SQL Cheatsheet  
  
DDL (Data Definition Language)  
  
Definition: Deals with descriptions of the database schema and is used to create and modify the structure of database objects  
  
Examples of DDL commands:

* CREATE – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
* DROP – is used to delete objects from the database.
* ALTER - is used to alter the structure of the database.
* TRUNCATE – is used to remove all records from a table, including all spaces allocated for the records are removed.
* COMMENT – is used to add comments to the data dictionary.
* RENAME – is used to rename an object existing in the database.

CREATE  
CREATE DATABASE database\_name  
CREATE TABLE (col def,…,PRIMARY KEY(col),FOREIGN KEY (col) REFERENCES table(col2))  
CREATE PROCEDURE procedure\_name AS sql\_statement GO;  
CREATE INDEX idx\_city ON table\_customers(field\_city);  
DROP INDEX idx\_city ON table\_customers;  
CREATE VIEW view\_name AS SELECT column1, column2 FROM table\_name WHERE condition;  
CREATE TRIGGER [TRIGGER\_NAME] [before | after] {insert | update | delete} on [table\_name] [FOR/AFTER/INSTEAD OF] [INSERT/UPDATE/DELETE] AS [trigger\_body]  
  
ALTER  
ADD: add a column  
ALTER TABLE table\_name ADD column\_name column\_definition;  
MODIFY: change data type of column  
ALTER TABLE table\_name MODIFY column\_name column\_type;  
DROP: delete a column  
ALTER TABLE table\_name DROP COLUMN column\_name;  
  
DROP  
DROP database\_name  
DROP VIEW view\_name;  
  
TRUNCATE  
TRUNCATE TABLE table\_name;  
  
RENAME  
RENAME TABLE table\_name TO new\_table\_name;  
RENAME DATABASE database\_name TO new\_database\_name;  
  
EXECUTE PROCEDURE  
EXEC procedure\_name;  
  
  
DML (Data Manipulation Language)  
  
Definition: Deals with the manipulation of data present in database  
  
Examples of DML:

* SELECT – is used to retrieve data from the a database.
* INSERT – is used to insert data into a table.
* UPDATE – is used to update existing data within a table.
* DELETE – is used to delete records from a database table.

SELECT  
SELECT: used to select data from a database  
SELECT \* FROM table\_name;  
SELECT \* FROM view\_name;  
DISTINCT: returns distinct values only (filters away duplicate values and returns rows of specified column)  
SELECT DISTINCT column\_name;  
WHERE: used to filter records/rows  
Where modifiers  
  
SELECT column1, column2 FROM table\_name WHERE condition;  
SELECT \* FROM table\_name WHERE condition1 AND condition2;  
SELECT \* FROM table\_name WHERE condition1 OR condition2;  
SELECT \* FROM table\_name WHERE NOT condition;  
SELECT \* FROM table\_name WHERE condition1 AND (condition2 OR condition3);  
SELECT \* FROM table\_name WHERE EXISTS (SELECT column\_name FROM table\_name WHERE condition);  
WHERE exp AND|OR exp AND|OR exp…  
  
where exp can be one of the following:  
  
column = value  
column > value  
column >= value  
column < value  
column <= value  
column BETWEEN value1 AND value2  
column IN (value1,value2,…)  
column NOT IN (value1,value2,…)  
column LIKE value  
column NOT LIKE value  
ORDER BY: used to sort the result-set in ascending or descending order  
SELECT \* FROM table\_name ORDER BY column;  
SELECT \* FROM table\_name ORDER BY column DESC;  
SELECT \* FROM table\_name ORDER BY column1 ASC, column2 DESC;  
SELECT TOP: used to specify the number of records to return from top of table  
SELECT TOP number columns\_names FROM table\_name WHERE condition;  
SELECT TOP percent columns\_names FROM table\_name WHERE condition;  
Not all database systems support SELECT TOP. The MySQL equivalent is the LIMIT clause  
SELECT column\_names FROM table\_name LIMIT offset, count;  
LIKE: operator used in a WHERE clause to search for a specific pattern in a column  
% (percent sign) is a wildcard character that represents zero or more characters  
\_ (underscore) is a wildcard character that represents a single character (\_ is exactly one character in the LIKE statement)  
SELECT column\_names FROM table\_name WHERE column\_name LIKE pattern;  
LIKE ‘a%’ (find any values that start with “a”)  
LIKE ‘%a’ (find any values that end with “a”)  
LIKE ‘%or%’ (find any values that have “or” in any position)  
LIKE ‘\_r%’ (find any values that have “r” in the second position)  
LIKE ‘a\_\_%’ (find any values that start with “a” and are at least 3 characters in length)  
LIKE ‘[a-c]%’ (find any values starting with “a”, “b”, or “c”  
LIKE '%[^0-9]%' (match all strings that don't have a digit)  
IN: operator that allows you to specify multiple values in a WHERE clause  
essentially the IN operator is shorthand for multiple OR conditions  
SELECT column\_names FROM table\_name WHERE column\_name IN (value1, value2, …);  
SELECT column\_names FROM table\_name WHERE column\_name IN (SELECT STATEMENT);  
BETWEEN: operator selects values within a given range inclusive  
SELECT column\_names FROM table\_name WHERE column\_name BETWEEN value1 AND value2;  
SELECT \* FROM Products WHERE (column\_name BETWEEN value1 AND value2) AND NOT column\_name2 IN (value3, value4);  
SELECT \* FROM Products WHERE column\_name BETWEEN #01/07/1999# AND #03/12/1999#;  
NULL: values in a field with no value  
SELECT \* FROM table\_name WHERE column\_name IS NULL;  
SELECT \* FROM table\_name WHERE column\_name IS NOT NULL;  
AS: aliases are used to assign a temporary name to a table or column  
SELECT column\_name AS alias\_name FROM table\_name;  
SELECT column\_name FROM table\_name AS alias\_name;  
SELECT column\_name AS alias\_name1, column\_name2 AS alias\_name2;  
SELECT column\_name1, column\_name2 + ‘, ‘ + column\_name3 AS alias\_name;  
UNION: Unions combine data into new rows  
Each SELECT statement within UNION must have the same number of columns  
The columns must have similar data types  
The columns in each SELECT statement must also be in the same order  
SELECT columns\_names FROM table1 UNION SELECT column\_name FROM table2;  
UNION operator only selects distinct values, UNION ALL will allow duplicates  
JOIN: Joins combine data into new columns  
Diagrams  
  
- INNER JOIN: returns records that have matching value in both tables  
SELECT column\_names FROM table1 INNER JOIN table2 ON table1.column\_name=table2.column\_name;  
SELECT table1.column\_name1, table2.column\_name2, table3.column\_name3 FROM ((table1 INNER JOIN table2 ON relationship) INNER JOIN table3 ON relationship);  
- LEFT (OUTER) JOIN: returns all records from the left table (table1), and the matched records from the right table (table2)  
SELECT column\_names FROM table1 LEFT JOIN table2 ON table1.column\_name=table2.column\_name;  
- RIGHT (OUTER) JOIN: returns all records from the right table (table2), and the matched records from the left table (table1)  
SELECT column\_names FROM table1 RIGHT JOIN table2 ON table1.column\_name=table2.column\_name;  
- FULL (OUTER) JOIN: returns all records when there is a match in either left or right table  
SELECT column\_names FROM table1 FULL OUTER JOIN table2 ON table1.column\_name=table2.column\_name;  
- Self JOIN: a regular join, but the table is joined with itself  
SELECT column\_names FROM table1 T1, table1 T2 WHERE condition;  
- CROSS JOIN  
Each row from 1st table joins with all the rows of 2nd table.  
Example: | table\_a | table\_b | | ------- | ------- | | 1 | a | | 2 | b | | 3 | c |  
  
SELECT a.num, b.word FROM table\_a a CROSS JOIN table\_b b;  
  
Result: | num | word | | --- | ---- | | 1 | a | | 2 | a | | 3 | a | | 1 | b | | 2 | b | | 3 | b | | 1 | c | | 2 | c | | 3 | c |  
  
INTERSECT: set operator which is used to return the records that two SELECT statements have in common  
Generally used the same way as UNION above  
SELECT columns\_names FROM table1 INTERSECT SELECT column\_name FROM table2;  
EXCEPT: set operator used to return all the records in the first SELECT statement that are not found in the second SELECT statement  
Generally used the same way as UNION above  
SELECT columns\_names FROM table1 EXCEPT SELECT column\_name FROM table2;  
ANY|ALL: operator used to check subquery conditions used within a WHERE or HAVING clauses  
The ANY operator returns true if any subquery values meet the condition  
The ALL operator returns true if all subquery values meet the condition  
SELECT columns\_names FROM table1 WHERE column\_name operator (ANY|ALL) (SELECT column\_name FROM table\_name WHERE condition);  
GROUP BY: statement often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns  
SELECT column\_name1, COUNT(column\_name2) FROM table\_name WHERE condition GROUP BY column\_name1 ORDER BY COUNT(column\_name2) DESC;  
HAVING: this clause was added to SQL because the WHERE keyword could not be used with aggregate functions  
SELECT COUNT(column\_name1), column\_name2 FROM table GROUP BY column\_name2 HAVING COUNT(column\_name1) > 5;  
WITH: often used for retrieving hierarchical data or re-using temp result set several times in a query. Also referred to as "Common Table Expression"  
WITH RECURSIVE cte AS (  
SELECT c0.\* FROM categories AS c0 WHERE id = 1 # Starting point  
UNION ALL  
SELECT c1.\* FROM categories AS c1 JOIN cte ON c1.parent\_category\_id = cte.id  
)  
SELECT \*  
FROM cte  
COUNT: returns the # of occurrences  
SELECT COUNT (DISTINCT column\_name);  
MIN() and MAX(): returns the smallest/largest value of the selected column  
SELECT MIN (column\_names) FROM table\_name WHERE condition;  
SELECT MAX (column\_names) FROM table\_name WHERE condition;  
AVG(): returns the average value of a numeric column  
SELECT AVG (column\_name) FROM table\_name WHERE condition;  
SUM(): returns the total sum of a numeric column  
SELECT SUM (column\_name) FROM table\_name WHERE condition;  
  
INSERT  
Used to insert new records/rows in a table  
  
INSERT INTO table\_name (column1, column2) VALUES (value1, value2);  
INSERT INTO table\_name VALUES (value1, value2 …);  
  
UPDATE  
Used to modify the existing records in a table  
  
UPDATE table\_name SET column1 = value1, column2 = value2 WHERE condition;  
UPDATE table\_name SET column\_name = value;  
  
DELETE  
Used to delete existing records/rows in a table  
  
DELETE FROM table\_name WHERE condition;  
DELETE \* FROM table\_name;  
  
DCL (Data Control Language)  
Definition: Deals with the rights, permissions and other controls of the database system  
  
Examples of DCL commands:

* GRANT-gives user’s access privileges to database.
* REVOKE-withdraw user’s access privileges given by using the GRANT command.

GRANT  
GRANT privilege\_name  
ON object\_name {Database\_name|Table\_name|View\_name|Dashboard\_name}  
TO {user\_name |PUBLIC |role\_name}  
[WITH GRANT OPTION];  
privilege\_name[ALL|EXECUTE|SELECT|INSERT|UPDATE|DELETE|CREATE|ALTER|DROP]  
  
REVOKE  
REVOKE privilege\_name  
ON object\_name {Database\_name|Table\_name|View\_name|Dashboard\_name}  
FROM {user\_name |PUBLIC |role\_name}  
Example: Revoke ALL ANSI-92 permissions (ie: SELECT, INSERT, UPDATE, DELETE, and REFERENCES) on a table for a user named anderson  
  
REVOKE ALL ON employees FROM anderson;  
Privileges types by object type  
Database  
ACCESS  
ALL  
CREATE TABLE  
CREATE VIEW  
CREATE DASHBOARD  
DROP  
DROP VIEW  
DELETE DASHBOARD  
SELECT, INSERT, TRUNCATE, UPDATE, DELETE  
SELECT VIEW  
EDIT DASHBOARD  
VIEW DASHBOARD  
VIEW SQL EDITOR  
Table  
SELECT, INSERT, TRUNCATE, UPDATE, DELETE  
DROP  
TRIGGER  
REFERENCES  
EXECUTE  
View  
SELECT  
DROP  
Dashboard  
VIEW  
EDIT  
DELETE  
  
TCL (Transaction Control Language)  
Definition: Deals with the transaction within the database  
  
Examples of TCL commands:

* COMMIT– commits a Transaction.
* ROLLBACK– rollbacks a transaction in case of any error occurs.
* SAVEPOINT–sets a savepoint within a transaction.
* SET TRANSACTION–specify characteristics for the transaction.

Extras  
Select Random values  
Src  
  
SELECT col1 FROM tbl ORDER BY RAND() LIMIT 10;//returns 10 rows random  
Load data from file to table  
LOAD DATA INFILE ″filename″ INTO TABLE table  
List Databases & tables structure  
SHOW DATABASES|TABLES  
SHOW COLUMNS FROM table