**Data Warehousing and SQL : Assignment 1**

Siddhant Sapte

ALY 6030

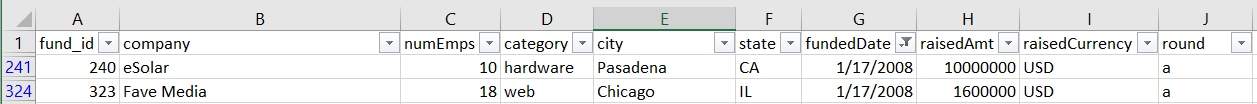
CRN: 21008

**Problem 1: Normalization**

In the first problem we have techcrunch.csv file where we have reports on the business of technology, startups, venture capital funding, and Silicon Valley from the website TechCrunch website.

1. What is a good choice for a primary key here? In contrast, give an example of an attribute (or composite) that would ***not*** be a valid primary key.

Primary key is the key in the database which cannot be null and should be unique. It represents each row of the database and in our case fund\_id is the only column which satisfies these constraints. No other column can be primary key, even the fundedDate cannot be primary key as the values are duplicating, for e.g.,



For the date, 1/17/2008 there are two entries and our primary key cannot contain duplicate values.

1. Does the table satisfy 1NF? Why or why not?

Normalization is a database technique which organizes tables in the database in such a manner that it reduces redundancy and dependency of data. It basically divides larger table into smaller table and links them using relationships. These table satisfies 1NF as each cell contains a single value and each record is unique.

1. Does the table satisfy 2NF? Why or why not?

The table does satisfy 2NF and we can reduce the main table into two more tables, one containing the company information and the other table containing the fund information. The company information contains company name, number of employees, category, location, city , state, company\_id. The fund table will contain fund\_id, funded date, raised amount, raised currency, round and company\_id which will be the foreign key referencing the company table. Company\_id and fund\_id is the primary key in both the tables. Here there is no partial dependency between the tables.

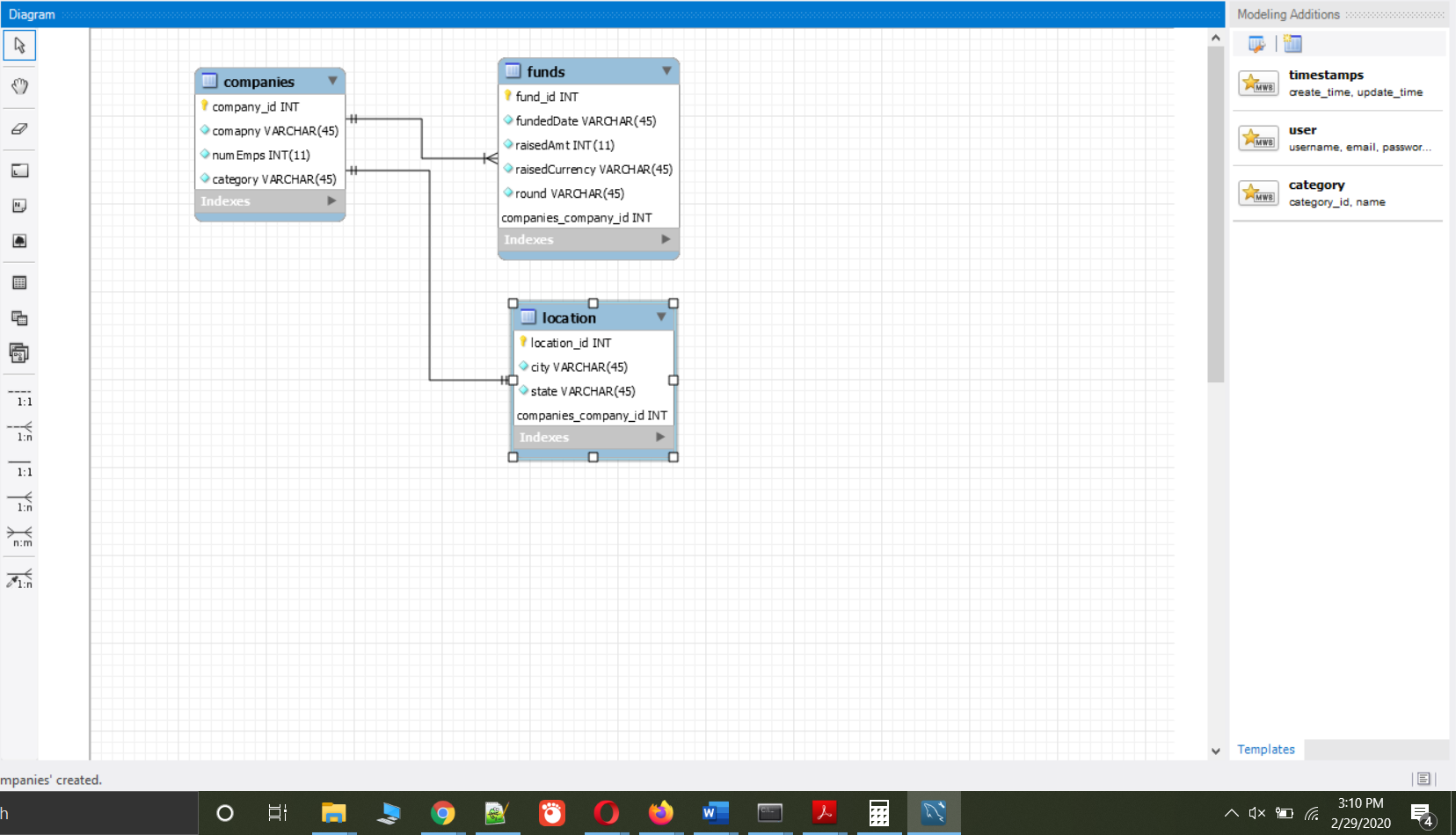
1. Does the table satisfy 3NF? Why or why not?

The table satisfies 3NF as we can further reduce the tables we created in 2NF. Three tables can be made with company information, fund information and location information. In the companys table we will have company\_id, company, num of employees and category. Here company\_id will be primary key. Fund information will have fund\_id, funded date, raised amount, raised currency, round and company\_id, here fund\_id is a primary key and company\_id will be the foreign key which will have a reference of company table. The companys table and funds table will have one to many relationship.

Similarly the locations table will have location\_id, city, state and company\_id where location\_id will be the primary key and company\_id will be the foreign key referencing the companys table. The companys table and locations table will have one to one relationship according to our data.

1. Sketch a proposed Entity-Relationship diagram that would bring this dataset into 3NF. If you answered “yes” to (4), for example, your ERD would just be the raw data table with no changes. If however your ERD requires multiple tables to be in 3NF, you should draw all relationships between them and indicate their type (one-to-one, one-to-many, etc.)

Here is the ER diagram for the 3NF.



**Problem 2:**

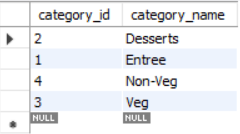
1. In the first problem we need to insert two recipes.

We will first select the database,

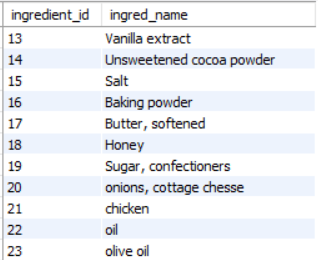
**use recipes;**

Then run the queries,

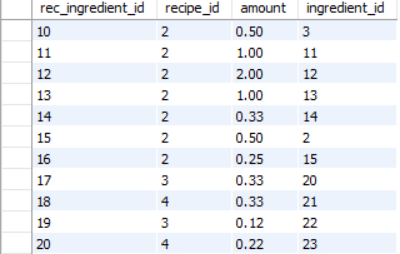
**select \* from categories;  
insert into categories (category\_id,category\_name) values ('3', 'Veg'),('4','Non-Veg');**



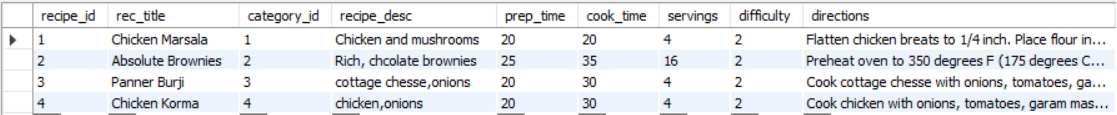
**select \* from ingredients order by ingredient\_id;  
insert into ingredients (ingredient\_id,ingred\_name) values ('20', 'onions, cottage chesse'),('21','chicken'),('22','oil'),('23','olive oil');**



**select \* from rec\_ingredients;  
insert into rec\_ingredients (rec\_ingredient\_id,recipe\_id,amount,ingredient\_id) values ('17', '3','0.33','20'),('18','4','0.33','21'),('19','3','0.12','22'),('20','4','0.22','23');**

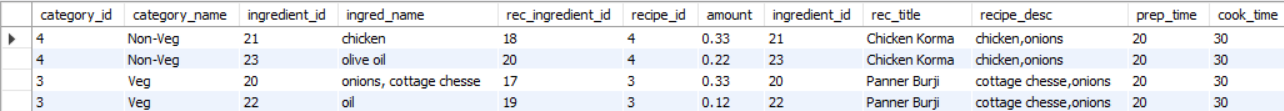


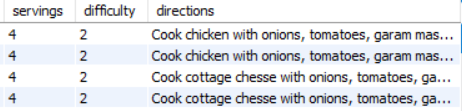
**select \* from recipe\_main;  
insert into recipe\_main (recipe\_id,rec\_title,category\_id,recipe\_desc,prep\_time,cook\_time,servings,difficulty,directions) values ('3', 'Panner Burji','3','cottage chesse,onions','20','30','4','2','Cook cottage chesse with onions, tomatoes, garam masala, red masala and fried onions and garnish with fried onions and corriander'),('4', 'Chicken Korma','4','chicken,onions','20','30','4','2','Cook chicken with onions, tomatoes, garam masala, red masala and fried onions and garnish with fried onions and corriander');**



1. Write only one SQL query that returns all information on only the two new recipes you inserted from all the tables you created in step 1 above.

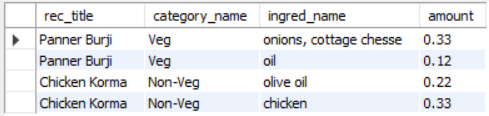
**select c.category\_id, c.category\_name,i.ingredient\_id,i.ingred\_name, r.rec\_ingredient\_id, r.recipe\_id, r.amount, r.ingredient\_id, m.rec\_title,m.recipe\_desc,m.prep\_time,m.cook\_time,m.servings,m.difficulty,m.directions from(((recipe\_main as m   
join categories as c on c.category\_id = m.category\_id)  
join rec\_ingredients as r on r.recipe\_id = m.recipe\_id)  
join ingredients as i on i.ingredient\_id = r.ingredient\_id)  
where c.category\_id = '3' or c.category\_id = '4';**





1. Write a SELECT query that identifies the recipe name, category name, and ingredient name, and ingredient amount. No other variables should be included

**select m.rec\_title, c.category\_name,i.ingred\_name, r.amount from  
(((recipe\_main as m   
join categories as c on c.category\_id = m.category\_id)  
join rec\_ingredients as r on r.recipe\_id = m.recipe\_id)  
join ingredients as i on i.ingredient\_id = r.ingredient\_id)  
where c.category\_id = '3' or c.category\_id = '4'  
order by c.category\_name desc,i.ingred\_name desc;**



**Reference:**

Normalization of Database. (n.d.). Retrieved from <https://www.studytonight.com/dbms/database-normalization.php>