CSCI 303

Introduction to Data Science

13 - Working with SQL Databases (2)

This Lecture

- More SELECT queries
  - Operators and functions
  - Sorting
  - Grouping and aggregating
  - Joining tables

The obligatory setup code...

In [9]:
```
import pandas as pd
import sqlite3   # We'll be using a simple file-based SQLite3 database
from pandas import Series, DataFrame

dburi = 'sqlite:///csci303.sqlite3'
```

Let's also display info about the tables we'll be using in our examples:

In [10]:
```
from sqlalchemy import create_engine, inspect
inspector = inspect(create_engine(dburi))
```

In [11]:
```
1 pd.DataFrame(inspector.get_columns('scifi_author'))
```

Out[11]:

	autoincrement	default	name	nullable	primary_key	type
0	auto	None	id	False	1	TEXT
1	auto	None	name	False	0	TEXT
2	auto	None	birth_year	True	0	INTEGER
3	auto	None	death_year	True	0	INTEGER

In [12]:
```
1 pd.DataFrame(inspector.get_columns('scifi_work'), columns=['name','type'])
```

Out[12]:

	name	type
0	id	TEXT
1	author_id	TEXT
2	title	TEXT
3	publication_year	INTEGER

Operators and Functions

Wildcard matching using `LIKE` :

In [13]:
```
1 pd.read_sql_query("SELECT * FROM scifi_author WHERE name LIKE 'A%'", dburi) # sea
```

Out[13]:

	id	name	birth_year	death_year
0	/authors/OL3399168A	Ann Leckie	NaN	NaN
1	/authors/OL24708A	A. E. van Vogt	1912.0	2000.0
2	/authors/OL7304221A	Arthur C. Clarke	NaN	NaN
3	/authors/OL2623461A	Arthur C. Clarke	1917.0	2008.0
4	/authors/OL7228383A	Ann Leckie	NaN	NaN
5	/authors/OL7099704A	Andre Norton	1912.0	2005.0
6	/authors/OL7319169A	Arthur C. Clarke	NaN	NaN
7	/authors/OL218124A	Alfred Bester	1913.0	1987.0

The '%' wildcard matches any string of any length, so the above query asks for all authors where the author name starts with an 'A'.

Note that the wildcard in SQL is just a single '%', but that has meaning in Python strings, so we have to escape it by using an extra '%'.

You can also use '_' to stand in for any single character.

So, for instance, we could get authors for whom we have only a first initial:

In [14]:
```
pd.read_sql_query("SELECT * FROM scifi_author WHERE name LIKE '_. %%'", dburi)
```

Out[14]:

	id	name	birth_year	death_year
0	/authors/OL7046811A	C. J. Cherryh	NaN	NaN
1	/authors/OL24708A	A. E. van Vogt	1912.0	2000.0
2	/authors/OL7313078A	C. J. Cherryh	NaN	NaN
3	/authors/OL7342190A	C. J. Cherryh	NaN	NaN
4	/authors/OL7319469A	C. J. Cherryh	NaN	NaN

SQL has numerous functions and operators. For instance, pretty much any mathematical expression is allowed:

In [15]:
```
# gets the name of the author
# also gets the age of the author by subtracting birth year from death yaer (as lo
query = """
 SELECT name, death_year - birth_year AS approx_age
 FROM scifi_author
 WHERE death_year IS NOT NULL
"""

pd.read_sql_query(query, dburi)
```

Out[15]:

	name	approx_age
0	A. E. van Vogt	88
1	Arthur C. Clarke	91
2	Cordwainer Smith	53
3	Robert A. Heinlein	81
4	Clifford D. Simak	84
5	Roger Zelazny	58
6	Isaac Asimov	72
7	Andre Norton	93
8	Alfred Bester	74
9	Philip K. Dick	54
10	Poul Anderson	75

We slipped in a couple of other SQL things there. Let's explain those.

First, the `AS` keyword lets us rename or "alias" a column. That's handy when doing an expression like above.

We'll see other uses for `AS` later.

Also, we used another operator, `IS NOT NULL`.

NULL values in SQL are not comparable - any attempt to compare them using a relational operator (such as =) will always return false!

Instead, use `IS NULL` and `IS NOT NULL` to accept/reject NULL values.

There are a number of useful string functions and operators.

For example, is it "A. E. Van Vogt", or "A. E. van Vogt"?

We can convert strings to all lowercase using `lower`:

In [16]:
```
pd.read_sql_query("SELECT * FROM scifi_author WHERE lower(name) LIKE '%%van vogt'"
#pd.read_sql_query("SELECT * FROM scifi_author WHERE name LIKE '%%van vogt'", dbur
```

Out[16]:

	id	name	birth_year	death_year
0	/authors/OL24708A	A. E. van Vogt	1912	2000

All of the functions and operators mentioned above are standard SQL.

Your database may supply (many) others in addition.

For example, here's a link to documentation on PostgreSQL's functions and operators:
https://www.postgresql.org/docs/9.5/static/functions.html
(https://www.postgresql.org/docs/9.5/static/functions.html)

Sorting

To sort the results of a query, add an `ORDER BY` clause.

`ORDER BY` is followed by the columns you want to sort by. Each column name can optionally be followed with `ASC` (the default) or `DESC` to determine whether the sort is ascending or descending.

In [17]:
```
pd.read_sql_query("SELECT * FROM scifi_author ORDER BY name", dburi)[:10] # sorts
```

Out[17]:

	id	name	birth_year	death_year
0	/authors/OL24708A	A. E. van Vogt	1912.0	2000.0
1	/authors/OL218124A	Alfred Bester	1913.0	1987.0
2	/authors/OL7099704A	Andre Norton	1912.0	2005.0
3	/authors/OL3399168A	Ann Leckie	NaN	NaN
4	/authors/OL7228383A	Ann Leckie	NaN	NaN
5	/authors/OL7304221A	Arthur C. Clarke	NaN	NaN
6	/authors/OL2623461A	Arthur C. Clarke	1917.0	2008.0
7	/authors/OL7319169A	Arthur C. Clarke	NaN	NaN
8	/authors/OL7046811A	C. J. Cherryh	NaN	NaN
9	/authors/OL7313078A	C. J. Cherryh	NaN	NaN

In [18]:

```
# sorts the age of the author in descending order
query = """
 SELECT name, death_year - birth_year AS approx_age
 FROM scifi_author
 WHERE death_year IS NOT NULL
 ORDER BY approx_age DESC
"""

pd.read_sql_query(query, dburi)
```

Out[18]:

	name	approx_age
0	Andre Norton	93
1	Arthur C. Clarke	91
2	A. E. van Vogt	88
3	Clifford D. Simak	84
4	Robert A. Heinlein	81
5	Poul Anderson	75
6	Alfred Bester	74
7	Isaac Asimov	72
8	Roger Zelazny	58
9	Philip K. Dick	54
10	Cordwainer Smith	53

DISTINCT

Somewhat related to sorting, sometimes you want to get only a unique set of records back.

This is particularly useful when trying to find the unique settings for a particular column.

For instance, our data unfortunately has a lot of duplicates.

Let's see just the unique author names starting with 'C':

In [19]:

```
query = """
SELECT DISTINCT name
FROM scifi_author
WHERE name LIKE 'C%%'
"""

pd.read_sql_query(query, dburi)
```

Out[19]:

	name
0	Clifford D. Simak
1	C. J. Cherryh
2	Connie Willis
3	Cordwainer Smith
4	Charles Stross

Grouping and Aggregating

SQL has capabilities for grouping data and computing aggregate functions on the groups, very similar to what we saw pandas doing in the Advanced pandas lecture.

The basic syntax is

```
SELECT a1, a2, ..., fn1(a3), fn2(a4), ...
FROM table
GROUP BY a1, a2, ...;
```

where `fn1` etc. compute some kind of aggregate. Some of the functions available are `COUNT`, `SUM`, `AVG`, `MIN`, and `MAX`.

It is important that every attribute selected participate in the group operation.

For example, this query is valid and finds out the multiplicity of each author in our database:

In [20]:

```
query = """
SELECT name, COUNT(name) FROM scifi_author
GROUP BY name
"""

pd.read_sql_query(query, dburi)
```

Out[20]:

	name	COUNT(name)
0	A. E. van Vogt	1
1	Alfred Bester	1
2	Andre Norton	1
3	Ann Leckie	2
4	Arthur C. Clarke	3
5	C. J. Cherryh	4
6	Charles Stross	1
7	Clifford D. Simak	3
8	Connie Willis	1
9	Cordwainer Smith	2
10	Frederik Pohl	2
11	Isaac Asimov	2
12	Julian May	2
13	Philip K. Dick	2
14	Poul Anderson	1
15	Robert A. Heinlein	1
16	Robert Silverberg	2
17	Roger Zelazny	2

whereas this query is invalid, because we are not grouping by birth_year:

In [21]:

```
query = """
SELECT name, COUNT(name), birth_year FROM scifi_author
GROUP BY name
"""
pd.read_sql_query(query, dburi)
```

Out[21]:

	name	COUNT(name)	birth_year
0	A. E. van Vogt	1	1912.0
1	Alfred Bester	1	1913.0
2	Andre Norton	1	1912.0
3	Ann Leckie	2	NaN
4	Arthur C. Clarke	3	NaN
5	C. J. Cherryh	4	NaN
6	Charles Stross	1	1964.0
7	Clifford D. Simak	3	1904.0
8	Connie Willis	1	NaN
9	Cordwainer Smith	2	NaN
10	Frederik Pohl	2	1919.0
11	Isaac Asimov	2	1920.0
12	Julian May	2	1931.0
13	Philip K. Dick	2	1928.0
14	Poul Anderson	1	1926.0
15	Robert A. Heinlein	1	1907.0
16	Robert Silverberg	2	NaN
17	Roger Zelazny	2	1937.0

Usefully, we can also `ORDER BY` aggregate expressions. Suppose we want the most duplicated authors (to see where we need to do the most cleanup):

In [22]:
```
query = """
SELECT name, COUNT(name) FROM scifi_author
GROUP BY name
ORDER BY COUNT(name) DESC
"""
pd.read_sql_query(query, dburi)
```

Out[22]:

	name	COUNT(name)
0	C. J. Cherryh	4
1	Arthur C. Clarke	3
2	Clifford D. Simak	3
3	Ann Leckie	2
4	Cordwainer Smith	2
5	Frederik Pohl	2
6	Isaac Asimov	2
7	Julian May	2
8	Philip K. Dick	2
9	Robert Silverberg	2
10	Roger Zelazny	2
11	A. E. van Vogt	1
12	Alfred Bester	1
13	Andre Norton	1
14	Charles Stross	1
15	Connie Willis	1
16	Poul Anderson	1
17	Robert A. Heinlein	1

We can also use another clause with the keyword `HAVING`, which lets us filter the grouped results according to aggregate values. Let's only see the authors with multiplicity greater than 2:

In [23]:
```
query = """
SELECT name, COUNT(name) FROM scifi_author
GROUP BY name
HAVING COUNT(name) > 2
"""
pd.read_sql_query(query, dburi)
```

Out[23]:

	name	COUNT(name)
0	Arthur C. Clarke	3
1	C. J. Cherryh	4
2	Clifford D. Simak	3

Aggregate functions can also be applied without grouping:

In [24]:
```
query = """
 SELECT
   MIN(death_year - birth_year),
   AVG(death_year - birth_year),
   MAX(death_year - birth_year)
 FROM scifi_author
 WHERE death_year IS NOT NULL
"""

pd.read_sql_query(query, dburi)
```

Out[24]:

	MIN(death_year - birth_year)	AVG(death_year - birth_year)	MAX(death_year - birth_year)
0	53	74.818182	93

Joins

Databases are often factored in such a way as to minimize duplicate information. (This database didn't succeed so well in that.)

For example, we have a table of Science Fiction works, but it doesn't include author names or dates.

Rather, the authors live in a separate table, and are simply referenced by a key field from the works table.

In [25]:
```
pd.DataFrame(inspector.get_columns('scifi_work'), columns=['name','type'])
```

Out[25]:

	name	type
0	id	TEXT
1	author_id	TEXT
2	title	TEXT
3	publication_year	INTEGER

The `author_id` field provides our linkage to the `scifi_author` table.

There are two ways to do inner joins, which are the joins we most commonly want to do.

Here's the "wordy" way to join `scifi_author` and `scifi_work`:

In [26]:

```
query = """
SELECT scifi_author.name, scifi_work.title, scifi_work.publication_year
FROM   scifi_author JOIN scifi_work ON scifi_author.id = scifi_work.author_id
"""

pd.read_sql_query(query, dburi)
```

Out[26]:

	name	title	publication_year
0	Poul Anderson	Genesis	NaN
1	Julian May	Metaconcert	1989.0
2	Roger Zelazny	Sign of Chaos	1987.0
3	Arthur C. Clarke	A Meeting With Medusa/Green Mars (Special Doub...	NaN
4	Arthur C. Clarke	The Space Trilogy	NaN
5	Robert A. Heinlein	Von Stern zu Stern	NaN
6	C. J. Cherryh	The Well of Shiuan	NaN
7	Roger Zelazny	Amber Collectors Series III	NaN
8	Robert Silverberg	The world of the ocean depths	NaN
9	Isaac Asimov	"X" representa lo desconocido	NaN
10	Isaac Asimov	An Isaac Asimov second omnibus	NaN
11	Isaac Asimov	Vybor katastrof	NaN
12	Isaac Asimov	Robots e imperio	NaN
13	Robert Silverberg	Reflections and refractions	1997.0
14	Robert Silverberg	New Dimensions 9	1979.0
15	A. E. van Vogt	The Battle of Forever	NaN
16	Andre Norton	Leopard in exile	NaN
17	Frederik Pohl	Jem	NaN
18	Clifford D. Simak	Tous les pièges de la terre	NaN
19	Arthur C. Clarke	Preludio allo spazio	NaN
20	Isaac Asimov	How Do Aeroplanes Fly? (Ask Isaac Asimov)	NaN
21	Isaac Asimov	The intelligent man's guide to the biological ...	NaN
22	Isaac Asimov	Los griegos	NaN
23	Poul Anderson	Galaxy Science Fiction, April 1951 Kornbluth's...	NaN
24	Poul Anderson	The worlds of Poul Anderson	NaN
25	Poul Anderson	Hrolf Kraki's saga	NaN
26	Poul Anderson	The sign of the raven	NaN
27	Philip K. Dick	The man in the high castle	2001.0
28	A. E. van Vogt	Monsters	NaN
29	Roger Zelazny	Une rose pour l'Ecclesiaste	NaN
...	...	...	...
4905	Isaac Asimov	Isaac Asimov Las Amenazas De Nuestro Mundo/Isa...	NaN
4906	Poul Anderson	KING YS:GALLICENAE	NaN
4907	Poul Anderson	Three in Time: A White Wolf Rediscovery Trio	NaN

	name	title	publication_year
4908	Isaac Asimov	Saturno	NaN
4909	C. J. Cherryh	Gate of Ivrel (Morgaine Saga, Book 1)	NaN
4910	Philip K. Dick	Pacific park	NaN
4911	Robert Silverberg	Clocks for the ages	NaN
4912	Philip K. Dick	Five novels of the 1960s & 70s	NaN
4913	Robert Silverberg	Stormy voyager	1968.0
4914	Philip K. Dick	The Selected Letters of Philip K. Dick 1972-19...	1994.0
4915	Philip K. Dick	Galactic pot-healer	1969.0
4916	Julian May	Die Schulter des Orion	2002.0
4917	Julian May	The life cycle of an opossum	1973.0
4918	Isaac Asimov	Our solar system	NaN
4919	Isaac Asimov	How was the universe born?	NaN
4920	Poul Anderson	Planet of no return	NaN
4921	Isaac Asimov	Isaac Asimov's 21st Century Library of the Uni...	NaN
4922	Isaac Asimov	Positronic Man, The	NaN
4923	Poul Anderson	Ensign Flandry, Volume 3	NaN
4924	Philip K. Dick	Pacific park	NaN
4925	Clifford D. Simak	Bring Back Yesterday / The Trouble With Tycho	NaN
4926	Philip K. Dick	Pacific park	NaN
4927	Frederik Pohl	Beyond the blue event horizon	NaN
4928	Robert Silverberg	Mammoths Mastoons And Man	NaN
4929	Robert Silverberg	The Planet Killers Three Novels Of The Spaceways	NaN
4930	Robert Silverberg	Le Seigneur des ténèbres, tome 2	1996.0
4931	Robert Silverberg	Star of Gypsies	1986.0
4932	Philip K. Dick	Time Out of Joint	2003.0
4933	A. E. van Vogt	The Silkie	NaN
4934	Isaac Asimov	Sueños De Robot	NaN

4935 rows × 3 columns

Some of the table name specifiers are unnecessary in the above query, as SQL can work out what table you mean if the column name is unique.

The only column we really *need* the specifier on is the `scifi_author.id` column, since there is also a column named `id` in `scifi_work`.

However, it makes things clearer if we do something to specify what tables columns are coming from.

This is a common use of table aliases using the `AS` keyword.

In [27]:

```
query = """
SELECT a.name, w.title, w.publication_year
FROM   scifi_author AS a JOIN scifi_work AS w
ON a.id = w.author_id
"""

pd.read_sql_query(query, dburi)
```

Out[27]:

	name	title	publication_year
0	Poul Anderson	Genesis	NaN
1	Julian May	Metaconcert	1989.0
2	Roger Zelazny	Sign of Chaos	1987.0
3	Arthur C. Clarke	A Meeting With Medusa/Green Mars (Special Doub...	NaN
4	Arthur C. Clarke	The Space Trilogy	NaN
5	Robert A. Heinlein	Von Stern zu Stern	NaN
6	C. J. Cherryh	The Well of Shiuan	NaN
7	Roger Zelazny	Amber Collectors Series III	NaN
8	Robert Silverberg	The world of the ocean depths	NaN
9	Isaac Asimov	"X" representa lo desconocido	NaN
10	Isaac Asimov	An Isaac Asimov second omnibus	NaN
11	Isaac Asimov	Vybor katastrof	NaN
12	Isaac Asimov	Robots e imperio	NaN
13	Robert Silverberg	Reflections and refractions	1997.0
14	Robert Silverberg	New Dimensions 9	1979.0
15	A. E. van Vogt	The Battle of Forever	NaN
16	Andre Norton	Leopard in exile	NaN
17	Frederik Pohl	Jem	NaN
18	Clifford D. Simak	Tous les pièges de la terre	NaN
19	Arthur C. Clarke	Preludio allo spazio	NaN
20	Isaac Asimov	How Do Aeroplanes Fly? (Ask Isaac Asimov)	NaN
21	Isaac Asimov	The intelligent man's guide to the biological ...	NaN
22	Isaac Asimov	Los griegos	NaN
23	Poul Anderson	Galaxy Science Fiction, April 1951 Kornbluth's...	NaN
24	Poul Anderson	The worlds of Poul Anderson	NaN
25	Poul Anderson	Hrolf Kraki's saga	NaN
26	Poul Anderson	The sign of the raven	NaN
27	Philip K. Dick	The man in the high castle	2001.0
28	A. E. van Vogt	Monsters	NaN
29	Roger Zelazny	Une rose pour l'Ecclesiaste	NaN
...	...	...	...
4905	Isaac Asimov	Isaac Asimov Las Amenazas De Nuestro Mundo/Isa...	NaN

	name	title	publication_year
4906	Poul Anderson	KING YS:GALLICENAE	NaN
4907	Poul Anderson	Three in Time: A White Wolf Rediscovery Trio	NaN
4908	Isaac Asimov	Saturno	NaN
4909	C. J. Cherryh	Gate of Ivrel (Morgaine Saga, Book 1)	NaN
4910	Philip K. Dick	Pacific park	NaN
4911	Robert Silverberg	Clocks for the ages	NaN
4912	Philip K. Dick	Five novels of the 1960s & 70s	NaN
4913	Robert Silverberg	Stormy voyager	1968.0
4914	Philip K. Dick	The Selected Letters of Philip K. Dick 1972-19...	1994.0
4915	Philip K. Dick	Galactic pot-healer	1969.0
4916	Julian May	Die Schulter des Orion	2002.0
4917	Julian May	The life cycle of an opossum	1973.0
4918	Isaac Asimov	Our solar system	NaN
4919	Isaac Asimov	How was the universe born?	NaN
4920	Poul Anderson	Planet of no return	NaN
4921	Isaac Asimov	Isaac Asimov's 21st Century Library of the Uni...	NaN
4922	Isaac Asimov	Positronic Man, The	NaN
4923	Poul Anderson	Ensign Flandry, Volume 3	NaN
4924	Philip K. Dick	Pacific park	NaN
4925	Clifford D. Simak	Bring Back Yesterday / The Trouble With Tycho	NaN
4926	Philip K. Dick	Pacific park	NaN
4927	Frederik Pohl	Beyond the blue event horizon	NaN
4928	Robert Silverberg	Mammoths Mastoons And Man	NaN
4929	Robert Silverberg	The Planet Killers Three Novels Of The Spaceways	NaN
4930	Robert Silverberg	Le Seigneur des ténèbres, tome 2	1996.0
4931	Robert Silverberg	Star of Gypsies	1986.0
4932	Philip K. Dick	Time Out of Joint	2003.0
4933	A. E. van Vogt	The Silkie	NaN
4934	Isaac Asimov	Sueños De Robot	NaN

4935 rows × 3 columns

An equivalent, and more compact way to write the same query is to move the join condition(s) into the WHERE clause, and simply list the tables you want data from:

In [28]:

```
query = """
SELECT a.name, w.title, w.publication_year
FROM scifi_author AS a, scifi_work AS w
WHERE a.id = w.author_id
"""
pd.read_sql_query(query, dburi)
```

Out[28]:

	name	title	publication_year
0	Poul Anderson	Genesis	NaN
1	Julian May	Metaconcert	1989.0
2	Roger Zelazny	Sign of Chaos	1987.0
3	Arthur C. Clarke	A Meeting With Medusa/Green Mars (Special Doub...	NaN
4	Arthur C. Clarke	The Space Trilogy	NaN
5	Robert A. Heinlein	Von Stern zu Stern	NaN
6	C. J. Cherryh	The Well of Shiuan	NaN
7	Roger Zelazny	Amber Collectors Series III	NaN
8	Robert Silverberg	The world of the ocean depths	NaN
9	Isaac Asimov	"X" representa lo desconocido	NaN
10	Isaac Asimov	An Isaac Asimov second omnibus	NaN
11	Isaac Asimov	Vybor katastrof	NaN
12	Isaac Asimov	Robots e imperio	NaN
13	Robert Silverberg	Reflections and refractions	1997.0
14	Robert Silverberg	New Dimensions 9	1979.0
15	A. E. van Vogt	The Battle of Forever	NaN
16	Andre Norton	Leopard in exile	NaN
17	Frederik Pohl	Jem	NaN
18	Clifford D. Simak	Tous les pièges de la terre	NaN
19	Arthur C. Clarke	Preludio allo spazio	NaN
20	Isaac Asimov	How Do Aeroplanes Fly? (Ask Isaac Asimov)	NaN
21	Isaac Asimov	The intelligent man's guide to the biological ...	NaN
22	Isaac Asimov	Los griegos	NaN
23	Poul Anderson	Galaxy Science Fiction, April 1951 Kornbluth's...	NaN
24	Poul Anderson	The worlds of Poul Anderson	NaN
25	Poul Anderson	Hrolf Kraki's saga	NaN
26	Poul Anderson	The sign of the raven	NaN
27	Philip K. Dick	The man in the high castle	2001.0
28	A. E. van Vogt	Monsters	NaN
29	Roger Zelazny	Une rose pour l'Ecclesiaste	NaN
...	...	...	...
4905	Isaac Asimov	Isaac Asimov Las Amenazas De Nuestro Mundo/Isa...	NaN
4906	Poul Anderson	KING YS:GALLICENAE	NaN
4907	Poul Anderson	Three in Time: A White Wolf Rediscovery Trio	NaN

	name	title	publication_year
4908	Isaac Asimov	Saturno	NaN
4909	C. J. Cherryh	Gate of Ivrel (Morgaine Saga, Book 1)	NaN
4910	Philip K. Dick	Pacific park	NaN
4911	Robert Silverberg	Clocks for the ages	NaN
4912	Philip K. Dick	Five novels of the 1960s & 70s	NaN
4913	Robert Silverberg	Stormy voyager	1968.0
4914	Philip K. Dick	The Selected Letters of Philip K. Dick 1972-19...	1994.0
4915	Philip K. Dick	Galactic pot-healer	1969.0
4916	Julian May	Die Schulter des Orion	2002.0
4917	Julian May	The life cycle of an opossum	1973.0
4918	Isaac Asimov	Our solar system	NaN
4919	Isaac Asimov	How was the universe born?	NaN
4920	Poul Anderson	Planet of no return	NaN
4921	Isaac Asimov	Isaac Asimov's 21st Century Library of the Uni...	NaN
4922	Isaac Asimov	Positronic Man, The	NaN
4923	Poul Anderson	Ensign Flandry, Volume 3	NaN
4924	Philip K. Dick	Pacific park	NaN
4925	Clifford D. Simak	Bring Back Yesterday / The Trouble With Tycho	NaN
4926	Philip K. Dick	Pacific park	NaN
4927	Frederik Pohl	Beyond the blue event horizon	NaN
4928	Robert Silverberg	Mammoths Mastoons And Man	NaN
4929	Robert Silverberg	The Planet Killers Three Novels Of The Spaceways	NaN
4930	Robert Silverberg	Le Seigneur des ténèbres, tome 2	1996.0
4931	Robert Silverberg	Star of Gypsies	1986.0
4932	Philip K. Dick	Time Out of Joint	2003.0
4933	A. E. van Vogt	The Silkie	NaN
4934	Isaac Asimov	Sueños De Robot	NaN

4935 rows × 3 columns

Now we can ask questions like, "What books did Ann Leckie write?"

In [29]:

```
query = """
SELECT a.name, w.title, w.publication_year
FROM scifi_author AS a, scifi_work AS w
WHERE a.id = w.author_id
AND a.name = 'Ann Leckie'
"""
pd.read_sql_query(query, dburi)
```

Out[29]:

	name	title	publication_year
0	Ann Leckie	Ancillary Justice	None
1	Ann Leckie	Ancillary Mercy	None
2	Ann Leckie	Ancillary Sword	None

Or, "Who wrote *I, Robot*?"

In [30]:

```
query = """
SELECT a.name
FROM scifi_author AS a, scifi_work AS w
WHERE a.id = w.author_id
AND w.title = 'I, Robot'
"""
pd.read_sql_query(query, dburi)
```

Out[30]:

	name
0	Isaac Asimov
1	Isaac Asimov

Putting It All Together

Now that we know the basics, let's try some queries with the Sci-fi books dataset.

Let's start with, "How many books are listed for each author entry?"

In [31]:

```
query = """
SELECT a.id, a.name, COUNT(w.title)
FROM   scifi_author AS a, scifi_work AS w
WHERE  a.id = w.author_id
GROUP BY a.id, a.name
ORDER BY a.name
"""
pd.read_sql_query(query, dburi)
```

Out[31]:

	id	name	COUNT(w.title)
0	/authors/OL24708A	A. E. van Vogt	208
1	/authors/OL218124A	Alfred Bester	34
2	/authors/OL7099704A	Andre Norton	285
3	/authors/OL7228383A	Ann Leckie	3
4	/authors/OL2623461A	Arthur C. Clarke	376
5	/authors/OL7304221A	Arthur C. Clarke	1
6	/authors/OL7319169A	Arthur C. Clarke	2
7	/authors/OL7046811A	C. J. Cherryh	165
8	/authors/OL7313078A	C. J. Cherryh	1
9	/authors/OL7319469A	C. J. Cherryh	1
10	/authors/OL7342190A	C. J. Cherryh	1
11	/authors/OL343157A	Charles Stross	38
12	/authors/OL765755A	Clifford D. Simak	138
13	/authors/OL20934A	Connie Willis	51
14	/authors/OL4392224A	Cordwainer Smith	6
15	/authors/OL593082A	Cordwainer Smith	21
16	/authors/OL23364A	Frederik Pohl	216
17	/authors/OL34221A	Isaac Asimov	1504
18	/authors/OL320739A	Julian May	222
19	/authors/OL6829346A	Julian May	15
20	/authors/OL274606A	Philip K. Dick	364
21	/authors/OL7208702A	Philip K. Dick	4
22	/authors/OL24072A	Poul Anderson	310
23	/authors/OL28641A	Robert A. Heinlein	201
24	/authors/OL6890242A	Robert Silverberg	539
25	/authors/OL4587302A	Roger Zelazny	227
26	/authors/OL7211371A	Roger Zelazny	2

That's probably not what we wanted; our data has some bad duplication in it.

Let's try just grouping by author name:

In [32]:

```
query = """
SELECT a.name, COUNT(w.title)
FROM   scifi_author AS a, scifi_work AS w
WHERE  a.id = w.author_id
GROUP BY a.name
ORDER BY a.name
"""
pd.read_sql_query(query, dburi)
```

Out[32]:

	name	COUNT(w.title)
0	A. E. van Vogt	208
1	Alfred Bester	34
2	Andre Norton	285
3	Ann Leckie	3
4	Arthur C. Clarke	379
5	C. J. Cherryh	168
6	Charles Stross	38
7	Clifford D. Simak	138
8	Connie Willis	51
9	Cordwainer Smith	27
10	Frederik Pohl	216
11	Isaac Asimov	1504
12	Julian May	237
13	Philip K. Dick	368
14	Poul Anderson	310
15	Robert A. Heinlein	201
16	Robert Silverberg	539
17	Roger Zelazny	229

I wonder if duplication is a problem in the works data, too?

Let's look closer at one of our more prolific authors:

In [33]:

```
query = """
SELECT w.title, COUNT(w.title)
FROM   scifi_author AS a, scifi_work AS w
WHERE  a.id = w.author_id
AND    a.name = 'Arthur C. Clarke'
GROUP BY w.title
HAVING COUNT(w.title) > 1
ORDER BY w.title
"""
pd.read_sql_query(query, dburi)
```

Out[33]:

	title	COUNT(w.title)
0	1984	2
1	2001	6
2	2001, a space odyssey	2
3	2010	3
4	2061, [i.e. Dos mil sesenta y uno] odisea tres	5
5	2061: Odisei III	5
6	Across the Sea of Stars	2
7	American heritage	2
8	Beyond the fall of night	2
9	Cánticos de la lejanatierra	5
10	Childhood's End	3
11	Cradle	3
12	Deep Range	2
13	El centinela	5
14	Firstborn	2
15	Glide Path	2
16	Indian Ocean treasure	2
17	Interplanetary flight	3
18	Islands in the sky	2
19	La ciudad y las estrellas	5
20	Lama mi jing	5
21	Lama ren wu	5
22	Lama zai xian	5
23	Lama zhen xiang	5
24	Le guide del tramonto	5
25	Miestas ir zvaigdes	5
26	Rama revealed	2
27	Reach for Tomorrow	2
28	Rendezvous with Rama	2
29	Report on Planet 3	2
30	Sunstorm	2

	title	COUNT(w.title)
31	Tales from the White Hart	2
32	Tales of ten worlds	3
33	The City and the Stars	3
34	The Ghost from the Grand Banks and the Deep Range	5
35	The Sands of Mars	2
36	The Sentinel	2
37	The Songs of Distant Earth	2
38	The city and the stars	3
39	The other side of the sky	3

Hm, maybe we can filter this down a bit. It turns out you can combine COUNT and DISTINCT - this will at least let us remove books with exact duplicate titles.

(It still won't help with books that are the same but are listed differently, or books in other languages.)

In [34]:

```
query = """
SELECT a.name, COUNT(w.title), COUNT(DISTINCT w.title)
FROM   scifi_author AS a, scifi_work AS w
WHERE  a.id = w.author_id
GROUP BY a.name
ORDER BY a.name
"""
pd.read_sql_query(query, dburi)
```

Out[34]:

	name	COUNT(w.title)	COUNT(DISTINCT w.title)
0	A. E. van Vogt	208	153
1	Alfred Bester	34	31
2	Andre Norton	285	275
3	Ann Leckie	3	3
4	Arthur C. Clarke	379	291
5	C. J. Cherryh	168	163
6	Charles Stross	38	32
7	Clifford D. Simak	138	111
8	Connie Willis	51	44
9	Cordwainer Smith	27	24
10	Frederik Pohl	216	179
11	Isaac Asimov	1504	1226
12	Julian May	237	222
13	Philip K. Dick	368	252
14	Poul Anderson	310	254
15	Robert A. Heinlein	201	160
16	Robert Silverberg	539	475
17	Roger Zelazny	229	193

If we want to see the actual distinct titles, we can just use DISTINCT without grouping:

In [35]:

```
query = """
SELECT DISTINCT a.name, w.title
FROM   scifi_author AS a, scifi_work AS w
WHERE  a.id = w.author_id
ORDER BY a.name, w.title
"""
pd.read_sql_query(query, dburi)
```

Out[35]:

	name	title
0	A. E. van Vogt	'O Kosmos tou Meden-A
1	A. E. van Vogt	A Report on the Violent Male
2	A. E. van Vogt	A Van Vogt omnibus
3	A. E. van Vogt	A la poursuite des Slans
4	A. E. van Vogt	Anagennese
5	A. E. van Vogt	Anazetese tou mellontos
6	A. E. van Vogt	Astounding Science Fiction, August 1946 (Volum...
7	A. E. van Vogt	Au-delà du village enchanté
8	A. E. van Vogt	Away and Beyond
9	A. E. van Vogt	Aytokratoria tou atomou
10	A. E. van Vogt	Briefe von den Sternen
11	A. E. van Vogt	Children of Tomorrow
12	A. E. van Vogt	Computerworld
13	A. E. van Vogt	Cosmic Encounter
14	A. E. van Vogt	Créateur d'Univers
15	A. E. van Vogt	Cycle de linn
16	A. E. van Vogt	Das verzauverte Dorf
17	A. E. van Vogt	Der wunde Punkt
18	A. E. van Vogt	Des lendemains qui scintillent
19	A. E. van Vogt	Destinatia univers
20	A. E. van Vogt	Destination
21	A. E. van Vogt	Destination Univers
22	A. E. van Vogt	Destination: Universe!
23	A. E. van Vogt	Die Waffenhändler von Isher
24	A. E. van Vogt	Earth Factor X
25	A. E. van Vogt	Earth factor X
26	A. E. van Vogt	Earth's Last Fortress
27	A. E. van Vogt	El Viaje de La Beagle Espacial
28	A. E. van Vogt	Empire of the Atom
29	A. E. van Vogt	Empire of the atom
...	...	...
4058	Roger Zelazny	The hand of Oberon
4059	Roger Zelazny	This Immortal

	name	title
4060	Roger Zelazny	Threshold
4061	Roger Zelazny	To Die in Italbar/A Dark Travelling
4062	Roger Zelazny	To Spin is Miracle Cat
4063	Roger Zelazny	To die in Italbar
4064	Roger Zelazny	Today We Choose Faces / Bridge of Ashes
4065	Roger Zelazny	Today We Choose a Face
4066	Roger Zelazny	Today we choose faces
4067	Roger Zelazny	Today we choose faces ; and, Bridge of ashes
4068	Roger Zelazny	Toi, l'immortel
4069	Roger Zelazny	Trumps of Doom
4070	Roger Zelazny	Trumps of Doom #6 Amber Series
4071	Roger Zelazny	Trumps of Doom (Chronicles of Amber)
4072	Roger Zelazny	Trumps of doom
4073	Roger Zelazny	Tvoretz snovideniĭ
4074	Roger Zelazny	Un pont de cendres
4075	Roger Zelazny	Une rose pour l'Ecclesiaste
4076	Roger Zelazny	Une rose pour l'Ecclésiaste
4077	Roger Zelazny	Unicorn Variations
4078	Roger Zelazny	Unicorn variations
4079	Roger Zelazny	Warriors of Blood and Dream
4080	Roger Zelazny	Way up high
4081	Roger Zelazny	Wheel of Fortune
4082	Roger Zelazny	When Pussywillows Last in the Catyard Bloomed ...
4083	Roger Zelazny	Wilderness
4084	Roger Zelazny	Wizard World
4085	Roger Zelazny	Wizard world
4086	Roger Zelazny	l'île des morts
4087	Roger Zelazny	le serum de la deesse bleue

4088 rows × 2 columns

"Which of our authors is most prolific?"

In [36]:

```
query = """
SELECT a.name, COUNT(DISTINCT w.title) AS unique_count
FROM   scifi_author AS a, scifi_work AS w
WHERE  a.id = w.author_id
GROUP BY a.name
ORDER BY unique_count DESC
"""
pd.read_sql_query(query, dburi)[:5]
```

Out[36]:

	name	unique_count
0	Isaac Asimov	1226
1	Robert Silverberg	475
2	Arthur C. Clarke	291
3	Andre Norton	275
4	Poul Anderson	254

"What year had the most works published?"

In [37]:

```
query = """
SELECT publication_year, COUNT(DISTINCT title) AS unique_count
FROM scifi_work
GROUP BY publication_year
ORDER BY unique_count DESC
"""
pd.read_sql_query(query, dburi)[:5]
```

Out[37]:

	publication_year	unique_count
0	NaN	3267
1	1972.0	47
2	1975.0	37
3	1973.0	36
4	1976.0	33

"What years did Arthur C. Clarke publish the most in?"

In [38]:

```
query = """
SELECT w.publication_year, COUNT(DISTINCT w.title) AS unique_count
FROM scifi_work AS w, scifi_author AS a
WHERE a.id = w.author_id AND a.name = 'Arthur C. Clarke'
GROUP BY publication_year
ORDER BY unique_count DESC
"""
pd.read_sql_query(query, dburi)[:5]
```

Out[38]:

	publication_year	unique_count
0	NaN	241
1	1957.0	5
2	1972.0	5
3	1953.0	4
4	1989.0	4

We clearly don't know more than we know...

What other questions can you think to ask of this data?

Extra Stuff

- SQL odds and ends
  - Set operations
  - Subqueries
  - Outer joins