Cheatography

SQL Cheat Sheet

by DarioPittera (aggialavura) via cheatography.com/83764/cs/44620/

General commands

create database db_name	create a database
use db_name	use a database
select distinct	select unique values
select * into new_tbl	copy a table
select top (n)	select the first X rows
select top (0) * into new_tbl from original_tbl	copy only the headers
order by	order by a column
and / or	AND OR operators
where	conditional selection
between	to choose values between two
in ("","")	check if contained in a list
insert into tbl | values ()	insert values into a table
where is null	select null values
update | set | where col is null	insert values into null cells of a col
delete from tbl where xx	delete certain values
sum(), count(), avg() - groupby	sum, count, and avg functions
having	condition on grouped data
inner join / join | on	return rows with match in both tables
left / right join | on	return common rows + left or right unique values
full outer join / outer join | on	return matching and non-matching values

General commands (cont)

inner join/ join | on with same tbl	= self join - duplicate entries
union all	merge two tables into one WITH DUPLICATES
union	merge two tables into one WITHOUT DUPLICATES
like | % - _	to match a condition with multiple chars (%) or one char (_)
case	create new col based on value of a different one
create table tbl_name	create a new table

Data types

STRINGS:
- char() - when we know it is a set len we can provide it between parenthesis
- varchar() - when the length is variable - we can set the max length
- nchar() - like char but supports unicode chars
- nvarchar() - like varchar but supports unicode chars

NUMBERS
- int
- bigint
- smallint
- tinyint
- decimal(p,s) - here p = total digits, s = digits after comma -> 999.99 = decimal(5,2)

DATE:
{{link="https://learn.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver16
"}}link text{{/link}}
- data
- time data

Data types (cont)

- datetime

Constraints

not null \ alter column col_name type not null	value must be not null
unique \ add unique(col)	must have a unique value
check(col_name >= x) \ add check(col_name>=x)	check if satisfy a condition
default \ add default col_name x	add a default if nothing has been provided
primary key \ add primary key (col)	assign the primary key to a col - has to be not null
foreign key references tbl_name(primary_key_col)\ add foreign key (col) references tbl(primary_key_col)	assign foreign key to a col - cannot have values outside the primary key

Order of execution

1. from and join
2. where
3. group by
4. having
5. select
6. distinct
7. order by
8. top

This means that for example if you give an alias in SELECT and then you try to use it in HAVING, it will throw an error because the select piece did not happened yet.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Cheatography

SQL Cheat Sheet

by DarioPittera (aggialavura) via cheatography.com/83764/cs/44620/

Window functions

raw_number() over()	assign a unique number to each row, if there are ties, the numbers are assigned randomly
rank() over()	assign a number based on ranking. If there are ties, it will skip a number/s
dense_rank() over()	assign a number based on ranking. If there are ties, it won't skip the number/s
lead(col) over()	allows you to access data from the subsequent row
lag() over()	access the previous row
isnull(col, 'value_to-_insert')	accepts two parame-ters: where to look, what to input instea-dinput instead
coalescence(col-1,col2,col3,'value-_to_insert')	accepts multiple parameters and it will output the first NON-NULL value
first_value(col) over()	will create a new column where the first cell value of an expression is written
last_value(col) over(rows between unbounded preceding and unbounded following)	will create a new column where the last cell value of an expression is written

Stored procedures

Stored procedures are precompiled set of SQL statements stored in the database that can be executed as a single unit.
To create them we will use **CREATE PROCEDURE** _name_ **AS BEGIN** _code_ **END**.
Example:

```
create procedure see_all -- or create proc
as
begin
select * from Employees
end
-- call the procedure
execute see_all
exec see_all
see_all
```

It can also accept parameters:

```
-- create a procedure that accepts
parameters
CREATE PROCEDURE GetEmployees-
ByDepartment
@DepartmentID INT -- = 101 to set default
to 101 for instance
AS
BEGIN
SELECT EmployeeID, FirstName,
LastName
FROM Employees
WHERE DepartmentID = @DepartmentID;
END;
exec GetEmployeesByDepartment 101
```

Views

A **view** in SQL is a virtual table based on the result of a SELECT query. Unlike physical tables, views don't store data themselves; they display data dynamically from the underlying tables whenever queried.
Views are helpful for:
- simplifying complex queries,
- improving code maintainability, and
- enhancing security by controlling data access.

```
CREATE VIEW view1 AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
select * from view1
```

NOTE!!! If you make changes on the View, they will be reflected on the original table as well!

By **DarioPittera** (aggialavura)

cheatography.com/aggialavura/
www.dariopittera.com

Not published yet.
Last updated 16th October, 2024.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com