**https://leanpub.com/aprimeronsql/read**

**Some of The Most Important SQL Commands**

* **SELECT** - extracts data from a database
* **UPDATE** - updates data in a database
* **DELETE** - deletes data from a database
* **INSERT INTO** - inserts new data into a database
* **CREATE DATABASE** - creates a new database
* **ALTER DATABASE** - modifies a database
* **CREATE TABLE** - creates a new table
* **ALTER TABLE** - modifies a table
* **DROP TABLE** - deletes a table
* **CREATE INDEX** - creates an index (search key)
* **DROP INDEX** - deletes an index

**Manipulating Data**

In above examples, it was implicitly assumed that all fields would contain single value. For example, all the languages have only one author. What if a language has more than one author? We may try to insert comma separated value or insert values with ‘&’ but this will cause more problems if we were to make queries. We can’t even create fields such as ‘author1’ or ‘author2’ as we don’t beforehand how many authors a language might have. To solve all these problems we split the tables as describe below –

**Splitting the Tables -**

Figure: a table holding author details

| **author\_id** | **Author** | **language\_id** |
| --- | --- | --- |
| 1 | Colmerauer | 1 |
| 2 | Wall | 2 |
| 3 | Ousterhout | 4 |
| 4 | Iverson | 3 |
| 5 | Kemeny | 5 |
| 6 | Kurtz | 5 |

Figure: a table holding programming language details

| **id** | **language** | **year** | **Standard** |
| --- | --- | --- | --- |
| 1 | Prolog | 1972 | ISO |
| 2 | Perl | 1987 | (null) |
| 3 | APL | 1964 | ANSI |
| 4 | Tcl | 1988 | (null) |
| 5 | BASIC | 1964 | ANSI |

### Order of execution of SELECT queries

A query is not evaluated from left to right, there is a specific sequence in which its various parts are evaluated as given below.

1. FROM clause
2. WHERE clause
3. GROUP BY clause
4. HAVING clause
5. SELECT clause
6. ORDER BY clause

**Literal values -**

**sqlite> SELECT language, year, 'AD', 44 FROM mydata;**

language year 'AD' 44

---------- ---------- ---------- ----------

Prolog 1972 AD 44

Perl 1987 AD 44

APL 1964 AD 44

Tcl 1988 AD 44

#works differently in Postgres. It does fill the values like above but doesn’t use these values as column name. Instead it uses ?column? as placeholder column name.

Note – Here WHERE and EXISTS are two different keywords.

**ifnull() function - # not for Postgres**

sqlite> SELECT language, standard, ifnull(standard,'NA') FROM lang;

language standard ifnull(standard,'NA')

---------- ---------- ---------------------

Prolog ISO ISO

Perl NA

APL ISO ISO

Tcl NA

BASIC ANSI ANSI

Pascal ISO ISO

**PostgreSQL**

**SOME DATA TYPES from PostgreSQL**

|  |  |
| --- | --- |
| **NAME** | **DESCRIPTION** |
| character [(n)] | Fixed length character string |
| character varying [(n)] | Variable length character string |
| Data | Calender data (y, m, d) |
| Boolean |  |
| Inet | IPv4 or IPv6 addresses |
| Integer | 4 byte signed integer |
| Json | Textual JSON data |
| Jsonb | Binary JSON data |
| real | Single precision 4 byte floating point number |
| Serial | Autoincrementing 4 bytes integer |
| Text | Variable length character string |
| Point |  |
| Date |  |
|  |  |
|  |  |
|  |  |

**Lessons from** [**www.pgexercises.com**](http://www.pgexercises.com)

**Question:** How can you produce a list of facilities, with each labelled as ‘cheap’ or ‘expensive’ (in a new column named ‘cost’)depending on if their montly maintenance cost is more than 100?

**Hint – Use CASE WHEN…THEN…ELSE…END AS..**

**exercises=# select name, monthlymaintenance from cd.facilities;**

name | monthlymaintenance

-----------------+--------------------

Tennis Court 1 | 200

Tennis Court 2 | 200

Badminton Court | 50

Table Tennis | 10

---snipped output---

(9 rows)

**exercises=# SELECT name, CASE WHEN monthlymaintenance > 100 THEN 'expensive' ELS**

**E 'cheap' END AS cost from cd.facilities;**

name | cost

-----------------+-----------

Tennis Court 1 | expensive

Tennis Court 2 | expensive

Badminton Court | cheap

Table Tennis | cheap

--snipped output---

(9 rows)

**Question:** Print out the list of members who joined after the start of September 2012.

**exercises=# select memid, surname, joindate from cd.members; --this is the data we have**

memid | surname | joindate

-------+-------------------+---------------------

0 | GUEST | 2012-07-01 00:00:00

1 | Smith | 2012-07-02 12:02:05

2 | Smith | 2012-07-02 12:08:23

3 | Rownam | 2012-07-03 09:32:15

4 | Joplette | 2012-07-03 10:25:05

5 | Butters | 2012-07-09 10:44:09

6 | Tracy | 2012-07-15 08:52:55

7 | Dare | 2012-07-25 08:59:12

8 | Boothe | 2012-07-25 16:02:35

9 | Stibbons | 2012-07-25 17:09:05

---snipped output---

**exercises=# select memid, surname, joindate from cd.members WHERE joindate >= '2012-09-01';**

memid | surname | joindate

-------+-------------------+---------------------

24 | Sarwin | 2012-09-01 08:44:42

26 | Jones | 2012-09-02 18:43:05

27 | Rumney | 2012-09-05 08:42:35

28 | Farrell | 2012-09-15 08:22:05

29 | Worthington-Smyth | 2012-09-17 12:27:15

30 | Purview | 2012-09-18 19:04:01

---snipped output---

The datatype is **[**timestamp without time zone ]. Above, we only provided the date part of Timestamp. Postgres automatically converted it to full timestamp 2012-09-01 00:00:00

**Question:** Produce an ordered list of distinct surnames. This list should produce only first 10 results

Hint – See the use of ORDER BY, DISTINCT and LIMIT

**exercises=# SELECT DISTINCT surname from cd.members ORDER BY surname LIMIT 10;**

surname

---------

Bader

Baker

Boothe

Butters

Coplin

Crumpet

Dare

Farrell

Genting

GUEST

(10 rows)

**Qestion:** You want a single combined list of all the members in 2 columns.

HINT - See the use of UNION (and UNION ALL). It combines the result of two SELECT statements in one column.

**exercises=# SELECT surname from cd.members UNION SELECT name from cd.facilities;**

surname

-------------------

Jones

GUEST

Table Tennis

Owen

Snooker Table

Tennis Court 1

---snipped output----

**Question –** Find out the latest joining date.

Hint - use ‘max’j and aggregation

**exercises=# SELECT joindate FROM cd.members WHERE joindate = (SELECT max(joindate) FROM cd.members);**

joindate

---------------------

2012-09-26 18:08:45

(1 row)

Alternatively –

Select max(joindate) as latest from cd.members;

**Question** - How can you produce a list of the start times for bookings for tennis courts, for the date '2012-09-21'? Return a list of start time and facility name pairings, ordered by the time.

select bks.starttime as start, facs.name as name

from

cd.facilities facs

inner join cd.bookings bks

on facs.facid = bks.facid

where

facs.facid in (0,1) and

bks.starttime >= '2012-09-21' and

bks.starttime < '2012-09-22'

order by bks.starttime;

**Question** - How can you output a list of all members who have recommended another member? Ensure that there are no duplicates in the list, and that results are ordered by (surname, firstname).

**SELECT DISTINCT mem2.firstname, mem2.surname FROM cd.members mem1 INNER JOIN cd.members mem2 on mem2.memid = mem1.recommendedby ORDER BY mem2.surname, mem2.firstname;**