

SQL DATA MANIPULATION – SELECT STATEMENT EXAMPLES

SELECT STATEMENT: read rows from a table

- SQL SELECT statement can be split over several lines for clarity
- Not case-sensitive, except within a string delimited by apostrophes
- Results returned by a SELECT statement constitute a temporary table;
If no rows returned, then it's considered an empty table

	SQL SELECT STATEMENT	Explanation/Grammar
1	Select LastName from Youth2A -outputs one field from one table	SELECT Column Name {table field name} FROM TableName
2	Select FirstName, LastName from Youth2A -outputs multiple fields from one table	List columns in desired order Separated with a comma
3	Select * from Youth2D -outputs all fields from a table	* selects all of the columns in a table
4	Select City, City from Youth2B -outputs the same field twice from one table	a column can be selected more than once and will produce multiple columns in the result
5	Select ID, Location, Leader, DupUnit*9 -the field multiplied by 9 gives the result in output from Youth2N order by ID -sorts in ascending order by ID number	▪ an arithmetic expression can be used to produce a column; operators + - * / ▪ each element (result) in the list is one column, even if several fields are used in the arithmetic expression ▪ expressions must be of the same datatype: numeric, character, or date ▪ the result table can be sorted
6	Select City ' , ' State from Youth2a -example output would be: Wausau, WI	is the concatenation operator apostrophes delimit string
7	Select City ' , ' State ' ' Zip from Youth2B -example output would be: Wausau, WI 54402-6097	You can concatenate multiple strings
8	Select ProjectID, ClubID from Youth2D where MemberID = '12345' -(case sensitive)	▪ specify criteria with WHERE ▪ query produces a list of project ID's and club ID's for 4H member '12345'

9	Select MemberID, LastName, FirstName from Youth2A Where LastName = 'Jones' and FirstName like '%J%' -looks for J in the FirstName field; could be Jim, or James, or John...	Multiple conditions can be joined by • and • or <i>column_name LIKE pattern</i> % percent sign: wildcard representing zero or more characters _ underscore: signifies a <u>single</u> character wildcard
10	Select CountyID, ClubID, Zip from Youth2A where LastName like 'Jo%' Order by ClubID, CountyID - Order By column(s) has to be in SELECT statement or query will result in an error.	▪ ORDER BY determines the sorted order of the list returned. ▪ Without ORDER BY the list order is arbitrary even if the table is indexed. ▪ <u>an order column does not have to be an index</u>
11	Select CountyID, ClubID from Youth2A Order by ClubID DESC, -descending order CountyID -ascending order	▪ ascending order is default (small to large) ▪ use DESC after the column name to sort in descending order (large to small)
12	Select count (*) from Youth2G -outputs a count of all the activities in the Youth2G (Activity Code File)	▪ counts all records in the table ▪ returns the COUNT as an integer ▪ if a column name is specified instead of the * then it counts only the rows where there is data in the specified column (not null)
13	Select (FirstName ' ' LastName) AS MemberName from Youth2A where State = 'WI' -combines FirstName and LastName into one field, takes all records with state of 'WI'	▪ Can use "AS" to create a new result field which combines 2 or more fields into one ▪ Can rename any single field or a group of fields together into one column name
14	Select FirstName, Middlel, LastName, MemberID, Age, Birthdate from Youth2A where School is NULL -for "is NOT NULL" -must use the word 'is,' an '=' sign doesn't work -selects all members with no school	NULL value ▪ value for the column is not applicable for the row in question ▪ no value is assigned to the column yet ▪ query returns all members who don't have a specified school in the file NOT NULL if you want members <u>with</u> schools, use where School is NOT NULL
15	Select HomePhone, WorkPhone, Fax, Cell, Beeper, FirstName, LastName, SpouseFirstName from Youth2b	LIKE ▪ LastName will include the word "Johnson" for retrieved records ▪ Matches case

	where LastName like '%Johnson%' -all records in LastName field must have Johnson as their value	Essentially the same as saying Where LastName = 'Johnson'
16	Select AwardID, Youth2e."Desc", ES237, FedInit, StateInit from Youth2E where upper (Youth2E."Desc") like '%DONALD%MEMORIAL%' -case sensitive all caps {alternative example using LOWER function where lower (Youth2E."Desc") like '%donald%memorial%' } -case sensitive all lower case -quotes ("") around Desc are necessary because DESC is a reserved SQL keyword	UPPER - the UPPER function is applied to the data Description field in order to <u>disregard case when comparing</u> - Description will include the string "Donald" and the string "Memorial" for selected records - retrieves any case combination of Donald-Memorial or Donald Kerndt Memorial or Donald_Memorial LOWER function available - use UPPER or LOWER functions to make comparison not case sensitive
17	Select AwardID, Youth2e."Desc", ES237 from Youth2E where upper (Youth2e."Desc") like '%DONALD_KERNDT%' - _single character wildcard	LIKE use single underscore as a single character wildcard to rule out unrelated words in the "Desc" field
18	Select LastName, FirstName, State from Youth2A where State IN ('CA', "WI", 'AZ') -case sensitive -true if state is in the list, displays all valid records	IN returns a Boolean value: TRUE if the column value/ expression matches one of the values in the condition list FALSE if the column value/ expression is not equal to one of the values in the condition list
19	Select LastName, FirstName, State from Youth2A where State NOT IN ('CA', 'WI', 'AZ') -case sensitive -displays value that doesn't match these three	NOT IN condition is true if state not in the list
20	Select LastName, FirstName, State from Youth2A where State !='CA' and - (State <> 'CA' and) State !='WI' and - (State <> 'WI' and) State !='AZ' - (State <> 'AZ')	!= not equal operator (can also use the <> not equal operator) condition is true if state not in the list essentially the same statement as #22 except #22 will find records with a null value in state field - This query only returns records with values in the state field
21	Select ClubID, CountyID, ClubName, Leader, Address1 from Youth2X where ClubID between '200' and '250'	BETWEEN - Between operator; inclusive - works with numeric, string and date values

22	Select ClubID, CountyID, ClubName, Leader, Address1 from Youth2X where ClubID between '200' and '250' or ClubID between '700' and '750'	BETWEEN Between operator used with compound condition
23	Select LastName, FirstName, State from Youth2A where LastName between 'A' and 'M'	BETWEEN - used with string data - includes all records with last names from beginning with 'A' to beginning 'M'

SELECT DISTINCT - only outputs a value one time, eliminates the repetition of multiple fields that meet the requirements (displays one distinct name).

end