**RETRIEVING DATA FROM A SINGLE TABLE**

SELECT         columns to be retrieved from the database

FROM             names of table(s) in which required data is located

WHERE          filters to limit the rows retrieved

ORDER BY sort the rows in a specified order

ESSENTIAL - Questions 1-9,    VERY USEFUL – Questions 10-16,     ADVANCED - Question 17

1. SELECT \*

FROM dept;

Expected result:

|  |  |  |
| --- | --- | --- |
| DEPTNO | DNAME | LOCATION |
| 10 | ACCOUNTING | OXFORD |
| 20 | RESEARCH | SOTON |
| 30 | SALES | BRISTOL |
| 40 | OPERATIONS | SOTON |

(4 rows returned)

2.           SELECT ename, monthly\_sal, commission, deptno

FROM emp

ORDER BY ename;

Expected result:

|  |  |  |  |
| --- | --- | --- | --- |
| ENAME | MONTHLY\_SAL | COMMISSION | DEPTNO |
| ADAMS | 1100 | NULL | 20 |
| ALLEN | 1600 | 300 | 30 |
| BLAKE | 2850 | NULL | 30 |
| CLARK | 2450 | NULL | 10 |
| FORD | 3000 | NULL | 20 |
| JAMES | 950 | NULL | 30 |
| JONES | 2975 | NULL | 20 |
| KING | 5000 | NULL | 10 |
| MARTIN | 1250 | 1400 | 30 |
| MILLER | 1300 | NULL | 10 |
| SCOTT | 3000 | NULL | 20 |
| SMITH | 800 | NULL | 20 |
| TURNER | 1500 | 0 | 30 |
| WARD | 1250 | 500 | 30 |

  (14 rows returned)

3.         SELECT ename, monthly\_sal, deptno

FROM emp

WHERE monthly\_sal> 2900

ORDER BY ename;

Expected result:

|  |  |  |
| --- | --- | --- |
| ENAME | MONTHLY\_SAL | DEPTNO |
| FORD | 3000 | 20 |
| JONES | 2975 | 20 |
| KING | 5000 | 10 |
| SCOTT | 3000 | 20 |

  (4 rows returned)

Note filters in the where clause can use any of the following relational operators:

is equal to                    =

is not equal to              !=

is greater than              >

is less than                   <

is greater than or equal to   >=

is less than or equal to        <=

IMPORTANT NOTE.    The relational operators above are useless for identifying blank values – known in databases as NULL values.

4a. SELECT ename, monthly\_sal, commission, deptno

FROM  emp

WHERE commission = NULL

ORDER BY ename;

Expected result:

(no rows will be returned)

To work with NULL values, use the relational operator IS NULL (or IS NOT NULL).

4b.       SELECT ename, job, monthly\_sal, commission, deptno

FROM emp

WHERE commission IS  NULL

ORDER BY   ename;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ENAME | JOB | MONTHLY\_SAL | COMMISSION | DEPTNO |
| ADAMS | CLERK | 1100 | NULL | 20 |
| BLAKE | MANAGER | 2850 | NULL | 30 |
| CLARK | MANAGER | 2450 | NULL | 10 |
| FORD | ANALYST | 3000 | NULL | 20 |
| JAMES | CLERK | 950 | NULL | 30 |
| JONES | MANAGER | 2975 | NULL | 20 |
| KING | PRESIDENT | 5000 | NULL | 10 |
| MILLER | CLERK | 1300 | NULL | 10 |
| SCOTT | ANALYST | 3000 | NULL | 20 |
| SMITH | CLERK | 800 | NULL | 20 |

  (10 rows returned)

4c.       SELECT ename, job, monthly\_sal, commission, deptno

FROM emp

WHERE commission IS NOT NULL

ORDER BY ename;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ENAME | JOB | MONTHLY\_SAL | COMMISSION | DEPTNO |
| ALLEN | SALESMAN | 1600 | 300 | 30 |
| MARTIN | SALESMAN | 1250 | 1400 | 30 |
| TURNER | SALESMAN | 1500 | 0 | 30 |
| WARD | SALESMAN | 1250 | 500 | 30 |

(4 rows returned)

Note that 0 is a different value to NULL

5.         SELECT ename, job, monthly\_sal, deptno, hiredate

FROM emp

WHERE monthly\_sal > 2900

AND deptno = 20

AND job= 'ANALYST'

AND hiredate  > '1981-08-01'

ORDER BY ename;

Expected result:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ENAME | JOB | MONTHLY\_SAL | DEPTNO | HIREDATE |
| FORD | ANALYST | 3000 | 20 | 1981-12-03 |
| SCOTT | ANALYST | 3000 | 20 | 1982-12-09 |

(2 rows returned)

In SQLite, by default, date fields are stored in the format YYYY-MM-DD

Multiple where conditions can be applied to a query using a combination of AND and OR logical operators.

NOTE

(i)                 for string comparisons such as job =  'ANALYST', the test performed is case sensitive.  job = 'analyst' would return no rows.

(ii)                both date and string values must be given in inverted commas

6.         SELECT ename, monthly\_sal, deptno

FROM emp

WHERE deptno  = 10

OR monthly\_sal     > 2900

ORDER BY  ename;

Expected result:

|  |  |  |
| --- | --- | --- |
| ENAME | MONTHLY\_SAL | DEPTNO |
| CLARK | 2450 | 10 |
| FORD | 3000 | 20 |
| JONES | 2975 | 20 |
| KING | 5000 | 10 |
| MILLER | 1300 | 10 |
| SCOTT | 3000 | 20 |

(6 rows returned)

      Here OR is used to combine conditions (filters) in the WHERE clause.

7.         SELECT ename, monthly\_sal, deptno

FROM emp

WHERE deptno IN (10,20)

ORDER BY ename;

Expected result:

|  |  |  |
| --- | --- | --- |
| ENAME | MONTHLY\_SAL | DEPTNO |
| ADAMS | 1100 | 20 |
| CLARK | 2450 | 10 |
| FORD | 3000 | 20 |
| JONES | 2975 | 20 |
| KING | 5000 | 10 |
| MILLER | 1300 | 10 |
| SCOTT | 3000 | 20 |
| SMITH | 800 | 20 |

(8 rows returned)

IN can be used to reduce the number of OR's in a WHERE statement.

This condition is equivalent to WHERE deptno = 10    OR   deptno = 20.

8a.       SELECT ename, monthly\_sal, deptno

FROM emp

WHERE deptno  = 10

OR deptno  = 20

AND monthly\_sal > 2900

ORDER BY ename;

Expected result:

|  |  |  |
| --- | --- | --- |
| ENAME | MONTHLY\_SAL | DEPTNO |
| CLARK | 2450 | 10 |
| FORD | 3000 | 20 |
| JONES | 2975 | 20 |
| KING | 5000 | 10 |
| MILLER | 1300 | 10 |
| SCOTT | 3000 | 20 |

      (6 rows returned)

Here the monthly\_sal condition is only applied to employees in department no 20.

Note when using AND with OR together in the same WHERE statement, beware of the order of precedence of these logical operators.   AND is always applied before OR.   If in doubt use brackets to force the order of execution.

8b.       SELECT ename, monthly\_sal, deptno

FROM emp

WHERE (deptno  = 10

OR deptno  = 20)

AND monthly\_sal > 2900

ORDER BY ename;

Expected result:

|  |  |  |
| --- | --- | --- |
| ENAME | MONTHLY\_SAL | DEPTNO |
| FORD | 3000 | 20 |
| JONES | 2975 | 20 |
| KING | 5000 | 10 |
| SCOTT | 3000 | 20 |

       (4 rows returned)

Here the monthly\_sal condition applies to employees from both departments

Note that each condition must be a complete statement that can be evaluated to true or false.   You can NOT use deptno = 10 OR 20.    It must be deptno = 10 OR deptno = 20.

9. To select rows based on user input at runtime, you need to specify parameters prefixed by @ as follows:

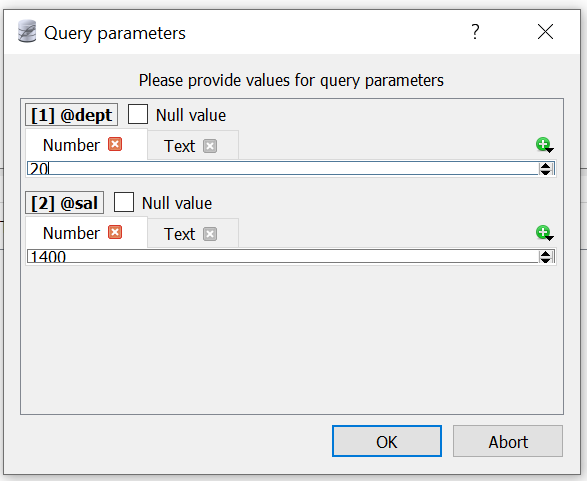
SELECT ename, deptno, monthly\_sal

FROM emp

WHERE deptno=@dept

AND monthly\_sal>=@sal;

Enter the parameter values as below:



|  |  |  |
| --- | --- | --- |
| ENAME | DEPTNO | MONTHLY\_SAL |
| JONES | 20 | 2975 |
| SCOTT | 20 | 3000 |
| FORD | 20 | 3000 |

10.         The standard arithmetic operators (addition +, subtraction -, multiplication \*, division /) can be used to calculate new values based on one or more stored numeric values.

SELECT ename, monthly\_sal\*12, commission, deptno

FROM emp

WHERE deptno = 30

ORDER BY ename;

Expected result:

|  |  |  |  |
| --- | --- | --- | --- |
| ENAME | MONTHLY\_SAL\*12 | COMMISSION | DEPTNO |
| ALLEN | 19200 | 300 | 30 |
| BLAKE | 34200 | NULL | 30 |
| JAMES | 11400 | NULL | 30 |
| MARTIN | 15000 | 1400 | 30 |
| TURNER | 18000 | 0 | 30 |
| WARD | 15000 | 500 | 30 |

  (6 rows returned)

The order of precedence of the arithmetic operators is as for standard arithmetic.

Division

Multiplication

Addition

Subtraction

Therefore   6 \* 2 + 3 = 15              (because 6 \* 2 = 12,   12 + 3 = 15)

Again, as in normal arithmetic, use brackets to over-ride default order of execution.

e.g.             6 \* (2 + 3) = 25             (because 2 + 3 = 5,    6 \* 5 = 30)

11a.       Beware the potential impact of NULL values on arithmetic calculations.

In database logic a NULL value arithmetically combined (using +, -, \*, /) with other values will always result in a NULL.

SELECT ename, monthly\_sal\*12, commission, monthly\_sal\*12 + commission

FROM emp

WHERE deptno = 30

ORDER BY ename;

Expected result:

|  |  |  |  |
| --- | --- | --- | --- |
| ENAME | MONTHLY\_SAL\*12 | COMMISSION | MONTHLY\_SAL\*12+COMMISSION |
| ALLEN | 19200 | 300 | 19500 |
| BLAKE | 34200 | NULL | NULL |
| JAMES | 11400 | NULL | NULL |
| MARTIN | 15000 | 1400 | 16400 |
| TURNER | 18000 | 0 | 18000 |
| WARD | 15000 | 500 | 15500 |

(6 rows returned)

Here an integer + NULL = NULL  (eg for BLAKE, 34200 + NULL = NULL)

To avoid such problems use the IFNULL() function which treats any NULL values found in the database as though they were the value specified in the second parameter.

11b. SELECT ename, monthly\_sal\*12, IFNULL(commission, 'N/A'), monthly\_sal\*12 + IFNULL(commission,0)

FROM emp

WHERE deptno = 30

ORDER BY ename;

 Expected result:

|  |  |  |  |
| --- | --- | --- | --- |
| ENAME | MONTHLY\_SAL\*12 | IFNULL(COMMISSION ,’N/A’) | MONTHLY\_SAL\*12+IFNULL(COMMISSION,0) |
| ALLEN | 19200 | 300 | 19500 |
| BLAKE | 34200 | N/A | 34200 |
| JAMES | 11400 | N/A | 11400 |
| MARTIN | 15000 | 1400 | 16400 |
| TURNER | 18000 | 0 | 18000 |
| WARD | 15000 | 500 | 15500 |

(6 rows returned)

12.       SELECT empno || ' : ' || ename, hiredate

FROM emp

WHERE deptno = 10

ORDER BY empno;

Expected result:

|  |  |
| --- | --- |
| EMPNO||':'||ENAME | HIREDATE |
| 7782 : CLARK | 1981-06-09 |
| 7839 : KING | 1981-11-17 |
| 7934 : MILLER | 1982-01-23 |

(3 rows returned)

Values from more than one database attribute (column) and/or string literals can be combined into a single output column using concatenation. The concatenation symbol is the two 'pipes'   (upper case key to the left of the Z key).

13.       SELECT empno || ' : ' || ename  AS 'Employee', hiredate AS 'Hired On'

FROM emp

WHERE deptno = 10

ORDER BY empno;

Expected result:

|  |  |
| --- | --- |
| Employee | Hired On |
| 7782 : CLARK | 1981-06-09 |
| 7839 : KING | 1981-11-17 |
| 7934 : MILLER | 1982-01-23 |

  (3 rows returned)

Output columns can be renamed by adding a AS “column alias” after the column name in the Select clause.

 14.       SELECT empno,  ename,  job, hiredate

FROM emp

WHERE hiredate BETWEEN '1981-05-01' AND '1981-12-03'

ORDER BY hiredate;

Expected result:

|  |  |  |  |
| --- | --- | --- | --- |
| EMPNO | ENAME | JOB | HIREDATE |
| 7698 | BLAKE | MANAGER | 1981-05-01 |
| 7782 | CLARK | MANAGER | 1981-06-09 |
| 7844 | TURNER | SALESMAN | 1981-09-08 |
| 7654 | MARTIN | SALESMAN | 1981-09-28 |
| 7839 | KING | PRESIDENT | 1981-11-17 |
| 7900 | JAMES | CLERK | 1981-12-03 |
| 7902 | FORD | ANALYST | 1981-12-03 |

  (7 rows returned)

Note the 'BETWEEN … AND …' construct is inclusive of the values quoted.

The following WHERE clause would therefore give the same result:

WHERE hiredate >= '1981-05-01'

            AND      hiredate < = '1981-12-03'

15.       SELECT ename, job, deptno

FROM emp

WHERE ename LIKE '%S%';

Expected result:

|  |  |  |
| --- | --- | --- |
| ENAME | JOB | DEPTNO |
| SMITH | CLERK | 20 |
| JONES | MANAGER | 20 |
| SCOTT | ANALYST | 20 |
| ADAMS | CLERK | 20 |
| JAMES | CLERK | 30 |

  (5 rows returned)

The LIKE operator combined with the wildcard symbol % can be used to search for parts of strings.  Here (with wildcard before and after the S) for any name with an S in.   This kind of search is very inefficient and should be avoided where possible.

16.       SELECT DISTINCT job

FROM emp

ORDER BY job;

Expected result:

|  |
| --- |
| JOB |
| ANALYST |
| CLERK |
| MANAGER |
| PRESIDENT |
| SALESMAN |

(5 rows returned)

For most SQL queries one line of output will be produced for each database row that meets the conditions in the where clause. DISTINCT will ensure that duplicate values are not displayed.

17.       SELECT deptno, job, ename, hiredate, STRFTIME('%d-%m-%Y',hiredate) AS 'HIREDATE in DD-MM-YYYY format'

FROM emp

WHERE deptno IN (10,20)

ORDER BY deptno DESC, job ASC

Expected result:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DEPTNO | JOB | ENAME | HIREDATE | HIREDATE in DD-MM-YYYY format |
| 20 | ANALYST | SCOTT | 1982-12-09 | 09-12-1982 |
| 20 | ANALYST | FORD | 1981-12-03 | 03-12-1981 |
| 20 | CLERK | SMITH | 1980-12-17 | 17-12-1980 |
| 20 | CLERK | ADAMS | 1983-01-12 | 12-01-1983 |
| 20 | MANAGER | JONES | 1981-04-02 | 02-04-1981 |
| 10 | CLERK | MILLER | 1982-01-23 | 23-01-1982 |
| 10 | MANAGER | CLARK | 1981-06-09 | 09-06-1981 |
| 10 | PRESIDENT | KING | 1981-11-17 | 17-11-1981 |

  (8 rows returned)

By default the ORDER BY clause will sort output into ascending order.  However, it is possible to specify either ascending (ASC) or descending (DESC) order for each separate element of the ORDER BY clause.

18. To display the length of service for each employee, we need to work out the difference between todays date and the employee’s hiredate. To do this we use the DATETIME function to get todays date and the JULIANDAY function to convert the date to the number of days since Nov 24, 4714 BC in order to calculate the number of days between the two dates.

**Note: You must have loaded the SQLite functions extensions library into your database (as explained in the SQLite Basic User Guide on SOL) before you can use the FLOOR function.**

SELECT ename, FLOOR((JULIANDAY(DATETIME('now')) - JULIANDAY(hiredate)) / 365) AS 'Length of service in years'

FROM emp

ORDER BY ename;

Expected result:

|  |  |
| --- | --- |
| ENAME | Length of service in years |
| ADAMS | 37 |
| ALLEN | 38 |
| BLAKE | 38 |
| CLARK | 38 |
| FORD | 38 |
| JAMES | 38 |
| JONES | 38 |
| KING | 38 |
| MARTIN | 38 |
| MILLER | 37 |
| SCOTT | 37 |
| SMITH | 39 |
| TURNER | 38 |
| WARD | 38 |

STRFTIME, DATETIME and JULIANDAY are all SQLite built-in functions for manipulating dates. See the SQLite Built Functions reference guide on SOL for further details of these as well as string, numeric and other functions.