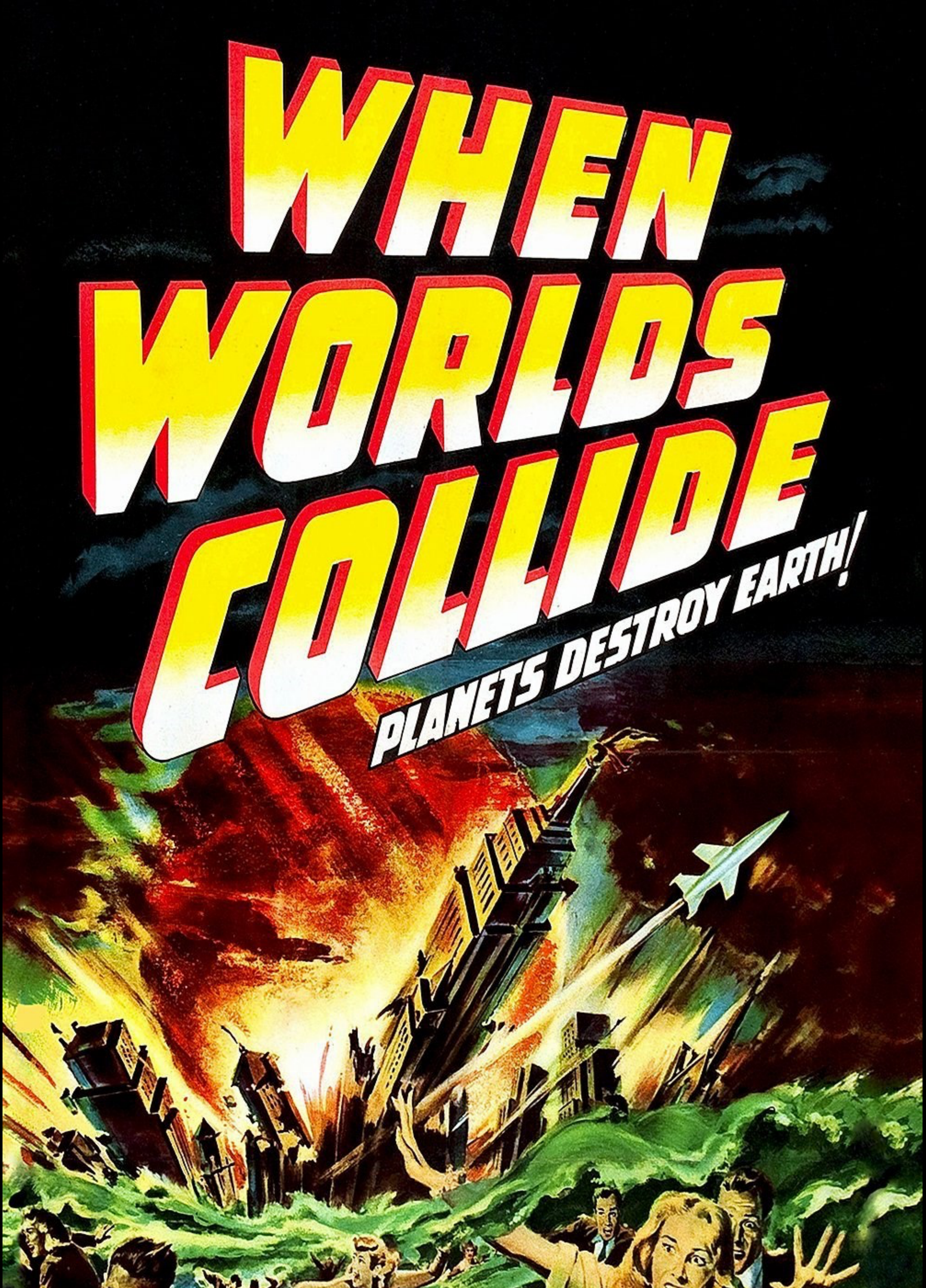
Disposable Databases for Development

Joe Francis

or Developing Against Shared Resources is Evil

(alternate working title)


WHEN
WORLDS
COLLIDE
PLANETS DESTROY EARTH!

Silver Bullets

Just kidding, there aren't

QUIT TESTING IN PRODUCTION
Troll.me

SafePlace℠

SHACKLES
SHACKLES EVERYWHERE
memegenerator.net

I FIND YOUR LACK OF AUTOMATION
DISTURBING
imgflip.com

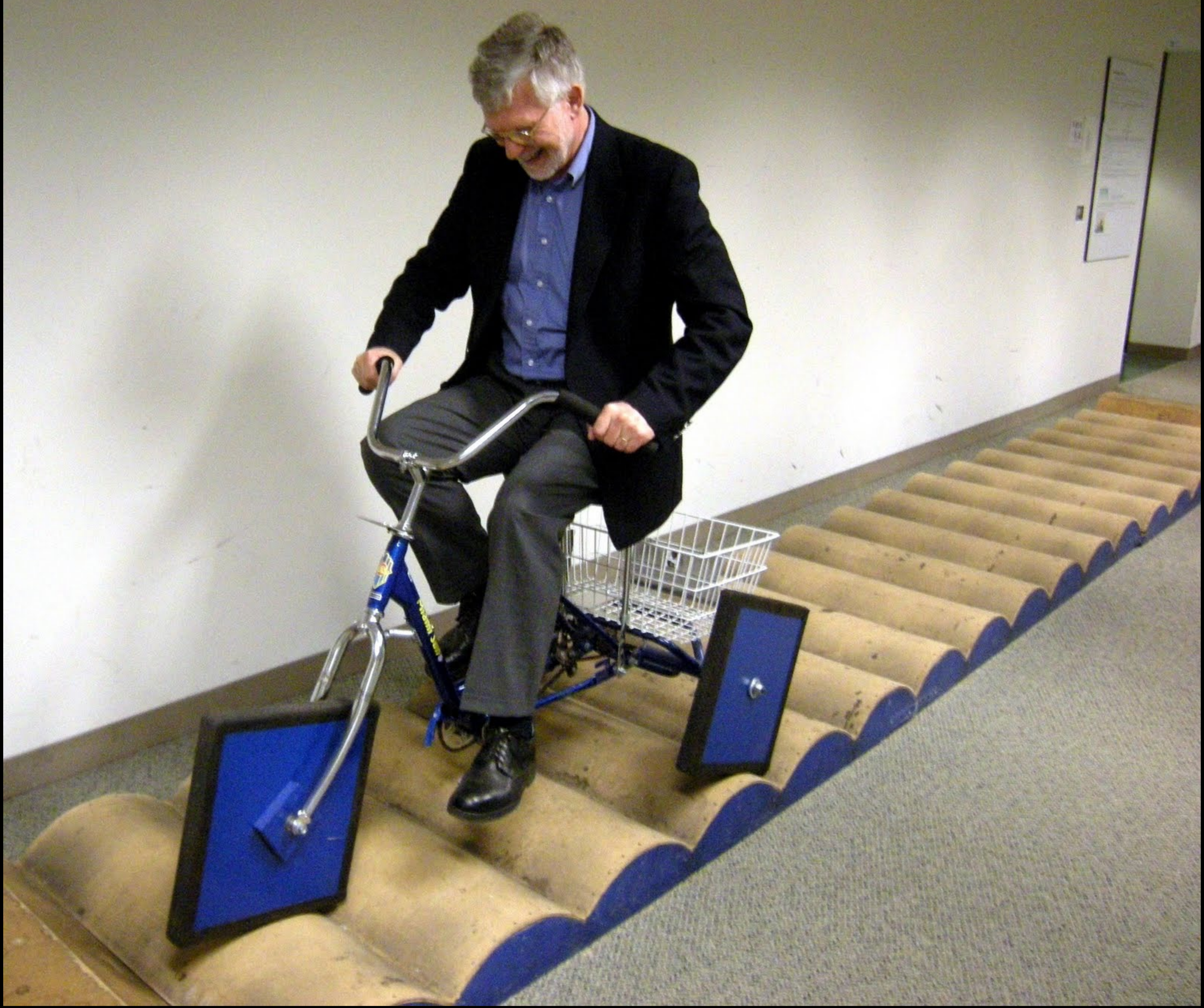
STOP/RESET

Debug like a scientist

⚠ WARNING

MAY CAUSE
NAUSEA, CONSTIPATION,
DIARRHEA, VOMITING,
ADDICTION AND
EVEN DEATH

```
$ mysqldump -h prod_db app_prod | mysql app_dev
```



What's wrong with everyone else's wheels?

May the bridges I burn light the way.

HARFORD COUNTY
maryland grown
SUPPORT HARFORD'S FARMS
BUY LOCAL
www.HarfordFarms.com

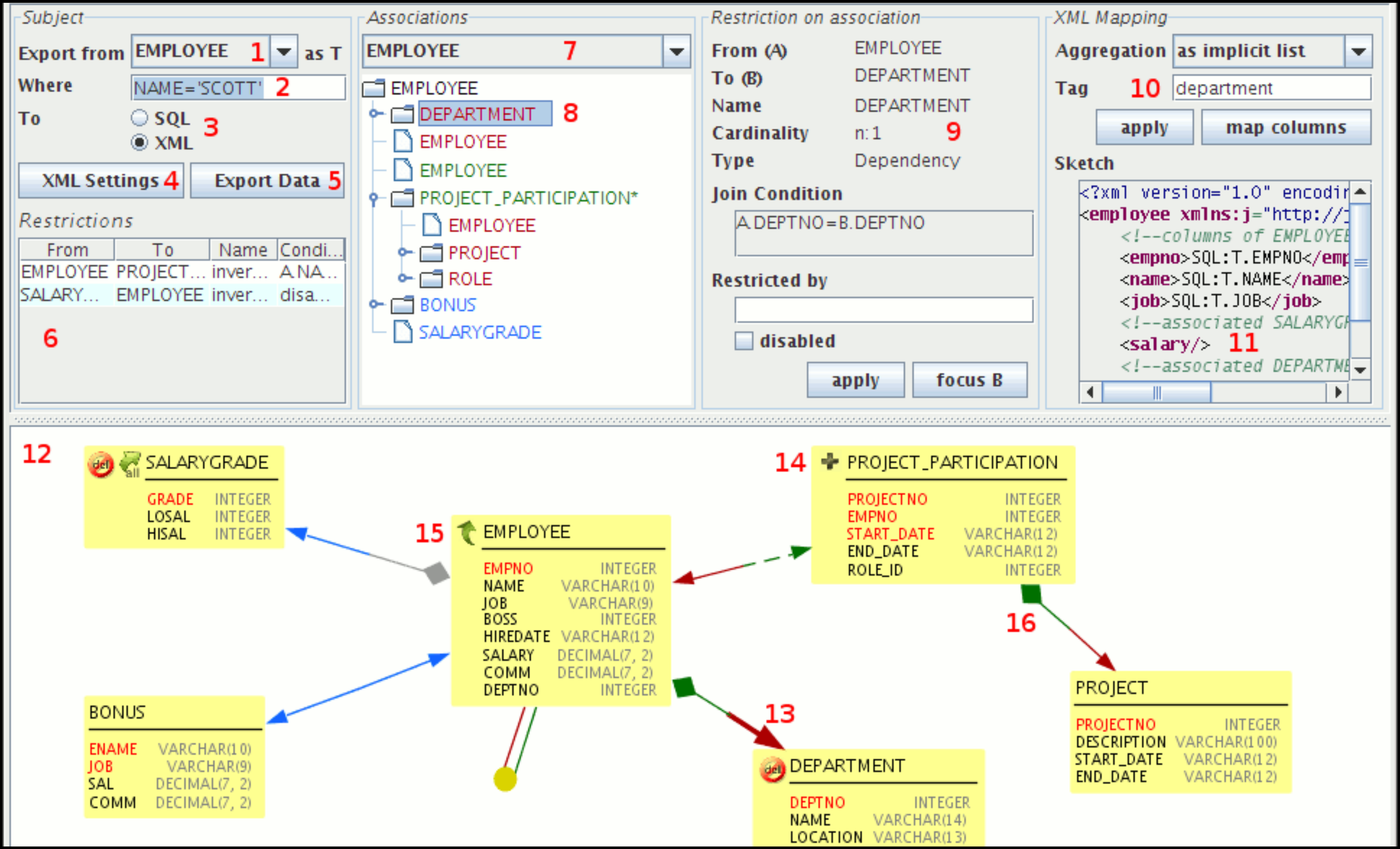
Work Local

Seed Your Database

Sidebar: is this all really a code smell?

Seed Data Wish List

- Referentially intact(-ish)
- Usefully broad
- Small and fast
- Quick and easy to refresh



From	To	Name	Condi...
EMPLOYEE	PROJECT...	inver...	A.NA...
SALARY...	EMPLOYEE	inver...	disa...

```
EMPLOYEE
  DEPARTMENT
    EMPLOYEE
    EMPLOYEE
  PROJECT_PARTICIPATION*
    EMPLOYEE
    PROJECT
    ROLE
  BONUS
  SALARYGRADE
```

From (A)	EMPLOYEE
To (B)	DEPARTMENT
Name	DEPARTMENT
Cardinality	n:1
Type	Dependency

Join Condition: A.DEPTNO=B.DEPTNO

Restricted by:

```
<?xml version="1.0" encodir
<employee xmlns:j="http://j
    <!--columns of EMPLOYEE
    <empno>SQL:T.EMPNO</emp
    <name>SQL:T.NAME</name>
    <job>SQL:T.JOB</job>
    <!--associated SALARYGR
    <salary/>
    <!--associated DEPARTME
```

SALARYGRADE: GRADE INTEGER; LOSAL INTEGER; HISAL INTEGER

EMPLOYEE: EMPNO INTEGER; NAME VARCHAR(10); JOB VARCHAR(9); BOSS INTEGER; HIREDATE VARCHAR(12); SALARY DECIMAL(7, 2); COMM DECIMAL(7, 2); DEPTNO INTEGER

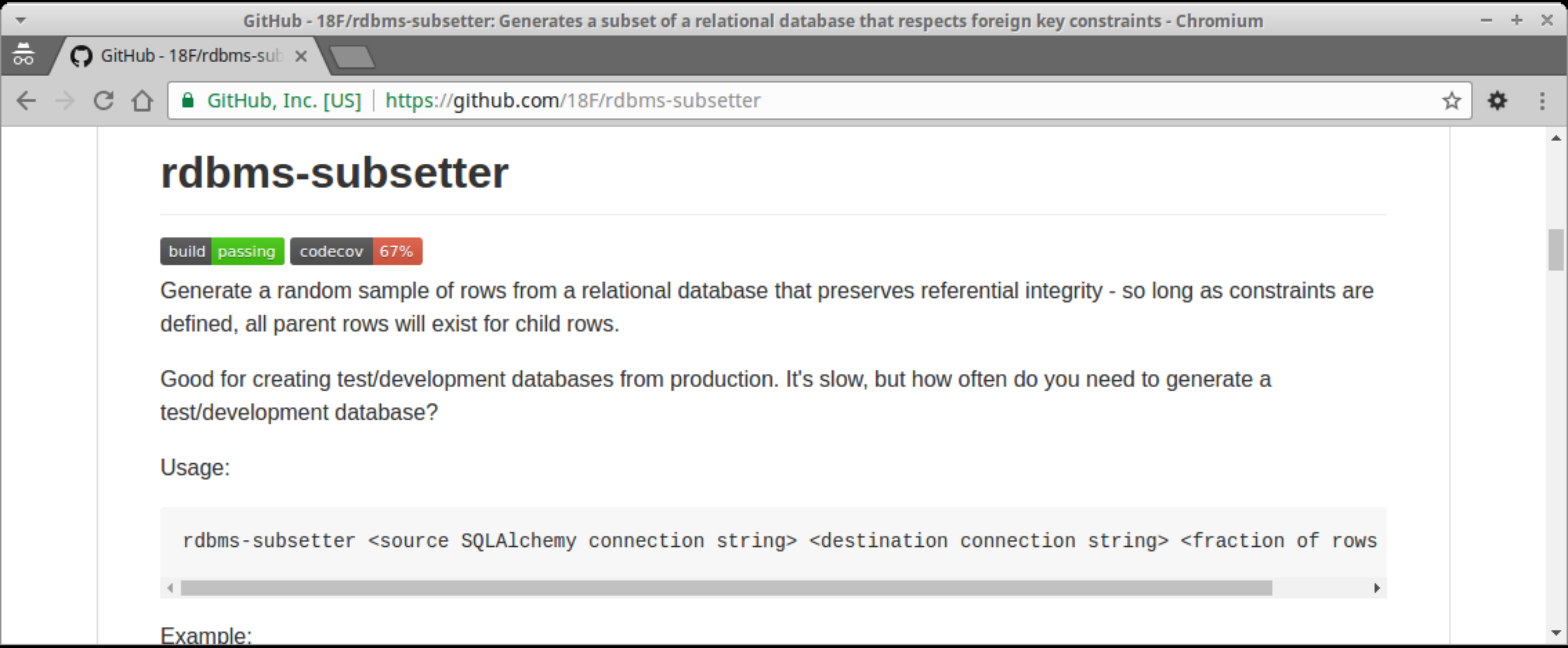
PROJECT_PARTICIPATION: PROJECTNO INTEGER; EMPNO INTEGER; START_DATE VARCHAR(12); END_DATE VARCHAR(12); ROLE_ID INTEGER

BONUS: ENAME VARCHAR(10); JOB VARCHAR(9); SAL DECIMAL(7, 2); COMM DECIMAL(7, 2)

DEPARTMENT: DEPTNO INTEGER; NAME VARCHAR(14); LOCATION VARCHAR(13)

PROJECT: PROJECTNO INTEGER; DESCRIPTION VARCHAR(100); START_DATE VARCHAR(12); END_DATE VARCHAR(12)

Jailer


GitHub - 18F/rdbms-subsetter: Generates a subset of a relational database that respects foreign key constraints - Chromium
GitHub - 18F/rdbms-sub
GitHub, Inc. [US] | https://github.com/18F/rdbms-subsetter
build passing
codecov 67%

rdbms-subsetter

Generate a random sample of rows from a relational database that preserves referential integrity - so long as constraints are defined, all parent rows will exist for child rows.

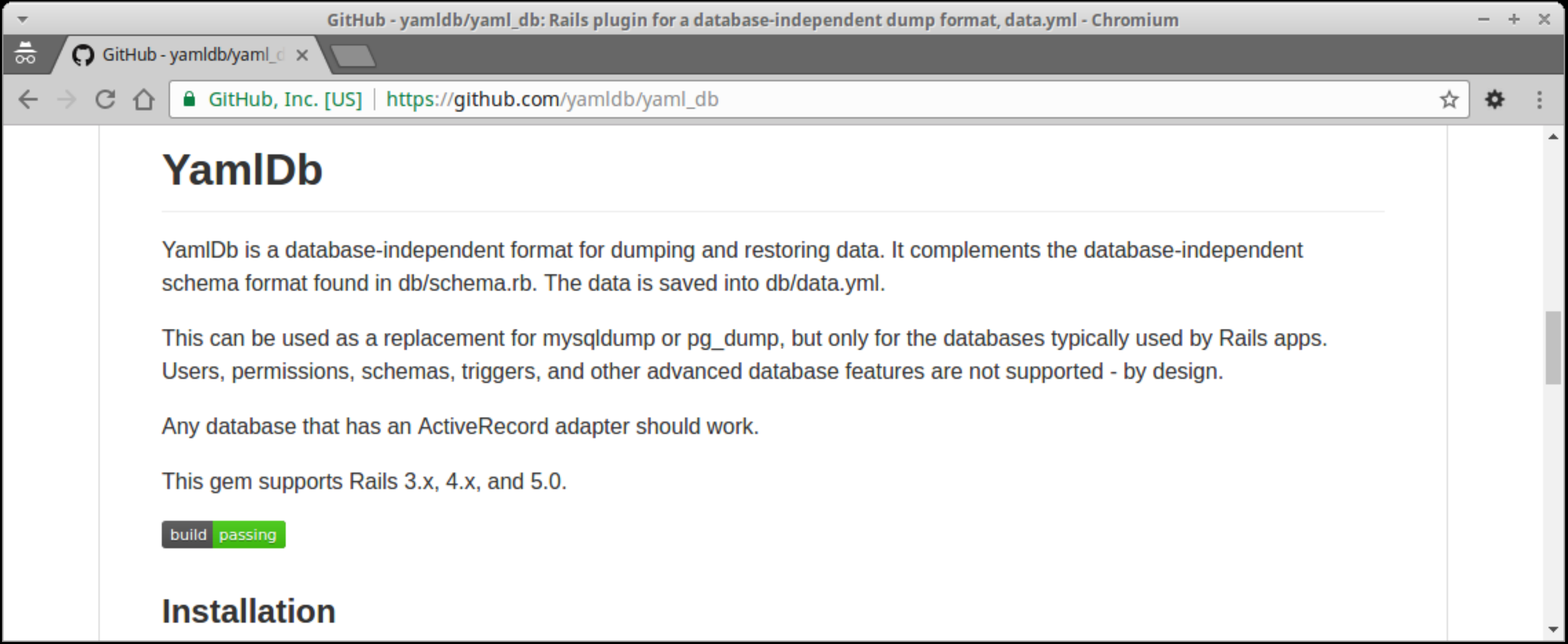
Good for creating test/development databases from production. It's slow, but how often do you need to generate a test/development database?

Usage:

```
rdbms-subsetter <source SQLAlchemy connection string> <destination connection string> <fraction of rows
```

Example:

rdbms-subsetter


GitHub - yamldb/yaml_db: Rails plugin for a database-independent dump format, data.yml - Chromium
GitHub - yamldb/yaml_d
GitHub, Inc. [US]
https://github.com/yamldb/yaml_db
build passing

YamlDb

YamlDb is a database-independent format for dumping and restoring data. It complements the database-independent schema format found in db/schema.rb. The data is saved into db/data.yml.

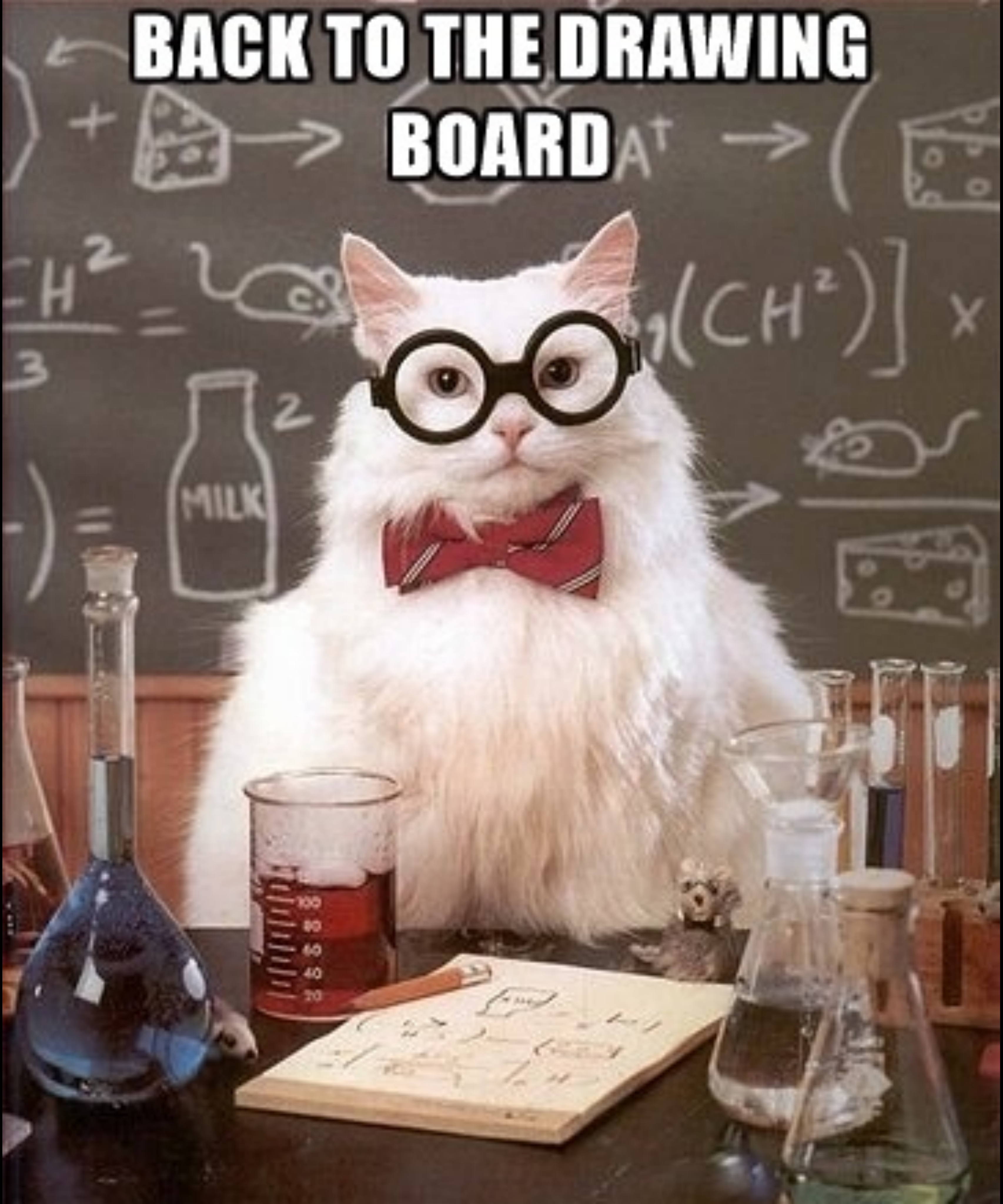
This can be used as a replacement for mysqldump or pg_dump, but only for the databases typically used by Rails apps. Users, permissions, schemas, triggers, and other advanced database features are not supported - by design.

Any database that has an ActiveRecord adapter should work.

This gem supports Rails 3.x, 4.x, and 5.0.

Installation

yaml_db


BACK TO THE DRAWING BOARD
MILK

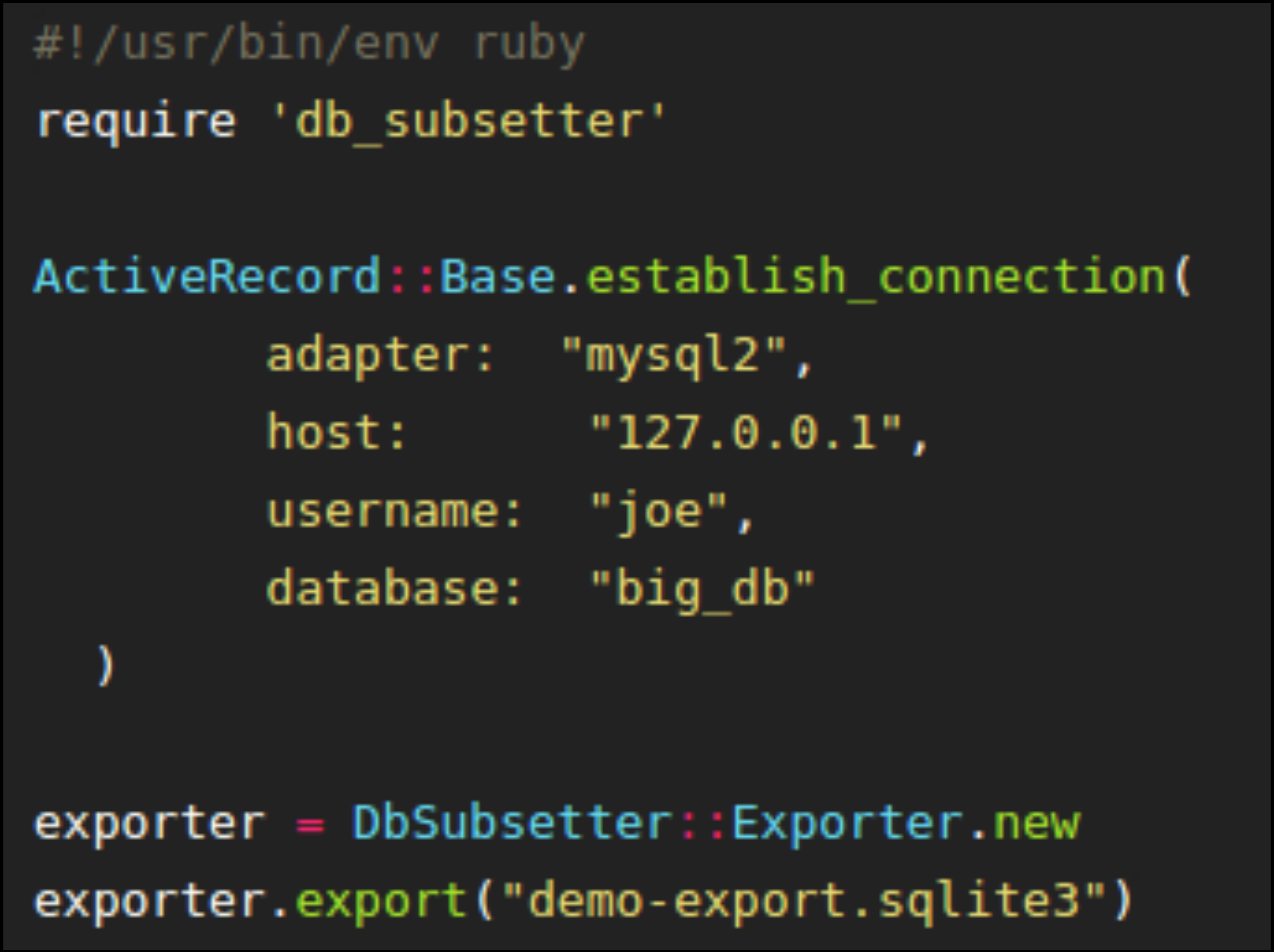
Found It!

Features

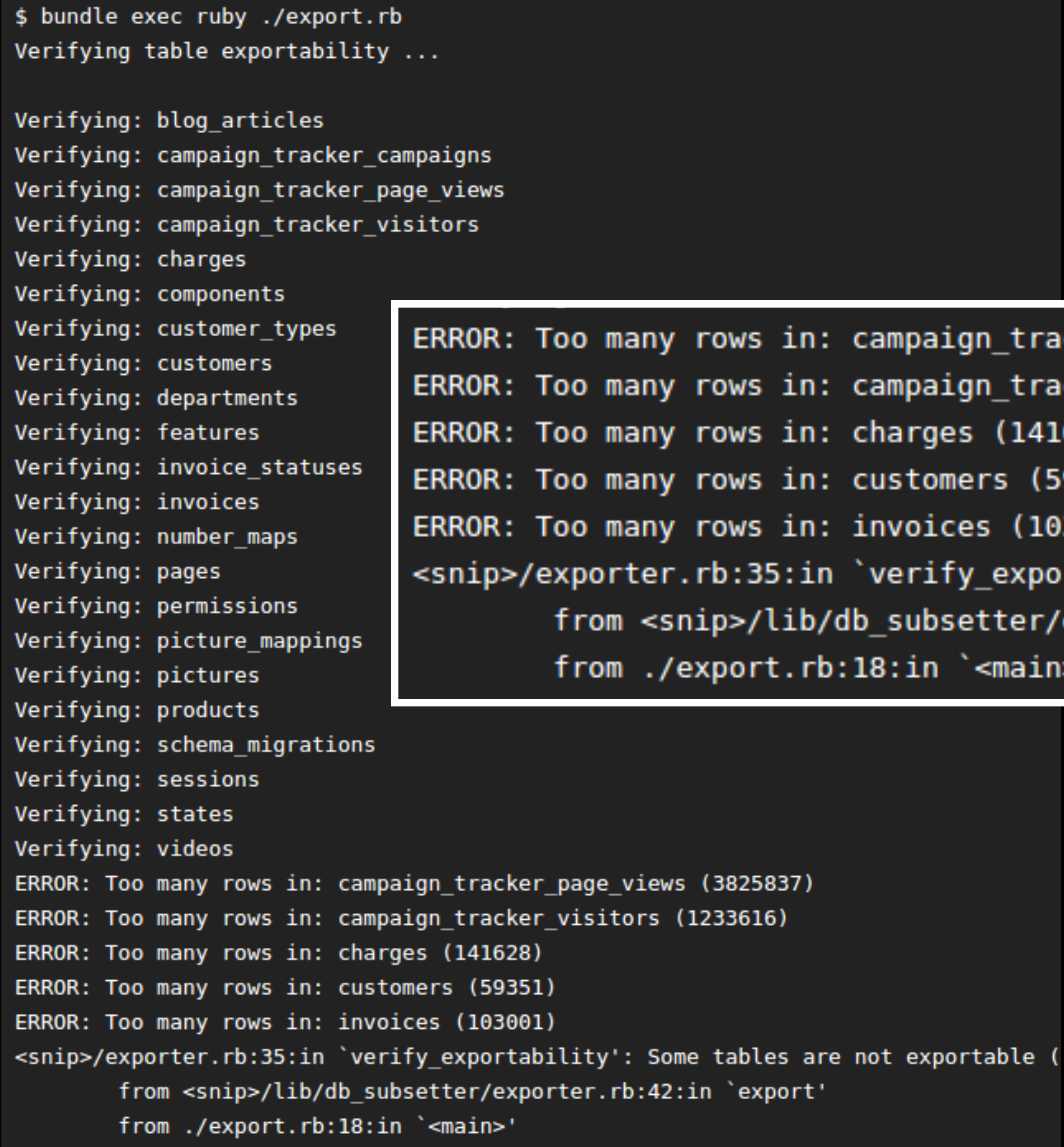
- DB agnostic
- Encode data as JSON
- Save via sqlite
- Random or not-random
- Tolerates imperfection

db_subsetter

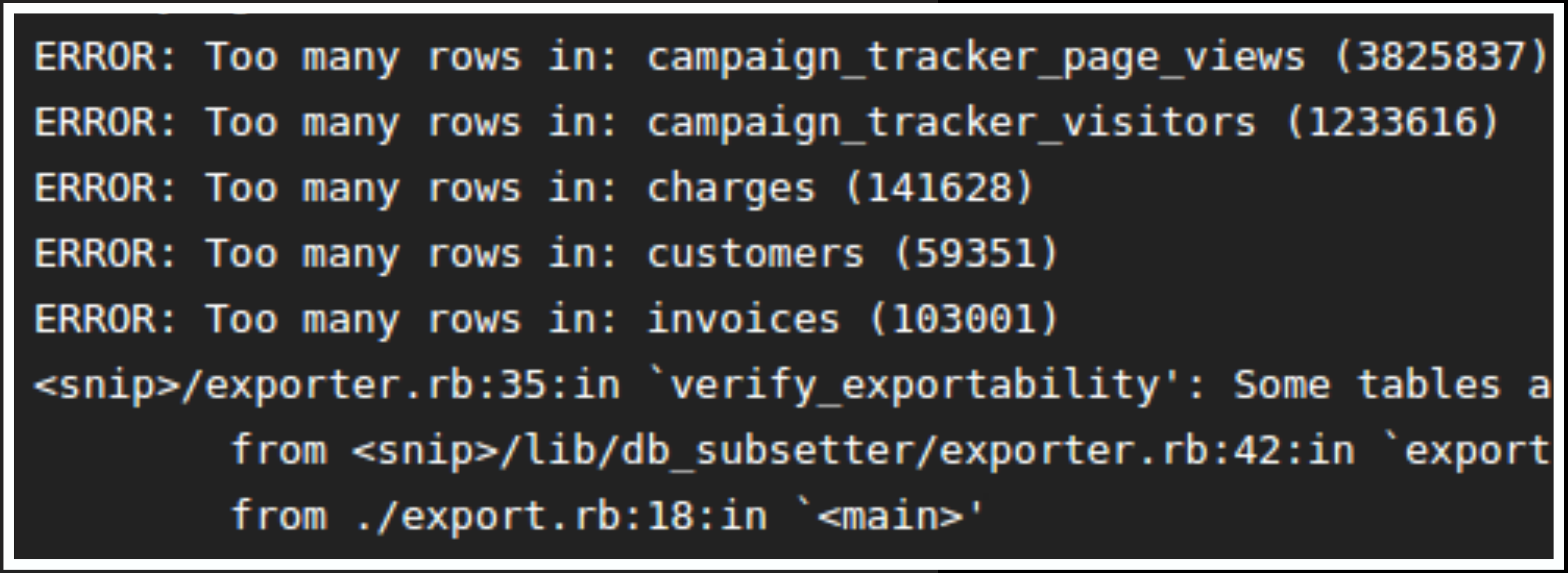
```
#Gemfile
source "https://rubygems.org"

gem "db_subsetter"
gem "mysql2"
```



```ruby
#!/usr/bin/env ruby
require 'db_subsetter'

ActiveRecord::Base.establish_connection(
        adapter:  "mysql2",
        host:      "127.0.0.1",
        username:  "joe",
        database:  "big_db"
  )

exporter = DbSubsetter::Exporter.new
exporter.export("demo-export.sqlite3")
```

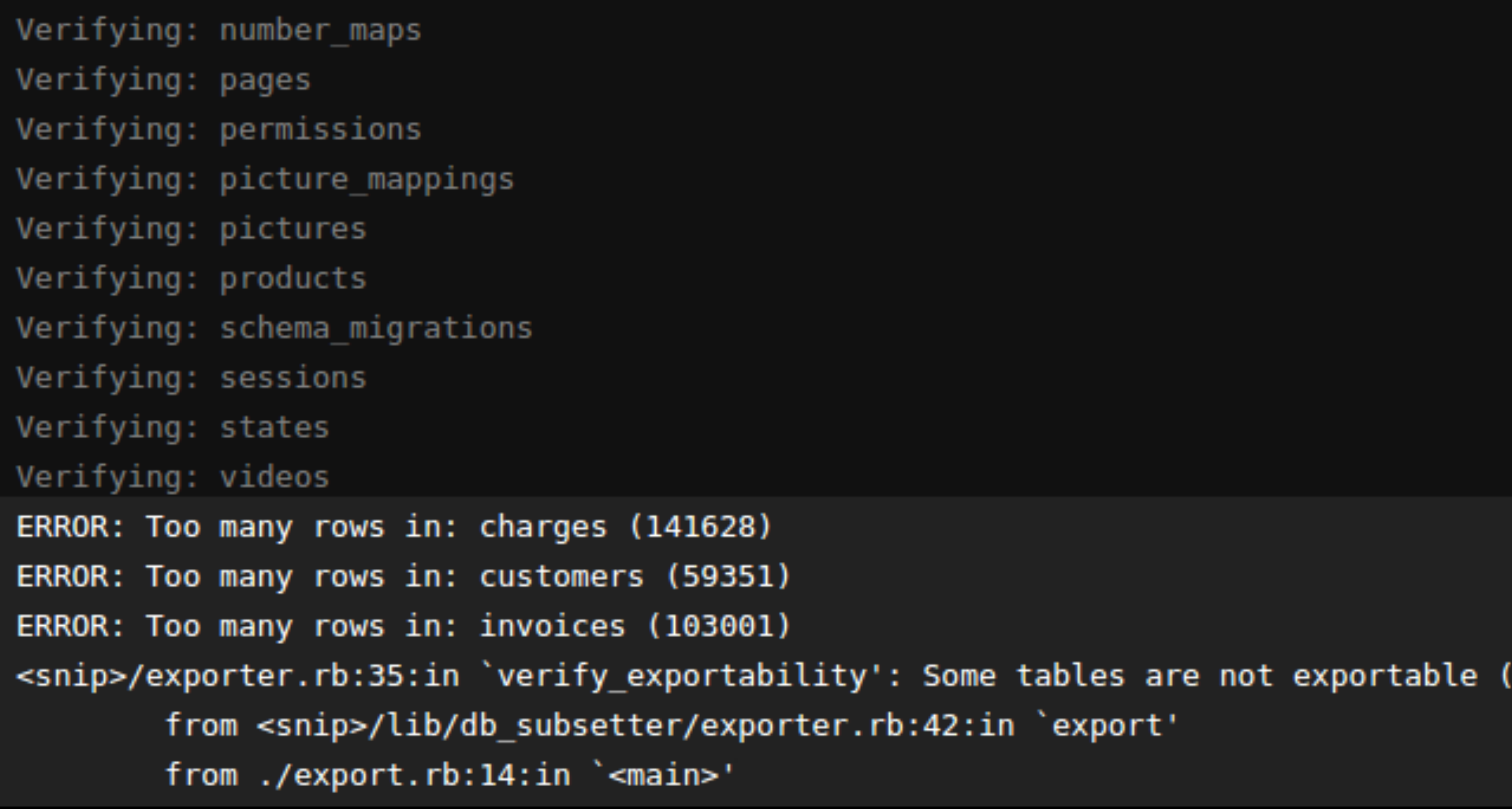


```
$ bundle exec ruby ./export.rb
Verifying table exportability ...

Verifying: blog_articles
Verifying: campaign_tracker_campaigns
Verifying: campaign_tracker_page_views
Verifying: campaign_tracker_visitors
Verifying: charges
Verifying: components
Verifying: customer_types
Verifying: customers
Verifying: departments
Verifying: features
Verifying: invoice_statuses
Verifying: invoices
Verifying: number_maps
Verifying: pages
Verifying: permissions
Verifying: picture_mappings
Verifying: pictures
Verifying: products
Verifying: schema_migrations
Verifying: sessions
Verifying: states
Verifying: videos
ERROR: Too many rows in: campaign_tracker_page_views (3825837)
ERROR: Too many rows in: campaign_tracker_visitors (1233616)
ERROR: Too many rows in: charges (141628)
ERROR: Too many rows in: customers (59351)
ERROR: Too many rows in: invoices (103001)
<snip>/exporter.rb:35:in `verify_exportability': Some tables are not exportable (
        from <snip>/lib/db_subsetter/exporter.rb:42:in `export'
        from ./export.rb:18:in `<main>'
```

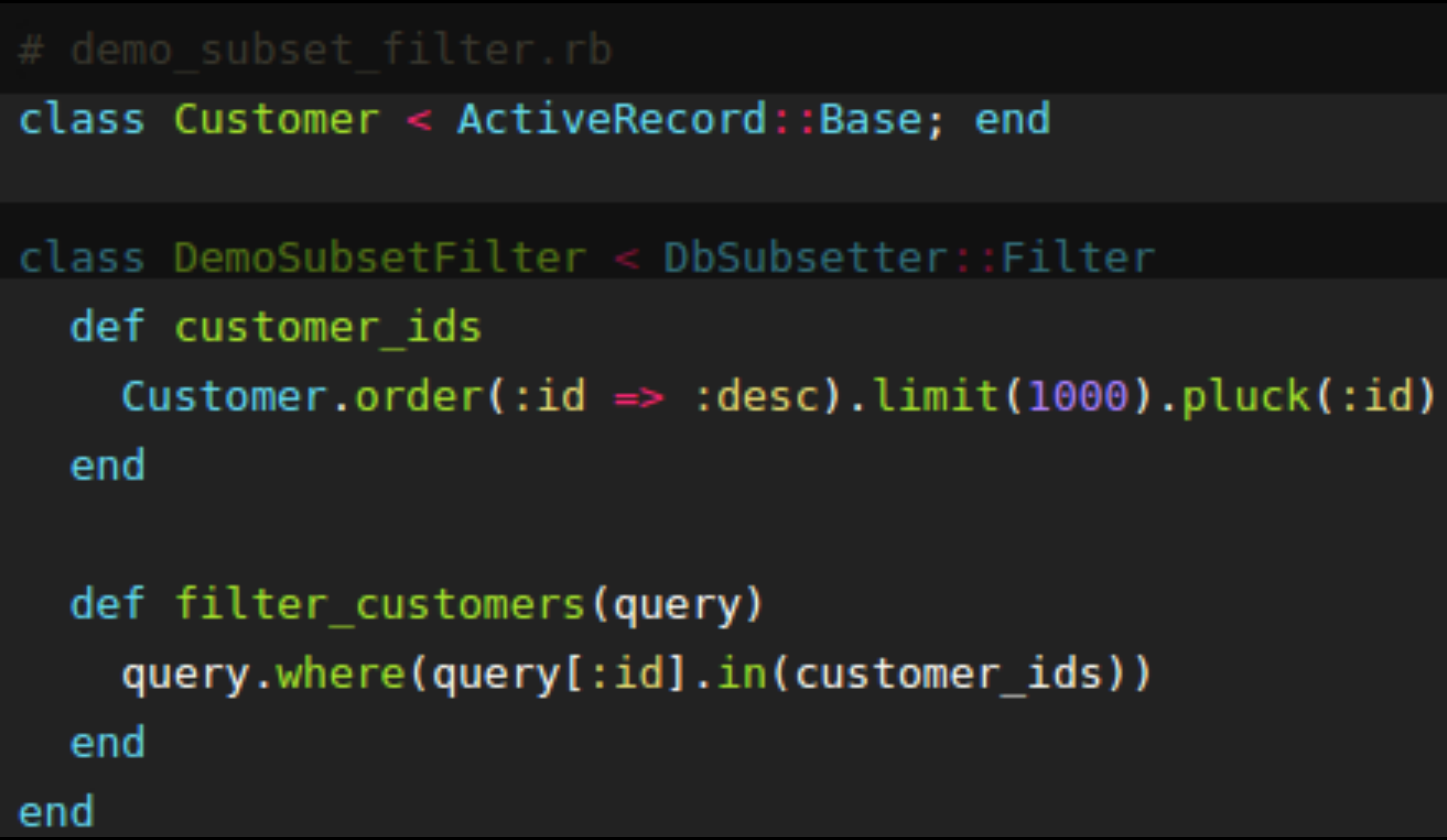


```
ERROR: Too many rows in: campaign_tracker_page_views (3825837)
ERROR: Too many rows in: campaign_tracker_visitors (1233616)
ERROR: Too many rows in: charges (141628)
ERROR: Too many rows in: customers (59351)
ERROR: Too many rows in: invoices (103001)
<snip>/exporter.rb:35:in `verify_exportability': Some tables a
        from <snip>/lib/db_subsetter/exporter.rb:42:in `export
        from ./export.rb:18:in `<main>'
```

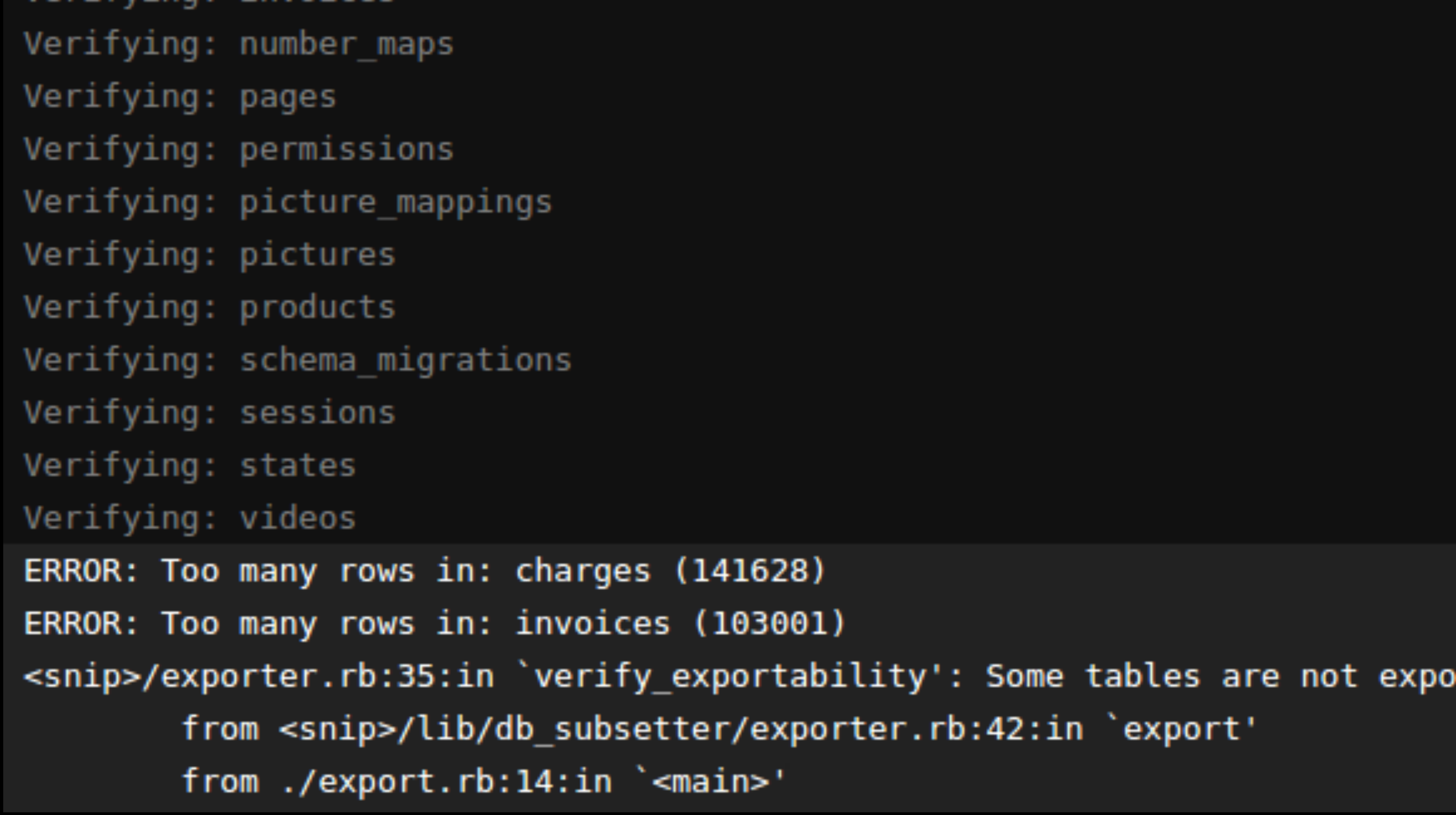
```ruby
# demo_subset_filter.rb
class DemoSubsetFilter < DbSubsetter::Filter
  def ignore_tables
    %w( campaign_tracker_page_views campaign_tracker_visitors )
  end
end


# export.rb
#!/usr/bin/env ruby
require 'db_subsetter'
require './demo_subset_filter'

ActiveRecord::Base.establish_connection(
        adapter:  "mysql2",
        host:      "127.0.0.1",
        username:  "joe",
        database:  "big_db"
  )

exporter = DbSubsetter::Exporter.new
exporter.filter = DemoSubsetFilter.new # <- Configure our new filter
exporter.export("demo-export.sqlite3")
```



```
Verifying: number_maps
Verifying: pages
Verifying: permissions
Verifying: picture_mappings
Verifying: pictures
Verifying: products
Verifying: schema_migrations
Verifying: sessions
Verifying: states
Verifying: videos
ERROR: Too many rows in: charges (141628)
ERROR: Too many rows in: customers (59351)
ERROR: Too many rows in: invoices (103001)
<snip>/exporter.rb:35:in `verify_exportability': Some tables are not exportable (
        from <snip>/lib/db_subsetter/exporter.rb:42:in `export'
        from ./export.rb:14:in `<main>'
```

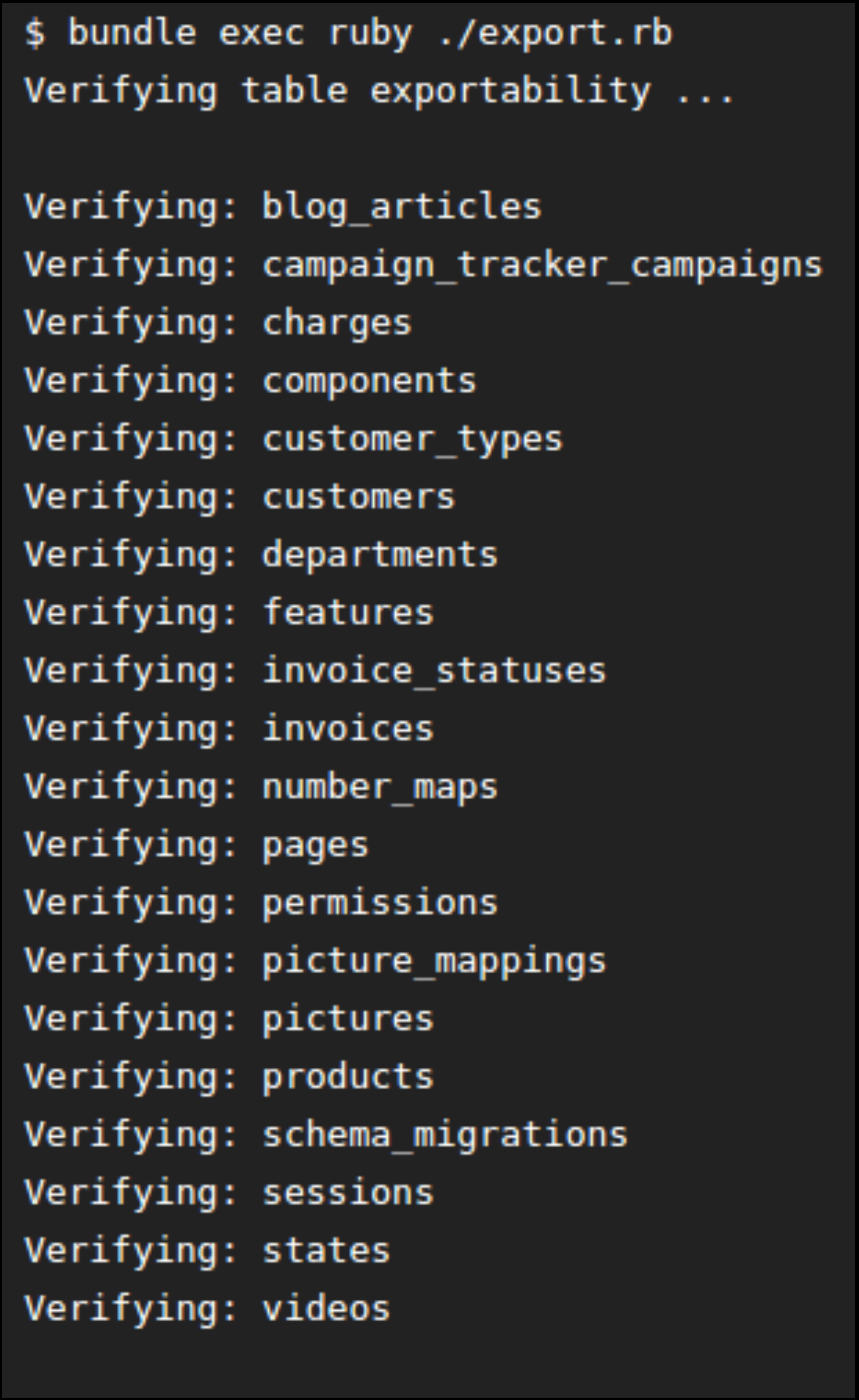


```ruby
# demo_subset_filter.rb
class Customer < ActiveRecord::Base; end

class DemoSubsetFilter < DbSubsetter::Filter
  def customer_ids
    Customer.order(:id => :desc).limit(1000).pluck(:id)
  end

  def filter_customers(query)
    query.where(query[:id].in(customer_ids))
  end
end
```

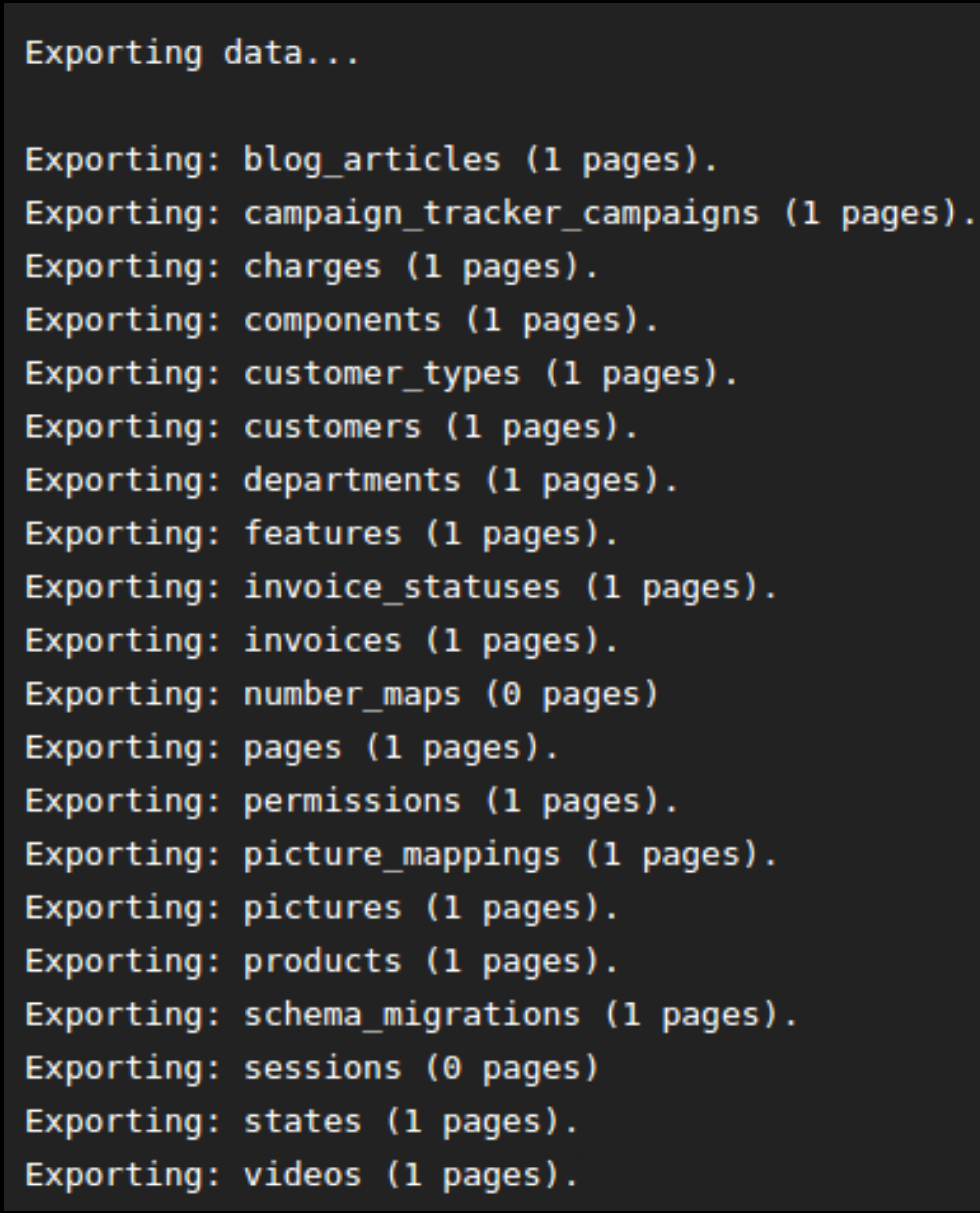


```
Verifying: number_maps
Verifying: pages
Verifying: permissions
Verifying: picture_mappings
Verifying: pictures
Verifying: products
Verifying: schema_migrations
Verifying: sessions
Verifying: states
Verifying: videos
ERROR: Too many rows in: charges (141628)
ERROR: Too many rows in: invoices (103001)
<snip>/exporter.rb:35:in `verify_exportability': Some tables are not expo
        from <snip>/lib/db_subsetter/exporter.rb:42:in `export'
        from ./export.rb:14:in `<main>'
```

```ruby
class Invoice < ActiveRecord::Bas; end
class Charge < ActiveRecord::Base; end

class DemoSubsetFilter < DbSubsetter::Filter
  def customer_ids
    Customer.order(:id => :desc).limit(1000).pluck(:id)
  end

  def invoice_ids
    Invoice.where(:customer_id => customer_ids).pluck(:id)
  end

  def charge_ids
    Charge.where(:invoice_id => invoice_ids).pluck(:id)
  end

  def filter_customers(query)
    query.where(query[:id].in(customer_ids))
  end

  def filter_invoices(query)
    query.where(query[:id].in(invoice_ids))
  end

  def filter_charges(query)
    query.where(query[:id].in(charge_ids))
  end
```



```
$ bundle exec ruby ./export.rb
Verifying table exportability ...

Verifying: blog_articles
Verifying: campaign_tracker_campaigns
Verifying: charges
Verifying: components
Verifying: customer_types
Verifying: customers
Verifying: departments
Verifying: features
Verifying: invoice_statuses
Verifying: invoices
Verifying: number_maps
Verifying: pages
Verifying: permissions
Verifying: picture_mappings
Verifying: pictures
Verifying: products
Verifying: schema_migrations
Verifying: sessions
Verifying: states
Verifying: videos
```



```
Exporting data...

Exporting: blog_articles (1 pages).
Exporting: campaign_tracker_campaigns (1 pages).
Exporting: charges (1 pages).
Exporting: components (1 pages).
Exporting: customer_types (1 pages).
Exporting: customers (1 pages).
Exporting: departments (1 pages).
Exporting: features (1 pages).
Exporting: invoice_statuses (1 pages).
Exporting: invoices (1 pages).
Exporting: number_maps (0 pages)
Exporting: pages (1 pages).
Exporting: permissions (1 pages).
Exporting: picture_mappings (1 pages).
Exporting: pictures (1 pages).
Exporting: products (1 pages).
Exporting: schema_migrations (1 pages).
Exporting: sessions (0 pages)
Exporting: states (1 pages).
Exporting: videos (1 pages).
```

```ruby
end

def invoice_ids
  Invoice.where(:customer_id => customer_ids).pluck(:id)
end

def charge_ids
  Charge.where(:invoice_id => invoice_ids).pluck(:id)
end

def product_ids
  Product.where(:discontinued => false).limit(500).pluck(:id)
end

def filter_customers(query)
  query.where(query[:id].in(customer_ids))
end

def filter_invoices(query)
  query.where(query[:id].in(invoice_ids))
end

def filter_charges(query)
```

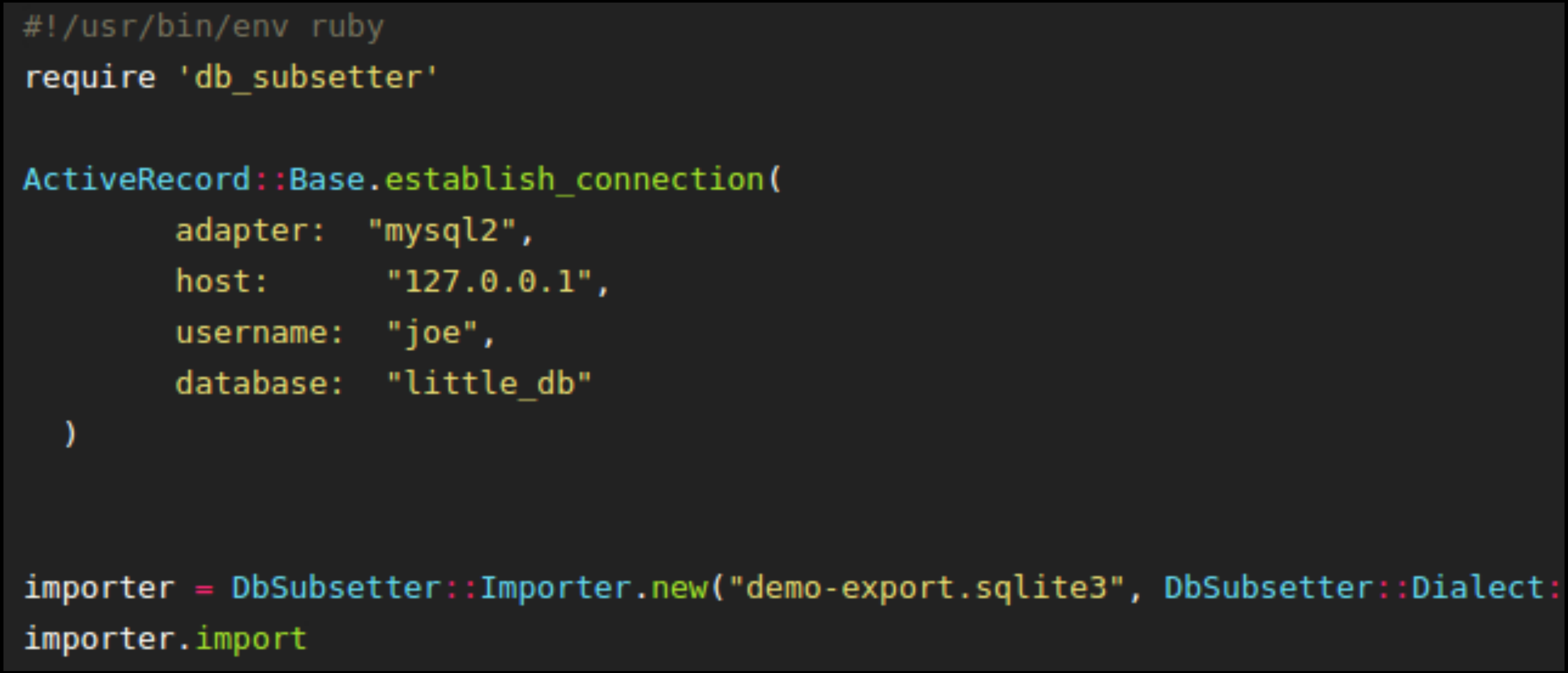
```
        adapter:  "mysql2",
        host:     "127.0.0.1",
        username: "joe",
        database: "big_db"
  )

exporter = DbSubsetter::Exporter.new
exporter.filter = DemoSubsetFilter.new
exporter.max_filtered_rows = 5000
exporter.export("demo-export.sqlite3")
```

```ruby
# demo_subset_scrambler.rb
class DemoSubsetScrambler < DbSubsetter::Scrambler
  def scramble_customers(row)
    scramble_column(:customers, :password, row, 'hashthisfirst')
    row
  end
end
```

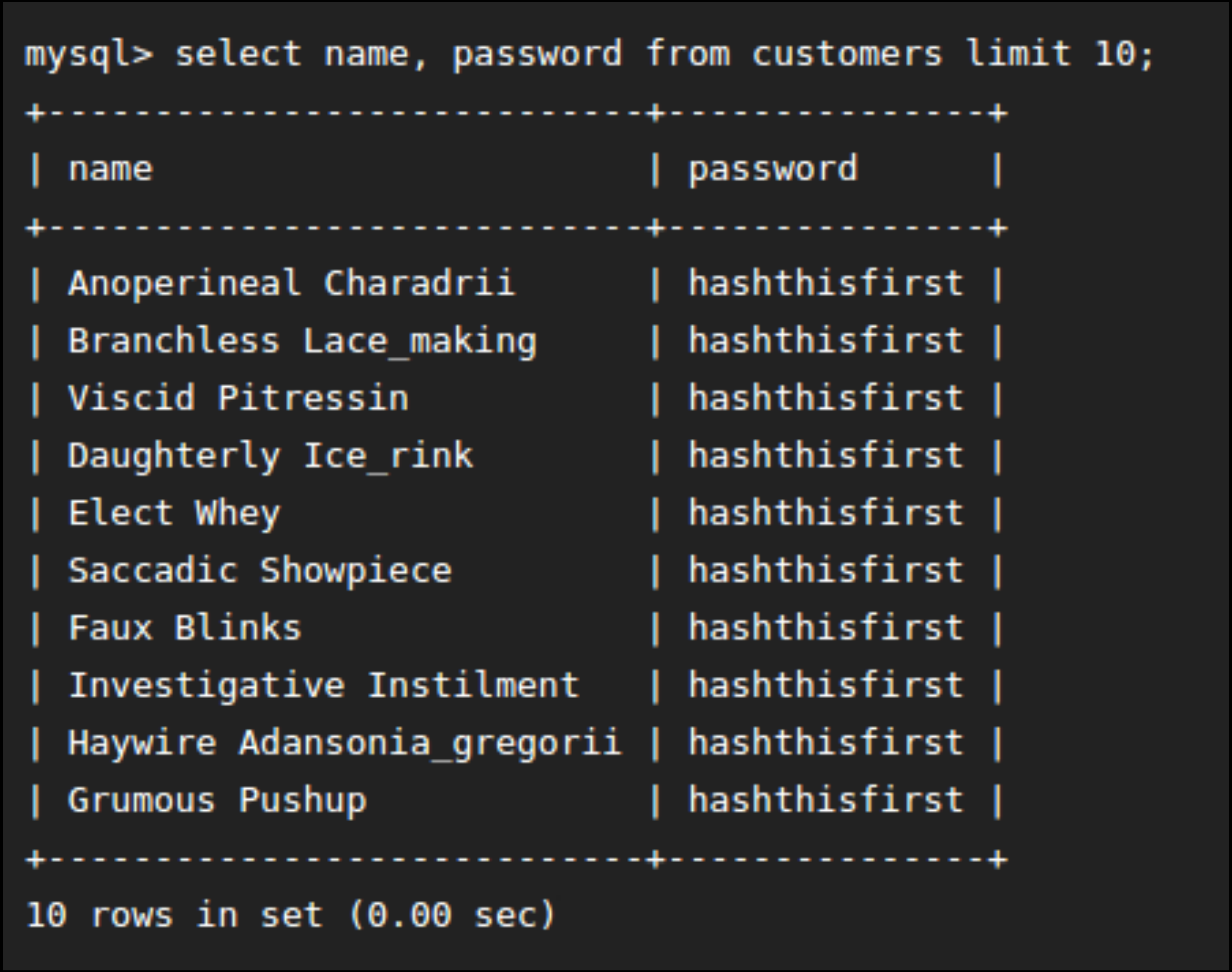
```ruby
# export.rb
#!/usr/bin/env ruby
require 'db_subsetter'
require './demo_subset_filter6'
require './demo_subset_scrambler6'


ActiveRecord::Base.establish_connection(
        adapter:  "mysql2",
        host:      "127.0.0.1",
        username:  "joe",
        database:  "big_db"
```

```
ActiveRecord::Base.establish_connection(
        adapter:  "mysql2",
        host:      "127.0.0.1",
        username:  "joe",
        database:  "big_db"
  )

exporter = DbSubsetter::Exporter.new
exporter.filter = DemoSubsetFilter.new
exporter.add_scrambler DemoSubsetScrambler.new # <- tell our exporter
exporter.max_filtered_rows = 5000
exporter.export("demo-export.sqlite3")
```

```ruby
# demo_subset_scrambler.rb
class DemoSubsetScrambler < DbSubsetter::Scrambler
  def scramble_customers(row)
    new_name = RandomWord.adjs.next.capitalize + " " + RandomWord.nouns.next.capi
    scramble_column(:customers, :name, row, new_name )

    scramble_column(:customers, :password, row, 'hashthisfirst')
    row
  end
end
```



```ruby
#!/usr/bin/env ruby
require 'db_subsetter'

ActiveRecord::Base.establish_connection(
        adapter:  "mysql2",
        host:      "127.0.0.1",
        username:  "joe",
        database:  "little_db"
  )


importer = DbSubsetter::Importer.new("demo-export.sqlite3", DbSubsetter::Dialect:
importer.import
```



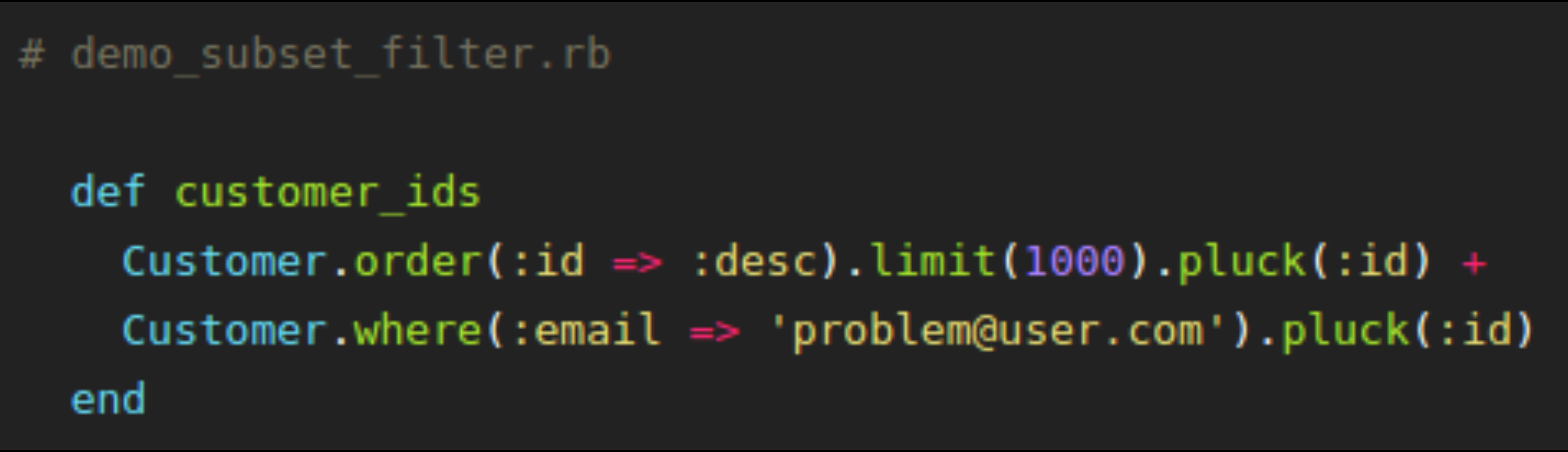
```
mysql> select name, password from customers limit 10;
+-----------------------------+---------------+
| name                        | password      |
+-----------------------------+---------------+
| Anoperineal Charadrii       | hashthisfirst |
| Branchless Lace_making      | hashthisfirst |
| Viscid Pitressin            | hashthisfirst |
| Daughterly Ice_rink         | hashthisfirst |
| Elect Whey                  | hashthisfirst |
| Saccadic Showpiece          | hashthisfirst |
| Faux Blinks                 | hashthisfirst |
| Investigative Instilment    | hashthisfirst |
| Haywire Adansonia_gregorii | hashthisfirst |
| Grumous Pushup              | hashthisfirst |
+-----------------------------+---------------+
10 rows in set (0.00 sec)
```

Performance?

- gzip the sqlite files
- Subsets 120GB -> 7MB in 55s (MS SQL)
- Imports inside VirtualBox in 150s
- MySQL imports much faster

Bonus Feature!

Our Killer App???



```
# demo_subset_filter.rb

  def customer_ids
    Customer.order(:id => :desc).limit(1000).pluck(:id) +
    Customer.where(:email => 'problem@user.com').pluck(:id)
  end
```

Bug-focused subsets

Future Directions

- Foreign key discovery
- Improve scrambler API
- Library of standards scramblers
- More supported DBs
- Test Suite
- More granular limits
- **Ideas please!**

Wrapping Up

- Version control is key
- If you can't be reckless with your dev DB, you're missing out
- Check out Jailer, rdbms-subsetter, or db_subsetter

Questions?

Joe Francis
@lostapathy
joe@lostapathy.com