PostgreSQL cheat sheet

Our practice databases

- Host: imperial-2021.ckp3dl3vzxoh.eu-west-2.rds.amazonaws.com
- Username: imperial
- Password: imperial-fdt-online-2019-colossal-shelf
- Port: 5432
- Database: **dvdrental** or **northwind** or **movies** or **world**

In Postgres

- Single quotes only - no double quotes
- This is a Postgres cheat sheet. MySQL or other versions of SQL may operate slightly differently.

Standard query structure

- SELECT (including window functions)
- FROM
- JOINs, each with an ON
- WHERE
- ORDER BY
- LIMIT

Query tips

- Build up your query slowly, running it as you go; start with a simple SELECT
- Capitalise all SQL keywords
- Use newlines clearly to break up the query

```
SELECT * FROM table_name
SELECT col_name FROM table_name
SELECT DISTINCT col_name FROM table_name
```

```
GROUP BY:
SELECT department, COUNT(employee_id)
FROM employees
GROUP BY department
```

Every **GROUP BY** needs an aggregate function: one of **COUNT**, **AVG**, **MAX**, **MIN**, **SUM**

Joins

```
SELECT col1, col2
FROM some_table
INNER JOIN some_other_table
ON some_table.key = some_other_table.key
```

A sample query

```
SELECT col1, col2
FROM some_table
INNER JOIN some_other_table
ON some_table.key = some_other_table.key
WHERE col1>4
ORDER BY col1 DESC
LIMIT 100
```

Updating

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Creating a new table

```
CREATE TABLE table_name(
         column1 datatype,
         column2 datatype,
         column3 datatype,
         .....
         columnN datatype
         );
```

Deleting a table

```
DROP TABLE table_name;
```

Inserting into a new table

```
insert into items_ver (item_id, name, item_group)
select item_id, name, item_group from items where
item_id=2;
```

Joins: shortcut syntax

```
SELECT *
    FROM weather w, cities c
    WHERE w.city = c.name;

SELECT *
    FROM weather, cities
    WHERE city = name;

SELECT *
FROM weather INNER JOIN cities
ON weather.city = cities.name;
```

Window functions

OVER indicates a window function

expression **OVER** (PARTITION BY *attribute*)

expression **OVER** (ORDER BY *attribute*)

```
SELECT depname, empno, salary, avg(salary)
OVER (PARTITION BY depname)
FROM empsalary;
```

Use LAG(column) to lag one column and LAG(column, n) to lag n columns. For example,

LAG(price, 2) OVER (ORDER BY date) will create a new column with price values lagged by 2.

LEAD is the opposite of LAG.

Subqueries

```
SELECT *
FROM film
WHERE rental_rate >
       (SELECT AVG(rental_rate) FROM film)

SELECT * FROM
(SELECT * FROM film) AS film_table
```

CTEs

```
WITH my_cte AS ( CTE query ),
my_second_cte AS ( second CTE query ),
main query
```

Views

```
CREATE VIEW my_view AS
view query
```

You can also use **CREATE OR REPLACE VIEW** or **DROP VIEW**

Loading data from a CSV

```
COPY movie
FROM '/path/to/data.csv'
DELIMITER ','
WITH CSV HEADER
```

\copy table_name **FROM** full_path **WITH CSV HEADER;**

Standard data types

varchar(*size*)	length-limited string
text	string of any length
integer	4-byte signed - 2147483648 to +2147483647
bigint	8-byte signed
serial	4-byte autoincrementing
real	4-byte floating point
double precision	8-byte floating point
money	currency
bool *or* boolean	true/false

Date/time types

date
timestamp
timestamp with timezone
time
time with timezone

Logical operators

AND, OR, NOT

Comparison operators

>, <, >=, <=, =, != or <>

a **BETWEEN** x **AND** y
a **IS DISTINCT FROM** b
a **IS NOT DISTINCT FROM** b
a **IS NULL**
a **IS NOT NULL**

expression = **NULL** will not work because nothing is equal to null (an unknown value)

Boolean expressions

expression **IS TRUE**
expression **IS NOT TRUE**
expression **IS FALSE**
expression **IS NOT FALSE**
expression **IS UNKNOWN**
expression **IS NOT UNKNOWN**
expression **IS NULL**

Set union: table_1 **UNION** table_2
Set intersection: table_1 **INTERSECT** table_2
Set difference: table_2 **EXCEPT** table_2

Check two queries for equality: (SELECT ...) = (SELECT ...)

Operators

\+ - * /
% (modulo)

You can use various mathematical functions like floor(), ceil(), round(). See the Postgres documentation for more.

|| string concatenation

String matching

string **LIKE** pattern

_ matches a single character
% matches a string of zero or more characters

Examples:
'abc' LIKE 'abc' *true*
'abc' LIKE 'a%' *true*
'abc' LIKE '_b_' *true*
'abc' LIKE 'c' *false*

string SIMILAR TO postgres_regexp

(The postgres regexp is similar to standard regexps, see the documentation)

Dates and times

time
time with timezone
date
timestamp (both date and time)
timestamp with timezone

date '2001-09-28' + time '03:00'
date '2001-09-28' + integer '7'
date '2001-09-28' + interval '1 hour'
timestamp '2001-09-28 01:00' + interval '23 hours'
timestamp '2001-09-29 03:00' - timestamp '2001-09-27 12:00'
time '05:00' - interval '2 hours'

DATE(col) converts to date
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
NOW() returns timestamp with timezone
date_trunc('hour', date) zeros all more granular parts
date_part('hour', date) extracts this particular part
(start1, end1) OVERLAPS (start2, end2)
(start1, length1) OVERLAPS (start2, length2)

Advanced aggregate functions

VARIANCE(col)
STDDEV(col)
EVERY(col) bool to bool

Subquery operators

EXISTS (subquery)
expression IN (subquery)
expression NOT IN (subquery)

expression operator ANY (subquery)
expression operator SOME (subquery)
expression operator ALL (subquery)

psql: command line options

-h *host*	set the host (such as localhost or the AWS server)
-p *port*	set the port (default 5432)
-U *user*	connect to Postgres with a particular Postgres username
-f *file*	run an SQL file (do queries or load data)
-E	show extra information about psql operations
-d *database*	connect to a particular database

psql: internal commands

help	show help
\q	quit
\l	list databases
\c *database*	connect to a particular database
\dt	list tables in current database
\lv	list views
\du	list database users