Introduction to Big Data
Group 5
Project Phase 2

Document Oriented Model

Collection Name - Business

```
{
        “_id”: “22 character unique alphanumeric string”,
        “name”: “String Business Name”,
        “address” : {
                “city”: “String city”
                “state”: “String State”
                “postalCode”: “String postal code”
        },
        “isTakeout”: Boolean value,
        “isOpen”: Boolean value,
        “stars”: float, rounded to half stars,
        “reviewCount”: integer, number of reviews,
        “categories”: [“”, “”, “”, .... ,””]
}
```

Collection Name - Users

```
{
        “_id”: “22 character unique alphanumeric string”,
        “name”: “String Business Name”,
        “reviewCount”: integer, number of reviews written by the user,
        “fans”: integer, number of fans the user has,
        “averageStars”: Float, average rating of all reviews,
}
```

Collection Name - Review

```
{
        “_id”:  “22 character unique alphanumeric string”,
        “userId”: “22 character unique alphanumeric string”
        “businessId”: “22 character unique alphanumeric string”
        “stars”: float, stars given by the user for the review
```

“**date”: “**string, date format YYYY-MM-DD”
}

Collection Name - Tip

{
“**_id**”: For each document an id is auto generated
“**businessId**”**: “**String**,** Refers to the ID in the Business collection”
“**userId**”**: “**String**,** Refers to the ID in the Business collection”
“**tip**”**: “**String tip written by the user**”**
“**tipDate”: “**string, date format YYYY-MM-DD”
“**complimentCount**” **:** integer, how many compliments the tip has
}

Relational schema - The relational schema in phase 1 had the following 8 tables:
Business(Primary key = Business_Id)
Address(foreign key = Business_Id)
Business_category(foreign key = Business_Id, Category_Id)
Category(primary key = Category_Id)
Rating(foreign key = Business_Id)
User(primary key = User_Id)
Review(primary key = Review_Id, foreign key = Business_Id, User_Id)
Tip(foreign key = Business_Id, User_Id)

The relational schema has 4 defined primary keys that are as -
Business_Id
Category_Id
User_Id
Review_Id

Document-oriented Model - The document oriented model contains the following collections
Business(id is businessId)
Users(id is userId)
Reviews(id is reviewId)
Tip(id is autogenerated)

In the document oriented model the relational tables Business, Address, Business_category, Category and Rating are all combined into one collection named Business.This is done so as to take advantage of the embedded documents and arrays in document oriented databases which reduce the number of collections.This leads to reduced number of joins which in turn makes querying easier and faster.
The other tables in the relational schema are converted to collections directly, without a lot of modifications.

Based on the relational schema five interesting queries are as follows:

1. Famous Restaurants by state that have rating greater than 4 and with a review count of at least > 100.
2. 5 star Pizza places in Las Vegas
3. All indian restaurants in toronto which have rating greater than 3
4. Best Beer Spots in Each City
5. States with the best police department.

The above stated queries are present in the python file Interesting_Queries.py
The time taken by these queries to run, without any indexing is as :

Query	Time
1. Famous Restaurants by state that have rating greater than 4 and with a review count of at least > 100.	283 msec
2. 5 star Pizza places in Las Vegas	128 msec
3. All indian restaurants in toronto which have rating greater than 3	143 msec
4. Best Beer Spots in Each City	162 msec
5. States with the best police department.	158 msec

The proposed indexes to make the queries faster are as:

1. Index on the star column of the rating table
   ```
   CREATE INDEX ON rating(star)
   ```
2. Index on the review_count column of the rating table
   ```
   CREATE INDEX ON rating(review_count)
   ```
3. Index on the city column in the address table
   ```
   CREATE INDEX ON address(city)
   ```
4. Index on the type column in the category table
   ```
   CREATE UNIQUE INDEX ON category(category_type)
   ```

The execution time of queries after indexing are as:

Query	Time
1. Famous Restaurants by state that have rating greater than 4 and with a review count of at least > 100.	278 msec
2. 5 star Pizza places in Las Vegas	83 msec
3. All indian restaurants in toronto which have rating greater than 3	77 msec
4. Best Beer Spots in Each City	96 msec
5. States with the best police department.	132 msec

Functional Dependency:
All the functional dependencies from our relational model contain the primary key. So if we prune the functional dependency where the primary key is on the left hand side, the final functional dependencies is zero.

Functional Dependency For Address:

1. Business_id -> city
2. Business_id -> state
3. Business_id -> postal_code
   and combinations involving business_id on the left hand side.

Functional Dependency For Business:

1. Business_id -> name
2. Business_id -> is_open
3. Business_id -> is_takeout
   and combinations involving business_id on the left hand side.

Functional Dependency for business_category:

1. Business_id -> category_id
   and combinations involving business_id on the left hand side.

Functional Dependency for Review:

1. Review_id -> user_id
2. Review_id -> business_id
3. Review_id - > stars

4. Review_id -> review_date
   and combinations involving review_id on the left hand side.

Functional Dependency for category:

1. id -> category_type

Functional Dependency for Rating:

1. Business_id -> star
2. Business_id -> review_count

Functional Dependency for Tip:

1. User_id,business_id -> tip
2. User_id,business_id -> complement_count
3. User_id,business_id -> tip_date
   And combinations involving user_id,business_id attribute on the left hand side.

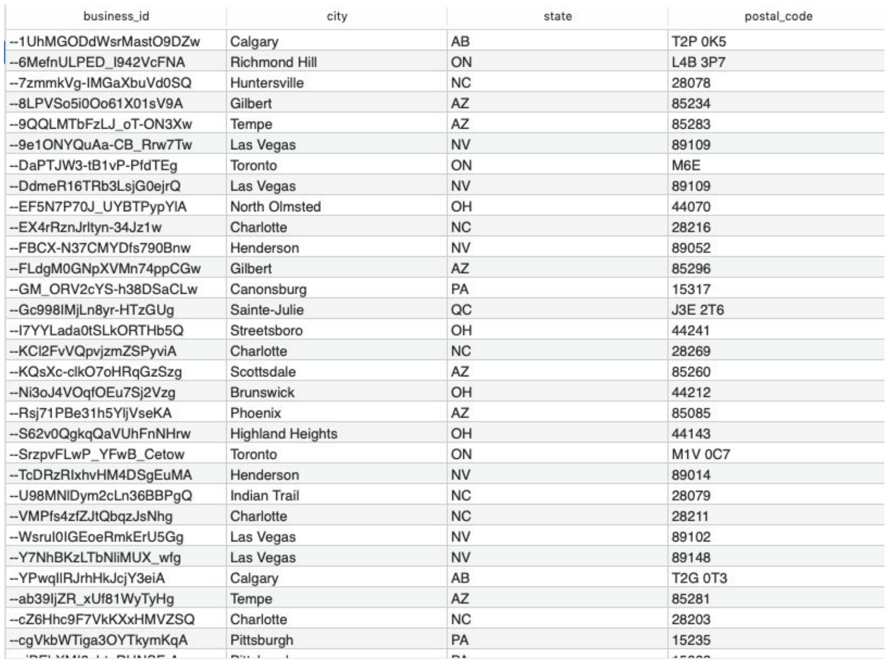
Functional Dependency for users:

1. User_id -> name
2. User_id -> review_count
3. User_id -> fans
4. User_id -> average_stars
   And combinations involving user_id on the left hand side.

Run FunctionalDependency.py

Normalisation:

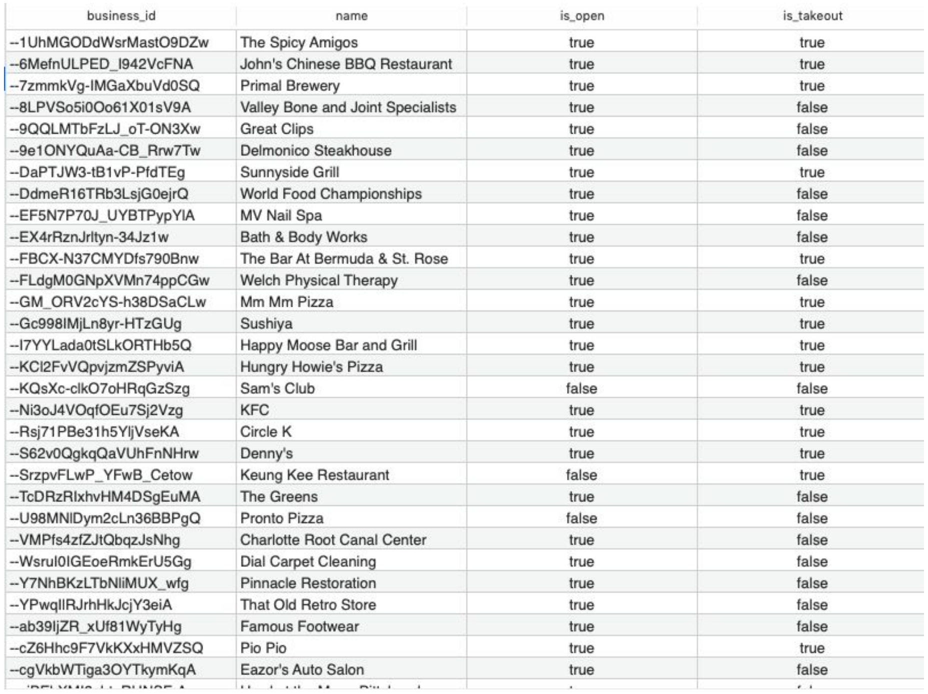
First Normal Form:
Our relational model is in the first normal form since all the values in each table are of atomic type(cannot be reduced any further) and also all the attributes in all the tables have values of the same datatype.
Eg Address Table



business_id	city	state	postal_code
--1UhMGODdWsrMastO9DZw	Calgary	AB	T2P 0K5
--6MefnULPED_I942VcFNA	Richmond Hill	ON	L4B 3P7
--7zmmkVg-IMGaXbuVd0SQ	Huntersville	NC	28078
--8LPVSo5i0Oo61X01sV9A	Gilbert	AZ	85234
--9QQLMTbFzLJ_oT-ON3Xw	Tempe	AZ	85283
--9e1ONYQuAa-CB_Rrw7Tw	Las Vegas	NV	89109
--DaPTJW3-tB1vP-PfdTEg	Toronto	ON	M6E
--DdmeR16TRb3LsjG0ejrQ	Las Vegas	NV	89109
--EF5N7P70J_UYBTPypYlA	North Olmsted	OH	44070
--EX4rRznJrltyn-34Jz1w	Charlotte	NC	28216
--FBCX-N37CMYDfs790Bnw	Henderson	NV	89052
--FLdgM0GNpXVMn74ppCGw	Gilbert	AZ	85296
--GM_ORV2cYS-h38DSaCLw	Canonsburg	PA	15317
--Gc998IMjLn8yr-HTzGUg	Sainte-Julie	QC	J3E 2T6
--I7YYLada0tSLkORTHb5Q	Streetsboro	OH	44241
--KCl2FvVQpvjzmZSPyviA	Charlotte	NC	28269
--KQsXc-clkO7oHRqGzSzg	Scottsdale	AZ	85260
--Ni3oJ4VOqfOEu7Sj2Vzg	Brunswick	OH	44212
--Rsj71PBe31h5YljVseKA	Phoenix	AZ	85085
--S62v0QgkqQaVUhFnNHrw	Highland Heights	OH	44143
--SrzpvFLwP_YFwB_Cetow	Toronto	ON	M1V 0C7
--TcDRzRIxhvHM4DSgEuMA	Henderson	NV	89014
--U98MNlDym2cLn36BBPgQ	Indian Trail	NC	28079
--VMPfs4zfZJtQbqzJsNhg	Charlotte	NC	28211
--WsruI0IGEoeRmkErU5Gg	Las Vegas	NV	89102
--Y7NhBKzLTbNliMUX_wfg	Las Vegas	NV	89148
--YPwqIIRJrhHkJcjY3eiA	Calgary	AB	T2G 0T3
--ab39IjZR_xUf81WyTyHg	Tempe	AZ	85281
--cZ6Hhc9F7VkKXxHMVZSQ	Charlotte	NC	28203
--cgVkbWTiga3OYTkymKqA	Pittsburgh	PA	15235

Second Normal Form:
Our relational model is in the second normal form as well because there exists no partial dependency. Partial Dependency is when a attribute is dependent on a part of the primary key.

Eg. Business Table

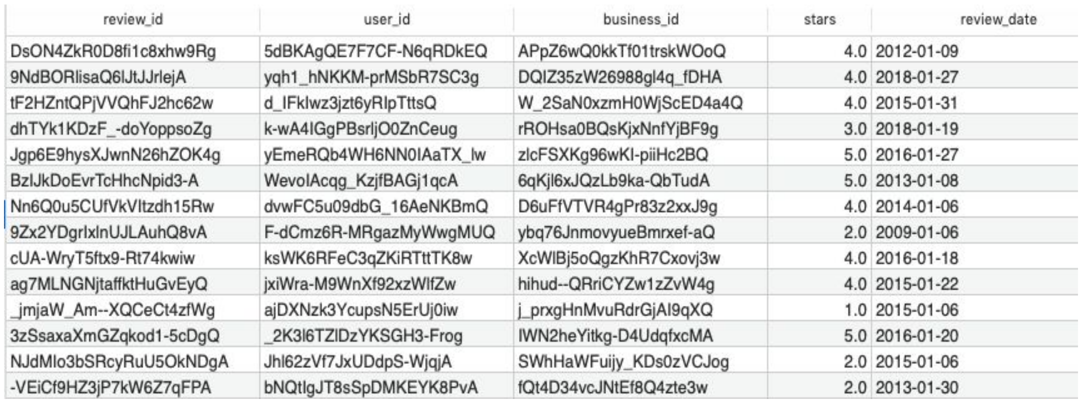


business_id	name	is_open	is_takeout
--1UhMGODdWsrMastO9DZw	The Spicy Amigos	true	true
--6MefnULPED_I942VcFNA	John's Chinese BBQ Restaurant	true	true
--7zmmkVg-IMGaXbuVd0SQ	Primal Brewery	true	true
--8LPVSo5i0Oo61X01sV9A	Valley Bone and Joint Specialists	true	false
--9QQLMTbFzLJ_oT-ON3Xw	Great Clips	true	false
--9e1ONYQuAa-CB_Rrw7Tw	Delmonico Steakhouse	true	false
--DaPTJW3-tB1vP-PfdTEg	Sunnyside Grill	true	true
--DdmeR16TRb3LsjG0ejrQ	World Food Championships	true	false
--EF5N7P70J_UYBTPypYlA	MV Nail Spa	true	false
--EX4rRznJrltyn-34Jz1w	Bath & Body Works	true	false
--FBCX-N37CMYDfs790Bnw	The Bar At Bermuda & St. Rose	true	true
--FLdgM0GNpXVMn74ppCGw	Welch Physical Therapy	true	false
--GM_ORV2cYS-h38DSaCLw	Mm Mm Pizza	true	true
--Gc998IMjLn8yr-HTzGUg	Sushiya	true	true
--I7YYLada0tSLkORTHb5Q	Happy Moose Bar and Grill	true	true
--KCl2FvVQpvjzmZSPyviA	Hungry Howie's Pizza	true	true
--KQsXc-clkO7oHRqGzSzg	Sam's Club	false	false
--Ni3oJ4VOqfOEu7Sj2Vzg	KFC	true	true
--Rsj71PBe31h5YljVseKA	Circle K	true	true
--S62v0QgkqQaVUhFnNHrw	Denny's	true	true
--SrzpvFLwP_YFwB_Cetow	Keung Kee Restaurant	false	true
--TcDRzRIxhvHM4DSgEuMA	The Greens	true	false
--U98MNlDym2cLn36BBPgQ	Pronto Pizza	false	false
--VMPfs4zfZJtQbqzJsNhg	Charlotte Root Canal Center	true	false
--WsruI0IGEoeRmkErU5Gg	Dial Carpet Cleaning	true	false
--Y7NhBKzLTbNliMUX_wfg	Pinnacle Restoration	true	false
--YPwqIIRJrhHkJcjY3eiA	That Old Retro Store	true	false
--ab39IjZR_xUf81WyTyHg	Famous Footwear	true	false
--cZ6Hhc9F7VkKXxHMVZSQ	Pio Pio	true	true
--cgVkbWTiga3OYTkymKqA	Eazor's Auto Salon	true	false

Third Normal Form:

The relational model is in 3rd Normal form as it does not contain any Transitive dependency. Transitive dependency is a condition where some attribute in some relation is dependent upon some non-primary key attribute of that relation.

Eg Review Table



review_id	user_id	business_id	stars	review_date
DsON4ZkR0D8fi1c8xhw9Rg	5dBKAgQE7F7CF-N6qRDkEQ	APpZ6wQ0kkTf01trskWOoQ	4.0	2012-01-09
9NdBORlisaQ6lJtJJrlejA	yqh1_hNKKM-prMSbR7SC3g	DQIZ35zW26988gl4q_fDHA	4.0	2018-01-27
tF2HZntQPjVVQhFJ2hc62w	d_IFklwz3jzt6yRlpTttsQ	W_2SaN0xzmH0WjScED4a4Q	4.0	2015-01-31
dhTYk1KDzF_-doYoppsoZg	k-wA4IGgPBsrljO0ZnCeug	rROHsa0BQsKjxNnfYjBF9g	3.0	2018-01-19
Jgp6E9hysXJwnN26hZOK4g	yEmeRQb4WH6NN0IAaTX_lw	zlcFSXKg96wKI-piiHc2BQ	5.0	2016-01-27
BzIJkDoEvrTcHhcNpid3-A	WevoIAcqg_KzjfBAGj1qcA	6qKjl6xJQzLb9ka-QbTudA	5.0	2013-01-08
Nn6Q0u5CUfVkVItzdh15Rw	dvwFC5u09dbG_16AeNKBmQ	D6uFfVTVR4gPr83z2xxJ9g	4.0	2014-01-06
9Zx2YDgrIxlnUJLAuhQ8vA	F-dCmz6R-MRgazMyWwgMUQ	ybq76JnmovyueBmrxef-aQ	2.0	2009-01-06
cUA-WryT5ftx9-Rt74kwiw	ksWK6RFeC3qZKiRTttTK8w	XcWlBj5oQgzKhR7Cxovj3w	4.0	2016-01-18
ag7MLNGNjtaffktHuGvEyQ	jxiWra-M9WnXf92xzWlfZw	hihud--QRriCYZw1zZvW4g	4.0	2015-01-22
_jmjaW_Am--XQCeCt4zfWg	ajDXNzk3YcupsN5ErUj0iw	j_prxgHnMvuRdrGjAI9qXQ	1.0	2015-01-06
3zSsaxaXmGZqkod1-5cDgQ	_2K3l6TZlDzYKSGH3-Frog	IWN2heYitkg-D4UdqfxcMA	5.0	2016-01-20
NJdMlo3bSRcyRuU5OkNDgA	Jhl62zVf7JxUDdpS-WjqjA	SWhHaWFuijy_KDs0zVCJog	2.0	2015-01-06
-VEiCf9HZ3jP7kW6Z7qFPA	bNQtlgJT8sSpDMKEYK8PvA	fQt4D34vcJNtEf8Q4zte3w	2.0	2013-01-30