Databases Final Project

Samantha Siow (415)
Daniel Jalova (315)

We created a database of startups, founders and investors.
This database allows users to query for startups or people within a given market type or location, and to find their success rates.

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.

Changes to our initial vision:

We had to do away including the amount and type of funding that startups have received because that information was not readily available on the API we were using.

We changed the format of our database such that investors are grouped together with startup founders and employees, and investing companies are group together with startups. This removed the need for an investorID and simplified our database.

Specialized Data Extraction:

Because we used the AngelList API[1], we found that a lot of the data we needed was in JSON format. There were several stages to extracting the data that we need. Using AngelList's RESTful interface, we created a Python script[2] to send http queries for startups and people, and then saved these as lists of JSON Objects. We then created a Java program that parsed the JSON into Startup and Person classes, and converted each object into a .csv file. Each .csv file represented a table in our database, resulting in tables for StartupMarkets, StartupListings, UserRoles, StartupLocations and StartupTypes. This made it much easier to load onto the MySQL database using the inbuilt LOAD LOCAL DATA INFILE method.

The Python script and Java program can be found in the PythonScripts folder and the AngelListJSONtoCSVParser folder respectively.

User Guide:

The user can run the code through their browser.

Major areas of Specialization:

- Complex extraction of real data from online sources: This was achieved through the complex parsing of data in JSON format from the AngelList API into MySQL.
- Specialized view or forms-based interface with sophisticated report generation: This was achieved through a CSS/HTML browser interface that would allow the user to view all information on a startup, the related people, and what their roles are within the startup. The user can also filter all the startups by location or market type, and a generated reported on any startup in the database.

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.

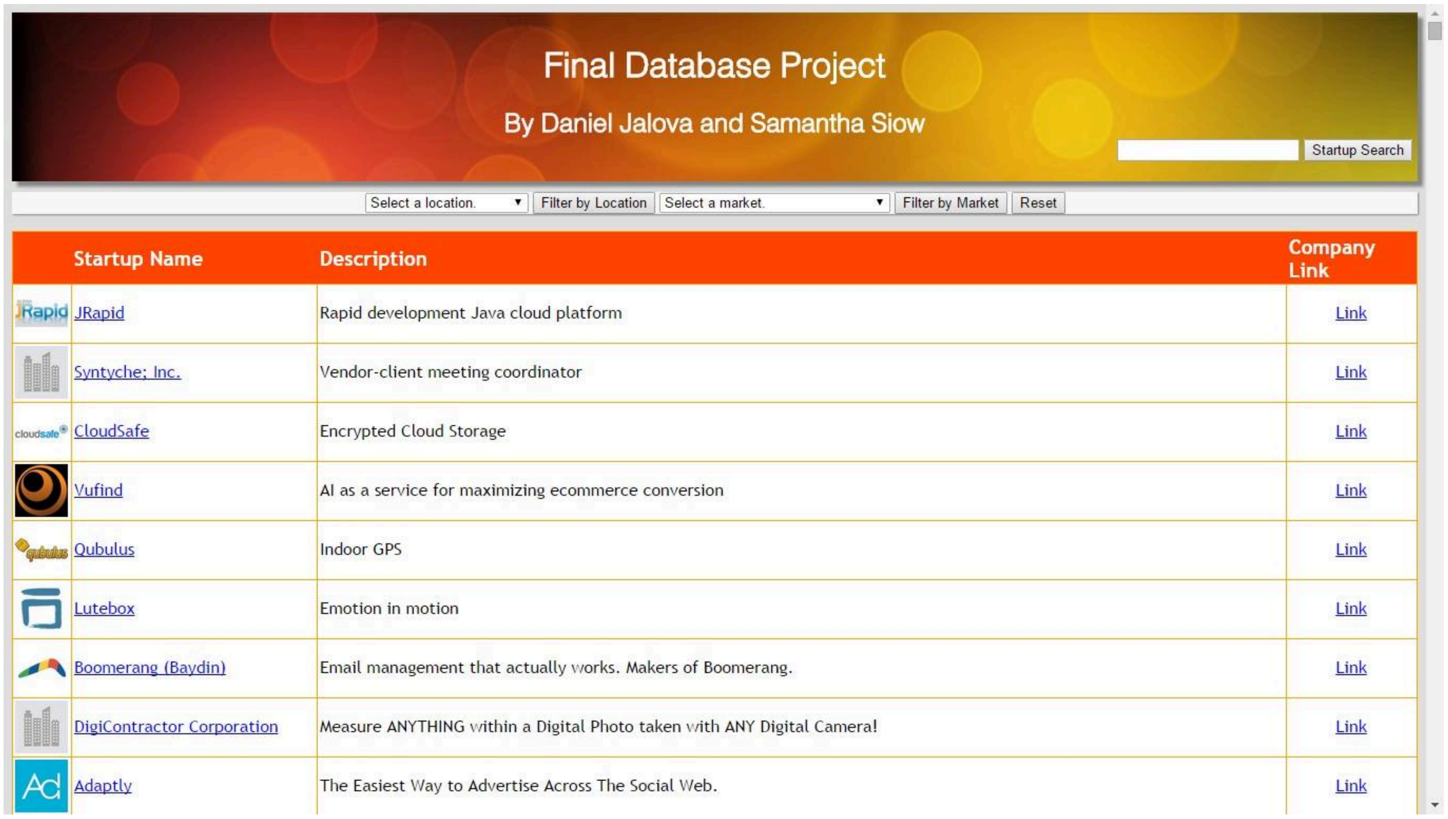
System Limitations and Suggested Possibilities for Improvement

- We are limited by the size of memory of our ugrad database. One way we could improve this was to move the database onto a different server, should we have needed more memory.
- We are also limited by the AngelList API, which only allowed us to pull 1000 queries per hour. Because of limited time and the large volume of data, we were only able to pull approximately 8,000 queries which might not be a good approximation of the full extent of all the Startups and related people in the world. One way we could have improved this was by utilizing different APIs, such as Crunchbase, in order to pull more information, but that would have involved more complex parsing methods because the format would be very different.
- Additionally, the AngelList API did not have some of the data we needed in order to make more complex queries and relations, such as the type and amount of funding startups had. In order to improve this, we could, as said above, have utilized a different API that would have provided this information.

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.

Sample Output

Home screen:



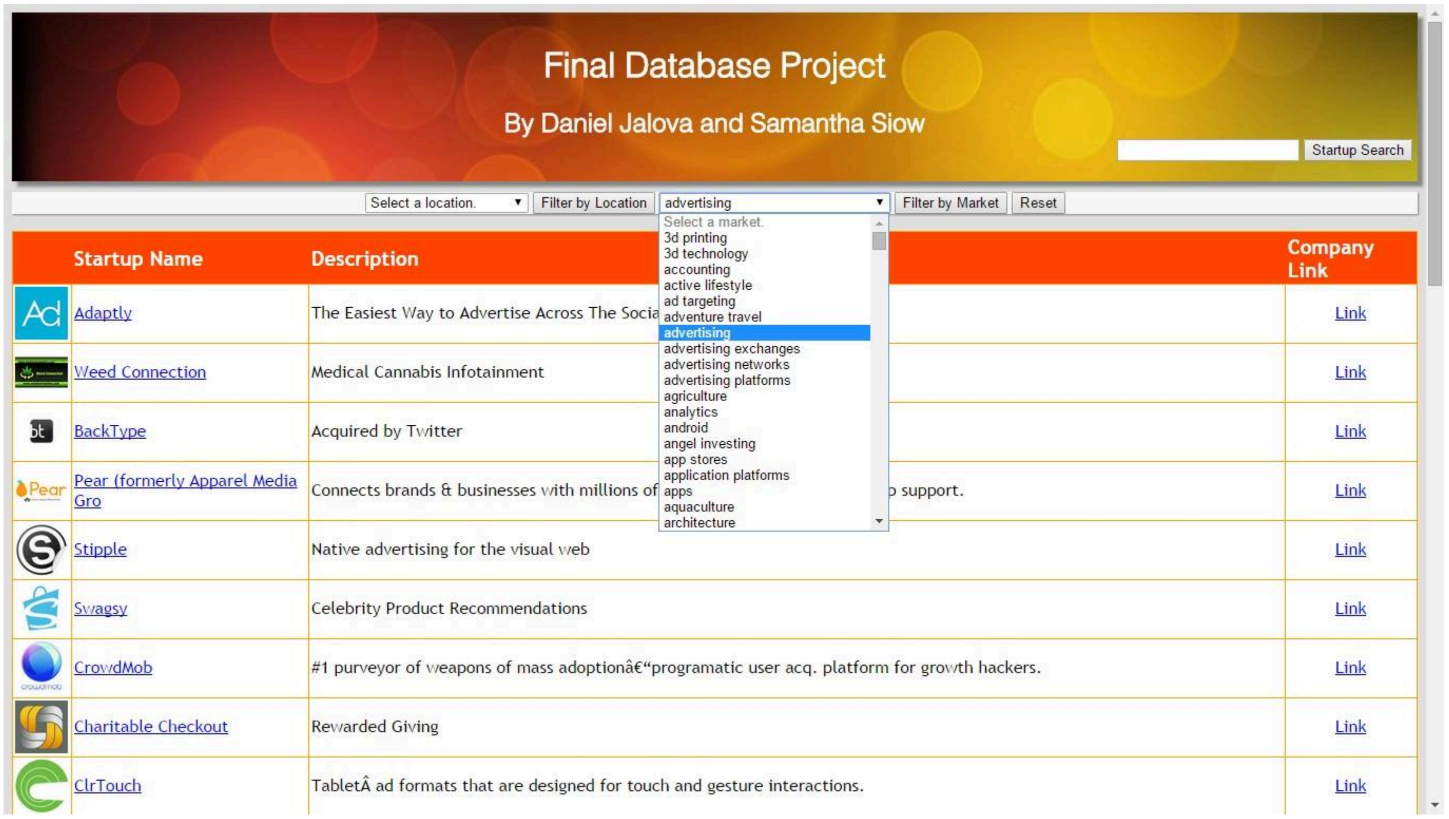
Final Database Project

By Daniel Jalova and Samantha Siow

Startup Name	Description	Company Link
JRapid	Rapid development Java cloud platform	Link
Syntyche; Inc.	Vendor-client meeting coordinator	Link
CloudSafe	Encrypted Cloud Storage	Link
Vufind	AI as a service for maximizing ecommerce conversion	Link
Qubulus	Indoor GPS	Link
Lutebox	Emotion in motion	Link
Boomerang (Baydin)	Email management that actually works. Makers of Boomerang.	Link
DigiContractor Corporation	Measure ANYTHING within a Digital Photo taken with ANY Digital Camera!	Link
Adaptly	The Easiest Way to Advertise Across The Social Web.	Link

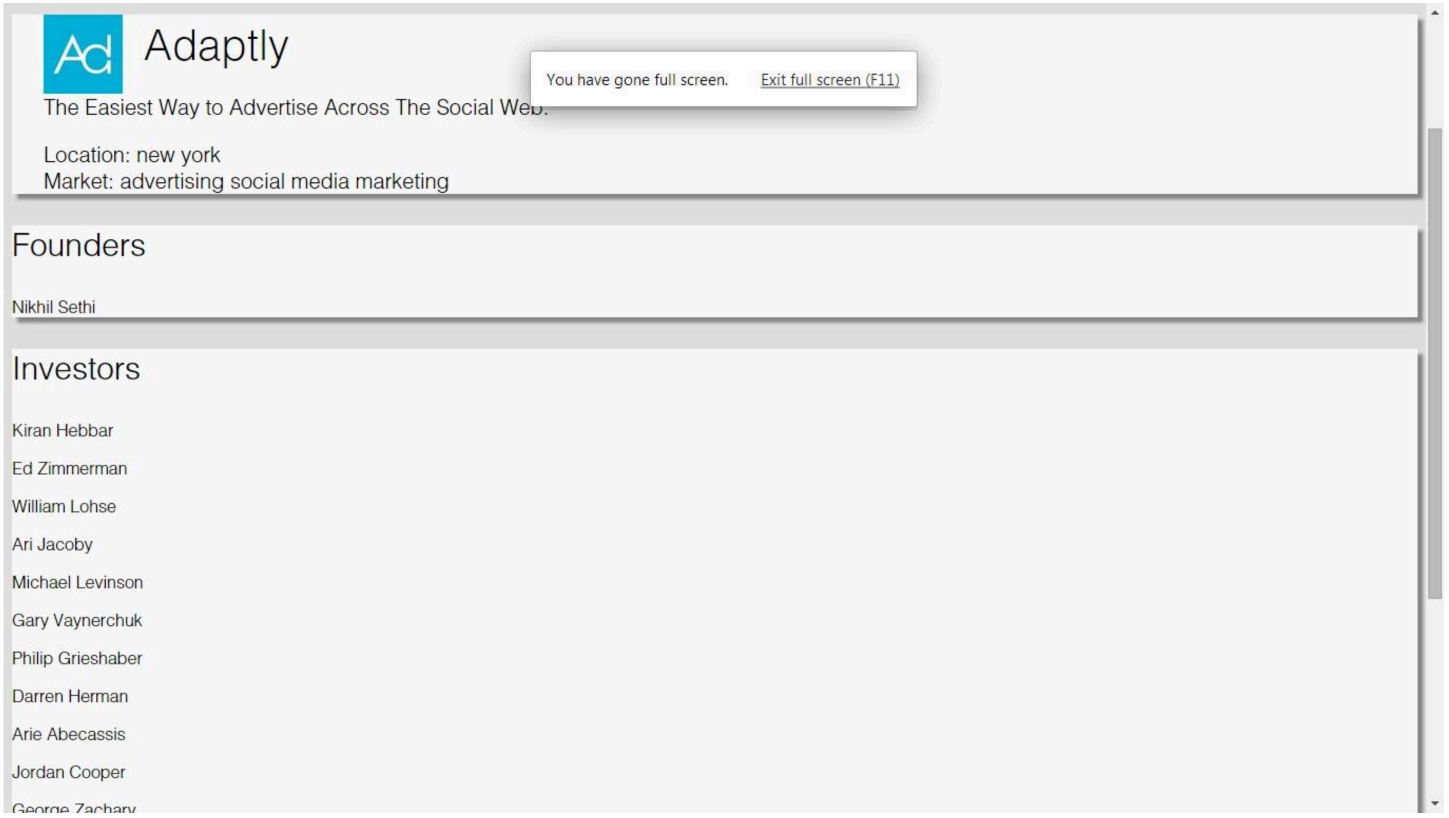
Filter by drop down boxes:

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.



Startup Name	Description	Company Link
Adaptly	The Easiest Way to Advertise Across The Socia	Link
Weed Connection	Medical Cannabis Infotainment	Link
BackType	Acquired by Twitter	Link
Pear (formerly Apparel Media Gro	Connects brands & businesses with millions of p support.	Link
Stipple	Native advertising for the visual web	Link
Swagsy	Celebrity Product Recommendations	Link
CrowdMob	#1 purveyor of weapons of mass adoptionâ€“programatic user acq. platform for growth hackers.	Link
Charitable Checkout	Rewarded Giving	Link
ClrTouch	TabletÂ ad formats that are designed for touch and gesture interactions.	Link

Resulting report generated for a startup:



Adaptly

The Easiest Way to Advertise Across The Social Web.

Location: new york
Market: advertising social media marketing

Founders

Nikhil Sethi

Investors

Kiran Hebbar
Ed Zimmerman
William Lohse
Ari Jacoby
Michael Levinson
Gary Vaynerchuk
Philip Grieshaber
Darren Herman
Arie Abecassis
Jordan Cooper
George Zachary

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.

Relational Table Specification [4]

```
DROP TABLE IF EXISTS StartupListing;                          # Startup data
        CREATE TABLE StartupListing
                (id INT NOT NULL PRIMARY KEY,                 # id = 123
                startupName VARCHAR(32) NOT NULL,             # startupName = 'AngelList'
                quality int NOT NULL,                         # quality = 10, based on AngelList rankings
                highConcept VARCHAR(140),                     # highConcept = 'A platform for startups'
                companyUrl VARCHAR(140),                      # companyUrl = 'http://angel.co'
                companySize VARCHAR(12),                      # companySize = '1-10'
                thumbUrl VARCHAR(200));                       # thumbUrl = 'http://....jpg' thumbnail image

DROP TABLE IF EXISTS StartupLocation;                         # Location data for each startup
        CREATE TABLE StartupLocation
                (id INT NOT NULL,                             # id for the startup, e.g. 123
                location VARCHAR(32) NOT NULL,                # location of a startup’s office, e.g. San Francisco
    FOREIGN KEY (id) REFERENCES StartupListing(id));

DROP TABLE IF EXISTS StartupMarkets;                          # Markets for the startup's products
        CREATE TABLE StartupMarkets
                (id INT NOT NULL,                             # id for the startup, e.g. 123
                market VARCHAR(32) NOT NULL,                  # market for the startup's product, e.g. mobile
                FOREIGN KEY (id) REFERENCES StartupListing(id));      # links to the startup listing

DROP TABLE IF EXISTS StartupTypes;                    # Type of startup = vc, incubator, closed, acquired
        CREATE TABLE StartupTypes
                (id INT NOT NULL PRIMARY KEY,                 # id for the startup, e.g. 123
                type VARCHAR(32) NOT NULL,                    # type of startup, e.g. acquired
                FOREIGN KEY (id) REFERENCES StartupListing(id));

DROP TABLE IF EXISTS StartupRoles;                            # People and their roles in startups
        CREATE TABLE StartupRoles
                (id INT NOT NULL PRIMARY KEY,                 # id of the user, e.g. 345
                name VARCHAR(32) NOT NULL,                    # name of the person, e.g. John Smith
                roleID INT NOT NULL,                          # the id of their role
                role VARCHAR(32) NOT NULL,            # their role in the startup, e.g. founder, past_investor
                startupID INT NOT NULL,               # id of the startup that they are related to, e.g. 123
                 FOREIGN KEY (id) REFERENCES StartupListing(id));
```

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.

MySQL Code

LoadTables.SQL

```
# Loads the database with startup entries from the .csv files.
DROP TABLE IF EXISTS StartupListing;                          # Startup data
        CREATE TABLE StartupListing
                (id INT NOT NULL PRIMARY KEY,                 # id = 123
                startupName VARCHAR(32) NOT NULL,             # startupName = 'AngelList'
                quality int NOT NULL,                         # quality = 10, based on AngelList rankings
                highConcept VARCHAR(140),                     # highConcept = 'A platform for startups'
                companyUrl VARCHAR(140),                      # companyUrl = 'http://angel.co'
                companySize VARCHAR(12),                      # companySize = '1-10'
                thumbUrl VARCHAR(200));                       # thumbUrl = 'http://....jpg' thumbnail image

LOAD DATA LOCAL INFILE '/Users/samanthasiow/Documents/Fall 2014/Databases/FinalProject/SQLScripts/AllStartupListings.csv' INTO TABLE
StartupListing
        FIELDS TERMINATED BY ','
        LINES TERMINATED BY '\n';

SELECT * FROM StartupListing;

#Load database with startup locations from csv files
DROP TABLE IF EXISTS StartupLocation;                         # Location data for each startup
        CREATE TABLE StartupLocation
                (id INT NOT NULL,                             # id for the startup, e.g. 123
                location VARCHAR(32) NOT NULL,                # location of a startup’s office, e.g. San Francisco
    FOREIGN KEY (id) REFERENCES StartupListing(id));

LOAD DATA LOCAL INFILE '/Users/samanthasiow/Documents/Fall 2014/Databases/FinalProject/SQLScripts/AllStartupLocations.csv' INTO TABLE
StartupLocation
        FIELDS TERMINATED BY ','
        LINES TERMINATED BY '\n';

SELECT * FROM StartupLocation;
# Load database with startup markets from the csv files
DROP TABLE IF EXISTS StartupMarkets;                          # Markets for the startup's products
        CREATE TABLE StartupMarkets
                (id INT NOT NULL,                             # id for the startup, e.g. 123
                market VARCHAR(32) NOT NULL,                  # market for the startup's product, e.g. mobile
                FOREIGN KEY (id) REFERENCES StartupListing(id));      # links to the startup listing

LOAD DATA LOCAL INFILE '/Users/samanthasiow/Documents/Fall 2014/Databases/FinalProject/SQLScripts/AllStartupMarkets.csv' INTO TABLE
StartupMarkets
        FIELDS TERMINATED BY ','
        LINES TERMINATED BY '\n';

SELECT * FROM StartupMarkets;
```

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.

```
# Load database with startup types from the csv files
DROP TABLE IF EXISTS StartupTypes;                  # Type of startup = vc, incubator, closed, acquired
        CREATE TABLE StartupTypes
                (id INT NOT NULL PRIMARY KEY,              # id for the startup, e.g. 123
                type VARCHAR(32) NOT NULL,                 # type of startup, e.g. acquired
                FOREIGN KEY (id) REFERENCES StartupListing(id));

LOAD DATA LOCAL INFILE '/Users/samanthasiow/Documents/Fall 2014/Databases/FinalProject/SQLScripts/AllStartupTypes.csv' INTO TABLE
StartupTypes
        FIELDS TERMINATED BY ','
        LINES TERMINATED BY '\n';

SELECT * FROM StartupTypes;
# Load database with people and their roles in startups.
DROP TABLE IF EXISTS StartupRoles;                         # People and their roles in startups
        CREATE TABLE StartupRoles
                (id INT NOT NULL PRIMARY KEY,              # id of the user, e.g. 345
                name VARCHAR(32) NOT NULL,                 # name of the person, e.g. John Smith
                roleID INT NOT NULL,                       # the id of their role
                role VARCHAR(32) NOT NULL,         # their role in the startup, e.g. founder, past_investor
                startupID INT NOT NULL,            # id of the startup that they are related to, e.g. 123
                 FOREIGN KEY (id) REFERENCES StartupListing(id));

LOAD DATA LOCAL INFILE '/Users/samanthasiow/Documents/Fall 2014/Databases/FinalProject/SQLScripts/AllUserRoles.csv' INTO TABLE
StartupRoles
        FIELDS TERMINATED BY ','
        LINES TERMINATED BY '\n';

SELECT * FROM StartupRoles;
```

ShowAcquiredStartups.sql

```
/* Display all acquired startups */
DELIMITER $$

DROP PROCEDURE IF EXISTS ShowAcquiredStartups $$

CREATE PROCEDURE ShowAcquiredStartups()
BEGIN
        SELECT startupName as StartupName, highConcept as HighConcept, companyUrl as URL
        FROM StartupListing as S, StartupTypes as T
        WHERE S.id = T.id and T.type = 'acquired';
END
$$
```

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.

ShowLargestMarketPerLocation.sql

```
/* Display the most popular startup market by location */
DELIMITER $$

DROP PROCEDURE IF EXISTS ShowLargestMarketPerLocation $$

CREATE PROCEDURE ShowLargestMarketPerLocation(IN locationName VARCHAR(255))
BEGIN

CREATE OR REPLACE VIEW NumStartupsPerLocation AS
        SELECT *
        FROM (SELECT M.market as Market, count(market) as NumStartups
                FROM StartupLocation as L, StartupMarkets as M
                WHERE L.id = M.id and L.location = locationName
                GROUP BY market
                ORDER BY NumStartups DESC) as M
        WHERE M.NumStartups >= ALL (SELECT count(market) as NumStartups
                                                        FROM StartupLocation as L, StartupMarkets as M
                                                        WHERE L.id = M.id and L.location = locationName
                                                        GROUP BY market
                                                        ORDER BY NumStartups DESC);
END;
$$
```

ShowNumStartupsAtLocation.sql

```
/* Takes a location param, and displays the number of startups at that location */
DELIMITER $$

DROP PROCEDURE IF EXISTS ShowNumStartupsAtLocation $$

CREATE PROCEDURE ShowNumStartupsAtLocation(IN locationName VARCHAR(255))
BEGIN
        SELECT L.location, count(L.location) as NumStartups
        FROM StartupLocation as L, StartupListing as S
        WHERE L.id = S.id and L.location = locationName;
END;
$$
```

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.

ShowNumStartupsPerLocation.sql

```
/* Display the number of startups per location */
DELIMITER $$

DROP PROCEDURE IF EXISTS ShowNumStartupsPerLocation $$

CREATE PROCEDURE ShowNumStartupsPerLocation()
BEGIN
        SELECT L.location, count(L.location) as NumStartups
        FROM StartupLocation as L, StartupListing as S
        WHERE L.id = S.id
        GROUP BY location
        ORDER BY NumStartups DESC;
END;
$$
```

ShowPercentageAcquiredPerLocation.sql

```
/* Display an ordered list of the percentage startups acquired by location */
DELIMITER $$

DROP PROCEDURE IF EXISTS ShowPercentageAcquiredPerLocation $$

CREATE PROCEDURE ShowPercentageAcquiredPerLocation()
BEGIN

CREATE OR REPLACE VIEW NumStartupsPerLocation AS
        SELECT DISTINCT  L.location, count(L.location) as numPerCity
        FROM StartupMarkets as M, StartupLocation as L
        WHERE M.id = L.id
        GROUP BY L.location;

SELECT DISTINCT A.location, (A.numAcquired/N.numPerCity)*100 as PercentageAcquired
FROM NumStartupsPerLocation as N,
        (SELECT DISTINCT L.location, count(location) as numAcquired
        FROM StartupTypes as T, StartupLocation as L
        WHERE T.id = L.id and T.type = 'acquired'
        GROUP BY L.Location) as A
WHERE A.location = N.location
ORDER BY PercentageAcquired DESC;

END;
$$
```

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.

ShowPercentageMarketPerLocation.sql

```
/* Given a market type, display the % of startups of that market in all locations */
DELIMITER $$

DROP PROCEDURE IF EXISTS ShowPercentageMarketPerLocation $$

CREATE PROCEDURE ShowPercentageMarketPerLocation(IN marketType VARCHAR(255))
BEGIN

CREATE OR REPLACE VIEW NumStartupsPerLocation AS
        SELECT DISTINCT  L.location, count(L.location) as numPerCity
        FROM StartupMarkets as M, StartupListing as S, StartupLocation as L
        WHERE M.id = S.id and S.id = L.id
        GROUP BY L.location;

  SELECT DISTINCT M.location, (M.numPerCity/N.numPerCity)*100 as PercentageMobile
        FROM NumStartupsPerLocation as N,
                (SELECT DISTINCT L.id, L.location, count(L.location) as numPerCity
                FROM StartupMarkets as M, StartupListing as S, StartupLocation as L
                WHERE M.id = S.id and S.id = L.id and M.market = marketType
                GROUP BY L.location) as M
        WHERE M.location = N.location
  ORDER BY PercentageMobile DESC;

END;
$$
```

ShowPersonsByStartup.sql

```
/* Show all the people related to the startup and what their relation is */
DELIMITER $$

DROP PROCEDURE IF EXISTS ShowPersonsByStartup $$

CREATE PROCEDURE ShowPersonsByStartup(IN startupID INT)
BEGIN
        IF EXISTS (SELECT * FROM StartupListing as SL WHERE SL.id = startupID)
                THEN
                        SELECT name as Name, role as Role, startupName as StartupName
                        FROM StartupListing as S, StartupRoles as R
                        WHERE R.startupID = S.id and S.id = startupID;
        ELSE
                SELECT 'No people found for that startup.' AS  'Error Message';
        END IF;
END
$$
```

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.

ShowRolesByStartupID.sql

```
/* Display all people with a given role, given the startup id*/
DELIMITER $$

DROP PROCEDURE IF EXISTS ShowRolesByStartupID $$

CREATE PROCEDURE ShowRolesByStartupID(IN startupID INT, IN queryBy VARCHAR(255))
BEGIN
	IF EXISTS (SELECT * FROM StartupListing as SL WHERE SL.id = startupID)
		THEN
			SELECT name as Name, role as Role, startupName as StartupName
			FROM StartupListing as S, StartupRoles as R
			WHERE R.startupID = S.id and S.id = startupID and R.role = queryBy;
	ELSE
		SELECT 'No people found for that role for that startup.' AS  'Error Message';
	END IF;
END;
$$
```

ShowStartupByLocation.sql

```
/* Display all startups at a given location */
DELIMITER $$

DROP PROCEDURE IF EXISTS ShowStartupByLocation $$

CREATE PROCEDURE ShowStartupByLocation(IN locationName VARCHAR(255))
BEGIN
	IF EXISTS (SELECT location FROM StartupLocation WHERE location = locationName)
		THEN
			SELECT startupName as StartupName, highConcept as HighConcept, companyUrl as URL
			FROM StartupListing as S, StartupLocation as L
      WHERE S.id = L.id and L.location = locationName;
	ELSE
		SELECT 'No Startups Located.' AS  'Error Message';
	END IF;
END
$$
```

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.

ShowStartupByMarket.sql

```
/* Display all startups in a given market */
DELIMITER $$

DROP PROCEDURE IF EXISTS ShowStartupByMarket $$

CREATE PROCEDURE ShowStartupByMarket(IN marketName VARCHAR(255))
BEGIN
        IF EXISTS (SELECT market FROM StartupMarkets WHERE marketName = market)
                THEN
                        SELECT startupName as StartupName, highConcept as HighConcept, companyUrl as URL
                        FROM StartupListing as S, StartupMarkets as M
      WHERE S.id = M.id and M.market = marketName;
        ELSE
                SELECT 'No Startups found in that Market.' AS  'Error Message';
        END IF;
END
$$
```

ShowStartupIDByName.sql

```
/* Given a startup id, return the name of the startup */
DELIMITER $$

DROP PROCEDURE IF EXISTS ShowStartupIDByName $$

CREATE PROCEDURE ShowStartupIDByName(IN startupName VARCHAR(255))
BEGIN
        IF EXISTS (SELECT * FROM StartupListing as SL WHERE SL.startupName = startupName)
                THEN
                        SELECT id as id
                        FROM StartupListing as S
                        WHERE S.startupName = startupName;
        ELSE
                SELECT 'No startups found for that ID.' AS  'Error Message';
        END IF;
END
$$
```

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.

ShowPersonRoles.sql

```
/* Show all roles a person has, given their name */
DELIMITER $$


DROP PROCEDURE IF EXISTS ShowPersonRoles $$


CREATE PROCEDURE ShowPersonRoles(IN personName VARCHAR(255))
BEGIN
	IF EXISTS (SELECT * FROM StartupRoles WHERE personName = name)
		THEN
			SELECT name as Name, role as Role, startupName as StartupName
			FROM StartupListing as S, StartupRoles as R
			WHERE R.startupID = S.id and R.name = personName;
	ELSE
		SELECT 'No Startups found in that Market.' AS  'Error Message';
	END IF;
END
$$
```

Python Scripts for Retrieving Data on the AngelList RESTful Interface

pullStartups.py

```python
import os,sys
import urllib
import json
from angellist import AngelList

def main(argv):

      if len(sys.argv) != 3 or not argv[1].isdigit() or not argv[2].isdigit():
        print "Usage is python read.py <start ID> <end ID>"
        exit()

      if ( int(argv[1]) > int(argv[2]) ):
        print "Invalid range"
        exit()

      filename = "Startups_" + argv[1] + "-" + argv[2]

      if not os.path.isfile(filename):

        angel = AngelList("6093fd1f428b52cd5e5f039e63b4870ce3b3c5884ff35e32")

      # angelapi.search({'method':'GET', 'query':'search-string'})

        f = open(filename, 'w')

        # Enter range of IDs here
        for x in range( int(argv[1]), int(argv[2]) ):
                try:
                        f.write(json.dumps( angel.startups({'method':'GET', 'id':str(x)}) ) )
                        f.write('\n')
                except Exception:
                        pass

if __name__ == "__main__":
      main(sys.argv)
```

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.

pullStartupRoles.py

```
import sys, os
import json
import urllib2
from angellist import AngelList

def main(argv):


      if len(sys.argv) != 3 or not argv[1].isdigit() or not argv[2].isdigit():
        print "Usage is python pullStartupRoles.py <start ID> <end ID>"
        exit()

      startID = int(argv[1])
      endID = int(argv[2])

      if ( startID >= endID ):
        print "Invalid Range"

      try:
        roles = open( "roles", 'r')

      except IOError:
        "File does not exist"

      filename = "Roles_" + argv[1] + "-" + argv[2]

      if not os.path.isfile(filename):

        angel = AngelList("6093fd1f428b52cd5e5f039e63b4870ce3b3c5884ff35e32")
        ANGELLIST_ROLES_URL = "https://api.angel.co/1/startup_roles?startup_id="

        f = open(filename, 'w')

        # Enter range of IDs here

        for startupID in roles:
                try:
                        if ( int(startupID) >= startID and int(startupID) <= endID ):
                                result = json.dumps(json.loads(urllib2.urlopen(ANGELLIST_ROLES_URL +
startupID).read()))
                                f.write( '{"startupID":' + startupID.rstrip('\n') + ', ' + result[1:])
                                f.write('\n')
                except Exception:
                        pass


if __name__ == "__main__":
      main(sys.argv)
```

---

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.

Java Program for parsing JSON to .csv files

StartupListing.java

```java
package StartupData;

/* Holds all the data for a startup */
public class StartupListing {
	private String id;
	private String name;
	private String quality;
	private String highConcept;
	private String companyUrl;
	private String companySize;
	private String thumbUrl;

	public StartupListing(String id,String name, String quality,
			String highConcept, String companyUrl, String companySize, String thumbUrl) {
		super();
		this.id = id;
		this.name = name;
		this.quality = quality;
		this.highConcept = highConcept;
		this.companyUrl = companyUrl;
		this.companySize = companySize;
		this.thumbUrl = thumbUrl;
	}

	@Override
	public String toString() {
		return  id.replace(",", ";")
				+ "," + name.replace(",", ";")
				+ "," + quality.replace(",", ";")
				+ "," + highConcept.replace(",", ";") + "," + companyUrl.replace(",", ";")
				+ "," + companySize.replace(",", ";") + "," + thumbUrl;
	}


}
```

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.

AngelListStartupJSONtoCSV.java

```
package StartupData;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;


public class AngelListStartupJSONtoCSV {
        private static ArrayList<StartupListing> allStartupListings;      // all startups
        private static ArrayList<String> allStartupLocations;     // relation for startup to locations
        private static ArrayList<String> allStartupMarkets;               // relation for startup to markets
        private static ArrayList<String> allStartupTypes;         // relation for startup to types
        private static String[] StartupFiles;               // files to read

        public static void main(String[] args) throws Exception {
                setupArrays();

                for (int i=0 ; i < StartupFiles.length; i++) {
                        readFromFile("Startups_" + StartupFiles[i]);
                }

                printToFile(allStartupListings, "StartupListings");
                printToFile(allStartupLocations, "StartupLocations");
                printToFile(allStartupMarkets, "StartupMarkets");
                printToFile(allStartupTypes, "StartupTypes");
        }

    // Used an open source JSON parser
    // Run with the .jar from http://mvnrepository.com/artifact/org.json/json
        /**
         * Parse the JSON string into a startupData object,
         * and add it to the appropriate list.
         * @param startupObject      json formatted string
         */
        private static void parseJSONtoList(String startupObject) throws Exception {
            String id, name, quality, highConcept, companyUrl, companySize, thumbUrl;
                StartupListing startupData;

                try {
                JSONObject startup = new JSONObject(startupObject);
                id = startup.get("id") + "";
                System.out.println(id);
                // if not hidden, get all the data
                if (!startup.getBoolean("hidden")) {

                        name = startup.get("name") + "";
                        quality = startup.get("quality") + "";
                        highConcept = startup.get("high_concept") + "";
                        companyUrl = startup.get("company_url") + "";
                        thumbUrl = startup.get("thumb_url") + "";

                        // for each market type the startup is listed as,
                        // add to the markets list.
                        JSONArray market = startup.getJSONArray("markets");
                        for (int k = 0; k < market.length(); k++) {
```

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.

```
                            JSONObject marketTag = market.getJSONObject(k);
                            allStartupMarkets.add(id + "," + marketTag.getString("name"));
                    }
                    // for each location the startup is listed as,
                    // add to the location list.
                    JSONArray location = startup.getJSONArray("locations");
                    for (int k = 0; k < location.length(); k++) {
                            JSONObject locationTag = location.getJSONObject(k);
                            allStartupLocations.add( id + "," + locationTag.getString("name"));
                    }

                    companySize = startup.get("company_size") + "";
                    // for each company type the startup is listed as,
                    // add to the type list.
                    JSONArray type = startup.getJSONArray("company_type");
                    for (int k = 0; k < type.length(); k++) {
                            JSONObject companyType = type.getJSONObject(k);
                            allStartupTypes.add(id + "," + companyType.getString("name"));
                    }

                    // crate a startupData object to hold all the information
                    startupData = new StartupListing(id,
                                    name, quality, highConcept, companyUrl, companySize, thumbUrl);
                    allStartupListings.add(startupData);
            }
            } catch (JSONException e) {
                e.printStackTrace();
            }
    }

    /**
     * Read from the file and parse line by line.
     * @param path       path to file
     * @throws Exception  file not found
     */
    private static void readFromFile(String path) throws Exception {
            FileReader fr = new FileReader(path);
        BufferedReader reader = new BufferedReader(fr);
        String next;
        while ((next = reader.readLine()) != null) {
            parseJSONtoList(next);
        }
    }




    /**
     * Print the contents of the array list into a file
     * @param listToPrint the list to print
     * @param fileName          the name of the file to print to
     */
    private static void printToFile(ArrayList listToPrint, String fileName) {
            PrintWriter writer;
            try {
                    writer = new PrintWriter("all" + fileName + ".csv", "UTF-8");
                    for(int i = 0; i < listToPrint.size(); i++) {
                            writer.println(listToPrint.get(i).toString());
                    }
                    writer.close();
            } catch (FileNotFoundException e) {
                    e.printStackTrace();
```

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.

```java
                } catch (UnsupportedEncodingException e) {
                        e.printStackTrace();
                }
        }

        private static void setupArrays() {
                allStartupListings = new ArrayList<>();
                allStartupLocations = new ArrayList<>();
                allStartupMarkets = new ArrayList<>();
                allStartupTypes = new ArrayList<>();
                StartupFiles = new String[6];
                StartupFiles[0] = "1-1000";
                StartupFiles[1] = "1001-2000";
                StartupFiles[2] = "2001-3000";
                StartupFiles[3] = "3001-3500";
                StartupFiles[4] = "6001-7000";
                StartupFiles[5] = "7001-8000";
        }
}
```

StartupRole.java

```java
package StartupRoles;
/* Holds all the data for an individual and their role in a startup */
public class StartupRole {
        String role;
        String name;;
        String userID;
        String roleID;
        String startupID;

        public StartupRole(String userID, String name, String role, String roleID, String startupID) {
                this.userID = userID;
                this.name = name.replace(',', ';');
                this.role = role.replace(',', ';');
                this.roleID = roleID;
                this.startupID = startupID;
        }

        public String toString() {
                return userID + "," + name + "," + roleID + "," + role + "," + startupID;
        }
}
```

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.

AngelListRoleJSONtoCSV.java

```java
package StartupRoles;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.HashMap;

import org.json.JSONArray;
import org.json.JSONObject;
/*Convert a JSON formatted string into an entry on a csv */
public class AngelListRoleJSONtoCSV {
	private static ArrayList<StartupRole> allUserRoles;

	public static void main(String[] args) throws Exception {
		allUserRoles = new ArrayList<>();
		String path = "Roles_1-2000";
		readFromFile(path);

		printToFile(allUserRoles, "UserRoles");
	}

	/**
	 * Parse the JSON string into a startupRole object,
	 * and add it to the allUserRoles list.
	 * @param startupRoles 		json formatted string
	 */
	private static void parseJSONToList(String startupRoles) {
		StartupRole user;
		String userID, startupID, userName, userRole, roleID;

		JSONObject object = new JSONObject(startupRoles);
		startupID = object.get("startupID") + "";
		JSONArray roles = object.getJSONArray("startup_roles");
		// for each role the startup has, create a user with that role
		// that is attached to the startup
		for (int i = 0; i < roles.length(); i++) {
			JSONObject role = roles.getJSONObject(i);
			roleID = role.get("id") + "";
			userRole = role.get("role") + "";
			JSONObject userInfo = role.getJSONObject("user");
			userName = userInfo.get("name") + "";
			userID = userInfo.get("id") + "";
			user = new StartupRole(userID, userName, userRole, roleID, startupID);
			allUserRoles.add(user);
		}
	}
```

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.

```
	/**
	 * Read from the file and parse line by line.
	 * @param path       path to file
	 * @throws Exception file not found
	 */
	@SuppressWarnings("resource")
	private static void readFromFile(String path) throws Exception {
		FileReader fr = new FileReader(path);
	    BufferedReader reader = new BufferedReader(fr);
	    String next;
	    next = reader.readLine();
	    parseJSONToList(next);
	    while ((next = reader.readLine()) != null) {
	        parseJSONToList(next);
	    }
	}

	/**
	 * Print the contents of the array list into a file
	 * @param listToPrint the list to print
	 * @param fileName           the name of the file to print to
	 */
	private static void printToFile(ArrayList listToPrint, String fileName) {
		PrintWriter writer;
		try {
			writer = new PrintWriter("all" + fileName + ".csv", "UTF-8");
			for(int i = 0; i < listToPrint.size(); i++) {
				writer.println(listToPrint.get(i).toString());
			}
			writer.close();
		} catch (FileNotFoundException e) {
			e.printStackTrace();
		} catch (UnsupportedEncodingException e) {
			e.printStackTrace();
		}
	}
}
```

[1] https://angel.co/api, and a sample query: https://api.angel.co/1/startups/6702.
[2] The Python script can be found in the PythonScripts folder.
[3] The Java program can be found in the AngelListJSONtoCSVParser folder.
[4] See the LoadTables.sql file in the SQLScripts folder.