FIT5137 Assignment 2

SHENYI ZHANG_30359953

MONASH University
Information Technology

ASSESSMENT COVER SHEET

Student ID number	
Given Name	
Family name	

Unit Name and Code:	**FIT5137 Advanced Database Technology**		
Campus:	**Clayton**		
Assignment Title:	**FIT5137 Individual Assignment - Sem 2/2020**		
Name of Lecturer:	**Agnes Haryanto**		
Name of Tutor:	Ningran Icey Li		
Tutorial Day and Time:	**Monday 14:00**		
Phone Number:	**0406958427**		
Email Address:	**Szha0177@student.monash.edu**		
Has any part of this assignment been previously submitted as part of another unit/course? ☐ Yes ☒ No			
Due Date:	**6/11/2020**	Date Submitted:	2/11/2020

All work must be submitted by the due date. If an extension of work is granted this must be specified with the signature of the lecturer/tutor.

Extension granted until (date)_____ Signature of lecturer/tutor _____

Please note that it is your responsibility to retain copies of your assessments.

Intentional plagiarism or collusion amounts to cheating under Part 7 of the Monash University (Council) Regulations

Plagiarism: Plagiarism means taking and using another person's ideas or manner of expressing them and passing them off as one's own. For example, by failing to give appropriate acknowledgement. The material used can be from any source (staff, students or the internet, published and unpublished works).

Collusion: Collusion means unauthorised collaboration with another person on assessable written, oral or practical work and includes paying another person to complete all or part of the work.

Where there are reasonable grounds for believing that intentional plagiarism or collusion has occurred, this will be reported to the Associate Dean (Education) or delegate, who may disallow the work concerned by prohibiting assessment or refer the matter to the Faculty Discipline Panel for a hearing.

Student Statement:

- I have read the university's Student Academic Integrity Policy and Procedures.
- I understand the consequences of engaging in plagiarism and collusion as described in Part 7 of the Monash University (Council) Regulations http://adm.monash.edu/legal/legislation/statutes
- have taken proper care to safeguard this work and made all reasonable efforts to ensure it could not be copied.
- No part of this assignment has been previously submitted as part of another unit/course.
- I acknowledge and agree that the assessor of this assignment may for the purposes of assessment, reproduce the assignment and:
  - i. provide to another member of faculty and any external marker; and/or
  - ii. submit it to a text matching software; and/or
  - iii. submit it to a text matching software which may then retain a copy of the assignment on its database for the purpose of future plagiarism checking.
- I certify that I have not plagiarised the work of others or participated in unauthorised collaboration when preparing this assignment.

Signature ...Shenyi Zhang..... Date.....2/11/2020.....

* delete (iii) if not applicable

The information on this form is collected for the primary purpose of assessing your assignment and ensuring the academic integrity requirements of the University are met. Other purposes of collection include recording your plagiarism and collusion declaration, attending to course and administrative matters and statistical analyses. If you choose not to complete all the questions on this form it may not be possible for Monash University to assess your assignment. You have a right to access personal information that Monash University holds about you, subject to any exceptions in relevant legislation. If you wish to seek access to your personal information or inquire about the handling of your personal information, please contact the University Privacy Officer: privacyofficer@adm.monash.edu.au

Updated: 17 Jun 2014

Task 1

1. **Database Design (codes in cypher file for this task):**
   a. Labels Identity:
      Rest,City, State, Country; (Rest means restaruants i.e. place)
      User, Employement, TypeOfWorking, Transport
      Cuisines, Day, Feature, Payment

   b. Relationship :
      To be shown in picture below

   c. Explaination:
      User and Restaurant nodes is related with relationship, with 3 rating properties stored in the relationship. The rating properties is created as list respectively in case one user reviews same place more than one time. Also a count property is also added.

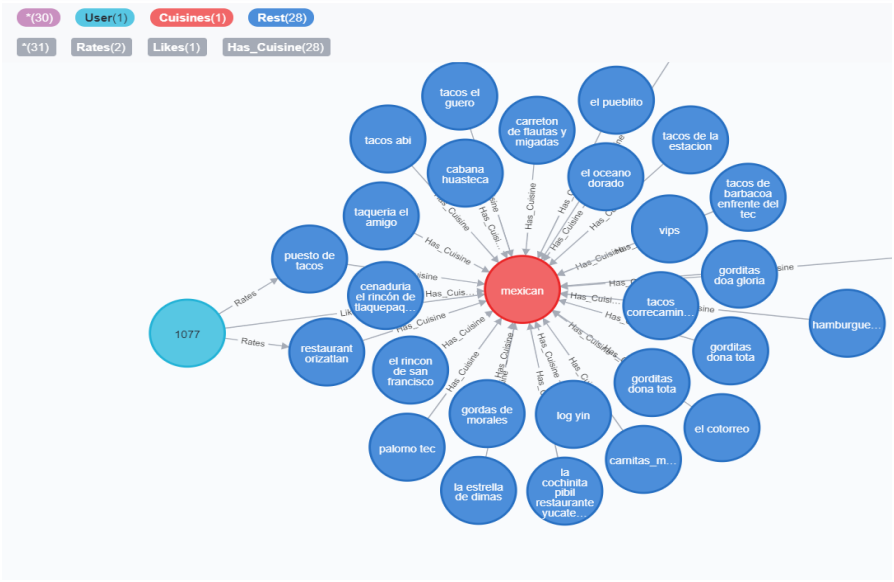
      Cuisines nodes is related with both Rest and User by two kinds of relationship respectively. Same as payment method; Note that if one place accept "any" payment, then this place will be created repationshp with all payment method exist in database.

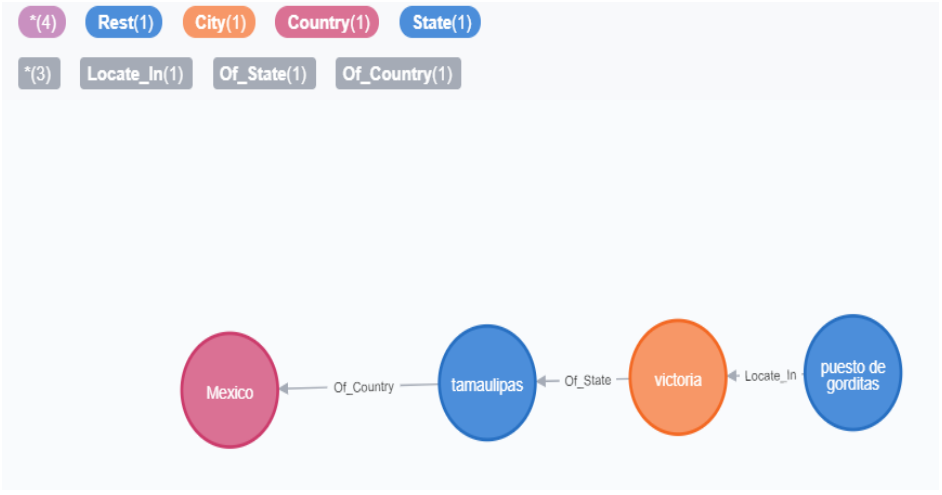
      To achieve the reduction of redundant data:
      The features of restaruants is created as a nodes with a common label "Feature", same as the Payment method, Transport etc.
      Opening Days is also created as nodes and the opening hours is stored as a property of the relationship between places and days.
      Rest is related with city nodes only instead of both city, state and country. To find the country of a restruants, we need to go through city nodes first, then city is related with state then country.

      Also, for places which exists in openingHour.csv but not in place.csv, they are also created in the database

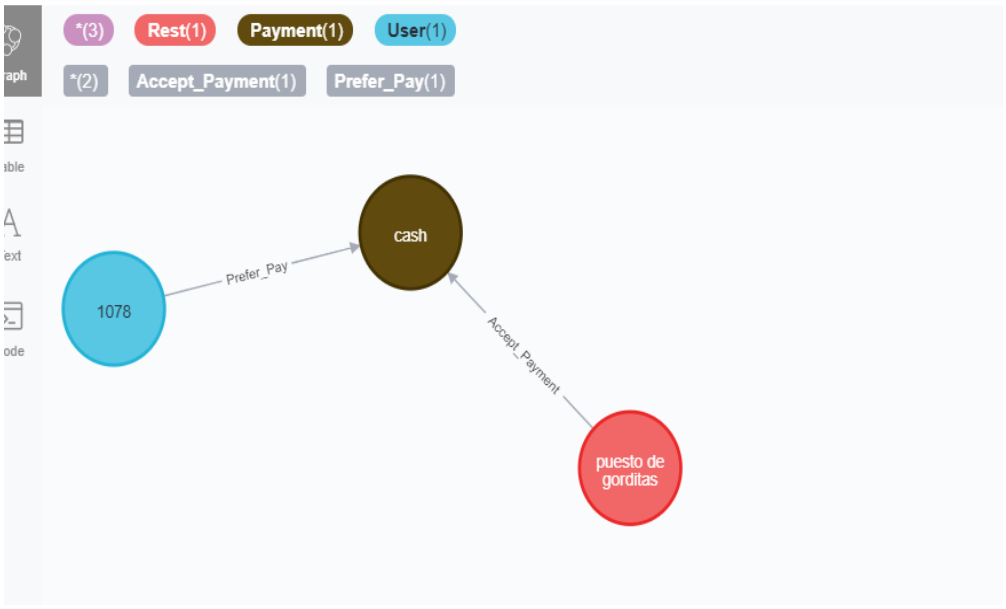
   d. Illustration of relationship logic:

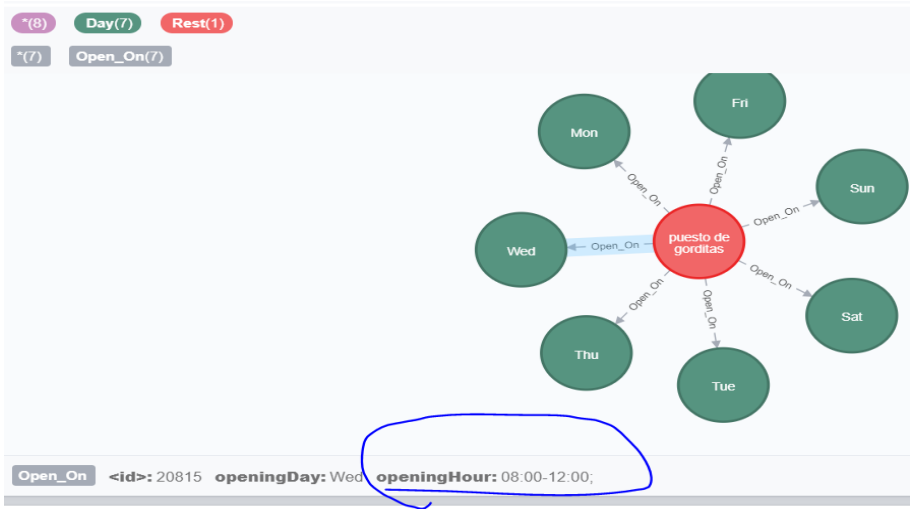

*(6)
User(1)
Rest(5)
*(5)
Rates(5)
restaurante marisco s...
restaurant la chalita
tortas locas hipocampo
1077
puesto de tacos
restaurant orizatlan
Rates
List
Rates
<id>: 16980 count: 1 rating_food: 2 rating_place: 2 rating_service: 2

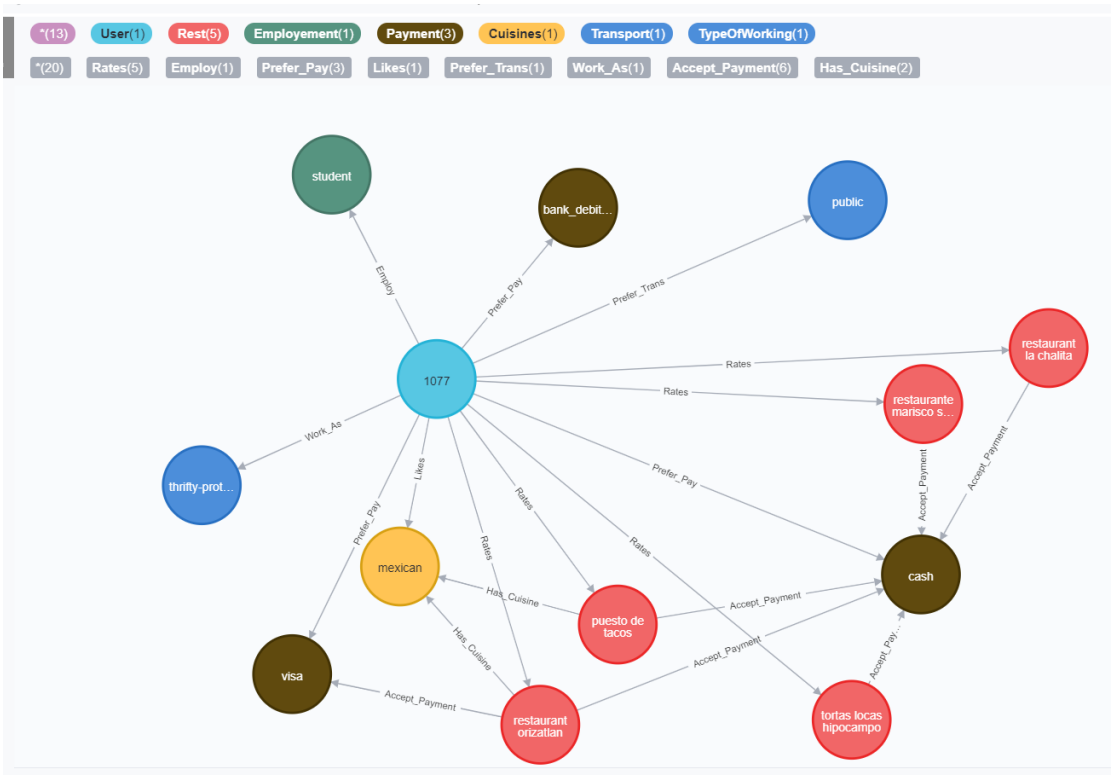

*(30)
User(1)
Cuisines(1)
Rest(28)
*(31)
Rates(2)
Likes(1)
Has_Cuisine(28)
tacos el guero
carreton de flautas y migadas
el pueblito
tacos abi
tacos de la estacion
cabana huasteca
el oceano dorado
tacos de barbacoa enfrente del tec
taqueria el amigo
vips
puesto de tacos
gorditas doa gloria
cenaduria el rincón de tlaquepaq...
mexican
tacos correcamin...
hamburgue...
1077
restaurant orizatlan
gorditas dona tota
el rincon de san francisco
gorditas dona tota
gordas de morales
log yin
el cotorreo
palomo tec
carnitas_m...
la estrella de dimas
la cochinita pibil restaurante yucate...
Rates
Has_Cuisine


*(4)
Rest(1)
City(1)
Country(1)
State(1)
*(3)
Locate_In(1)
Of_State(1)
Of_Country(1)
Mexico
Of_Country
tamaulipas
Of_State
victoria
Locate_In
puesto de gorditas

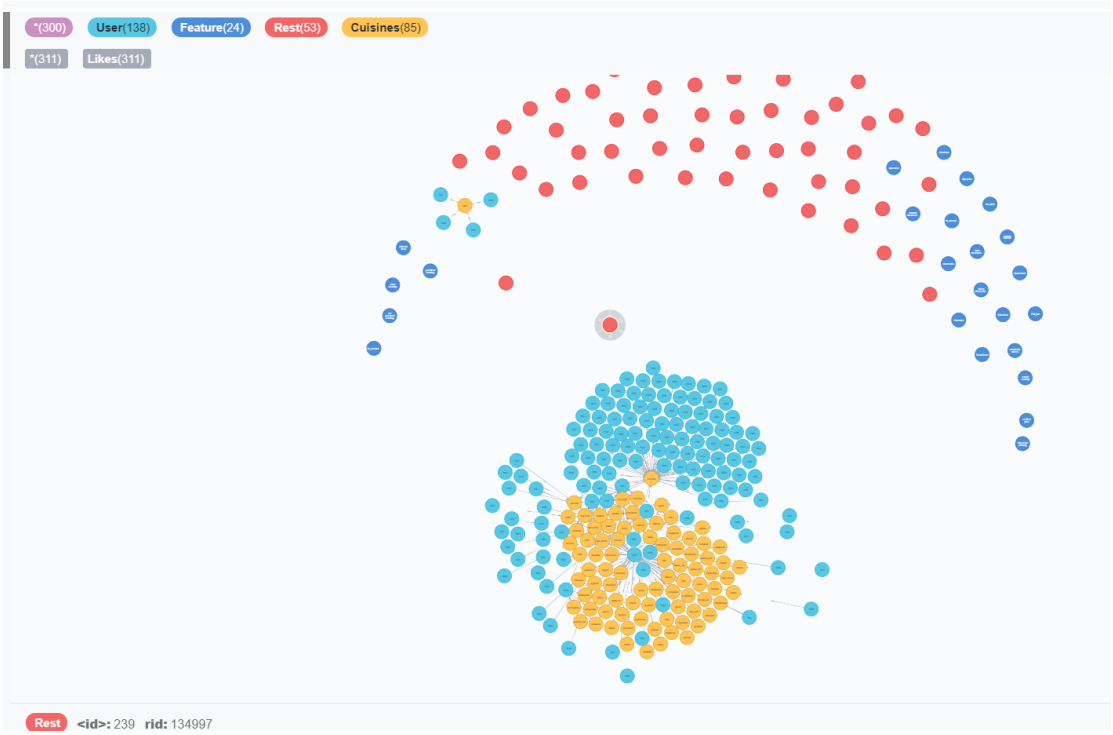
```
eo4j$ match (n:Rest {rid:"132560"}) -- (p:Payment) -- (u:User {uid:"1078"})
```







e. Overview of the whole database



Task 2

1. **Index choosen:**

   create index Rest for (r:Rest) on (r.rid,r.name);
   create index Rates for (r:Rates) on (r.rating_food,r.rating_service,r.rating_place);
   create index City for (c:City) on (c.city);
   Create compund indexes for restaruants' id and name, all three rating's property in Rates relationship respectively. Also an index on City nodes for city property. Reason is that these properties are heavily retrieved in the query. While querying rates in each places is one of the main purpuse of building the database, there are also lots of query about the places and cities.
   Therefore building indexes for them can store a copy of these properties in the memory which can contribute to improve the efficiency

2. **Query Results (Codes is in cypher file provided):**

(1) How many reviews does "Chilis Cuernavaca" have?

// match (r : Rest {name: "chilis cuernavaca"}) <-[r2]- () return sum(r2.count)

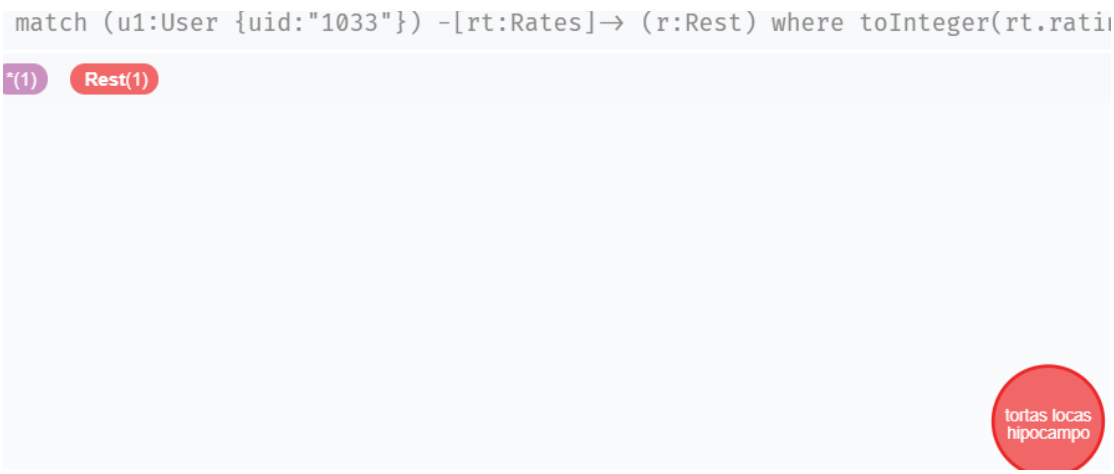
(2) Show all place, cuisines, and service ratings for restaurants in "Morelos" state

```
// match (c:City) -[:Of_State]-> (s:State {state:"morelos"})
with c,s
match (r:Rest) -[:Locate_In]-> (c)
with r,s
match (r) -[:Has_Cuisine]-> (cs:Cuisines)
with r, cs,s
match (r) <-[rt:Rates]- (u:User)
return s.state as state, r.rid as place_id,r.name as place_name, collect(distinct cs.cuisine) as cuisines, collect(rt.rating_service) as rating_service
```

(3) Can you recommend places that user 1003 has never been but user 1033 have been and gave ratings above 1?

//

```
match (u1:User {uid:"1033"}) -[rt:Rates]-> (r:Rest)
where toInteger(rt.rating_food[0]) > 1 and toInteger(rt.rating_place[0]) > 1 and toInteger(rt.rating_service[0]) > 1
with r
match (u2:User {uid:"1003"})
where not (u2) -[:Rates]-> (r)
return r
```



(4) List all restaurant names and locations that do not provide Mexican cuisines.

//

```
match (r:Rest)
where not (r) -[:Has_Cuisine]-> (:Cuisines {cuisine:"mexican"})
match (c:City)
with r,c
match (r) -- (c)
return {place_name:r.name, place_city:c.city, place_lat:r.loc_lat, place_long:r.loc_long}
```

{place_name:r.name, place_city:c.city, place_lat:r.loc_lat, place_long:r.loc_long}


```
{
  "place_lat": 23,
  "place_name": "pollo_frito_buenos_aires",
  "place_long": -99,
  "place_city": "victoria"
}
```

```
{
  "place_lat": 23,
  "place_name": "la perica hamburguesa",
  "place_long": -99,
  "place_city": "victoria"
}
```


ed streaming 102 records after 1 ms and completed after 6 ms.

(5) Count how many times each user provides ratings.

//
match (u:User) -[rt:Rates]-> (r:Rest)
return u.uid, sum(rt.count)
// or --- since there is no customers rates a same restaruant more than one times
match (u:User) -[rt:Rates]-> (r:Rest)
return u.uid, count(rt)

```
04j$ match (u:User) -[rt:Rates]→ (r:Rest) return u.uid, sum(rt.count)
```

	u.uid	sum(rt.count)
1	"1001"	9
2	"1002"	10
3	"1003"	13
4	"1004"	8
5	"1005"	9
6	"1006"	11
7	"1007"	9

(6) Display a list of pairs of restaurants having more than three features in common.

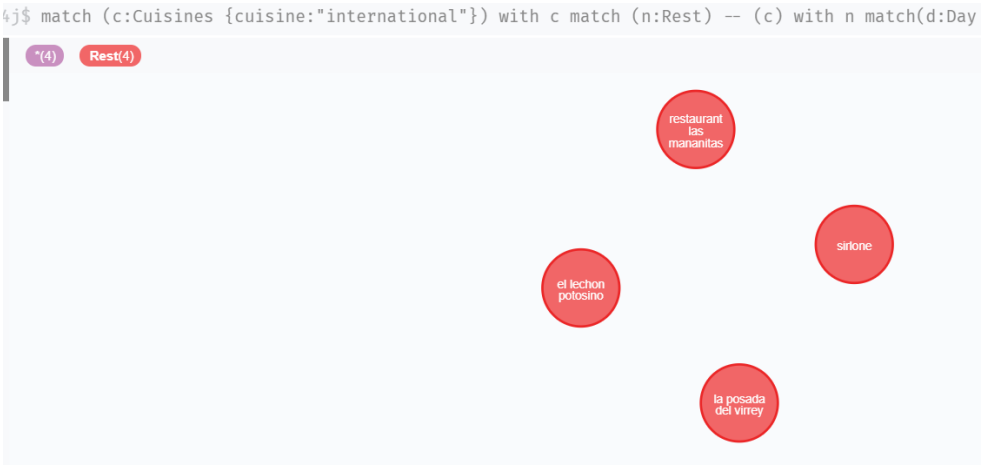
//

```
match (r1: Rest),(r2: Rest)
where size((r1) -[:Has_Feature]-> (:Feature) <-[:Has_Feature]- (r2)) >3
return{r1:r1.rid,r2:r2.rid}
```

(7) Display International restaurants that are open on Sunday.

//

```
match (c:Cuisines {cuisine:"international"})
with c
match (n:Rest) -- (c)
with n
match(d:Day {day:"Sun"}) <-[:Open_On]- (n)
return n
```

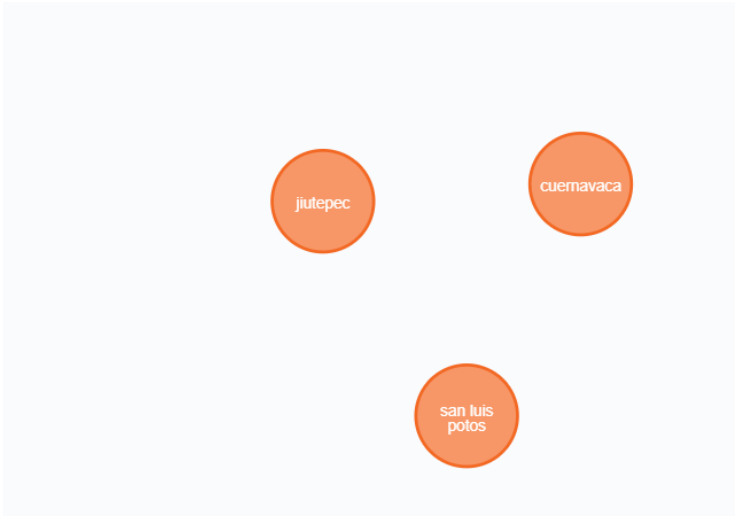


(8) What is the average food rating for restaurants in Victoria city?

```
//
match (c :City)
where c.city =~".*victoria.*"
with c
match (r :Rest) -- (c)
with r
match (r) -[rt:Rates]- (:User)
return avg(rt.rating_food[0])
```

(9) What are the top 3 most popular cities based on the total average service ratings

```
//
match (n:City) -- (r:Rest) -[rt:Rates]- (u:User)
with n, avg(rt.rating_service[0]) as rates
order by rates desc
limit 3
return n, rates
```



(10) For each place, rank other places that are close to each other by their locations. You will need to use the longitude and latitude to calculate the distance between places

//
match (n:Rest),(n2:Rest)
where n.rid<>n2.rid
with n.name as name1,n2, distance(point({longitude:n.loc_long,latitude:n.loc_lat}), point({ longitude: n2.loc_long, latitude: n2.loc_lat })) as dis
order by dis desc
return {place:name1, ranks: collect(n2.name) }

Task3

1. Create new restaruant

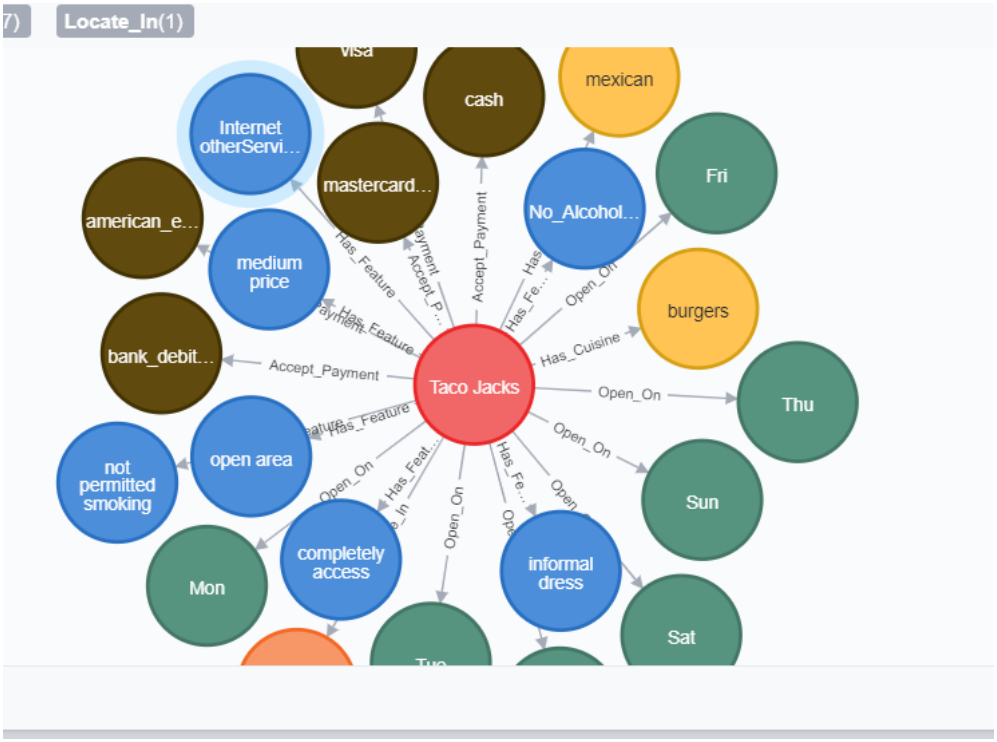
//
merge (n:Rest {rid:"70000"})
set n.name = "Taco Jacks",
n.loc_stree = "Carretera Central Sn",
n.parkingArrange = "none"

match (n:Rest {rid :"70000"}), (c:City {city:"slp"})
merge (n) -[:Locate_In]-> (c)

match (n:Rest {rid :"70000"}), (f1:Feature {feature:"No_Alcohol_Served"})
merge (n) -[:Has_Feature]-> (f1)

match (n:Rest {rid :"70000"}), (f1:Feature {feature:"not permitted smoking"})
merge (n) -[:Has_Feature]-> (f1)

```
match (n:Rest {rid :"70000"}), (f1:Feature {feature:"informal dress"})
merge (n) -[:Has_Feature]-> (f1)

match (n:Rest {rid :"70000"}), (f1:Feature {feature:"completely access"})
merge (n) -[:Has_Feature]-> (f1)

match (n:Rest {rid :"70000"}), (f1:Feature {feature:"medium price"})
merge (n) -[:Has_Feature]-> (f1)

match (n:Rest {rid :"70000"}),(f1:Feature {feature:"t franchise"})
merge (n) -[:Has_Feature]-> (f1)

match (n:Rest {rid :"70000"}), (f1:Feature {feature:"open area"})
merge (n) -[:Has_Feature]-> (f1)

match (n:Rest {rid :"70000"}), (f1:Feature {feature:"Internet otherService"})
merge (n) -[:Has_Feature]-> (f1)

match (n:Rest {rid :"70000"}),(f1:Payment)
merge (n) -[:Accept_Payment]-> (f1)

match (n:Rest {rid :"70000"}), (f1:Cuisines {cuisine:"mexican"})
merge (n) -[:Has_Cuisine]-> (f1)

match (n:Rest {rid :"70000"}), (f1:Cuisines {cuisine:"burgers"})
merge (n) -[:Has_Cuisine]-> (f1)

match (n:Rest {rid :"70000"}), (f1:Day)
merge (n) -[r:Open_On {openingHours:"9:00-20:00"}]- (f1);

match (:Day {day:"Sun"}) -[r1]- (n:Rest {rid :"70000"}) -[r]- (f1:Day {day:"Sat"})
set r.openingHours = "12:00-18:00",
    r1.openingHours = "12:00-18:00"
```



2. **They have also realised that the user with user_id 1108, no longer prefers Fast_Food and also prefers to pay using debit_cards instead of cash. You are required to update user 1108's favorite cuisines and favorite payment methods**

//

```
match (n:User {uid:"1108"}) -[r]- (u: Cuisines {cuisine:"fast_food"})
delete r

match (n:User {uid:"1108"}) -[r]- (u: Payment {payment:"cash"})
delete r;
match (n:User {uid:"1108"}) -- (u: Payment {payment:"debit cards"})
merge (n) -[:Prefer_Pay]- (u)
```

3. **The management has realised that the user with user_id 1063 was an error. Therefore delete the user 1063 from the database**

//

match (n:User{uid:"1063"})

detach delete n