Fascinating Reporting with Postgres psql and sendmail

Presented by Christopher L. Augustus
https://orcid.org/0000-0001-7297-2325

PostgresOpen 2019, Orlando, Florida
2:30 PM, Thursday, September 12, 2019
Salons 13-15

https://www.osti.gov/servlets/purl/1560062

About the Speaker

Tennessee Blue Book 2017-2018, page 685

IIA Information International Associates

https://www.iiaweb.com/

U.S. DEPARTMENT OF ENERGY | Office of Science | Office of Scientific and Technical Information

https://www.osti.gov

RFETN Radio Free East Tennessee PODCAST

https://archive.org/search.php?query=subject%3A%22RFETN%22&sort=-publicdate

The Plan

④ Automated Reporting System

③ UNIX/Linux sendmail

② Postgres SQL

① Postgres psql

Two Development Theories

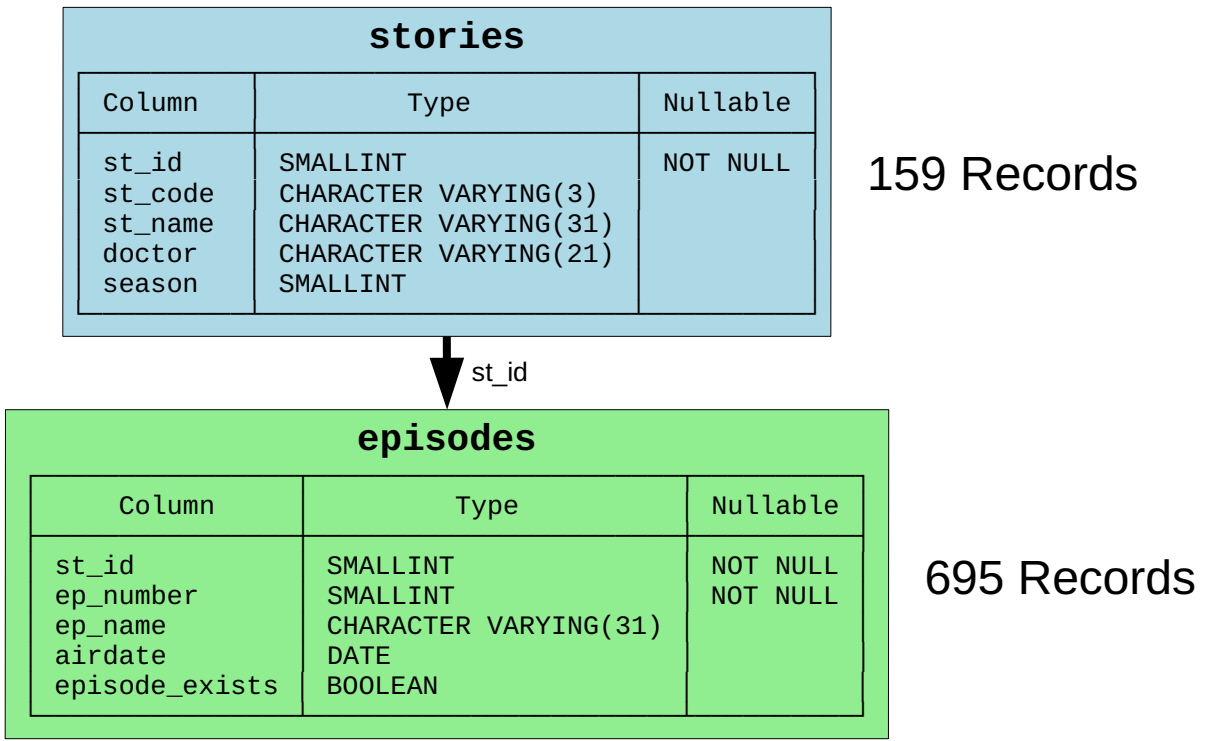
Paul McCartney

– or –

John Lennon

Simple Doctor Who Database

- Simple schema "dw" of two tables "stories" (every classic Doctor Who story intended for broadcast between 1963 and 1989) and "episodes" (every broadcast episode).
- A script to create the "dw" schema and the two tables can be found here:
  - http://www.knology.net/~augustus/presentations/build_dw.txt



① Postgres psql

What is psql?

- Official Webpage: https://www.postgresql.org/docs/current/app-psql.html
- PostgreSQL interactive terminal (psql)
  - A terminal-based front-end to Postgres.
  - Type in queries interactively, issue them to Postgres, and see the query results.
  - Alternatively, input can be from a file or from command line arguments.
  - Provides a number of meta-commands and various shell-like features to facilitate writing scripts and automating a wide variety of tasks.

One Way to Connect to Postgres with psql

- Setup a ".pgpass" file.
  - https://www.postgresql.org/docs/current/libpq-pgpass.html
- Set four environment variables:

```
export PGDATABASE=postgres
export PGHOST=localhost
export PGPORT=5432
export PGUSER=chris
```

Default psql Settings

Example of a simple query with all default settings.

```
SELECT   *
FROM     dw.stories
WHERE    season = 1
ORDER BY st_id;
 st_id | st_code |         st_name          |      doctor      | season
-------+---------+--------------------------+------------------+--------
     1 | A       | An Unearthly Child       | William Hartnell |      1
     2 | B       | The Daleks               | William Hartnell |      1
     3 | C       | The Edge of Destruction  | William Hartnell |      1
     4 | D       | Marco Polo               | William Hartnell |      1
     5 | E       | The Keys of Marinus      | William Hartnell |      1
     6 | F       | The Aztecs               | William Hartnell |      1
     7 | G       | The Sensorites           | William Hartnell |      1
     8 | H       | The Reign of Terror      | William Hartnell |      1
(8 rows)
```

psql Meta Commands

https://www.postgresql.org/docs/current/app-psql.html#APP-PSQL-META-COMMANDS

- Always start with a backslash \.
- Postgres 11 has 56 major backslash commands.
- Quick overview of some meta commands:
  - \q to cleanly exit psql.
  - \h or \help for getting help on SQL commands.
  - \d useful for showing details of database objects.
  - \echo for outputting strings and variables.
  - \copy is an awesome way to get data into and out of Postgres!!!
  - \if for conditional statements – new in Postgres 11.
  - \set is useful for setting and viewing psql variables.
  - \pset is useful for formatting query output.

\pset

- Sets the formatting of query output tables.
- In Postgres 11, typing \pset on a default configured psql returns:

```
border                 1
columns                0
expanded               off
fieldsep               '|'
fieldsep_zero          off
footer                 on
format                 aligned
linestyle              ascii
null                   ''
numericlocale          off
pager                  1
pager_min_lines        0
recordsep              '\n'
recordsep_zero         off
tableattr
title
tuples_only            off
unicode_border_linestyle single
unicode_column_linestyle single
unicode_header_linestyle single
```

\pset linestyle ascii & \pset border 0

Query with a border of zero.

```
\pset linestyle ascii
\pset border 0
SELECT   *
FROM     dw.stories
WHERE    season = 1
ORDER BY st_id;
st_id st_code         st_name              doctor      season
----- ------- ---------------------- ---------------- ------
    1 A       An Unearthly Child     William Hartnell      1
    2 B       The Daleks             William Hartnell      1
    3 C       The Edge of Destruction William Hartnell      1
    4 D       Marco Polo             William Hartnell      1
    5 E       The Keys of Marinus    William Hartnell      1
    6 F       The Aztecs             William Hartnell      1
    7 G       The Sensorites         William Hartnell      1
    8 H       The Reign of Terror    William Hartnell      1
(8 rows)
```

\pset linestyle ascii & \pset border 1

Query with a border of one which is the default.

```
\pset linestyle ascii
\pset border 1
SELECT   *
FROM     dw.stories
WHERE    season = 1
ORDER BY st_id;
 st_id | st_code |         st_name          |      doctor      | season
-------+---------+--------------------------+------------------+--------
     1 | A       | An Unearthly Child       | William Hartnell |      1
     2 | B       | The Daleks               | William Hartnell |      1
     3 | C       | The Edge of Destruction  | William Hartnell |      1
     4 | D       | Marco Polo               | William Hartnell |      1
     5 | E       | The Keys of Marinus      | William Hartnell |      1
     6 | F       | The Aztecs               | William Hartnell |      1
     7 | G       | The Sensorites           | William Hartnell |      1
     8 | H       | The Reign of Terror      | William Hartnell |      1
(8 rows)
```

\pset linestyle ascii & \pset border 2

Query with a border of two.

```
\pset linestyle ascii
\pset border 2
SELECT   *
FROM     dw.stories
WHERE    season = 1
ORDER BY st_id;
+-------+---------+-------------------------+------------------+--------+
| st_id | st_code |         st_name         |      doctor      | season |
+-------+---------+-------------------------+------------------+--------+
|     1 | A       | An Unearthly Child      | William Hartnell |      1 |
|     2 | B       | The Daleks              | William Hartnell |      1 |
|     3 | C       | The Edge of Destruction | William Hartnell |      1 |
|     4 | D       | Marco Polo              | William Hartnell |      1 |
|     5 | E       | The Keys of Marinus     | William Hartnell |      1 |
|     6 | F       | The Aztecs              | William Hartnell |      1 |
|     7 | G       | The Sensorites          | William Hartnell |      1 |
|     8 | H       | The Reign of Terror     | William Hartnell |      1 |
+-------+---------+-------------------------+------------------+--------+
(8 rows)
```

\pset linestyle unicode

- \pset linestyle has three options:
  - ascii – displays the borders with ASCII7 characters.
  - old-ascii – for compatibility to 8.4 and earlier.
  - unicode – displays the boarders with Unicode box drawing characters.

\pset linestyle unicode & \pset border 0

Query with a border of zero.

```
\pset linestyle unicode
\pset border 0
SELECT   *
FROM     dw.stories
WHERE    season = 1
ORDER BY st_id;
st_id st_code         st_name              doctor      season
═════ ═══════ ═══════════════════════ ═══════════════ ══════
    1 A       An Unearthly Child      William Hartnell      1
    2 B       The Daleks              William Hartnell      1
    3 C       The Edge of Destruction William Hartnell      1
    4 D       Marco Polo              William Hartnell      1
    5 E       The Keys of Marinus     William Hartnell      1
    6 F       The Aztecs              William Hartnell      1
    7 G       The Sensorites          William Hartnell      1
    8 H       The Reign of Terror     William Hartnell      1
(8 rows)
```

\pset linestyle unicode & \pset border 1

Query with a border of one.

```
\pset linestyle unicode
\pset border 1
SELECT   *
FROM     dw.stories
WHERE    season = 1
ORDER BY st_id;
 st_id │ st_code │         st_name          │      doctor       │ season
═══════╪═════════╪══════════════════════════╪═══════════════════╪════════
     1 │ A       │ An Unearthly Child       │ William Hartnell  │      1
     2 │ B       │ The Daleks               │ William Hartnell  │      1
     3 │ C       │ The Edge of Destruction  │ William Hartnell  │      1
     4 │ D       │ Marco Polo               │ William Hartnell  │      1
     5 │ E       │ The Keys of Marinus      │ William Hartnell  │      1
     6 │ F       │ The Aztecs               │ William Hartnell  │      1
     7 │ G       │ The Sensorites           │ William Hartnell  │      1
     8 │ H       │ The Reign of Terror      │ William Hartnell  │      1
(8 rows)
```

\pset linestyle unicode & \pset border 2

Query with a border of two.

```
\pset linestyle unicode
\pset border 2
SELECT   *
FROM     dw.stories
WHERE    season = 1
ORDER BY st_id;
```

st_id	st_code	st_name	doctor	season
1	A	An Unearthly Child	William Hartnell	1
2	B	The Daleks	William Hartnell	1
3	C	The Edge of Destruction	William Hartnell	1
4	D	Marco Polo	William Hartnell	1
5	E	The Keys of Marinus	William Hartnell	1
6	F	The Aztecs	William Hartnell	1
7	G	The Sensorites	William Hartnell	1
8	H	The Reign of Terror	William Hartnell	1

(8 rows)

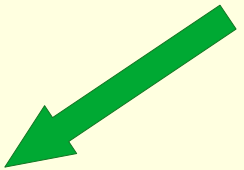
\pset title & \pset footer & \pset tuples_only

- \pset title puts a title above the tables.
  - If title is more than one word, enclose the title with single quotes.
  - To remove the title, do not pass any values.
  - Once set, a title will show up on every table until changed or removed!
  - \C is a shortcut.
- \pset footer turns on or off the "(x rows)" display at the bottom of a table.
- \pset tuples_only turns on viewing just the rows or rows and headers.
  - \t is a shortcut.

\pset title

Give the table a title.

```
\pset linestyle unicode
\pset border 2
\pset title 'Doctor Who Season One'
SELECT   *
FROM     dw.stories
WHERE    season = 1
ORDER BY st_id;
```



Doctor Who Season One

st_id	st_code	st_name	doctor	season
1	A	An Unearthly Child	William Hartnell	1
2	B	The Daleks	William Hartnell	1
3	C	The Edge of Destruction	William Hartnell	1
4	D	Marco Polo	William Hartnell	1
5	E	The Keys of Marinus	William Hartnell	1
6	F	The Aztecs	William Hartnell	1
7	G	The Sensorites	William Hartnell	1
8	H	The Reign of Terror	William Hartnell	1

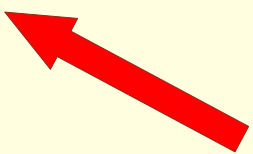
(8 rows)

\pset footer

Removes the footer.

```
\pset linestyle unicode
\pset border 2
\pset footer off
SELECT   *
FROM     dw.stories
WHERE    season = 1
ORDER BY st_id;
```

st_id	st_code	st_name	doctor	season
1	A	An Unearthly Child	William Hartnell	1
2	B	The Daleks	William Hartnell	1
3	C	The Edge of Destruction	William Hartnell	1
4	D	Marco Polo	William Hartnell	1
5	E	The Keys of Marinus	William Hartnell	1
6	F	The Aztecs	William Hartnell	1
7	G	The Sensorites	William Hartnell	1
8	H	The Reign of Terror	William Hartnell	1



\pset tuples_only

Show just the rows of the table. (Also set the boarder to zero.)

```
\pset border 0
\pset tuples_only on
SELECT   *
FROM     dw.stories
WHERE    season = 1
ORDER BY st_id;
    1 A       An Unearthly Child      William Hartnell      1
    2 B       The Daleks              William Hartnell      1
    3 C       The Edge of Destruction William Hartnell      1
    4 D       Marco Polo              William Hartnell      1
    5 E       The Keys of Marinus     William Hartnell      1
    6 F       The Aztecs              William Hartnell      1
    7 G       The Sensorites          William Hartnell      1
    8 H       The Reign of Terror     William Hartnell      1
```

\pset format

- Formats the output in one of eight major formats. So far all examples have been aligned. Possible values are:
  - unaligned
  - aligned
  - wrapped
  - html
  - asciidoc
  - latex
  - latex-longtable
  - troff-ms

\pset format asciidoc

Added in Postgres 9.5.

```
\pset format asciidoc
\pset title 'Doctor Who Season One'
SELECT   *
FROM     dw.stories
WHERE    season = 1
ORDER BY st_id;
.Doctor Who Season One
[options="header",cols=">l,<l,<l,<l,>l",frame="none"]
|====
^l|st_id ^l|st_code ^l|st_name ^l|doctor ^l|season
|1 |A |An Unearthly Child |William Hartnell |1
|2 |B |The Daleks |William Hartnell |1
|3 |C |The Edge of Destruction |William Hartnell |1
|4 |D |Marco Polo |William Hartnell |1
|5 |E |The Keys of Marinus |William Hartnell |1
|6 |F |The Aztecs |William Hartnell |1
|7 |G |The Sensorites |William Hartnell |1
|8 |H |The Reign of Terror |William Hartnell |1
|====

....
(8 rows)
....
```

\pset format unaligned

By default, unaligned data is pipe delimited. Can be configured.

```
\pset format unaligned
\pset title 'Doctor Who Season One'
SELECT   *
FROM     dw.stories
WHERE    season = 1
ORDER BY st_id;
Doctor Who Season One
st_id|st_code|st_name|doctor|season
1|A|An Unearthly Child|William Hartnell|1
2|B|The Daleks|William Hartnell|1
3|C|The Edge of Destruction|William Hartnell|1
4|D|Marco Polo|William Hartnell|1
5|E|The Keys of Marinus|William Hartnell|1
6|F|The Aztecs|William Hartnell|1
7|G|The Sensorites|William Hartnell|1
8|H|The Reign of Terror|William Hartnell|1
(8 rows)
```

\pset fieldsep (defaults to the pipe)
\pset recordsep (defaults to a newline)

\pset format html

Produces code for HTML table. Great format for e-mailed reports.

```
\pset format html
\pset title 'Doctor Who Season One'
SELECT   *
FROM     dw.stories
WHERE    season = 1
ORDER BY st_id;
<table border="1">
  <caption>Doctor Who Season One</caption>
  <tr>
    <th align="center">st_id</th>
    <th align="center">st_code</th>
    <th align="center">st_name</th>
    <th align="center">doctor</th>
    <th align="center">season</th>
  </tr>
  <tr valign="top">
    <td align="right">1</td>
    <td align="left">A</td>
    <td align="left">An Unearthly Child</td>
    <td align="left">William Hartnell</td>
    <td align="right">1</td>
  </tr>
  <tr valign="top">
    <td align="right">2</td>
    <td align="left">B</td>
    <td align="left">The Daleks</td>
    <td align="left">William Hartnell</td>
    <td align="right">1</td>
```

```
...
  <tr valign="top">
    <td align="right">6</td>
    <td align="left">F</td>
    <td align="left">The Aztecs</td>
    <td align="left">William Hartnell</td>
    <td align="right">1</td>
  </tr>
  <tr valign="top">
    <td align="right">7</td>
    <td align="left">G</td>
    <td align="left">The Sensorites</td>
    <td align="left">William Hartnell</td>
    <td align="right">1</td>
  </tr>
  <tr valign="top">
    <td align="right">8</td>
    <td align="left">H</td>
    <td align="left">The Reign of Terror</td>
    <td align="left">William Hartnell</td>
    <td align="right">1</td>
  </tr>
</table>
<p>(8 rows)<br />
</p>
```

\pset format html

Example of the output as an HTML table.

Doctor Who Season One

st_id	st_code	st_name	doctor	season
1	A	An Unearthly Child	William Hartnell	1
2	B	The Daleks	William Hartnell	1
3	C	The Edge of Destruction	William Hartnell	1
4	D	Marco Polo	William Hartnell	1
5	E	The Keys of Marinus	William Hartnell	1
6	F	The Aztecs	William Hartnell	1
7	G	The Sensorites	William Hartnell	1
8	H	The Reign of Terror	William Hartnell	1

(8 rows)

\pset format latex

Produces code in LaTeX format. How do you say LaTeX?

```
\pset format latex
\pset title 'Doctor Who Season One'
SELECT   *
FROM     dw.stories
WHERE    season = 1
ORDER BY st_id;
\begin{center}
Doctor Who Season One
\end{center}

\begin{tabular}{r | l | l | l | r}
\textit{st\_id} & \textit{st\_code} & \textit{st\_name} & \textit{doctor} & \textit{season} \\
\hline
1 & A & An Unearthly Child & William Hartnell & 1 \\
2 & B & The Daleks & William Hartnell & 1 \\
3 & C & The Edge of Destruction & William Hartnell & 1 \\
4 & D & Marco Polo & William Hartnell & 1 \\
5 & E & The Keys of Marinus & William Hartnell & 1 \\
6 & F & The Aztecs & William Hartnell & 1 \\
7 & G & The Sensorites & William Hartnell & 1 \\
8 & H & The Reign of Terror & William Hartnell & 1 \\
\end{tabular}

\noindent (8 rows) \\
```

\pset format troff-ms

An output format dating back to the 1960s!

```
\pset format troff-ms
\pset title 'Doctor Who Season One'
SELECT   *
FROM     dw.stories
WHERE    season = 1
ORDER BY st_id;
.LP
.DS C
Doctor Who Season One
.DE
.LP
.TS
center;
r | l | l | l | r.
\fIst_id\fP     \fIst_code\fP   \fIst_name\fP   \fIdoctor\fP    \fIseason\fP
_
1       A       An Unearthly Child      William Hartnell        1
2       B       The Daleks      William Hartnell        1
3       C       The Edge of Destruction William Hartnell        1
4       D       Marco Polo      William Hartnell        1
5       E       The Keys of Marinus     William Hartnell        1
6       F       The Aztecs      William Hartnell        1
7       G       The Sensorites  William Hartnell        1
8       H       The Reign of Terror     William Hartnell        1
.TE
.DS L
(8 rows)
.DE
```

\pset expanded

- Expanded can be on or off.
  - If on, expanded displays data in two columns.
    - The first column contains the field names.
    - The second column contains the field values.
  - Expanded display is useful for displaying data from very wide tables.
    - Shortcut \x

\pset expanded

```
\pset expanded off
SELECT   st_id,st_name
FROM     dw.stories
WHERE    season = 1
ORDER BY st_id;
 st_id |         st_name
-------+-------------------------
     1 | An Unearthly Child
     2 | The Daleks
     3 | The Edge of Destruction
     4 | Marco Polo
     5 | The Keys of Marinus
     6 | The Aztecs
     7 | The Sensorites
     8 | The Reign of Terror
(8 rows)
```

```
\pset expanded on
SELECT   st_id,st_name
FROM     dw.stories
WHERE    season = 1
ORDER BY st_id;
-[ RECORD 1 ]--------------------
st_id   | 1
st_name | An Unearthly Child
-[ RECORD 2 ]--------------------
st_id   | 2
st_name | The Daleks
-[ RECORD 3 ]--------------------
st_id   | 3
st_name | The Edge of Destruction
-[ RECORD 4 ]--------------------
st_id   | 4
st_name | Marco Polo
-[ RECORD 5 ]--------------------
st_id   | 5
st_name | The Keys of Marinus
-[ RECORD 6 ]--------------------
st_id   | 6
st_name | The Aztecs
-[ RECORD 7 ]--------------------
st_id   | 7
st_name | The Sensorites
-[ RECORD 8 ]--------------------
st_id   | 8
st_name | The Reign of Terror
```

\copy

Very powerful meta-command! Example outputting a CSV file.

```
-- Output to the screen.

\copy (SELECT * FROM dw.stories WHERE season = 1 ORDER BY st_id) TO STDOUT CSV HEADER;
st_id,st_code,st_name,doctor,season
1,A,An Unearthly Child,William Hartnell,1
2,B,The Daleks,William Hartnell,1
3,C,The Edge of Destruction,William Hartnell,1
4,D,Marco Polo,William Hartnell,1
5,E,The Keys of Marinus,William Hartnell,1
6,F,The Aztecs,William Hartnell,1
7,G,The Sensorites,William Hartnell,1
8,H,The Reign of Terror,William Hartnell,1
```

```
-- Output to a file from the machine where psql is running.

\copy (SELECT * FROM dw.stories WHERE season = 1 ORDER BY st_id) TO 'dw.csv' CSV HEADER;
COPY 8
```

Can output in three formats: CSV, text, and binary.

② Postgres SQL

Explore SQL

- SQL Command Reference:
  - https://www.postgresql.org/docs/current/sql-commands.html
- SELECT Statement Reference:
  - https://www.postgresql.org/docs/current/sql-select.html

Change the Column Headings

Put friendly column headings inside double quotes.

```
SELECT   st_id "#",
         st_name "Story Name"
FROM     dw.stories
WHERE    season = 1
ORDER BY "#";
 # |       Story Name
---+-------------------------
 1 | An Unearthly Child
 2 | The Daleks
 3 | The Edge of Destruction
 4 | Marco Polo
 5 | The Keys of Marinus
 6 | The Aztecs
 7 | The Sensorites
 8 | The Reign of Terror
(8 rows)
```

Subqueries in SELECT Clause

A subquery can be used to count the number of detail records.


```
SELECT   st_id "#",
         st_name "Story Name",
         (SELECT COUNT(*)
          FROM   dw.episodes e
          WHERE  e.st_id = s.st_id) "Episode Count"
FROM     dw.stories s
WHERE    season = 1
ORDER BY "#";
 # |       Story Name        | Episode Count
---+-------------------------+---------------
 1 | An Unearthly Child      |             4
 2 | The Daleks              |             7
 3 | The Edge of Destruction |             2
 4 | Marco Polo              |             7
 5 | The Keys of Marinus     |             6
 6 | The Aztecs              |             4
 7 | The Sensorites          |             6
 8 | The Reign of Terror     |             6
(8 rows)
```

Groups

Group by ID and name and use the aggregate function COUNT.

```
SELECT   s.st_id "#",
         st_name "Story Name",
         COUNT(*) "Episode Count"
FROM     dw.stories s JOIN dw.episodes e ON (s.st_id = e.st_id)
WHERE    season = 1
GROUP BY "#","Story Name"
ORDER BY "#";
 # |       Story Name         | Episode Count
---+--------------------------+---------------
 1 | An Unearthly Child       |             4
 2 | The Daleks               |             7
 3 | The Edge of Destruction  |             2
 4 | Marco Polo               |             7
 5 | The Keys of Marinus      |             6
 6 | The Aztecs               |             4
 7 | The Sensorites           |             6
 8 | The Reign of Terror      |             6
(8 rows)
```

USING

The USING (field) keyword cuts down on typing ON (field1 = field2).

```
SELECT   st_id "#",
         st_name "Story Name",
         COUNT(*) "Episode Count"
FROM     dw.stories JOIN dw.episodes USING (st_id)
WHERE    season = 1
GROUP BY "#","Story Name"
ORDER BY "#";
 # |       Story Name        | Episode Count
---+-------------------------+---------------
 1 | An Unearthly Child      |             4
 2 | The Daleks              |             7
 3 | The Edge of Destruction |             2
 4 | Marco Polo              |             7
 5 | The Keys of Marinus     |             6
 6 | The Aztecs              |             4
 7 | The Sensorites          |             6
 8 | The Reign of Terror     |             6
(8 rows)
```

Know Your Data!

- It is important to know how clean or dirty your data is.
- But what if there was a story without any episodes?

Subqueries

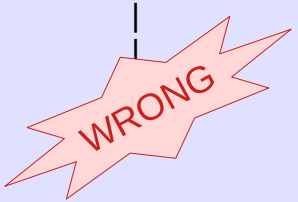
There are no episode detail records for Shada. This is correct.

```
SELECT   st_id "#",
         st_name "Story Name",
         (SELECT COUNT(*)
          FROM   dw.episodes e
          WHERE  e.st_id = s.st_id) "Episode Count"
FROM     dw.stories s
WHERE    season = 17
ORDER BY "#";
  #  |        Story Name          | Episode Count
-----+----------------------------+---------------
 104 | Destiny of the Daleks      |             4
 105 | City of Death              |             4
 106 | The Creature from the Pit  |             4
 107 | Nightmare of Eden          |             4
 108 | Horns of Nimon             |             4
 109 | Shada                      |             0
(6 rows)
```

JOIN

A normal join leaves off Shada.

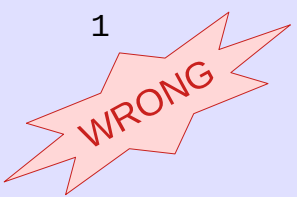
```
SELECT   st_id "#",
         st_name "Story Name",
         COUNT(*) "Episode Count"
FROM     dw.stories JOIN dw.episodes USING (st_id)
WHERE    season = 17
GROUP BY "#","Story Name"
ORDER BY "#";
  #  |        Story Name          | Episode Count
-----+----------------------------+---------------
 104 | Destiny of the Daleks      |             4
 105 | City of Death              |             4
 106 | The Creature from the Pit  |             4
 107 | Nightmare of Eden          |             4
 108 | Horns of Nimon             |             4
(5 rows)
```



OUTER JOIN

The outer join included Shada. But Shada shows one episode?

```
SELECT   st_id "#",
         st_name "Story Name",
         COUNT(*) "Episode Count"
FROM     dw.stories LEFT OUTER JOIN dw.episodes USING (st_id)
WHERE    season = 17
GROUP BY "#","Story Name"
ORDER BY "#";
  #  |        Story Name          | Episode Count
-----+----------------------------+---------------
 104 | Destiny of the Daleks      |             4
 105 | City of Death              |             4
 106 | The Creature from the Pit  |             4
 107 | Nightmare of Eden          |             4
 108 | Horns of Nimon             |             4
 109 | Shada                      |             1
(6 rows)
```



COUNT(field)

Counting the non NULL ep_number fields now returns zero.

```
SELECT   st_id "#",
         st_name "Story Name",
         COUNT(ep_number) "Episode Count"
FROM     dw.stories LEFT OUTER JOIN dw.episodes USING (st_id)
WHERE    season = 17
GROUP BY "#","Story Name"
ORDER BY "#";
  #  |        Story Name          | Episode Count
-----+----------------------------+---------------
 104 | Destiny of the Daleks      |             4
 105 | City of Death              |             4
 106 | The Creature from the Pit  |             4
 107 | Nightmare of Eden          |             4
 108 | Horns of Nimon             |             4
 109 | Shada                      |             0
(6 rows)
```

Sorting Results

Join the season 17 stories to episodes and sort the results.

```
SELECT   st_id "#",
         st_name "Story Name",
         ep_name "Episode Name"
FROM     dw.stories LEFT OUTER JOIN dw.episodes USING (st_id)
WHERE    season = 17
ORDER BY "#","Episode Name";
  #  |        Story Name          | Episode Name
-----+----------------------------+--------------
 104 | Destiny of the Daleks      | Part Four
 104 | Destiny of the Daleks      | Part One
 104 | Destiny of the Daleks      | Part Three
 104 | Destiny of the Daleks      | Part Two
 105 | City of Death              | Part Four
 105 | City of Death              | Part One
 105 | City of Death              | Part Three
 105 | City of Death              | Part Two
...
 108 | Horns of Nimon             | Part Four
 108 | Horns of Nimon             | Part One
 108 | Horns of Nimon             | Part Three
 108 | Horns of Nimon             | Part Two
 109 | Shada                      |
(21 rows)
```

Sorting Results

Episodes are sorted correctly.

```
SELECT   st_id "#",
         st_name "Story Name",
         ep_name "Episode Name"
FROM     dw.stories LEFT OUTER JOIN dw.episodes USING (st_id)
WHERE    season = 17
ORDER BY "#",ep_number;
  #  |        Story Name          | Episode Name
-----+----------------------------+--------------
 104 | Destiny of the Daleks      | Part One
 104 | Destiny of the Daleks      | Part Two
 104 | Destiny of the Daleks      | Part Three
 104 | Destiny of the Daleks      | Part Four
 105 | City of Death              | Part One
 105 | City of Death              | Part Two
 105 | City of Death              | Part Three
 105 | City of Death              | Part Four
...
 108 | Horns of Nimon             | Part One
 108 | Horns of Nimon             | Part Two
 108 | Horns of Nimon             | Part Three
 108 | Horns of Nimon             | Part Four
 109 | Shada                      |
(21 rows)
```

Arrays

- Nine years ago, I wondered who on earth needed arrays in a relational database.
- This was before discovering the rich set of functions in Postgres to handle arrays.
- The use of array fields in tables at OSTI are rare.
- Using arrays in reports has become quite common.
- For reference, see aggregate functions documentation:
  - https://www.postgresql.org/docs/current/functions-aggregate.html

ARRAY_AGG

Episodes are accurately shown as an array. Can it be prettied up?

```
SELECT   st_id "#",
         st_name "Story Name",
         ARRAY_AGG (ep_name ORDER BY ep_number) "Episode Names"
FROM     dw.stories LEFT OUTER JOIN dw.episodes USING (st_id)
WHERE    season = 17
GROUP BY "#","Story Name"
ORDER BY "#";
  #  |        Story Name         |                  Episode Names
-----+---------------------------+-------------------------------------------------
 104 | Destiny of the Daleks     | {"Part One","Part Two","Part Three","Part Four"}
 105 | City of Death             | {"Part One","Part Two","Part Three","Part Four"}
 106 | The Creature from the Pit | {"Part One","Part Two","Part Three","Part Four"}
 107 | Nightmare of Eden         | {"Part One","Part Two","Part Three","Part Four"}
 108 | Horns of Nimon            | {"Part One","Part Two","Part Three","Part Four"}
 109 | Shada                     | {NULL}
(6 rows)
```

ARRAY_TO_STRING

Episodes are now shown delimited by a semicolon and a space.

```
SELECT   st_id "#",
         st_name "Story Name",
         ARRAY_TO_STRING (ARRAY_AGG (ep_name ORDER BY ep_number),'; ') "Episode Names"
FROM     dw.stories LEFT OUTER JOIN dw.episodes USING (st_id)
WHERE    season = 17
GROUP BY "#","Story Name"
ORDER BY "#";
  #  |        Story Name         |                Episode Names
-----+---------------------------+-----------------------------------------
 104 | Destiny of the Daleks     | Part One; Part Two; Part Three; Part Four
 105 | City of Death             | Part One; Part Two; Part Three; Part Four
 106 | The Creature from the Pit | Part One; Part Two; Part Three; Part Four
 107 | Nightmare of Eden         | Part One; Part Two; Part Three; Part Four
 108 | Horns of Nimon            | Part One; Part Two; Part Three; Part Four
 109 | Shada                     |
(6 rows)
```

Temporary Table

Select the ID, story name, and an array of episode names.

```
SELECT   st_id,
         st_name,
         ARRAY_AGG (ep_name ORDER BY ep_number) ep_names
FROM     dw.stories LEFT OUTER JOIN dw.episodes USING (st_id)
GROUP BY st_id,st_name
ORDER BY st_id;
 st_id |            st_name             |                                                                                                    ep_names
-------+--------------------------------+-----------------------------------------------------------------------------------------------------------------------------
     1 | An Unearthly Child             | {"An Unearthly Child","The Cave of Skulls","The Forest of Fear","The Firemaker"}
     2 | The Daleks                     | {"The Dead Planet","The Survivors","The Escape","The Ambush","The Expedition","The Ordeal","The Rescue"}
     3 | The Edge of Destruction        | {"The Edge of Destruction","The Brink of Disaster"}
...
   143 | Revelation of the Daleks       | {"Part One","Part Two"}
   144 | The Mysterious Planet          | {"Part One","Part Two","Part Three","Part Four"}
   145 | Mindwarp                       | {"Part Five","Part Six","Part Seven","Part Eight"}
   146 | Terror of the Vervoids         | {"Part Nine","Part Ten","Part Eleven","Part Twelve"}
   147 | The Ultimate Foe               | {"Part Thirteen","Part Fourteen"}
   148 | Time and the Rani              | {"Part One","Part Two","Part Three","Part Four"}
   149 | Paradise Towers                | {"Part One","Part Two","Part Three","Part Four"}
   150 | Delta and the Bannermen        | {"Part One","Part Two","Part Three"}
   151 | Dragonfire                     | {"Part One","Part Two","Part Three"}
   152 | Remembrance of the Daleks      | {"Part One","Part Two","Part Three","Part Four"}
   153 | The Happiness Patrol           | {"Part One","Part Two","Part Three"}
   154 | Silver Nemesis                 | {"Part One","Part Two","Part Three"}
   155 | The Greatest Show in the Galaxy | {"Part One","Part Two","Part Three","Part Four"}
   156 | Battlefield                    | {"Part One","Part Two","Part Three","Part Four"}
   157 | Ghost Light                    | {"Part One","Part Two","Part Three"}
   158 | The Curse of Fenric            | {"Part One","Part Two","Part Three","Part Four"}
   159 | Survival                       | {"Part One","Part Two","Part Three"}
(159 rows)
```

CREATE TEMPORARY TABLE

Creates a temporary table that only lasts as long as the session.

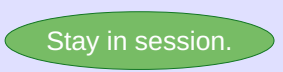
```
CREATE TEMPORARY TABLE stories_episodes AS
SELECT   st_id,
         st_name,
         ARRAY_AGG (ep_name ORDER BY ep_number) ep_names
FROM     dw.stories LEFT OUTER JOIN dw.episodes USING (st_id)
GROUP BY st_id,st_name
ORDER BY st_id;

SELECT   *
FROM     stories_episodes
ORDER BY st_id;
 st_id |           st_name            |                                                                                ep_names
-------+------------------------------+-------------------------------------------------------------------------------------------------------------------------
     1 | An Unearthly Child           | {"An Unearthly Child","The Cave of Skulls","The Forest of Fear","The Firemaker"}
...
   159 | Survival                     | {"Part One","Part Two","Part Three"}
(159 rows)

\d stories_episodes
                Table "pg_temp_3.stories_episodes"
  Column  |         Type          | Collation | Nullable | Default
----------+-----------------------+-----------+----------+---------
 st_id    | smallint              |           |          |
 st_name  | character varying(31) |           |          |
 ep_names | character varying[]   |           |          |
```

Stay in session.

UNNEST

Takes an array and makes separate rows for each element.

```
SELECT   st_id "#",
         st_name "Story",
         UNNEST (ep_names) "Episodes"
FROM     stories_episodes
WHERE    st_id IN (1,2,3)
ORDER BY st_id;
 # |          Story          |        Episodes
---+-------------------------+-------------------------
 1 | An Unearthly Child      | An Unearthly Child
 1 | An Unearthly Child      | The Cave of Skulls
 1 | An Unearthly Child      | The Forest of Fear
 1 | An Unearthly Child      | The Firemaker
 2 | The Daleks              | The Dead Planet
 2 | The Daleks              | The Survivors
 2 | The Daleks              | The Escape
 2 | The Daleks              | The Ambush
 2 | The Daleks              | The Expedition
 2 | The Daleks              | The Ordeal
 2 | The Daleks              | The Rescue
 3 | The Edge of Destruction | The Edge of Destruction
 3 | The Edge of Destruction | The Brink of Disaster
(13 rows)
```



UNNEST

See what happens with an empty array.

```
SELECT   st_id "#",
         st_name "Story",
         UNNEST (ep_names) "Episodes"
FROM     stories_episodes
WHERE    st_id IN (108,109,110)
ORDER BY st_id;
  #  |      Story       |  Episodes
-----+------------------+------------
 108 | Horns of Nimon   | Part One
 108 | Horns of Nimon   | Part Two
 108 | Horns of Nimon   | Part Three
 108 | Horns of Nimon   | Part Four
 109 | Shada            |
 110 | The Leisure Hive | Part One
 110 | The Leisure Hive | Part Two
 110 | The Leisure Hive | Part Three
 110 | The Leisure Hive | Part Four
(9 rows)
```

End session.

Create a New Temporary Table

This time the episode names column is semicolon delimited.

```
CREATE TEMPORARY TABLE new_stories_episodes AS
SELECT   st_id,
         st_name,
         ARRAY_TO_STRING (ARRAY_AGG (ep_name ORDER BY ep_number),'; ') ep_names
FROM     dw.stories LEFT OUTER JOIN dw.episodes USING (st_id)
GROUP BY st_id,st_name
ORDER BY st_id;

SELECT   *
FROM     new_stories_episodes
ORDER BY st_id;
 st_id |            st_name             |                                ep_names
-------+--------------------------------+-------------------------------------------------------------------------
     1 | An Unearthly Child             | An Unearthly Child; The Cave of Skulls; The Forest of Fear; The Firemaker
...
   158 | The Curse of Fenric            | Part One; Part Two; Part Three; Part Four
   159 | Survival                       | Part One; Part Two; Part Three
(159 rows)
```

Stay in session.

STRING_TO_ARRAY

Convert the semicolon delimited string into an array.

```
SELECT   st_id "#",
         st_name "Story",
         ep_names "Episodes",
         STRING_TO_ARRAY (ep_names,'; ') "Episodes Array"
FROM     new_stories_episodes
WHERE    st_id IN (108,109,110)
ORDER BY st_id;
  #  |     Story       |                Episodes                 |                  Episodes Array
-----+-----------------+-----------------------------------------+-------------------------------------------------
 108 | Horns of Nimon  | Part One; Part Two; Part Three; Part Four | {"Part One","Part Two","Part Three","Part Four"}
 109 | Shada           |                                         | {}
 110 | The Leisure Hive | Part One; Part Two; Part Three; Part Four | {"Part One","Part Two","Part Three","Part Four"}
(3 rows)
```

End session.

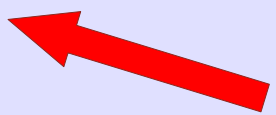
DISTINCT and DISTINCT ON

- For reference:
  - https://www.postgresql.org/docs/current/queries-select-lists.html#QUERIES-DISTINCT
  - https://www.postgresql.org/docs/current/sql-select.html
- DISTINCT is part of the SQL standard.
  - Very useful to show unique or distinct values only in a query.
- DISTINCT ON () is not part of the SQL standard.
  - Can be used as a shortcut of very basic windowing functions.

DISTINCT

Shows one record for each unique or distinct episode name.

```
SELECT   DISTINCT
         ep_name "Distinct Episode Names"
FROM     dw.stories LEFT OUTER JOIN dw.episodes USING (st_id)
WHERE    season = 17
ORDER BY "Distinct Episode Names";
 Distinct Episode Names
------------------------
 Part Four
 Part One
 Part Three
 Part Two

(5 rows)
```



COALESCE

Coalesce turns NULL values into a displayable value.

```
SELECT   DISTINCT
         COALESCE (ep_name,'None') "Distinct Episode Names"
FROM     dw.stories LEFT OUTER JOIN dw.episodes USING (st_id)
WHERE    season = 17
ORDER BY "Distinct Episode Names";
 Distinct Episode Names
------------------------
 None
 Part Four
 Part One
 Part Three
 Part Two
(5 rows)
```

DISTINCT ON ()

What is the first episode name of each story? Funny story!

```
SELECT   DISTINCT ON (st_id)
         st_name "Story",
         ep_number "Episode Number",
         ep_name "First Episode Name"
FROM     dw.stories LEFT OUTER JOIN dw.episodes USING (st_id)
WHERE    season = 1
ORDER BY st_id,ep_number;
          Story          | Episode Number |   First Episode Name
-------------------------+----------------+------------------------
 An Unearthly Child      |              1 | An Unearthly Child
 The Daleks              |              1 | The Dead Planet
 The Edge of Destruction |              1 | The Edge of Destruction
 Marco Polo              |              1 | The Roof of the World
 The Keys of Marinus     |              1 | The Sea of Death
 The Aztecs              |              1 | The Temple of Evil
 The Sensorites          |              1 | Strangers in Space
 The Reign of Terror     |              1 | A Land of Fear
(8 rows)
```

DISTINCT ON ()

What is the last episode name of each story?

```
SELECT   DISTINCT ON (st_id)
         st_name "Story",
         ep_number "Episode Number",
         ep_name "Last Episode Name"
FROM     dw.stories LEFT OUTER JOIN dw.episodes USING (st_id)
WHERE    season = 1
ORDER BY st_id,ep_number DESC;
          Story           | Episode Number |     Last Episode Name
--------------------------+----------------+---------------------------
 An Unearthly Child       |              4 | The Firemaker
 The Daleks               |              7 | The Rescue
 The Edge of Destruction  |              2 | The Brink of Disaster
 Marco Polo               |              7 | Assassin at Peking
 The Keys of Marinus      |              6 | The Keys of Marinus
 The Aztecs               |              4 | The Day of Darkness
 The Sensorites           |              6 | Kidnap
 The Reign of Terror      |              6 | Prisoners of Conciergerie
(8 rows)
```

③ UNIX/Linux sendmail

sendmail

- Sendmail is an electronic mail transport agent.
- Sendmail sends a message to one or more recipients, routing the message over whatever networks are necessary. Sendmail does internetwork forwarding as necessary to deliver the message to the correct place.
- Dates back to 1983!
- `sendmail -t` reads the message for recipients.

Hello World

Very simple "Hello world." e-mail message from sendmail.

```
cat >tmp.txt
To: chris@localhost
Subject: Test One
Hello world.  This is test one.
[Control-D]
$ cat tmp.txt | sendmail -t
```

```
From: chris@localhost
Sent: Sunday, September 1, 2019 3:14 PM
To: chris@localhost
Subject: Test One

Hello world. This is test one.
```

More Complex Example

Send the same e-mail to multiple recipients and provide a from.

```
cat >tmp.txt
To: apple@localhost
To: bag@localhost
Cc: cat@localhost
Cc: dog@localhost
Bcc: ear@localhost
Bcc: farm@localhost
From: goat@localhost
Subject: Test Two
This test shows how multiple recipients are specified.
[Control-D]
$ cat tmp.txt | sendmail -t
```

```
From: goat@localhost
Sent: Sunday, September 1, 2019 3:15 PM
To: apple@localhost, bag@localhost
Cc: cat@localhost, dog@localhost
Subject: Test Two

This test shows how multiple recipients are specified.
```

Send Text UTF-8

Send a simple plain character based UTF-8 e-mail.

```
cat >tmp.txt
To: chris@localhost
Subject: Test Three
MIME-Version: 1.0
Content-Type: text/plain; charset=UTF-8
Content-Disposition: inline


Just wanted to send you this note ♪
```

[Control-D]

```
$ cat tmp.txt | sendmail -t
```

```
From: chris@localhost
Sent: Sunday, September 1, 2019 3:16 PM
To: chris@localhost
Subject: Test Three



Just wanted to send you this note ♪
```

Send HTML

```
cat >tmp.txt
To: chris@localhost
Subject: Test Four
MIME-Version: 1.0
Content-Type: text/html; charset=UTF-8
Content-Disposition: inline
<html>
<body>
<b>Bold</b>
<br>
<i>Just testing out HTML.</i>
<br>
End of the page.
</body>
</html>
[Control-D]
$ cat tmp.txt | sendmail -t
```

From: chris@localhost
Sent: Sunday, September 1, 2019 3:17 PM
To: chris@localhost
Subject: Test Four

Bold
Just testing out HTML.
End of the page.

Attach a Binary File

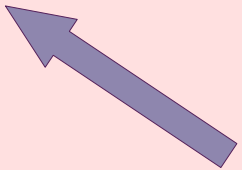
Complex example of sending a message and attachment part 1.

```
cat >tmp.txt
To: chris@localhost
Subject: Test Six
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="-q1w2e3r4t5"

---q1w2e3r4t5
Content-Type: text/html; charset=UTF-8
Content-Disposition: inline
<html>
<body>
Please see the attached PDF.
</body>
</html>

---q1w2e3r4t5
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="document.pdf"
[Control-D]
```

Attach a Binary File

Complex example of sending a message and attachment part 2.

```
cat document.pdf | uuencode --base64 document.pdf >>tmp.txt

cat >>tmp.txt
---q1w2e3r4t5--
[Control-D]

$ cat tmp.txt | sendmail -t
```



document.pdf (28 kb)

From: chris@localhost
Sent: Sunday, September 1, 2019 3:19 PM
To: chris@localhost
Subject: Test Six

Please see the attached PDF.

④ Automated Reporting System

A Complex Automated Reporting System

- Earlier this decade, a developer wanted to create a complex reporting system.
- He wanted to store these things in database tables:
  - Recipients
  - Titles
  - Schedule
  - Queries
  - Parameters
- No one had time to design it nor implement it.
- Would have only handled the simplest reports.
- Complex reports would still have to be custom written.
- Ultimately never progressed beyond wishes and talk.

A Simple Automated Reporting System

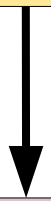
- I had a pain a few summers ago:
  - I maintained handful of shell script and psql reports sent on a cron schedule via e-mail.
  - To change recipients, I had to edit the shell scripts.
  - It was sometimes hard to match the e-mail to the shell script.
- There was not a lot of time to create a simple automated reporting system, so baby steps were taken.

Phase 1

- The requirements for phase 1:
  - Come up with database tables to store recipients.
  - Come up with a function to get recipients out of database tables into sendmail format.
  - Assign each report a unique ID.
- Then to test phase 1, create or retrofit an existing low profile report to prove the idea works.

Phase 1 Database Tables

```
CREATE TABLE dw.reports
(
  report_id          INTEGER                NOT NULL,
  report_title       CHARACTER VARYING(200) NOT NULL,
  report_description TEXT,
  CONSTRAINT reports_pk PRIMARY KEY (report_id)
);
```



```
CREATE TABLE dw.reports_email
(
  report_id     INTEGER               NOT NULL,
  fields        CHARACTER VARYING(4)  NOT NULL,
  email_address CHARACTER VARYING(64) NOT NULL,
  active_flag   BOOLEAN               NOT NULL DEFAULT TRUE,
  CONSTRAINT reports_email_pk PRIMARY KEY (report_id,fields,email_address),
  CONSTRAINT "'fields' can only equal TO, CC, BCC, or FROM"
             CHECK (fields IN ('TO','CC','BCC','FROM'))
);
```

Phase 1 Database Function

```
CREATE OR REPLACE FUNCTION dw.report_sendmail_recipients (INTEGER)
  RETURNS SETOF text AS
$report_sendmail_recipients$
  SELECT   CASE fields
             WHEN 'FROM' THEN 'From: '
             WHEN 'TO'   THEN 'To: '
             WHEN 'CC'   THEN 'Cc: '
             WHEN 'BCC'  THEN 'Bcc: '
           END || email_address
  FROM     dw.reports_email
  WHERE    report_id = $1 AND
           active_flag = TRUE
  ORDER BY CASE fields
             WHEN 'FROM' THEN 1
             WHEN 'TO'   THEN 2
             WHEN 'CC'   THEN 3
             WHEN 'BCC'  THEN 4
           END,
           email_address;
$report_sendmail_recipients$
  LANGUAGE SQL VOLATILE;
```

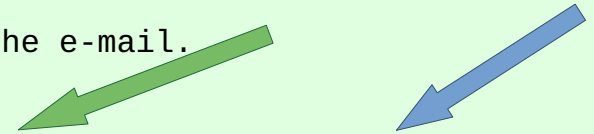
Phase 1 Populate the Tables

```
INSERT INTO dw.reports
  (report_id,report_title,report_description)
VALUES
  (1,'Doctor Who Season Report','A report of a Doctor Who season based on passed in parameter.');

INSERT INTO dw.reports_email
  (report_id,fields,email_address)
VALUES
  (1,'TO','chris@localhost'),
  (1,'FROM','thedoctor@localhost');
```

Phase 1 Shell Script (1 of 4)

```
#
# Get the season from parameter 1.
#
export DW_SEASON=$1
#
# Set environment variables for connection.  Password is in ".pgpass".
#
export PGDATABASE=postgres
export PGHOST=localhost
export PGPORT=5432
export PGUSER=chris
#
# Set the report ID to be the "Doctor Who Season" report.
#
export REPORT_ID=1
```

Phase 1 Shell Script (2 of 4)



```
#
# Begin the e-mail.
#
(
psql --quiet <<EndOfSql | awk 'NF'
--
-- Output only tuples, set thte border to zero, and turn off the footer.
--
\pset tuples_only on
\pset border 0
\pset footer off
--
-- Output the sendmail recipients from the report e-mails database table.
--
SELECT dw.report_sendmail_recipients (${REPORT_ID});
--
-- Output the subject line from the report database table.
--
SELECT 'Subject: ' || report_title
FROM   dw.reports
WHERE  report_id = ${REPORT_ID};
--
-- Start an HTML e-mail.
--
\echo 'MIME-Version: 1.0'
\echo 'Content-Type: text/html; charset=UTF-8'
\echo 'Content-Disposition: inline'
```

Phase 1 Shell Script (3 of 4)

```
\echo '<html>'
\echo '<head><style>'
\echo 'th { background-color: silver; }'
\echo 'caption { background-color: silver; font-size: 120%; font-weight: bolder }'
\echo '</style></head>'
\echo '<body>'
--
-- Turn off tuples only, set the output format to HTML, set the HTML boarder to 1,
-- set several table attributes, and set the title.
--
\pset tuples_only off
\pset format html
\pset border 1
\pset tableattr 'cellspacing="0" cellpadding="2"'
\pset title 'Season ${DW_SEASON} Report'
--
-- Run the query
--
SELECT   st_id "#",
         st_name "Story Name",
         ARRAY_TO_STRING (ARRAY_AGG (ep_name ORDER BY ep_number),'; ') "Episode List"
FROM     dw.stories LEFT OUTER JOIN dw.episodes USING (st_id)
WHERE    season = ${DW_SEASON}
GROUP BY st_id,st_name
ORDER BY st_id;
```

Phase 1 Shell Script (4 of 4)

```
--
-- Output the report ID for easy identification of the report.
--
\echo '<br><font size=1>Report ID ${REPORT_ID}</font>'
\echo '</body></html>'
----------
EndOfSql
#
# E-Mail the info.
#
) | /usr/sbin/sendmail -t
```

Phase 1 Tests

```
./doctor_who_season 1
```

From: chris@localhost
Sent: Sunday, September 1, 2019 3:20 PM
To: chris@localhost
Subject: Doctor Who Season Report

Season 1 Report

#	Story Name	Episode List
1	An Unearthly Child	An Unearthly Child; The Cave of Skulls; The Forest of Fear; The Firemaker
2	The Daleks	The Dead Planet; The Survivors; The Escape; The Ambush; The Expedition; The Ordeal; The Rescue
3	The Edge of Destruction	The Edge of Destruction; The Brink of Disaster
4	Marco Polo	The Roof of the World; The Singing Sands; Five Hundred Eyes; The Wall of Lies; Rider From Shang-Tu; Mighty Kublai Khan; Assassin at Peking
5	The Keys of Marinus	The Sea of Death; The Velvet Web; The Screaming Jungle; The Snows of Terror; Sentence of Death; The Keys of Marinus
6	The Aztecs	The Temple of Evil; The Warriors of Death; The Bride of Sacrifice; The Day of Darkness
7	The Sensorites	Strangers in Space; The Unwilling Warriors; Hidden Danger; A Desperate Venture; A Race Against Death; Kidnap
8	The Reign of Terror	A Land of Fear; Guests of Madame Guillotine; A Change of Identity; The Tyrant of France; A Bargain of Necessity; Prisoners of Conciergerie

Report ID 1

Phase 1 Tests

```
./doctor_who_season 17
```

From: chris@localhost
Sent: Sunday, September 1, 2019 3:21 PM
To: chris@localhost
Subject: Doctor Who Season Report

Season 17 Report

#	Story Name	Episode List
104	Destiny of the Daleks	Part One; Part Two; Part Three; Part Four
105	City of Death	Part One; Part Two; Part Three; Part Four
106	The Creature from the Pit	Part One; Part Two; Part Three; Part Four
107	Nightmare of Eden	Part One; Part Two; Part Three; Part Four
108	Horns of Nimon	Part One; Part Two; Part Three; Part Four
109	Shada	

Report ID 1

Send E-Mail with CSV Attachment

- Send a report just like the phase 1 report.
- Attached a CSV file that can be opened with Microsoft Excel or LibreOffice Calc.
- The CSV file will be all the details kept in the database.
- The shell script pieces will be identical to phase 1 report.
- Will utilize the powerful \copy meta-command!
- Blue text will be unchanged lines.
- Red text will be new or changed lines.

CSV Attachment (1 of 4)

```
#
# Begin the e-mail.
#
(
psql --quiet <<EndOfSql | awk 'NF'
--
-- Output only tuples, set thte border to zero, and turn off the footer.
--
\pset tuples_only on
\pset border 0
\pset footer off
--
-- Output the sendmail recipients from the report e-mails database table.
--
SELECT dw.report_sendmail_recipients (${REPORT_ID});
--
-- Output the subject line from the report database table.
--
SELECT 'Subject: ' || report_title
FROM   dw.reports
WHERE  report_id = ${REPORT_ID};
```

CSV Attachment (2 of 4)

```
--
-- Start a multipart e-mail.
--
\echo 'MIME-Version: 1.0'
\echo 'Content-Type: multipart/mixed; boundary="-q1w2e3r4t5"'
\echo
\echo '---q1w2e3r4t5'
--
-- The HTML half of the e-mail.
--
\echo 'Content-Type: text/html; charset=UTF-8'
\echo 'Content-Disposition: inline'
\echo '<html>'
\echo '<head><style>'
\echo 'th { background-color: silver; }'
\echo 'caption { background-color: silver; font-size: 120%; font-weight: bolder }'
\echo '</style></head>'
\echo '<body>'
--
-- Turn off tuples only, set the output format to HTML, set the HTML boarder to 1,
-- set several table attributes, and set the title.
--
\pset tuples_only off
\pset format html
\pset border 1
\pset tableattr 'cellspacing="0" cellpadding="2"'
\pset title 'Season ${DW_SEASON} Report'
```

CSV Attachment (3 of 4)

```
--
-- Run the query
--
SELECT   st_id "#",
         st_name "Story Name",
         ARRAY_TO_STRING (ARRAY_AGG (ep_name ORDER BY ep_number),'; ') "Episode List"
FROM     dw.stories LEFT OUTER JOIN dw.episodes USING (st_id)
WHERE    season = ${DW_SEASON}
GROUP BY st_id,st_name
ORDER BY st_id;
--
-- Output the report ID for easy identification of the report.
--
\echo '<br><font size=1>Report ID ${REPORT_ID}</font>'
\echo '</body></html>'
\echo '---q1w2e3r4t5'
```

CSV Attachment (4 of 4)

```
--
-- The CSV attachment half of the e-mail.
--
\echo 'Content-Type: text/plain; charset=UTF-8'
\echo 'Content-Transfer-Encoding: base64'
\echo 'Content-Disposition: attachment; filename="season_detail.csv"'
--
-- Create a temporary table of all story and episode records for a season.
--
CREATE TEMPORARY TABLE temp_season_table AS
SELECT   *
FROM     dw.stories LEFT OUTER JOIN dw.episodes USING (st_id)
WHERE    season = ${DW_SEASON}
ORDER BY st_id,ep_number;
--
-- Output the temporary table to uuencode
--
\COPY temp_season_table TO PROGRAM 'uuencode --base64 season_detail.csv' CSV HEADER
\echo '---q1w2e3r4t5--'
----------
EndOfSql
#
# E-Mail the info.
#
) | /usr/sbin/sendmail -t
```

CSV Attachment Test

```
./doctor_who_season 1
```

season_detail.csv (4 kB)

From: chris@localhost
Sent: Sunday, September 1, 2019 3:20 PM
To: chris@localhost
Subject: Doctor Who Season Report

Season 1 Report		
#	**Story Name**	**Episode List**
1	An Unearthly Child	An Unearthly Child; The Cave of Skulls; The Forest of Fear; The Firemaker
2	The Daleks	The Dead Planet; The Survivors; The Escape; The Ambush; The Expedition; The Ordeal; The Rescue
3	The Edge of Destruction	The Edge of Destruction; The Brink of Disaster
4	Marco Polo	The Roof of the World; The Singing Sands; Five Hundred Eyes; The Wall of Lies; Rider From Shang-Tu; Mighty Kublai Khan; Assassin at Peking
5	The Keys of Marinus	The Sea of Death; The Velvet Web; The Screaming Jungle; The Snows of Terror; Sentence of Death; The Keys of Marinus
6	The Aztecs	The Temple of Evil; The Warriors of Death; The Bride of Sacrifice; The Day of Darkness
7	The Sensorites	Strangers in Space; The Unwilling Warriors; Hidden Danger; A Desperate Venture; A Race Against Death; Kidnap
8	The Reign of Terror	A Land of Fear; Guests of Madame Guillotine; A Change of Identity; The Tyrant of France; A Bargain of Necessity; Prisoners of Conciergerie

Report ID 1

Phase 2

- A Java web application was written to allow end users to maintain the recipients.
- Recipients can be pulled from other tables.
- More than just shell script + psql reports use the tables.
- Just assigned report ID 79!

Future Phases?

- Automate some more pieces of the shell script.
  - Set database connection information from common environment file.
  - Directly call a certain version of psql from common environment file.
- Still no plans to create the complex system.

Fascinating Reporting with Postgres psql and sendmail

Thank You!

Download this presentation from OSTI.GOV:

https://www.osti.gov/servlets/purl/1560062