\_count DESC

Subqueries

SELECT name, cost FROM items WHERE cost >(

SELECT AVG(cost) FROM items)

ORDER BY cost DESC

This is not a great example, but this shows how you can use subqueries

How to join tables

SELECT customers.id, customers.name, items.name,items.cost

FROM customers, items

WHERE customers.id = seller\_id //Tell SQL where table is related

ORDER BY customers.id

Outer joins

SELECT customers.name, items.name FROM customers LEFT OUTER JOIN items ON customers.id = seller\_id

This example shows all customers with items even if they aren’t selling any items.

LEFT – lists all of the column names selected from the table listed left of the LEFT syntax no matter what!

RIGHT – does the opposite to LEFT.

UNION ALL

Union is used to combine the result-set of 2 or more select statements. Each SELECT statement with UNION must have the same number of columns; the columns must also have similar data types; the columns in each SELECT statement must also be in the same order.

SELECT columnname(s) FROM table1

UNION

SELECT columnname(s) FROM table2

E.g. SELECT name, cost, bids FROM items WHERE bids > 190

UNION

SELECT name, cost, bids FROM items WHERE cost > 1000

UNION ALL

Used to combine the results-set of 2 or more SELECT statements. It does not remove duplicate records between the various SELECT statements like UNION would – this is the only difference between the two.

Full text searching

To enable full text searching:

ALTER TABLE tablename ADD FULLTEXT(columnname)

Then…

SELECT name, cost FROM items WHERE MATCH(columnname) AGAINST(‘word to search for’)

FULLTEXT is considered better than LIKE

**Part 2**

INSERT INTO

INSERT INTO tablename(columnnames you want to fill with data) VALUES(row data to fill column data with)

INSERT INTO items(id, name, cost, seller\_id,bids) VALUES(‘1’,’beef chops’,’7.99’,’1’,’0’)

To insert multiple rows at the same time – separate value parenthesis with a comma:

…VALUES(row data),(2nd row data),(3rd row data)

UPDATE & DELETE

To edit a row:

UPDATE items SET name = ‘new value’ WHERE id=106

You can edit more than one column using a comma to separate data.

To delete rows:

DELETE FROM items WHERE id=106

Without the WHERE clause you would edit/delete the data from the full table

CREATE TABLE

You can create tables via phpmyadmin – however, this is counterintuitive to learning MySQL.

Via MySQL:

CREATE TABLE tablename(columnnames + datatypes)

E.g:

CREATE TABLE users(

Id int NOT NULL AUTO\_INCREMENT, // int is data type = integar

Username varchar(30) NOT NULL, //varchar() is data type = string.

Password varchar(20) NOT NULL, //number in brackets is memory allowed

PRIMARY KEY(id) //specify what column is the primary key

)

If data is 100% needed in the row then use the keyword NOT NULL, which means ‘not empty’. For example – all users would need a username and password to enter their account, therefore data needs to be added into these rows

AUTO\_INCREMENT is used for auto generating values for a particular column whenever a new row is being inserted. For example, if id is primary key, you don’t want the user to enter this data, or the programmer to enter the data manually in case it is incorrect – therefore AUTO\_INCREMENT starts at 1 and +1 every new row that is added.

\*You can use insert ab on phpmyadmin to quickly insert rows into table to practice\*

ALTER/DROP/RENAME TABLE

Add new column: ALTER TABLE tablename ADD newcolumnname varchar(10)

Delete column: ALTER TABLE tablename DROP COLUMN columnname

Rename table: RENAME TABLE tablename TO newname

Delete full table and all data: DROP TABLE tablename

Views

View is like a temporary dynamic table which updates automatically with the data from the original table if updated/changed. It is good to use if you always use the same SELECT statement to get data. For example, when creating a users mailing address (which could change if user moves location).

CREATE VIEW mailing AS

SELECT CONCAT(city,’, ‘,state) AS address FROM users

Or if you want to view the top 10 highest bidders for example:

CREATE VIEW mostbids AS

SELECT id, name, bids FROM items ORDER BY bids DESC LIMIT 10

Backup database

You can back up database via phpmyadmin by clicking the export tab and downloading to PC

**TEST**

**Intro**

1. What is the difference between SQL and MySQL
2. What is a primary key?
3. How do you create a database?
4. How do you – show tables and columns
5. Define fully qualified names

**P1**

1. Select multiple columns
2. How do you select all column names during a SELECT statement
3. What does LIMIT do? Give example
4. Use ORDER BY on 2 columns
5. Show a list of names in descending order
6. Give examples of: WHERE, AND, OR, IN & NOT IN
7. How do you name a custom column?
8. Explain and give example of LIKE, buzzwords associated with LIKE & REGEXP
9. List 3 functions and give examples
10. What is an aggregate function and list 5 with examples
11. Give an example of the GROUP BY query including HAVING keyword
12. Define UNION and UNION ALL? What is the difference?
13. Define sub queries and give example
14. How do you join tables?
15. Give example of outer joins
16. What is full text searching and give example

**P2**

1. How do you insert multiple rows at once
2. How do you: edit and delete a row
3. Create a table consisting of id, username and password – ensure to set primary key and correct data types for each column with keywords AUTO\_INCREMENT and NOT NULL
4. How do you: add a new column, delete a column, rename a table and delete a full table
5. What is a view?
6. Create a view for a mailing list
7. How do you back up your database using phpmyadmin