PORTLT

```
rails new portlt
cd portlt
```

Tickers table

Run ruby program to create tickers from name-ttl & output tickers to **BBB\data** and **portlt\db**

```
CD C:\BBB\ruby
ruby crt-tickers-fm-fs.rb

rails g model Ticker name full_name sector subsector market website
```

Edit create_tickers.rb

```
add_index :tickers, [:name], unique: true

rails db:migrate
```

Create crt_tickers.rb

```
require 'csv'

class CrtTickers < ActiveRecord::Base
    records_ins = 0
    records_upd = 0
    CSV.foreach(Rails.root.join("db/tickers.csv"), { col_sep: '|', headers: false}) do |row|
        tickers = Ticker.where(name: row[0])
        ticker = tickers.first
        if ticker
            ticker.full_name = row[1]
            ticker.sector = row[2]
            ticker.subsector = row[3]
            ticker.market = row[4]
            ticker.website = row[5]
            ticker.save
            records_upd += 1
        else
            Ticker.find_or_create_by(
            name: row[0],
            full_name: row[1],
            sector: row[2],
            subsector: row[3],
            market: row[4],
            website: row[5])
            records_ins += 1
        end
    end

    printf "%2d records added, %2d records updated.", records_ins, records_upd
end

rails runner db\crt_tickers.rb
```

Stocks table

Run ruby program to create stocks from name-ttl & output stocks to **BBB\data** and **portlt\db**

```
CD C:\BBB\ruby
ruby crt-stocks-fm-fs.rb

rails g model stock name market price:decimal{6,2} max_price:decimal{6,2} min_price:decimal{6,2} pe:decimal pbv:decimal paid_up:decimal
```

Edit create_stocks.rb

```
add_index :stocks, [:name], unique: true

rails db:migrate
```

Create crt_stocks.rb

```
require 'csv'

class CrtStocks < ActiveRecord::Base
    records_ins = 0
    records_upd = 0
    CSV.foreach(Rails.root.join("db/stocks.csv"), col_sep: '|', headers: false) do |row|
        tickers = Ticker.where(name: row[0])
        ticker = tickers.first
        if ticker
            ticker_id = ticker.id
            stocks = Stock.where(name: row[0])
            stock = stocks.first
            if stock
                stock.market = row[1]
                stock.price = row[2]
                stock.max_price = row[3]
                stock.min_price = row[4]
                stock.pe = row[5]
                stock.pbv = row[6]
                stock.paid_up = row[7]
                stock.market_cap = row[8]
                stock.daily_volume = row[9]
                stock.beta = row[10]
                stock.save
                records_upd += 1
            else
                Stock.find_or_create_by(
                  name: row[0],
                  market: row[1],
                  price: row[2],
                  max_price: row[3],
                  min_price: row[4],
                  pe: row[5],
                  pbv: row[6],
                  paid_up: row[7],
                  market_cap: row[8],
                  daily_volume: row[9],
                  beta: row[10],
                  ticker_id: ticker_id)
                  records_ins += 1
            end
        end
    end
    printf "%3d records added, %3d records updated.", records_ins, records_upd
end

rails runner db\crt_stocks.rb
```

Consensus table

Run ruby program to create consensus from name-ttl & output consensus to **BBB\data** and **portlt\db**

```
CD C:\BBB\ruby
ruby crt-consensus-fm-iaa.rb

rails g model consensu name price:decimal{6,2} buy:integer hold:integer sell:integer eps_a:decimal eps_b:decimal pe:decimal pbv:decimal
```

Edit create_consensus.rb

```
add_index :consensus, [:name], unique: true

rails db:migrate
```

Create crt_consensus.rb

```
require 'csv'

class CrtStocks < ActiveRecord::Base
    records_ins = 0
    records_upd = 0
    CSV.foreach(Rails.root.join("db/stocks.csv"), col_sep: '|', headers: false) do |row|
        tickers = Ticker.where(name: row[0])
        ticker = tickers.first
        if ticker
            ticker_id = ticker.id
```

```
            stocks = Stock.where(name: row[0])
            stock = stocks.first
            if stock
                stock.market = row[1]
                stock.price = row[2]
                stock.max_price = row[3]
                stock.min_price = row[4]
                stock.pe = row[5]
                stock.pbv = row[6]
                stock.paid_up = row[7]
                stock.market_cap = row[8]
                stock.daily_volume = row[9]
                stock.beta = row[10]
                stock.save
                records_upd += 1
            else
                Stock.find_or_create_by(
                  name: row[0],
                  market: row[1],
                  price: row[2],
                  max_price: row[3],
                  min_price: row[4],
                  pe: row[5],
                  pbv: row[6],
                  paid_up: row[7],
                  market_cap: row[8],
                  daily_volume: row[9],
                  beta: row[10],
                  ticker_id: ticker_id)
                  records_ins += 1
            end
        end
    end
    printf "%3d records added, %3d records updated.", records_ins, records_upd
end

rails runner db\crt_consensus.rb
```

EPS Table

Edit config/initializers/inflections.rb

```
ActiveSupport::Inflector.inflections(:en) do |inflect|
 inflect.irregular 'eps', 'epss'
 inflect.irregular 'tmp_eps', 'tmp_epss'
end

rails g model eps name year:integer quarter:integer q_amt:integer y_amt:integer aq_amt:integer ay_amt:integer q_eps:decimal{8,6} y_eps:d
```

Edit create_eps.rb

```
add_index :epss, [:name, :year, :quarter], unique: true

rails db:migrate
```

Run python program **Export-EPS-fm-PG-to-LT**

```
import pandas as pd
from sqlalchemy import create_engine

engine = create_engine('sqlite:///c:\\ruby\\portlt\\db\\development.sqlite3')
conlt = engine.connect()

engine = create_engine('postgresql+psycopg2://postgres:admin@localhost:5432/port_development')
conpg = engine.connect()

sql = 'SELECT name, id AS ticker_id FROM tickers'
sql

tickers = pd.read_sql(sql, conlt)
tickers.shape

sql = 'SELECT name,year,quarter,q_amt,y_amt,aq_amt,ay_amt,q_eps,y_eps,aq_eps,ay_eps \
FROM epss \
ORDER BY year, quarter, name'
sql
```

```
df_inp = pd.read_sql(sql, conpg)
df_inp.shape

df_merge = pd.merge(df_inp, tickers, on='name', how='outer', indicator = True)
df_merge.head()

df_ins = df_merge[df_merge['_merge'] == 'both']
df_ins.shape

df_eps = df_ins[['name','year','quarter',\
 'q_amt','y_amt',\
 'aq_amt','ay_amt',\
 'q_eps','y_eps',\
 'aq_eps','ay_eps','ticker_id']]
df_eps.shape

df_out = df_eps.set_index(['name','year','quarter'])

df_out.to_csv('../data/epss-new.csv')

rcds = df_eps.values.tolist()
len(rcds)

for rcd in rcds:
    print(rcd)

for rcd in rcds:
    conlt.execute("""INSERT INTO epss (name, year, quarter, \
q_amt, y_amt, aq_amt, ay_amt, \
q_eps, y_eps, aq_eps, ay_eps, ticker_id) \
VALUES( ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)""", rcd)

rails g scaffold eps name year:integer quarter:integer q_amt y_amt aq_amt ay_amt q_eps y_eps aq_eps ay_eps --skip-migration
```

Edit views\epss\index.html.erb

```
<table id="epss" class="table table-striped table-hover">

    <td><%= 'Q' + eps.quarter.to_s %></td>
    <td><%= number_with_precision(eps.q_amt, precision: 0, delimiter: ',') %></td>
    <td><%= number_with_precision(eps.y_amt, precision: 0, delimiter: ',') %></td>

    <td><%= number_with_precision(eps.aq_amt, precision: 0, delimiter: ',') %></td>
    <td><%= number_with_precision(eps.ay_amt, precision: 0, delimiter: ',') %></td>

    <td><%= number_with_precision(eps.q_eps, precision: 4) %></td>
    <td><%= number_with_precision(eps.y_eps, precision: 4) %></td>
    <td><%= number_with_precision(eps.aq_eps, precision: 4) %></td>
    <td><%= number_with_precision(eps.ay_eps, precision: 4) %></td>
    <% if (eps.y_amt != 0) %>
      <td><%= number_with_precision((eps.q_amt-eps.y_amt)/eps.y_amt.abs.to_f*100, precision: 2, delimiter: ',') %></td>
      <td><%= number_with_precision((eps.aq_amt-eps.ay_amt)/eps.ay_amt.abs.to_f*100, precision: 2, delimiter: ',') %></td>
    <% else %>
      <td><%= 0.00 %></td>
      <td><%= 0.00 %></td>
    <% end %>
```

Edit stylesheets\epss.coffee

```
jQuery ->
        $('#epss').dataTable({
        pagingType: 'full_numbers',
        order: [[ 0, "asc" ],[ 1, "desc"],[ 2, "desc"]]
        })
```

Edit models/epss.rb

```
belongs_to :ticker
before_save :assign_names
default_scope { order(name: 'asc', year: 'desc', quarter: 'desc') }

YEAR = [
        "2017",
        "2016",
        "2015"
        ]

QUARTER = [
```

```
        "1",
        "4",
        "3",
        "2"
        ]

def q_amt_in_million
    q_amt.div(1000)
end

def y_amt_in_million
    y_amt.div(1000)
end

def aq_amt_in_million
    aq_amt.div(1000)
end

def ay_amt_in_million
    ay_amt.div(1000)
end

private

def assign_names
    ticker = Ticker.find(self.ticker_id)
    self.name = ticker.name

    if (self.quarter == 4)
        eps = Eps.where(name: self.name, year: self.year, quarter: '3')
        if eps
            self.q_amt = self.aq_amt - eps.first.aq_amt
            self.y_amt = self.ay_amt - eps.first.ay_amt
            self.q_eps = self.aq_eps - eps.first.aq_eps
            self.y_eps = self.ay_eps - eps.first.ay_eps
        end
    end

    if (self.quarter == 1)
        self.aq_amt  = self.q_amt
        self.ay_amt  = self.y_amt
        self.aq_eps  = self.q_eps
        self.ay_eps  = self.y_eps
    end
end
```

Edit views/epss/_form.html.erb

```
<%= form_for(@eps) do |f| %>
    <% if @eps.errors.any? %>
        <div id="error_explanation">
            <h2><%= pluralize(@eps.errors.count, "error") %> prohibited this eps from being saved:</h2>
            <ul>
                <% @eps.errors.full_messages.each do |message| %>
                    <li><%= message %></li>
                <% end %>
            </ul>
        </div>
    <% end %>
    <!-- row 1 -->
    <div class="row">
        <div class="col-xs-12 col-sm-4">
            <label class="control-label col-sm-4"><%= f.label :ticker_id %></label>
            <%= f.collection_select :ticker_id, Ticker.all, :id, :name %>
        </div>
        <div class="col-xs-12 col-sm-4">
            <label class="control-label col-sm-4"><%= f.label :year %></label>
            <%=f.select :year, Eps::YEAR, {} %>
        </div>
        <div class="col-xs-12 col-sm-4">
            <label class="control-label col-sm-4"><%= f.label :quarter %></label>
            <%=f.select :quarter, Eps::QUARTER, {} %>
        </div>
    </div>
    <!-- row 2 -->
    <div class="row">
        <div class="col-xs-6">
            <label class="control-label col-xs-4"><%= f.label :q_amt %></label>
            <%= f.text_field :q_amt %>
        </div>
```

```
        <div class="col-xs-6">
            <label class="control-label col-xs-5"><%= f.label :y_amt %></label>
            <%= f.text_field :y_amt %>
        </div>
    </div>
    <!-- row 3 -->
    <div class="row">
        <div class="col-xs-6">
            <label class="control-label col-xs-4"><%= f.label :aq_amt %></label>
            <%= f.text_field :aq_amt %>
        </div>
        <div class="col-xs-6">
            <label class="control-label col-xs-5"><%= f.label :ay_amt %></label>
            <%= f.text_field :ay_amt %>
        </div>
    </div>
    <!-- row 4 -->
    <div class="row">
        <div class="col-xs-6">
            <label class="control-label col-xs-4"><%= f.label :q_eps %></label>
            <%= f.text_field :q_eps %>
        </div>
        <div class="col-xs-6">
            <label class="control-label col-xs-5"><%= f.label :y_eps %></label>
            <%= f.text_field :y_eps %>
        </div>
    </div>
    <!-- row 5 -->
    <div class="row">
        <div class="col-xs-6">
            <label class="control-label col-xs-4"><%= f.label :aq_eps %></label>
            <%= f.text_field :aq_eps %>
        </div>
        <div class="col-xs-6">
            <label class="control-label col-xs-5"><%= f.label :ay_eps %></label>
            <%= f.text_field :ay_eps %>
        </div>
    </div>
    <!-- row 6 -->
    <div class="row">
        <div class="col-xs-12">
            <label class="control-label col-xs-2"></label>
            <%= f.submit %>
        </div>
    </div>

<% end %>
```

Qt_Profits Table

```
rails g model qt_profit name year:integer quarter latest_amt:integer previous_amt:integer inc_amt:integer inc_pct:decimal ticker:belongs
```

Edit db\migrate\create*qt*profits.rb

```
add_index :qt_profits, [:name, :year, :quarter], unique: true

rails db:migrate
```

Run python program **Export-Qt_Profits-fm-PG-to-LT**

```
import pandas as pd
from sqlalchemy import create_engine

engine = create_engine('sqlite:///c:\\ruby\\portlt\\db\\development.sqlite3')
conlt = engine.connect()

engine = create_engine('postgresql+psycopg2://postgres:admin@localhost:5432/port_development')
conpg = engine.connect()

sql = 'SELECT name, id AS ticker_id FROM tickers'
sql

tickers = pd.read_sql(sql, conlt)
tickers.shape

sql = 'SELECT name,year,quarter,latest_profit,previous_profit, \
```

```
inc_profit,inc_percent \
FROM profits \
ORDER BY year desc, quarter desc, name'
sql

df_inp = pd.read_sql(sql, conpg)
df_inp.shape

df_merge = pd.merge(df_inp, tickers, on='name', how='outer', indicator = True)
df_merge.head()

df_ins = df_merge[df_merge['_merge'] == 'both']
df_ins.shape

df_prf = df_ins[['name','year','quarter',\
 'latest_profit','previous_profit',\
 'inc_profit','inc_percent','ticker_id']]
df_prf.shape

rcds = df_prf.values.tolist()
len(rcds)

for rcd in rcds:
    print(rcd)

for rcd in rcds:
    conlt.execute("""INSERT INTO qt_profits (name, year, quarter, \
latest_amt, previous_amt, inc_amt, inc_pct, ticker_id) \
VALUES( ?, ?, ?, ?, ?, ?, ?, ?)""", rcd)

rails g scaffold qt_profit name year:integer quarter latest_amt previous_amt inc_amt inc_pct --skip-migration
```

Edit views\qt_profits\index.html.erb

```
<h2>Quarterly Profits</h2>

<table id="qt_profits" class="table table-striped table-hover">
    <tbody>
        <% @qt_profits.each do |qt_profit| %>
            <tr>
            <td><%= qt_profit.name %></td>
            <td><%= qt_profit.year %></td>
            <td><%= qt_profit.quarter %></td>
            <td><%= number_with_precision(qt_profit.inc_amt, precision: 0, delimiter: ',') %></td>
            <td><%= number_with_precision(qt_profit.inc_pct, precision: 1, delimiter: ',') %></td>
            <td><%= number_with_precision(qt_profit.ticker.stock.price, precision: 2) %></td>
            <td><%= number_with_precision(qt_profit.ticker.stock.max_price, precision: 2) %></td>
            <td><%= number_with_precision((qt_profit.ticker.stock.max_price-qt_profit.ticker.stock.price)/
            qt_profit.ticker.stock.price*100, precision: 2) %></td>
            <% consensu = Consensu.find_by_name(qt_profit.name) %>

            <% if consensu %>
              <td><%= number_with_precision(consensu.target_price, precision: 2) %></td>
              <td><%= consensu.buy %></td>
              <td><%= consensu.hold %></td>
              <td><%= consensu.sell %></td>
            <% else %>
              <td><%= 0.00 %></td>
              <td><%= 0 %></td>
              <td><%= 0 %></td>
              <td><%= 0 %></td>
            <% end %>

            <td><%= number_with_precision(qt_profit.ticker.stock.pe, precision: 2) %></td>
            <td><%= number_with_precision(qt_profit.ticker.stock.pbv, precision: 2) %></td>
            <td><%= number_with_precision(qt_profit.ticker.stock.daily_volume, precision: 2, delimiter: ',') %></td>
            <td><%= number_with_precision(qt_profit.ticker.stock.beta, precision: 2) %></td>
            <td><%= link_to 'S', qt_profit %></td>
            <td><%= link_to 'E', edit_qt_profit_path(qt_profit) %></td>
        </tr>
        <% end %>
    </tbody>
</table>
```

Edit stylesheets\epss.coffee

```
jQuery ->
        $('#qt_profits').dataTable({
        pagingType: 'full_numbers',
```

```
        order: [[ 0, "asc" ],[ 1, "desc"],[ 2, "desc"]]
        })
```

Edit models\ticker.rb

```
has_one :stock
```

Yr_Profits Table

```
rails g model yr_profit name year:integer quarter latest_amt:integer previous_amt:integer inc_amt:integer inc_pct:decimal ticker:belongs
```

Edit db\migrate\create*yr*profits.rb

```
add_index :yr_profits, [:name, :year, :quarter], unique: true

rails db:migrate
```

Run python program **Export-Yr_Profits-fm-PG-to-LT**

```
import pandas as pd
from sqlalchemy import create_engine

engine = create_engine('sqlite:///c:\\ruby\\portlt\\db\\development.sqlite3')
conlt = engine.connect()

engine = create_engine('postgresql+psycopg2://postgres:admin@localhost:5432/port_development')
conpg = engine.connect()

sql = 'SELECT name, id AS ticker_id FROM tickers'
sql

tickers = pd.read_sql(sql, conlt)
tickers.shape

sql = 'SELECT name,year,quarter,latest_profit,previous_profit, \
inc_profit,inc_percent \
FROM yr_profits \
ORDER BY year desc, quarter desc, name'
sql

df_inp = pd.read_sql(sql, conpg)
df_inp.shape

df_merge = pd.merge(df_inp, tickers, on='name', how='outer', indicator = True)
df_merge.head()

df_ins = df_merge[df_merge['_merge'] == 'both']
df_ins.shape

df_prf = df_ins[['name','year','quarter',\
 'latest_profit','previous_profit',\
 'inc_profit','inc_percent','ticker_id']]
df_prf.shape

rcds = df_prf.values.tolist()
len(rcds)

for rcd in rcds:
    print(rcd)

for rcd in rcds:
    conlt.execute("""INSERT INTO yr_profits (name, year, quarter, \
latest_amt, previous_amt, inc_amt, inc_pct, ticker_id) \
VALUES( ?, ?, ?, ?, ?, ?, ?, ?)""", rcd)

rails g scaffold yr_profit name year:integer quarter latest_amt previous_amt inc_amt inc_pct --skip-migration
```

Bootstrap

Edit Gemfile

```
gem 'bootstrap-sass', '~> 3.3.5'
bundle install
```

```
rails g controller pages home
```

Edit config/routes.rb

```
root to:  'pages#home'
```

Edit app/views/pages/home.html.erb

```
<div class ="page-header text-center" >
  <h1>Home</h1>
</div>
```

Edit app/stylesheets/pages.scss

```
@import "bootstrap";
```

DataTables

Edit gem file

```
gem 'jquery-rails'
gem 'jquery-datatables-rails', github: 'rweng/jquery-datatables-rails'

bundle install
```

Edit app/assets/javascripts/application.js

```
//= require jquery
//= require jquery_ujs
```

Run the install generator:

```
rails g jquery:datatables:install bootstrap3
```

This will add to the corresponding asset files

app/assets/javascripts/application.js
//= require dataTables/jquery.dataTables
//= require dataTables/bootstrap/3/jquery.dataTables.bootstrap

app/assets/stylesheets/application.css
*= require dataTables/bootstrap/3/jquery.dataTables.bootstrap

Initialize your datatables using these option:

```
$('#datatable').dataTable({
  // ajax: ...,
  // autoWidth: false,
  // pagingType: 'full_numbers',
  // processing: true,
  // serverSide: true,

  // Optional, if you want full pagination controls.
  // Check dataTables documentation to learn more about available options.
  // http://datatables.net/reference/option/pagingType
});
```