Query.jl

Query.jl

David Anthoff
UC Berkeley

Installation instructions

To follow this talk on your own system, you need a number of packages:

```
Pkg.clone("https://github.com/davidanthoff/Dataverse.jl")
Pkg.add("RDatasets")
Pkg.add("DataTables")
Pkg.add("IndexedTables")
Pkg.add("TypedTables")
```

You should also execute the initial `using` cell in this notebook before this talk so that things get precompiled.

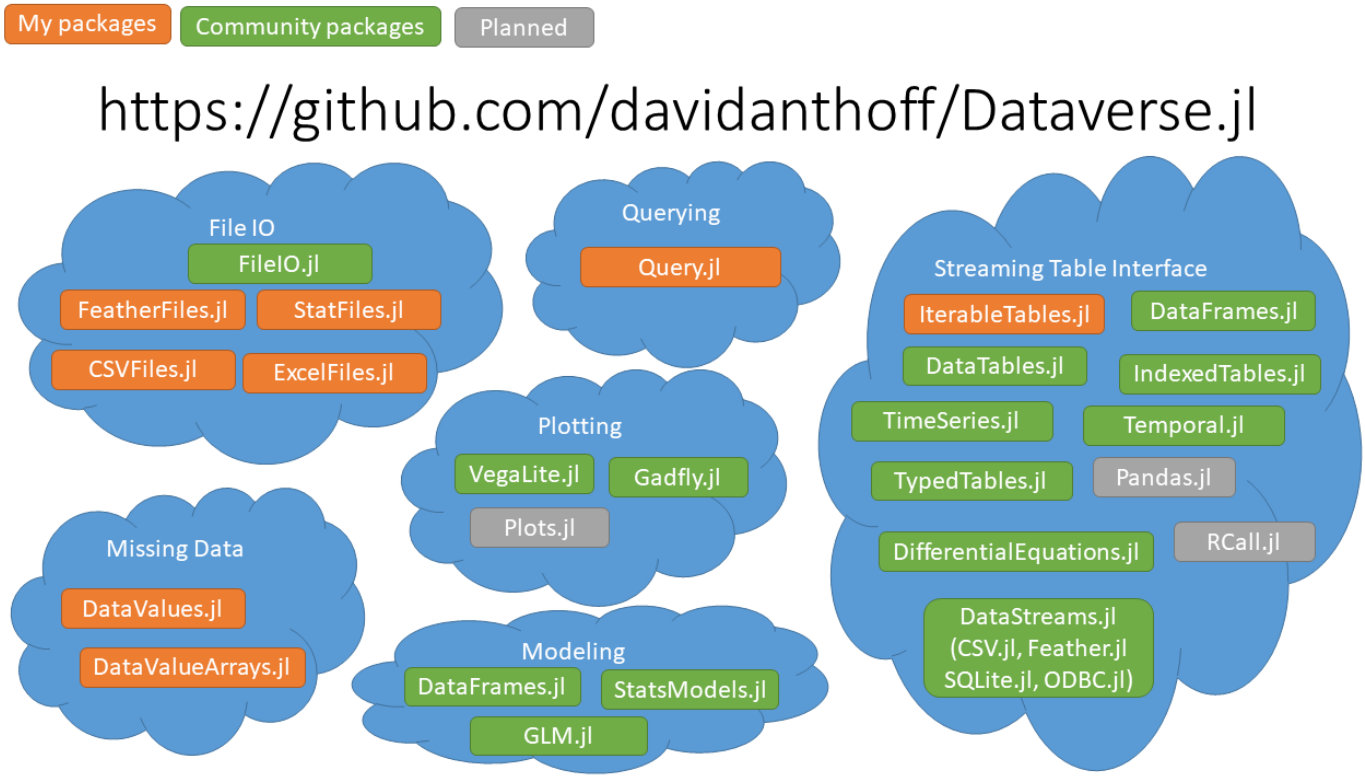
Outline

1. Feature show-off
2. Internals

Feature show-off

Intellectual debt

- .Net LINQ
- https://github.com/blackrock/NamedTuples.jl




```
In [2]: using Dataverse, RDatasets, DataFrames, DataTables, IndexedTables, TypedTables
```

Querying an array

```
In [3]: source = collect(1:20)
```

```
Out[3]: 20-element Array{Int64,1}:
          1
          2
          3
          4
          5
          6
          7
          8
          9
         10
         11
         12
         13
         14
         15
         16
         17
         18
         19
         20
```

In [4]:
```
@from i in source begin
    @where i%2==0
    @select i^2
    @collect
end
```

Out[4]:
```
10-element Array{Int64,1}:
   4
  16
  36
  64
 100
 144
 196
 256
 324
 400
```

Querying a DataFrame

In [5]:
```
df = dataset("ggplot2", "mpg")
```

Out[5]:

	Manufacturer	Model	Displ	Year	Cyl	Trans	Drv	Cty	Hwy	Fl	Class
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
4	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
7	audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact
8	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
9	audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
10	audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact
11	audi	a4 quattro	2.0	2008	4	auto(s6)	4	19	27	p	compact
12	audi	a4 quattro	2.8	1999	6	auto(l5)	4	15	25	p	compact
13	audi	a4 quattro	2.8	1999	6	manual(m5)	4	17	25	p	compact
14	audi	a4 quattro	3.1	2008	6	auto(s6)	4	17	25	p	compact
15	audi	a4 quattro	3.1	2008	6	manual(m6)	4	15	25	p	compact
16	audi	a6 quattro	2.8	1999	6	auto(l5)	4	15	24	p	midsize
17	audi	a6 quattro	3.1	2008	6	auto(s6)	4	17	25	p	midsize
18	audi	a6 quattro	4.2	2008	8	auto(s6)	4	16	23	p	midsize
19	chevrolet	c1500 suburban 2wd	5.3	2008	8	auto(l4)	r	14	20	r	suv
20	chevrolet	c1500 suburban 2wd	5.3	2008	8	auto(l4)	r	11	15	e	suv
21	chevrolet	c1500 suburban 2wd	5.3	2008	8	auto(l4)	r	14	20	r	suv
22	chevrolet	c1500 suburban 2wd	5.7	1999	8	auto(l4)	r	13	17	r	suv
23	chevrolet	c1500 suburban 2wd	6.0	2008	8	auto(l4)	r	12	17	r	suv
24	chevrolet	corvette	5.7	1999	8	manual(m6)	r	16	26	p	2seater
25	chevrolet	corvette	5.7	1999	8	auto(l4)	r	15	23	p	2seater

	Manufacturer	Model	Displ	Year	Cyl	Trans	Drv	Cty	Hwy	Fl	Class
26	chevrolet	corvette	6.2	2008	8	manual(m6)	r	16	26	p	2seater
27	chevrolet	corvette	6.2	2008	8	auto(s6)	r	15	25	p	2seater
28	chevrolet	corvette	7.0	2008	8	manual(m6)	r	15	24	p	2seater
29	chevrolet	k1500 tahoe 4wd	5.3	2008	8	auto(l4)	4	14	19	r	suv
30	chevrolet	k1500 tahoe 4wd	5.3	2008	8	auto(l4)	4	11	14	e	suv
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

In [6]:
```
@from i in df begin
    @where i.Manufacturer == "audi"
    @select i
    @collect DataFrame
end
```

Out[6]:

	Manufacturer	Model	Displ	Year	Cyl	Trans	Drv	Cty	Hwy	Fl	Class
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
4	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
7	audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact
8	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
9	audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
10	audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact
11	audi	a4 quattro	2.0	2008	4	auto(s6)	4	19	27	p	compact
12	audi	a4 quattro	2.8	1999	6	auto(l5)	4	15	25	p	compact
13	audi	a4 quattro	2.8	1999	6	manual(m5)	4	17	25	p	compact
14	audi	a4 quattro	3.1	2008	6	auto(s6)	4	17	25	p	compact
15	audi	a4 quattro	3.1	2008	6	manual(m6)	4	15	25	p	compact
16	audi	a6 quattro	2.8	1999	6	auto(l5)	4	15	24	p	midsize
17	audi	a6 quattro	3.1	2008	6	auto(s6)	4	17	25	p	midsize
18	audi	a6 quattro	4.2	2008	8	auto(s6)	4	16	23	p	midsize

In [7]:
```
@from i in df begin
    @where i.Manufacturer == "audi"
    @select {i.Model, years_since_95 = i.Year-1995, i.Trans}
    @collect DataFrame
end
```

Out[7]:

	Model	years_since_95	Trans
1	a4	4	auto(l5)
2	a4	4	manual(m5)
3	a4	13	manual(m6)
4	a4	13	auto(av)
5	a4	4	auto(l5)
6	a4	4	manual(m5)
7	a4	13	auto(av)
8	a4 quattro	4	manual(m5)
9	a4 quattro	4	auto(l5)
10	a4 quattro	13	manual(m6)
11	a4 quattro	13	auto(s6)
12	a4 quattro	4	auto(l5)
13	a4 quattro	4	manual(m5)
14	a4 quattro	13	auto(s6)
15	a4 quattro	13	manual(m6)
16	a6 quattro	4	auto(l5)
17	a6 quattro	13	auto(s6)
18	a6 quattro	13	auto(s6)

In [8]:
```
@from i in df begin
    @where i.Manufacturer == "audi"
    @orderby descending(i.Year), i.Model, i.Trans
    @select i
    @collect DataFrame
end
```

Out[8]:

	Manufacturer	Model	Displ	Year	Cyl	Trans	Drv	Cty	Hwy	Fl	Class
1	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
2	audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
4	audi	a4 quattro	2.0	2008	4	auto(s6)	4	19	27	p	compact
5	audi	a4 quattro	3.1	2008	6	auto(s6)	4	17	25	p	compact
6	audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact
7	audi	a4 quattro	3.1	2008	6	manual(m6)	4	15	25	p	compact
8	audi	a6 quattro	3.1	2008	6	auto(s6)	4	17	25	p	midsize
9	audi	a6 quattro	4.2	2008	8	auto(s6)	4	16	23	p	midsize
10	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
11	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
12	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
13	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
14	audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
15	audi	a4 quattro	2.8	1999	6	auto(l5)	4	15	25	p	compact
16	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
17	audi	a4 quattro	2.8	1999	6	manual(m5)	4	17	25	p	compact
18	audi	a6 quattro	2.8	1999	6	auto(l5)	4	15	24	p	midsize

In [9]:
```
@from i in df begin
    @group i.Cty by i.Manufacturer into g
    @select {Manufacturer = g.key, Cty = mean(g)}
    @collect DataFrame
end
```

Out[9]:

	Manufacturer	Cty
1	audi	17.61111111111111
2	chevrolet	15.0
3	dodge	13.135135135135135
4	ford	14.0
5	honda	24.444444444444443
6	hyundai	18.642857142857142
7	jeep	13.5
8	land rover	11.5
9	lincoln	11.333333333333334
10	mercury	13.25
11	nissan	18.076923076923077
12	pontiac	17.0
13	subaru	19.285714285714285
14	toyota	18.529411764705884
15	volkswagen	20.925925925925927

File IO

In [10]:
```
q = @from i in df begin
    @group i.Cty by i.Manufacturer into g
    @select {Manufacturer = g.key, Cty = mean(g)}
end
save("average_mpg.csv", q)
```

In [11]:
```
dt = DataTable(load("average_mpg.csv"))
```

Out[11]:

	Manufacturer	Cty
1	audi	17.6111
2	chevrolet	15.0
3	dodge	13.1351
4	ford	14.0
5	honda	24.4444
6	hyundai	18.6429
7	jeep	13.5
8	land rover	11.5
9	lincoln	11.3333
10	mercury	13.25
11	nissan	18.0769
12	pontiac	17.0
13	subaru	19.2857
14	toyota	18.5294
15	volkswagen	20.9259

In [12]:
```
save("average_mpg.feather", dt);
```

Tabular File IO (via FileIO.jl)

load

File extension	Implementation package
.csv	CSVFiles.jl, TextParse.jl
.xls .xlsx	ExcelFiles.jl, ExcelReader.jl
.feather	FeatherFiles.jl, Feather.jl
.dta .sas7bdat .sav	StatFiles.jl, ReadStat.jl

save

File extension	Implementation package
.csv	CSVFiles.jl
.feather	FeatherFiles.jl, Feather.jl

More data structures

In [13]:
```
avg_mpg = @from i in dt begin
    @select i.Manufacturer => i.Cty
    @collect Dict
end
```

Out[13]:
```
Dict{String,Float64} with 15 entries:
  "jeep"       => 13.5
  "hyundai"    => 18.6429
  "lincoln"    => 11.3333
  "mercury"    => 13.25
  "nissan"     => 18.0769
  "toyota"     => 18.5294
  "subaru"     => 19.2857
  "ford"       => 14.0
  "land rover" => 11.5
  "audi"       => 17.6111
  "dodge"      => 13.1351
  "honda"      => 24.4444
  "volkswagen" => 20.9259
  "pontiac"    => 17.0
  "chevrolet"  => 15.0
```

In [14]:
```
@from i in avg_mpg begin
    @where i.second > 20
    @select i.first
    @collect
end
```

Out[14]:
```
2-element Array{String,1}:
 "honda"
 "volkswagen"
```

In [15]:
```
@from i in df begin
    @join j in avg_mpg on i.Manufacturer equals j.first
    @select {i.Manufacturer, i.Year, avg_mpg = j.second}
    @collect IndexedTable
end
```

Out[15]:

Manufacturer	Year	avg_mpg
"audi"	1999	17.6111
"audi"	1999	17.6111
"audi"	1999	17.6111
"audi"	1999	17.6111
"audi"	1999	17.6111
"audi"	1999	17.6111
"audi"	1999	17.6111
"audi"	1999	17.6111
"audi"	1999	17.6111
"audi"	2008	17.6111
⋮		
"volkswagen"	2008	20.9259
"volkswagen"	2008	20.9259
"volkswagen"	2008	20.9259
"volkswagen"	2008	20.9259
"volkswagen"	2008	20.9259
"volkswagen"	2008	20.9259
"volkswagen"	2008	20.9259
"volkswagen"	2008	20.9259
"volkswagen"	2008	20.9259
"volkswagen"	2008	20.9259

IterableTables.jl

Source & Sink

- DataFrames.jl
- DataStreams.jl (CSV.jl, Feather.jl, SQLite.jl, ODBC.jl)
- DataTables.jl
- IndexedTables.jl
- TimeSeries.jl
- Temporal.jl
- TypedTables.jl

Sources

- Any named tuple iterator
- DifferentialEquations.jl

Sinks

- Gadfly.jl
- VegaLite.jl
- StatsModels.jl (→ GLM.jl)

Piping

In [16]:
```
load("average_mpg.csv") |>
save("testfile2.feather")
```

Out[16]:
```
Feather.Sink(Feather.Metadata.CTable("", 15, Feather.Metadata.Column[Feather.
Metadata.Column("Manufacturer", Feather.Metadata.PrimitiveArray(UTF8, PLAIN,
 8, 15, 0, 168), 0, nothing, ""), Feather.Metadata.Column("Cty", Feather.Meta
data.PrimitiveArray(DOUBLE, PLAIN, 176, 15, 0, 120), 0, nothing, "")], 2,
 ""), "C:\\Users\\david\\Google Drive\\Talks\\2017 juliacon\\testfile2.feathe
r", IOBuffer(data=UInt8[...], readable=true, writable=true, seekable=true, ap
pend=false, size=0, maxsize=Inf, ptr=1, mark=-1), "", "", 15×2 DataFrames.Dat
aFrame
```

Row	Manufacturer	Cty
1	"audi"	17.6111
2	"chevrolet"	15.0
3	"dodge"	13.1351
4	"ford"	14.0
5	"honda"	24.4444
6	"hyundai"	18.6429
7	"jeep"	13.5
8	"land rover"	11.5
9	"lincoln"	11.3333
10	"mercury"	13.25
11	"nissan"	18.0769
12	"pontiac"	17.0
13	"subaru"	19.2857
14	"toyota"	18.5294
15	"volkswagen"	20.9259

)

In [17]:
```
df |> save("testfile2.csv")
```

In [18]:
```
df |> DataTable |> DataFrame |>
@query(i, begin
    @where i.Class=="compact"
    @select i
end) |>
@sub(save("testfile3.csv")) |> IndexedTable
```

Out[18]:

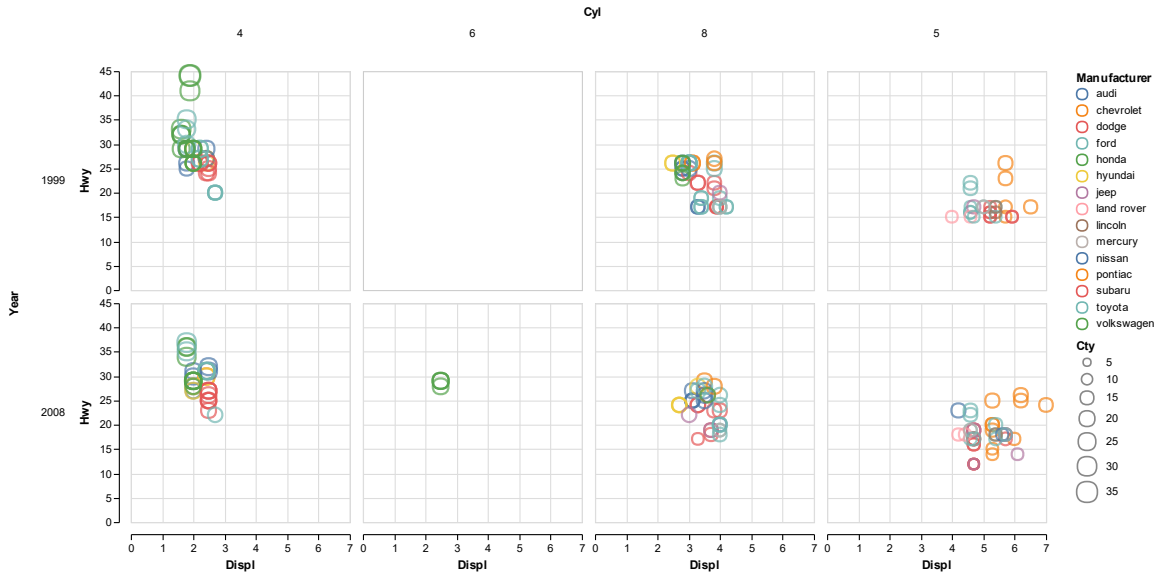
```
Manufacturer  Model         Displ  Year  Cyl  Trans         Drv  Cty  Hwy  Fl
  │ Class
──────────────────────────────────────────────────────────────────────────────
──┼──────────
"audi"        "a4"          1.8    1999  4    "auto(l5)"    "f"  18   29
 "p" │ "compact"
"audi"        "a4"          1.8    1999  4    "manual(m5)"  "f"  21   29
 "p" │ "compact"
"audi"        "a4"          2.0    2008  4    "auto(av)"    "f"  21   30
 "p" │ "compact"
"audi"        "a4"          2.0    2008  4    "manual(m6)"  "f"  20   31
 "p" │ "compact"
"audi"        "a4"          2.8    1999  6    "auto(l5)"    "f"  16   26
 "p" │ "compact"
"audi"        "a4"          2.8    1999  6    "manual(m5)"  "f"  18   26
 "p" │ "compact"
"audi"        "a4"          3.1    2008  6    "auto(av)"    "f"  18   27
 "p" │ "compact"
"audi"        "a4 quattro"  1.8    1999  4    "auto(l5)"    "4"  16   25
 "p" │ "compact"
"audi"        "a4 quattro"  1.8    1999  4    "manual(m5)"  "4"  18   26
 "p" │ "compact"
"audi"        "a4 quattro"  2.0    2008  4    "auto(s6)"    "4"  19   27
 "p" │ "compact"

  ⋮
"volkswagen"  "gti"         2.8    1999  6    "manual(m5)"  "f"  17   24
 "r" │ "compact"
"volkswagen"  "jetta"       1.9    1999  4    "manual(m5)"  "f"  33   44
 "d" │ "compact"
"volkswagen"  "jetta"       2.0    1999  4    "auto(l4)"    "f"  19   26
 "r" │ "compact"
"volkswagen"  "jetta"       2.0    1999  4    "manual(m5)"  "f"  21   29
 "r" │ "compact"
"volkswagen"  "jetta"       2.0    2008  4    "auto(s6)"    "f"  22   29
 "p" │ "compact"
"volkswagen"  "jetta"       2.0    2008  4    "manual(m6)"  "f"  21   29
 "p" │ "compact"
"volkswagen"  "jetta"       2.5    2008  5    "auto(s6)"    "f"  21   29
 "r" │ "compact"
"volkswagen"  "jetta"       2.5    2008  5    "manual(m5)"  "f"  21   29
 "r" │ "compact"
"volkswagen"  "jetta"       2.8    1999  6    "auto(l4)"    "f"  16   23
 "r" │ "compact"
"volkswagen"  "jetta"       2.8    1999  6    "manual(m5)"  "f"  17   24
 "r" │ "compact"
```

Plotting

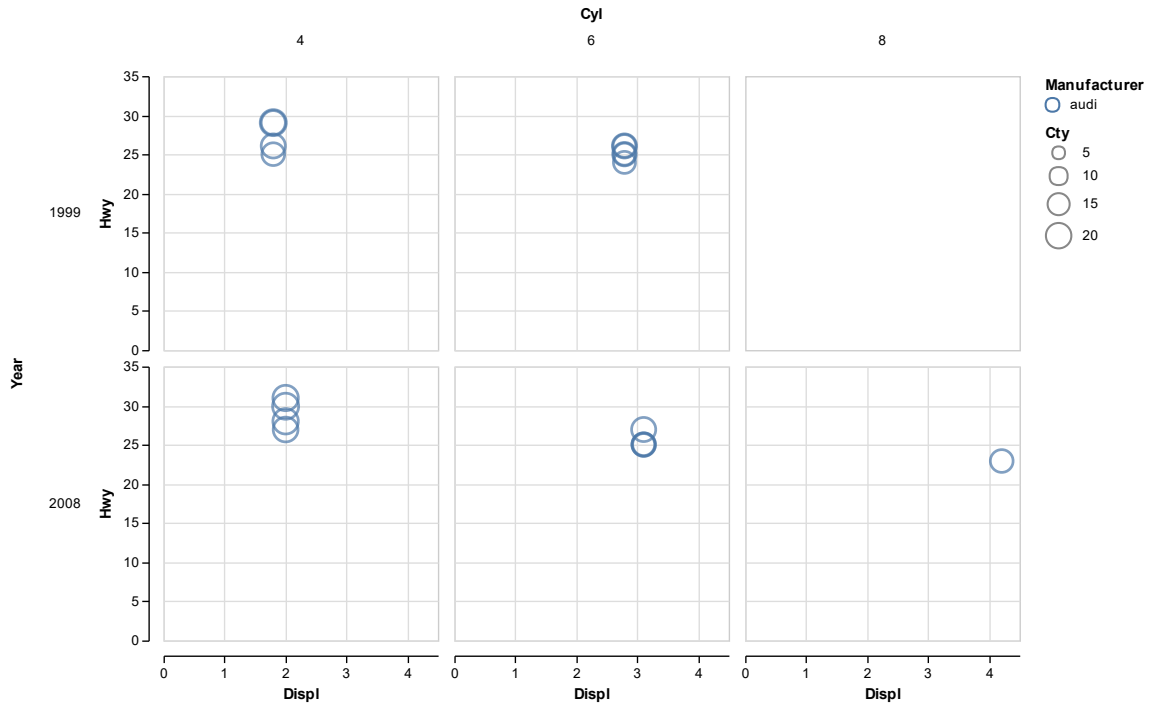
In [19]:
```
df |>
vlplot() |>
mark_point() |>
encoding_column_ord(:Cyl) |>
encoding_row_ord(:Year) |>
encoding_x_quant(:Displ) |>
encoding_y_quant(:Hwy) |>
encoding_size_quant(:Cty) |>
encoding_color_nominal(:Manufacturer)
```

Out[19]:



In [20]:
```
df |>
@query(i, begin
    @where i.Manufacturer == "audi"
    @select i
end) |>
vlplot() |>
mark_point() |>
encoding_column_ord(:Cyl) |>
encoding_row_ord(:Year) |>
encoding_x_quant(:Displ) |>
encoding_y_quant(:Hwy) |>
encoding_size_quant(:Cty) |>
encoding_color_nominal(:Manufacturer)
```

Out[20]:



In [21]:
```
df |>
@query(i, begin
    @where i.Manufacturer == "audi"
    @select i
end) |>
vlplot() |>
mark_point() |>
encoding_column_ord(:Cyl) |>
encoding_row_ord(:Year) |>
encoding_x_quant(:Displ) |>
encoding_y_quant(:Hwy) |>
encoding_size_quant(:Cty) |>
encoding_color_nominal(:Manufacturer) |>
save("graph.svg")
```

In [22]:
```
df |>
@query(i, begin
    @where i.Trans=="auto(l5)"
    @group i by i.Manufacturer into g
    @let p = (vlplot(g) |> mark_point() |> encoding_x_ord(:Year) |> encoding_y
_quant(:Hwy))
    @select {filename="$(g.key).pdf", plot=p}
end ) |>
res -> begin
    for i in res
        save(i.filename, i.plot)
    end
end
```

```
WARNING: Mapping to the storage type failed; perhaps your data had out-of-ran
ge values?
Try `map(clamp01nan, img)` to clamp values to a valid range.
WARNING: Mapping to the storage type failed; perhaps your data had out-of-ran
ge values?
Try `map(clamp01nan, img)` to clamp values to a valid range.
WARNING: Mapping to the storage type failed; perhaps your data had out-of-ran
ge values?
Try `map(clamp01nan, img)` to clamp values to a valid range.
WARNING: Mapping to the storage type failed; perhaps your data had out-of-ran
ge values?
Try `map(clamp01nan, img)` to clamp values to a valid range.
WARNING: Mapping to the storage type failed; perhaps your data had out-of-ran
ge values?
Try `map(clamp01nan, img)` to clamp values to a valid range.
WARNING: Mapping to the storage type failed; perhaps your data had out-of-ran
ge values?
Try `map(clamp01nan, img)` to clamp values to a valid range.
WARNING: Mapping to the storage type failed; perhaps your data had out-of-ran
ge values?
Try `map(clamp01nan, img)` to clamp values to a valid range.
WARNING: Mapping to the storage type failed; perhaps your data had out-of-ran
ge values?
Try `map(clamp01nan, img)` to clamp values to a valid range.
WARNING: Mapping to the storage type failed; perhaps your data had out-of-ran
ge values?
Try `map(clamp01nan, img)` to clamp values to a valid range.
WARNING: Mapping to the storage type failed; perhaps your data had out-of-ran
ge values?
Try `map(clamp01nan, img)` to clamp values to a valid range.
```

Internals

User syntax

LINQ style

```
x = @from i in source begin
    @where i.age > 40
    @select {i.Firstname, i.Lastname}
    @collect DataFrame
end
```

dplyr style (julia 1.0 timeframe)

```
x = source |> @filter(age>40) |>
    @select(Firstname, Lastname) |> DataFrame
```

Generator style (maybe julia 2.0+)

```
x = DataFrame({i.Firstname, i.Lastname} for
    i in source if i.age > 40)
```

Method interface

```
DataFrame(
          select_internal(
                          where_internal(
                                         query(source),
                                         i->i.age > 40),
                          i->@NT(Firstname=i.Firstname, Lastname=i.Lastname)))
```

Backends

Iterator based

- Chained iterators
- Main current implementation
- Works with a lot of sources
- Doesn't exploit source data structure

Query translation based (prototype)

- Builds an AST of the query
- A source can translate this into e.g. SQL
- Barely functional SQLite prototype

Lots of potential!

IndexedTables.jl
JuliaDB.jl
DryadLINQ
PLINQ

SQL

In [23]:
```
using SQLite
```

In [24]:
```
db = SQLite.DB(joinpath(Pkg.dir("SQLite"), "test", "Chinook_Sqlite.sqlite"))
```

Out[24]:
```
SQLite.DB("C:\Users\david\.julia\v0.6\SQLite\test\Chinook_Sqlite.sqlite")
```

In [25]:
```
t = Query.table(db, "Employee")
```

Out[25]:
```
Query.SQLiteTable(SQLite.DB("C:\Users\david\.julia\v0.6\SQLite\test\Chinook_S
qlite.sqlite"), "Employee")
```

In [26]:
```
@from i in t begin
    @where i.ReportsTo==2
    @select {Name=i.LastName, Adr=i.Address}
    @collect DataFrame
end
```

```
SELECT LastName AS Name, Address AS Adr FROM Employee WHERE ReportsTo="2"
```

Out[26]:

	Name	Adr
1	Nullable{String}("Peacock")	Nullable{String}("1111 6 Ave SW")
2	Nullable{String}("Park")	Nullable{String}("683 10 Street SW")
3	Nullable{String}("Johnson")	Nullable{String}("7727B 41 Ave")

```
DataFrame(
        select_internal(
                where_internal(
                        query(source),
                        i->i.age > 40),
                i->@NT(Firstname=i.Firstname, Lastname=i.Lastname)))
```

```
query(source::IterSource)
→ <: BaseIterImpl
```

```
Query(source::SQLTable)
→ <: CustomSQLImpl
```

```
where_internal(source:: BaseIterImpl,…)
→ <: BaseIterImpl
```

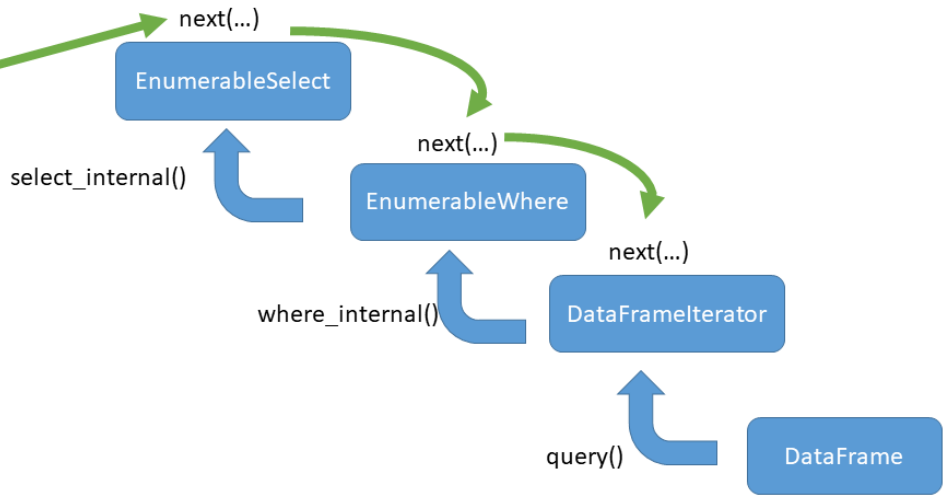
```
where_internal(source:: CustomSQLImpl,…)
→ <: CustomSQLImpl
```

```
select_internal(source:: BaseIterImpl,…)
→ <: BaseIterImpl
```

```
select_internal(source:: CustomSQLImpl,…)
→ <: CustomSQLImpl
```

```
DataFrame(
        select_internal(
                where_internal(
                        query(source),
                        i->i.age > 40),
                i->@NT(Firstname=i.Firstname, Lastname=i.Lastname)))
```

```
function DataFrame(…)
  …
  for row in source
    # Put into df
  end
end
```



```
where_internal(query(source), i -> i.age>40)
```

```
where_internal(query(source), i -> i.age>40, :( i -> i.age>40 ) )
```

Anonymous Function

Expression

Conclusion

Status

- Stable
- Tested
- Documented
- No major (or minor) redesign on the horizon

Performance

`size(data) == (100_000_000, 2)`

DataTables.jl

```
by(data, :B, d -> mean(d[:A]))
```

Query.jl

```
x = @from i in data begin
    @group i.A by i.B into g
    @select {m = mean(g)}
    @collect DataTable
end
```

Pandas.jl

```
mean(groupby(data, "B"))
```

Package	Runtime
DataTables.jl (PR #76)	5.4s
Query.jl	5.2s
Pandas.jl	2.7s

Pain points (and their solution)

- Dealing with columns
  → named tuples in base will solve that (julia 1.0)
- Boxing of named tuples in certain situations
  → #18632 (julia 1.0)
- Currently discussed julia 1.0 named tuple syntax `(;a=3, b=2)`
  → Begging Jeff
- Current (official?) missing values plan `Union{T,Null}`
  → DataValues.jl

Future

- dplyr like interface
- Performance of iterator based implementation
- More operators
- Please help and contribute, this is a big project!

Thank you!

David Anthoff
anthoff@berkeley.edu
www.david-anthoff.com