Cheatography

SQL Cheat Sheet

by ArturPuiu via cheatography.com/133667/cs/39867/

General Commands

SELECT	Select data from a database.	SELECT column1, column2 FROM table_name;
SELECT DISTINCT	Return only distinct values.	SELECT DISTINCT column1, column2 FROM table_name;
WHERE	Extract records that fulfill a specified condition.	SELECT column1, column2 FROM table_name WHERE condition;
AND, OR and NOT	Filter records based on more than one condition. Combined with WHERE.	
The AND operator displays a record if all the conditions separated by AND are TRUE.	SELECT column1, column2 FROM table_name WHERE condition1 AND condition2;	
The OR operator displays a record if any of the conditions separated by OR is TRUE.	SELECT column1, column2 FROM table_name WHERE condition1 OR condition2;	
The NOT operator displays a record if the condit-ion(s) is NOT TRUE	SELECT column1, column2 FROM table_name WHERE NOT condition;	
ORDER BY	Sort the result-set in ascending or descending order. *Ascending order is by default.*	SELECT column1, column2 FROM table_name ORDER BY column1 ASC|DESC;

General Commands (cont)

INSERT INTO	Insert new records in a table.	
1. Specify both the column names and the values.	INSERT INTO table_name (column1, column2, column3) VALUES (value1, value2)	
If you are adding values for all the columns of the table, no need to specify the column names.	INSERT INTO table_name VALUES (value1, value2);	
NULL	A field with a NULL value is a field with no value. A field with a NULL value is one that has been left blank during record creation.	
IS NULL Syntax	SELECT column_names FROM table_name WHERE column_name IS NULL;	
IS NOT NULL Syntax	SELECT column_names FROM table_name WHERE column_name IS NOT NULL;	
UPDATE	Modify the existing records in a table.	UPDATE table_name SET column1 = value1, column2 = value2 WHERE condition;
DELETE	Delete existing records in a table.	DELETE FROM table_name WHERE condition;

General Commands (cont)

LIMIT	Specify the number of records to return.	SELECT column_name(s) FROM table_name WHERE condition LIMIT number;
IN	Allows to specify multiple values in a WHERE clause. The IN operator is a shorthand for multiple OR conditions.	SELECT column_name(s) FROM table_name WHERE column_name IN (value1, value2);
	SELECT column_name(s) FROM table_name WHERE column_name IN (SELECT STATEM- ENT);	
BETWEEN	Selects values within a given range. The values can be numbers, text, or dates. *Is inclusive*.	SELECT column_name(s) FROM table_name WHERE column_name BETWEEN value1 AND value2;
GROUP BY	Groups rows that have the same values into summary rows. Is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.	SELECT column_name(s) FROM table_name WHERE condition GROUP BY column_na- me(s) ORDER BY column_na- me(s);

General Commands (cont)

HAVING	Was added to SQL because the WHERE keyword cannot be used with aggregate functions	SELECT column_name(s) FROM table_name WHERE condition GROUP BY column_name(s) HAVING condition ORDER BY column_name(s);
EXISTS	Test for the existence of any record in a subquery. Returns TRUE if the subquery returns one or more records.	SELECT column_name(s) FROM table_name WHERE EXISTS (SELECT column_name FROM table_name WHERE condition);
ANY	Allow to perform a comparison between a single column value and a range of other values. *1.Returns a boolean value as a result 2. Returns TRUE if ANY of the subquery values meet the condition.* ANY means that the condition will be true if the operation is true for any of the values in the range.	SELECT column_name(s) FROM table_name WHERE column_name operator ANY (SELECT column_name FROM table_name WHERE condition);

General Commands (cont)

ALL	Returns: *1. A boolean value as a result* *2. Returns TRUE if ALL of the subquery values meet the condition* *3. Is used with SELECT, WHERE and HAVING statements.* ALL means that the condition will be true only if the operation is true for all values in the range.
ALL Syntax With SELECT	SELECT ALL column_name(s) FROM table_name WHERE condition;
ALL Syntax With WHERE or HAVING	SELECT column_name(s) FROM table_name WHERE column_name operator ALL (SELECT column_name FROM table_name WHERE condition);

General Commands (cont)

INSERT INTO SELECT	Copies data from one table and Inserts it into another table. Requires that the data types in source and target tables matches.The existing records in the target table are unaffected.	
	Copy all columns from one table to another table	INSERT INTO table2 SELECT * FROM table1 WHERE condition;
	Copy only some columns from one table into another table	INSERT INTO table2 (col1, col2 ...) SELECT col1, col2 ... FROM table1 WHERE condition;

General Commands (cont)

CASE	Goes through conditions and returns a value when the first condition is met (like an if-then-else statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause. If there is no ELSE part and no conditions are true, it returns NULL.	CASE WHEN condition1 THEN result1 WHEN condition2 THEN result2 WHEN conditionN THEN resultN ELSE result END;

DATABASE/TABLE

CREATE SCHEMA	Create a new SQL schema	CREATE SCHEMA *sch*;
DROP SCHEMA	Drop a SQL schema	DROP SCHEMA *Sch*;
SET search_path = sch;		
CREATE DATABASE	Create a new SQL database	CREATE DATABASE *db*;
DROP DATABASE	Drop a SQL database	DROP DATABASE *db*;
BACKUP DATABASE	Create a full back up of an existing SQL database. Ex: 'E:\testD-B.bak'	BACKUP DATABASE *db* TO DISK = *'filepath'*;
	A differential back up only backs up the parts of the database that have changed since the last full database backup.	BACKUP DATABASE *db* TO DISK = *'filepath'* WITH DIFFERENTIAL;

DATABASE/TABLE (cont)

CREATE TABLE	Create a new table in a database	CREATE TABLE *tb* (*col1 datatype*, *col2 datatype*, *col3 datatype*);
A copy of an existing table can also be created using CREATE TABLE. The new table will be filled with the existing values from the old table	CREATE TABLE *tb* AS SELECT *col1, col2,...* FROM *existing tb* WHERE ....;	
DROP TABLE	Drop an existing table in a database	DROP TABLE *tb*;
TRUNCATE TABLE	Delete the data inside a table, but not the table itself.	TRUNCATE TABLE *tb*;
ALTER TABLE	Is used to add, delete, or modify columns in an existing table	
Add a column in a table	ALTER TABLE *tb* ADD *col datatype*;	
Delete a column in a table	ALTER TABLE *tb* DROP COLUMN *col*;	
Rename a column in a table	ALTER TABLE *tb* RENAME COLUMN *old_name* TO *new_name*;	
Rename a column in a table in SQL Server	EXEC sp_rename *'tb_name.old_name'*, *'new_name'*, *'COLUMN'*;	

DATABASE/TABLE (cont)

Change the data type of a column in a table	**SQL Server / MS Access** ALTER TABLE *tb_name* ALTER COLUMN *col_name datatype*;	
	My SQL / Oracle ALTER TABLE *tb_name* MODIFY COLUMN *col_name datatype*;	
	Oracle 10G and later ALTER TABLE *tb_name* MODIFY *col_name datatype*;	
Constr-aints	Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement.	CREATE TABLE *tb_name* (*col1 datatype constraint,* *col2 datatype constraint*);

Numeric Functions

MIN()	Returns the minimum value in a set of values.
MAX()	Returns the maximum value in a set of values.
COUNT()	Returns the number of records. NULL values are not counted.
AVG()	Returns the average value of an expression. NULL values are ignored.
SUM()	Calculates the sum of a set of values. NULL values are ignored.

Arithmetic Operators

+	Add
-	Subtract
*	Multiply
/	Divide

Arithmetic Operators (cont)

%	Modulo

Bitwise Operators

&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR

Comparison Operators

=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<>	Not equal to

Compound Operators

+=	Add equals
-=	Subtract equals
*=	Multiply equals
/=	Divide equals
%=	Modulo equals
&=	Bitwise AND equals
^-=	Bitwise exclusive equals
|*=	Bitwise OR equals

Logical Operators

ALL	TRUE if all of the subquery values meet the condition
AND	TRUE if all the conditions separated by AND is TRUE
ANY	TRUE if any of the subquery values meet the condition
BETWEEN	TRUE if the operand is within the range of comparisons
EXISTS	TRUE if the subquery returns one or more records
IN	TRUE if the operand is equal to one of a list of expressions

Logical Operators (cont)

LIKE	TRUE if the operand matches a pattern
NOT	Displays a record if the condition(s) is NOT TRUE
OR	TRUE if any of the conditions separated by OR is TRUE
SOME	TRUE if any of the subquery values meet the condition

Joining Tables

INNER JOIN	Selects records that have matching values in both tables.	SELECT column_name(s) FROM table1 INNER JOIN table2 ON table1.column_name = table2.column_name;
LEFT JOIN	Returns all records from the left table (table1), and the matching records (if any) from the right table (table2).	SELECT column_name(s) FROM table1 LEFT JOIN table2 ON table1.column_name = table2.column_name;
RIGHT JOIN	Returns all records from the right table (table2), and the matching records (if any) from the left table (table1).	SELECT column_name(s) FROM table1 RIGHT JOIN table2 ON table1.column_name = table2.column_name;
CROSS JOIN	Returns all records from both tables (table1 and table2).	SELECT column_name(s) FROM table1 CROSS JOIN table2;
SELF JOIN	A self join is a regular join, but the table is joined with itself.	SELECT column_name(s) FROM table1 T1, table1 T2 WHERE condition;

Joining Tables (cont)

UNION	Combine the result-set of two or more SELECT statements. *1) Every SELECT statement within UNION must have the same number of columns 2) The columns must also have similar data types 3) The columns in every SELECT statement must also be in the same order*
UNION Syntax. *Selects only distinct values by default. To allow duplicate values, use UNION ALL*	SELECT column_name(s) FROM table1 UNION SELECT column_name(s) FROM table2;
UNION ALL	SELECT column_name(s) FROM table1 UNION ALL SELECT column_name(s) FROM table2;

Date Functions

ADDDATE() OR DATE_ADD()	adds a time/date interval to a date and then returns the date. *ADDDATE(date, INTERVAL value addunit)* *ADDDATE(date, days)* *DATE_ADD(date, INTERVAL value addunit)*
ADDTIME()	Adds a time interval to a time/datetime and then returns the time/datetime. *ADDTIME(datetime, addtime)*

Date Functions (cont)	
CURDATE()	Returns the current date. The date is returned as "-YYYY-MM-DD" (string) or as YYYYMMDD (numeric). This function equals the CURDATE() function. *CURDATE()*
CURRENT_DATE()	Returns the current date. The date is returned as "-YYYY-MM-DD" (string) or as YYYYMMDD (numeric). This function equals the CURRENT_DATE() function. *CURRENT_DATE()*
CURRENT_TIME()	Returns the current time. The time is returned as "-HH-MM-SS" (string) or as HHMMSS.uuuuuu (numeric). This function equals the CURTIME() function. *CURRENT_TIME()*
CURTIME()	Returns the current time. The time is returned as "-HH-MM-SS" (string) or as HHMMSS.uuuuuu (numeric). This function equals the CURRENT_TIME() function. *CURTIME()*
CURRENT_TIMESTAMP()	Returns the current date and time. The date and time is returned as "YYYY-MM-DD HH-MM-SS" (string) or as YYYYMMDDHHMMSS.uuuuuu (numeric). *CURRENT_TIMESTAMP()*
DATE()	Extracts the date part from a datetime expression. *DATE(expression)*
DATEDIFF()	Returns the number of days between two date values. *DATEDIFF(date1, date2)*

Date Functions (cont)	
DATE_FORMAT()	Formats a date as specified. *DATE_FORMAT(-date, format)*
DATE_SUB()	Subtracts a time/date interval from a date and then returns the date. *DATE_SUB(date, INTERVAL value interval)*
DAY() OR DAYOFMONTH	returns the day of the month for a given date (a number from 1 to 31). *DAY(date) DAYOFMONTH(date)*
DAYNAME()	returns the weekday name for a given date. *DAYNAME(date)*
DAYOFWEEK()	returns the weekday index for a given date (a number from 1 to 7). 1=Sunday, 2=Monday, 3=Tuesday, 4=Wednesday, 5=Thursday, 6=Friday, 7=Saturday. *DAYOFWEEK(date)*
DAYOFWEEK-(date)	returns the day of the year for a given date (a number from 1 to 366). *DAYOFYEAR(date)*
EXTRACT()	extracts a part from a given date. *EXTRACT(part FROM date)*
FROM_DAYS()	returns a date from a numeric datevalue.is to be used only with dates within the Gregorian calendar. is the opposite of the TO_DAYS() function. *FROM_DAYS(number)*
HOUR()	returns the hour part for a given date (from 0 to 838). HOUR(datetime)
LAST_DAY()	extracts the last day of the month for a given date. LAST_DAY(date)

Date Functions (cont)	
LOCALTIME() OR LOCALTIMESTAMP()	returns the current date and time. The date and time is returned as "YYYY-MM-DD HH-MM-SS" (string) or as YYYYMMDDHHMMSS.uuuuuu (numeric). LOCALTIME()
MAKEDATE()	creates and returns a date based on a year and a number of days value. MAKEDATE(year, day)
MAKETIME()	Create and return a time value based on an hour, minute, and second value. MAKETIME(hour, minute, second)
MICROSECOND()	Return the microsecond part of a datetime. MICROSECOND(datetime)