Quick Reference for:

**Data Manipulation Language (Basics)**

Contents

[**INSERT, UPDATE, and DELETE** 1](#_Toc82581968)

[Create a “test” table with a SELECT statement 1](#_Toc82581969)

[INSERT data into table 1](#_Toc82581970)

[Using a SUBQUERY to INSERT MANY rows 2](#_Toc82581971)

[UPDATE data in a table 2](#_Toc82581972)

[Using a SUBUERY in an UPDATE statement 3](#_Toc82581973)

[DELETE data from a table 3](#_Toc82581974)

[Commit and Rollback 3](#_Toc82581975)

[**SELECT clause** 4](#_Toc82581976)

[Basic SELECT from a single table 4](#_Toc82581977)

[Rename column with COLUMN ALIAS 4](#_Toc82581978)

[Select with STRING EXPRESSIONS 4](#_Toc82581979)

[Select with ARITHMETIC EXPRESSIONS 5](#_Toc82581980)

[Select with SCALAR FUNCTIONS 5](#_Toc82581981)

[Use of the DUAL table 5](#_Toc82581982)

[**Filter with WHERE** 6](#_Toc82581983)

[Filter rows using WHERE causes 6](#_Toc82581984)

[Comparison operators (=, >, <, >=, <=, <>) 6](#_Toc82581985)

[Logical operators (AND, OR, NOT) 7](#_Toc82581986)

[IN operator 7](#_Toc82581987)

[BETWEEN operator 8](#_Toc82581988)

[LIKE operator 8](#_Toc82581989)

[IN NULL operator 9](#_Toc82581990)

[**Sort with ORDER BY** 10](#_Toc82581991)

[Sort data by COLUMN NAME 10](#_Toc82581992)

[Sort by COLUMN ALIAS 10](#_Toc82581993)

[Sort by EXPRESSION 10](#_Toc82581994)

[Sort by COLUMN POSITION 10](#_Toc82581995)

[**More Resources** 11](#_Toc82581996)

# **INSERT, UPDATE, and DELETE**

## Create a “test” table with a SELECT statement

|  |  |
| --- | --- |
| **Syntax** | --This is a quick way to copy a table for purposes of testing on a non-production table  CREATE TABLE *tablename* AS  *SELECT statement here* |
| **Example** | CREATE TABLE invoices\_copy AS  SELECT \*  FROM invoices; |
| **More info** | NOTE: When you use this method to copy a table, it only copies the table name, columns, and data types. It does not copy over any constraints into the newly created test table |

## INSERT data into table

|  |  |
| --- | --- |
| **Syntax** | **--Option 1: Insert with a column list**  INSERT INTO table\_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...);  **--Option 2: Insert without a column list. The number of values must equal the number of columns and be inserted in same order as columns**  INSERT INTO table\_name VALUES (value1, value2, value3, ...); |
| **Example** | **--Option 1: Insert without using column list**  INSERT INTO invoices  VALUES (115, 97, '456789', '01-AUG-14', 8344.50, 0, 0, 1, '31-AUG-14', NULL);  **--Option 2: Insert without using column list**  INSERT INTO invoices (invoice\_id, vendor\_id, invoice\_number, invoice\_total, payment\_total, credit\_total, terms\_id, invoice\_date, invoice\_due\_date)  VALUES (115, 97, '456789', 8344.50, 0, 0, 1, '01-AUG-14', '31-AUG-14'); |
| **More info** | <https://www.w3schools.com/sql/sql_insert.asp>   * Quotes - In Oracle, all text and dates are wrapped with single quotes. Do not use quotes on numeric values. See example above * Date Format – The default format of dates in Oracle follow a DD-MON-YY format. See above * Inserting NULL – use the keyword NULL if you want to insert NULL values * Inserting DEFAULT – use the keyword NULL if you want to insert NULL values * Note that changes are not durable (permanent) until you run the COMMIT command. If you choose undo the change before you commit, run the ROLLBACK command. |

## Using a SUBQUERY to INSERT MANY rows

|  |  |
| --- | --- |
| **Syntax** | INSERT into *tablename*  *SELECT statement here* |
| **Example** | INSERT INTO invoice\_archive (invoice\_id, vendor\_id, invoice\_number, invoice\_total, credit\_total, payment\_total, terms\_id, invoice\_date, invoice\_due\_date)  SELECT invoice\_id, vendor\_id, invoice\_number, invoice\_total, credit\_total, payment\_total, terms\_id, invoice\_date, invoice\_due\_date  FROM invoices  WHERE invoice\_total - payment\_total - credit\_total = 0; |
| **More info** | * Note: the number of columns in the SELECT must match the number of columns in your INSERT INTO clause * Note that changes are not durable (permanent) until you run the COMMIT command. If you choose undo the change before you commit, run the ROLLBACK command. |

## UPDATE data in a table

|  |  |
| --- | --- |
| **Syntax** | UPDATE table\_name SET column1 = value1, column2 = value2, ... WHERE condition; |
| **Example** | --update two column values for a single row  UPDATE invoices  SET payment\_date = '21-SEP-14',  payment\_total = 19351.18  WHERE invoice\_number = '97/522';  --update a single column for many rows  UPDATE invoices  SET terms\_id = 1  WHERE vendor\_id = 95;  --update a single column using artihmetic expression  UPDATE invoices  SET credit\_total = credit\_total + 100  WHERE invoice\_number = '97/522'; |
| **More info** | * NOTE: The WHERE condition determines how many records will be updated. * Note that changes are not durable (permanent) until you run the COMMIT command. If you choose undo the change before you commit, run the ROLLBACK command. * For more details see: <https://www.w3schools.com/sql/sql_update.asp> |

## Using a SUBUERY in an UPDATE statement

|  |  |
| --- | --- |
| **Syntax** | You are allowed to insert a select subquery in the WHERE or even set a column equal to the results of a Subquery. It’s best practice to wrap your subquery in parenthesis. |
| **Example** | --update all invoices based on a select for many records  UPDATE invoices  SET terms\_id = 1  WHERE vendor\_id IN  (SELECT vendor\_id  FROM vendors  WHERE vendor\_state IN ('CA', 'AZ', 'NV'));  --assign the max invoice\_due\_date from invoices to specific invoice  UPDATE invoices  SET credit\_total = credit\_total + 100,  invoice\_due\_date = (SELECT MAX(invoice\_due\_date) FROM invoices)  WHERE invoice\_number = '97/522'; |
| **More info** | Note that changes are not durable (permanent) until you run the COMMIT command. If you choose undo the change before you commit, run the ROLLBACK command. |

## DELETE data from a table

|  |  |
| --- | --- |
| **Syntax** | DELETE FROM *tablename*  WHERE *condition* |
| **Example** | --delete of a specific row  DELETE FROM invoice\_line\_items  WHERE invoice\_id = 100 AND invoice\_sequence = 1;  --delete of multiple rows  DELETE FROM invoice\_line\_items  WHERE invoice\_id = 114;  --delete with a subquery. Again...useful when spanning two tables  DELETE FROM invoice\_line\_items  WHERE invoice\_id IN  (SELECT invoice\_id  FROM invoices  WHERE vendor\_id = 115); |
| **More info** | If you want to delete all records, you can leave out the WHERE condition or just use TRUNCATE. For more info on DELETE: <https://www.w3schools.com/sql/sql_delete.asp> |

## Commit and Rollback

https://www.oracle-dba-online.com/sql/commit\_rollback\_savepoint.htm

|  |  |
| --- | --- |
| **Syntax** | COMMIT; --This makes your data changes to the table permanent (i.e. Saves)  ROLLACK;--This reverses your data changes if you haven’t already committed (i.e. undo) |
| **More** | Data Inserts, Updates, and Deletes are not officially saved to table for other users to see until you COMMIT. You can see your changes before Commit; If you rollback before commit it will undo changes  <https://www.oracle-dba-online.com/sql/commit_rollback_savepoint.htm> |

# **SELECT clause**

## Basic SELECT from a single table

|  |  |
| --- | --- |
| **Syntax** | SELECT *[\*] or [columns, expressions, subqueries]*  FROM *table(s) or subquery* |
| **Example** | --Select statement that retrieves all the data from Invoices table  SELECT \*  FROM invoices;  --Select statement that retrieves three columns from each row  SELECT invoice\_number, invoice\_date, invoice\_total  FROM invoices; |
| **More info** | <https://www.w3schools.com/sql/sql_select.asp> |

## Rename column with COLUMN ALIAS

|  |  |
| --- | --- |
| **Syntax** | SELECT column\_name AS alias\_name  FROM table\_name; |
| **Example** | --The first column alias uses standard syntax and doesn’t require quotes around the alias because there are not spaces  --The 2nd column has no alias so it will just use the default column name  --The 3rd column alias has spaces in it which requires the use of double quotes (note: not single quotes)  SELECT invoice\_number AS Invoice\_Number,  invoice\_date,  invoice\_total AS "Total of invoice"  FROM invoices; |
| **More info** | <https://www.w3schools.com/sql/sql_alias.asp> |

## Select with STRING EXPRESSIONS

|  |  |
| --- | --- |
| **Syntax** | You can use the || operator to concatenate columns and/or literal strings. |
| **Example** | --how to concat string data  SELECT vendor\_city,  vendor\_state, vendor\_city || vendor\_state as City\_State  FROM vendors;  --how to format strings using columns and literal string values  SELECT vendor\_name,  vendor\_city || ', ' || vendor\_state || ' ' || vendor\_zip\_code AS Location  FROM vendors;  --concatenating many literal string values  SELECT 'SQL' || ' ' || 'runs' || ' ' || 'the world!' as truth  from dual; |
| **More info** | FYI: Other DBMSs like Postgres claim to use this operator too but some like SQL Server will use the + operator to concatenate strings. Some DBMS support the use of a CONCAT() function. |

## Select with ARITHMETIC EXPRESSIONS

|  |  |
| --- | --- |
| **Syntax** | You can use any standard math operator (+ - \* /) to do math with values in the table. Normal order of operators (PEMDAS) apply |
| **Example** | --a statement that calculates the balance due to vendor  SELECT invoice\_total, payment\_total, credit\_total,  invoice\_total - payment\_total - credit\_total AS balance\_due  FROM invoices; |
| **More info** | SQL Operators - <https://www.w3schools.com/sql/sql_operators.asp> |

## Select with SCALAR FUNCTIONS

|  |  |
| --- | --- |
| **Syntax** | There are a number of functions that you’ve probably used in Excel or other coding languages. Many of these and other data-related functions are accessible in SQL and some unique to Oracle’s DBMS. These can be used in the SELECT and WHERE |
| **Example** | --Use of COUNT function  select COUNT(\*) as Count\_of\_Vendor\_Records  from vendors;  --SYSDATE and ROUND functions  SELECT invoice\_date,  SYSDATE AS today,  sysdate - invoice\_date,  ROUND(SYSDATE - invoice\_date) AS invoice\_age\_in\_days  FROM invoices;  --Example of SUBSTR function for selecting parts of a strings  SELECT vendor\_contact\_first\_name, vendor\_contact\_last\_name,  SUBSTR( vendor\_contact\_first\_name, 1, 1) ||  SUBSTR( vendor\_contact\_last\_name, 1, 1) AS initials  FROM vendors; |
| **More info** | Docs on all Oracle Functions - <https://docs.oracle.com/cd/B19306_01/server.102/b14200/functions001.htm>  Some basic functions used in SELECT   * MIN and MAX - <https://www.w3schools.com/sql/sql_min_max.asp> * COUNT, AVG, SUM - <https://www.w3schools.com/sql/sql_count_avg_sum.asp> |

## Use of the DUAL table

|  |  |
| --- | --- |
| **Syntax** | You cannot select with a FROM clause. In cases you don’t need a table, Dual serves as a “dummy” table that will allow you to test things like expressions in your SELECT clause. |
| **Example** | --Example of using the DUAL table  SELECT 'test' AS test\_string,  10-7 AS test\_calculation,  SYSDATE AS test\_date,  (10+10)\*2 as "testing"  from dual; |
| **More info** | <https://docs.oracle.com/cd/B19306_01/server.102/b14200/queries009.htm> |

# **Filter with WHERE**

## Filter rows using WHERE causes

|  |  |
| --- | --- |
| **Syntax** | SELECT column1, column2, ... FROM table\_name WHERE condition;  In a condition you can use a *column name* or *expression*. You cannot use a *column* *alias*. See below |
| **More info** | * NOTE: The WHERE clause cannot use a column alias to filter rows. This is due to the order in which clauses execute for a SQL Query. [Click here to learn about SQL order of operations](https://sqlbolt.com/lesson/select_queries_order_of_execution) * More info on WHERE: <https://www.w3schools.com/sql/sql_where.asp> |

## Comparison operators (=, >, <, >=, <=, <>)

|  |  |
| --- | --- |
| **Syntax** | You can use all of the usual operators to compare columns or expressions.  =, >, <, >=, <=  <> and != act as a “not equal to” operator |
| **Example** | --Example 1: WHERE using a column name  SELECT \*  FROM VENDORS  WHERE VENDOR\_STATE = 'IA';  --Example 2: WHERE using an expression  select \*  from invoices  where invoice\_total - payment\_total - credit\_total > 0;  --Example 3: Using range operators to pull back a range of values  select \*  from invoices  where invoice\_date <= '31-MAY-14';  --Example 4: Not equal to  select \*  from invoices  where credit\_total <> 0; |
| **More info** | <https://www.w3schools.com/sql/sql_operators.asp> |

## Logical operators (AND, OR, NOT)

|  |  |
| --- | --- |
| **Syntax** | You can use the following operators to change the logical order/flow of a filter: AND, OR, NOT |
| **Example** | --Example 1: Using AND to return rows that meet **both** conditions  select \*  from vendors  where vendor\_state = 'NJ' AND vendor\_city = 'Springfield';  --Example 2: Using OR to return rows that meet **either** condition  select \*  from vendors  where vendor\_state = 'NJ' OR vendor\_city = 'Springfield';  --Example 3: Using NOT to return records that **don’t** meet condition at all  select \*  from invoices  where NOT (invoice\_total >= 5000);    TIP: If you plan to combine the use of “AND” or “OR”, we recommend you use parenthesis since “AND” take precedence to “OR” in order of operations. e.g. The two clauses below will return different results  WHERE invoice\_date > '01-MAY-2014' OR invoice\_total > 500  AND invoice\_total - payment\_total - credit\_total > 0  WHERE (invoice\_date > '01-MAY-2014' OR invoice\_total > 500)  AND invoice\_total - payment\_total - credit\_total > 0 |
| **More info** | <https://www.w3schools.com/sql/sql_and_or.asp> |

## IN operator

|  |  |
| --- | --- |
| **Syntax** | SELECT column\_name(s) FROM table\_name WHERE column\_name IN (value1, value2, ...);  **OR**  SELECT column\_name(s) FROM table\_name WHERE column\_name IN (*SELECT* STATEMENT); |
| **Example** | --Example 1: IN operator with numbers. NOTE that we don’t put quotes around numeric values  select \*  from invoices  where terms\_id in (1,2,3);  --Example 2: IN operator with NOT operator and string values.  select \*  from vendors  where vendor\_state NOT IN ('CA','NV','OR');  --Example 3: IN operator with a SUBQUERY.  select \*  from invoices  where vendor\_id in (select vendor\_id  from invoices  where invoice\_date = '01-MAY-2014'); |
| **More info** | <https://www.w3schools.com/sql/sql_in.asp> |

## BETWEEN operator

|  |  |
| --- | --- |
| **Syntax** | SELECT column\_name(s) FROM table\_name WHERE column\_name BETWEEN value1 AND value2; |
| **Example** | --Example 1: BETWEEN with literal values  select \*  from invoices  where invoice\_date BETWEEN '01-MAY-2014' AND '31-MAY-2014';  --Example 2: BETWEEN proceeded by NOT  select \*  from vendors  where vendor\_zip\_code NOT BETWEEN 93600 and 93799;  --Example 3: BETWEEN operator with a test expression coded as a calculated value  select \*  from invoices  where invoice\_total - payment\_total - credit\_total BETWEEN 200 and 500;  --Example 4: BETWEEN operator with calculated values  select \*  from invoices  where invoice\_due\_date BETWEEN SYSDATE AND (SYSDATE + 30); |
| **More info** | <https://www.w3schools.com/sql/sql_between.asp> |

## LIKE operator

|  |  |
| --- | --- |
| **Syntax** | SELECT column1, column2, ... FROM table\_name WHERE columnN LIKE pattern;  NOTE: See link below on all the various patterns you can use |
| **Example** | --Example 1: Looks for values that **starts with** a specific value. In this case, cities starting with “San”  SELECT \*  FROM VENDORS  WHERE VENDOR\_CITY LIKE ('San%');  --Example 2: Looks for values that **contain** specific value. In this case, vendor names with “cat” in it  SELECT \*  FROM VENDORS  WHERE VENDOR\_name like ('%cat%'); |
| **More info** | <https://www.w3schools.com/sql/sql_like.asp> |

## IN NULL operator

|  |  |
| --- | --- |
| **Syntax** | SELECT column\_namesFROM table\_name WHERE column\_name IS NULL;  **OR**  SELECT column\_namesFROM table\_name WHERE column\_name IS NOT NULL; |
| **Example** | --Example 1: Return all records that have only have NULL values in a specific column  select \*  from vendors  where vendor\_address2 is null;  --Example 2: Return all records that have a value (i.e. are not NULL or blank) in a specific column  select \*  from vendors  where vendor\_address2 is not null; |
| **More info** | <https://www.w3schools.com/sql/sql_null_values.asp> |

# **Sort with ORDER BY**

## Sort data by COLUMN NAME

|  |  |
| --- | --- |
| **Syntax** | SELECT column1, column2, ... FROM table\_name ORDER BY column1, column2, ... ASC|DESC; |
| **Example** | --Example 1: SORT BY ONE COLUMN  SELECT vendor\_name,  vendor\_city || ', ' || vendor\_state || ' ' || vendor\_zip\_code AS address  FROM vendors  ORDER BY vendor\_name;  --Example 2: SORT BY ONE COLUMN **DESCENDING**  SELECT vendor\_name,  vendor\_city || ', ' || vendor\_state || ' ' || vendor\_zip\_code AS address  FROM vendors  ORDER BY vendor\_name DESC;  --Example 3: SORT BY THREE COLUMNS  SELECT vendor\_name,  vendor\_city || ', ' || vendor\_state AS city\_state  FROM vendors  ORDER BY vendor\_state, vendor\_city, vendor\_name; |
| **More info** | <https://www.w3schools.com/sql/sql_orderby.asp> |

## Sort by COLUMN ALIAS

|  |  |
| --- | --- |
| **Example** | --SORT BY ALIAS  SELECT vendor\_name,  vendor\_city || ', ' || vendor\_state AS city\_state  FROM vendors  ORDER BY city\_state, vendor\_name; |

## Sort by EXPRESSION

|  |  |
| --- | --- |
| **Example** | --SORT BY EXPRESSION  SELECT vendor\_name,  vendor\_city || ', ' || vendor\_state AS city\_state  FROM vendors  ORDER BY vendor\_contact\_last\_name || vendor\_contact\_first\_name; |

## Sort by COLUMN POSITION

|  |  |
| --- | --- |
| **Example** | --SORT BY COLUMN POSITION  SELECT vendor\_name,  vendor\_city || ', ' || vendor\_state AS city\_state  FROM vendors  ORDER BY 2, 1; |

# **More Resources**

Intro to SQL Syntax - <https://www.w3schools.com/sql/sql_syntax.asp>