NOTES COMPILED FROM EXERCISES IN CODIO

MODULE 1: LAUNCHING INTO COMPUTER SCIENCE

SQL

OPERATOR	DESCRIPTION
=	Equal
<> or !=	Not equal.
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
AND	Displays a record if both the first condition AND the second condition are true
OR	Displays a record if either the first condition OR the second condition is true

TASK	KEYWORDS TO TYPE:	COMMENTS
To open MySQL monitor	mysql	
NOTE: IN SQL, COMMANDS END WITH ; OR \g		
To list existing databases	**SHOW** DATABASES;	
To change into a datebase	**USE** name_of_database;	
To list existing tables	**SHOW** Tables;	
To list the columns	SHOW COLUMNS FROM table_name;	
To read/select all rows from table	**SELECT *** FROM table_name;	
To select a column	**SELECT** column_name1, column_name2 **FROM** table_name;	
To limit the records to be requested	**SELECT** column_name1,column_name2 **FROM** table_name **LIMIT** number_of_rows-we want; **(example:SELECT name,surname FROM names LIMIT 5;)**	
To sort in ascending order with a limited number of rows	**SELECT column1, column2 FROM table_name ORDER BY which_column_you_want_in_ascending_order ASC LIMIT number_of_rows;(SELECT** email,country **FROM** basic_info **ORDER BY** email **ASC LIMIT** 7;)	CRUD functions/keywords in SQL: CREATE-READ-UPDATE-DELETE
example: To sort in descending order	**SELECT** name, id **FROM** names **ORDER BY** id **DESC**;	
To write comments in an SQL file, use --	*-- This is how SQL comments are written (just two dashes)*	
for efficiency reason, source the file	**source source-file**.sql	
to clear the terminal	ctrl+L	
example: SELECT from the *names* table only the *surname* that is equal to the text string, *Steele*	**SELECT** * **FROM names WHERE** surname = "Steele";	(remember that text strings are one of the many datatypes SQL recognizes and must be written inside single or double quotes):
example: to look for a specific data	**SELECT** * **FROM names WHERE** surname = "Steele";	
USING WHERE	**SELECT * FROM** table_name **WHERE** column_name **OPERATOR** matching_data;	
example: to select data from one specific year	**SELECT * FROM** basic_info **WHERE** birthday**>**'2016' **ORDER BY** birthday;	
if you need help with SQL	help contents;	When you click on help contents, a list will appear. You can then choose the item you need by retyping HELP item_name;

SQL

EXAMPLE:
SHOW DATABASES;
USE people
SELECT email FROM basic_info ORDER BY email ASC;
SELECT email FROM basic_info ORDER BY email DESC;
SELECT country,city FROM basic_info LIMIT 3;
SELECT * FROM names WHERE id>5;
SELECT * FROM names WHERE surname="Britt";
SELECT id, birthday, country FROM basic_info WHERE birthday between 2015 AND 2016 ORDER BY DESC;
SELECT * FROM basic_info WHERE birthday !=2014 AND id BETWEEN 3 AND 7 ORDER BY email ASC LIMIT ;

SQL DATATYPES

SQL DATATYPES	
VARCHAR(size)	The **variable character field** can contain letters, numbers, and special characters. Max. size is specified in parenthesis. Can store up to 255 characters
TEXT	Holds a string with a maximum length of 65,535 characters
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
INT(size)	FROM -2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED. The maximum number of digits may be specified in parenthesis. Adding the UNSIGNED attribute will move that range up so it starts at zero instead of a negative number Eg INT(2) or INT(7)
DATE()	A date. Format: YYYY-MM-DD Note: The supported range is from '1000-01-01' to '9999-12-31'
YEAR()	A year in two-digit or four-digit format. Note: Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069
FLOAT(size,d)	a small number with a floating decimal point The total number of digits is specified in ***size***. The number of digits after the decimal point is specified in the ***d*** parameter.
DATETIME	A date and time combination. Format: YYYY-MM-DD HH:MI:SS
TINYINT(size)	similar to the BOOLEAN data types in other SQL database management systems where the values can be literally true or false and generally used for conditional statements (if, else, else if). A value of 0 will always be considered a false value whereas any non-zero value -128 > -1 and 1 < 127 represent a true value. e.g TINYINT(1)