DATA WAREHOUSING AND SQL  
  
  
ALY6030, SPRING 2020  
MODULE 1 PROJECT ASSIGNMENT

WEEK 1: THE RELATIONAL MODEL AND BASIC SQL

SUBMITTED BY: SHIVANI ADSAR

NUID: 001399374

SUBMITTED TO: PROF. DAYA RUDRAMOORTHI

DATE: 05/31/2020

**Introduction**

The assignment aims at providing practical experience of working on normalizations and SQL queries by working on a database. Moreover, we have gained the importance of normalization by working on the ‘techcrunch’ dataset and referred to the ‘Recipes’ dataset for working on SQL queries. The practical experience of working on SQL queries has helped in understanding the concepts of JOINS, while, 1NF, 2NF and 3NF normalization forms provided knowledge of normalizations in SQL and databases.

**Analysis**

**Problem 1: Normalization**

* We have used the given file, named, ‘techcrunch.csv’ for understanding the possible normalizations that can be performed on the database.
* Normalization helps in reducing data redundancy and maintaining dtaa consistency for effective data retrieval.
* The dataset contains the funding records of companies as reported by TechCrunch, which is a website that reports on the business of technology, startups and venture capital funding.

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.**

After analyzing the ‘techcrunch’ dataset, we can note that there are 10 columns in the dataset and 523 records of companies. We can see that, the ‘fund\_id’ has 523 unique records that can be considered a better selection for a primary key. It is important for primary keys to have unique values. Also, the field, ‘fund\_id’ does not have NULL values.

The field, ‘fund\_id’ is a unique key which helps in identifying the records in the database.

In order for the data retrival process to be effective, the primary key should be unique. Moreover, we can note that there are some columns in the dataset that have redundant values. For example, we can observe that the fields, “company”, “numEmps”, “category”,”city”,”state”, “round” and “raisedCurrency” have duplicate and redundant values. These duplicate values make it difficult for retrieving every record from the database. Hence, these attributes cannot be used as primary keys. Therefore, we need to normalize the data into different tables for better data retrieval.

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

The dataset does not contain more than one value for a single field. It is noted that all values in the dataset are dependent on the ‘fund\_id’ which is a primary key. Every attribute has individual values in separate fields and not combined values. Therefore, there is no repetition in the cells of the database.

For instance, the “city” and “state” are segregated fields and not combined into a single cell. Also, the data type of values matches their attributes, like, the “fundedDate” has Date values, “raisedAmt” has numerical values and “raisedCurrency” has character values. Hence there is a uniformity observed amongst the values stored for every attribute and their associated data type. Hence, the table does satisfy the 1NF normalization form.

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

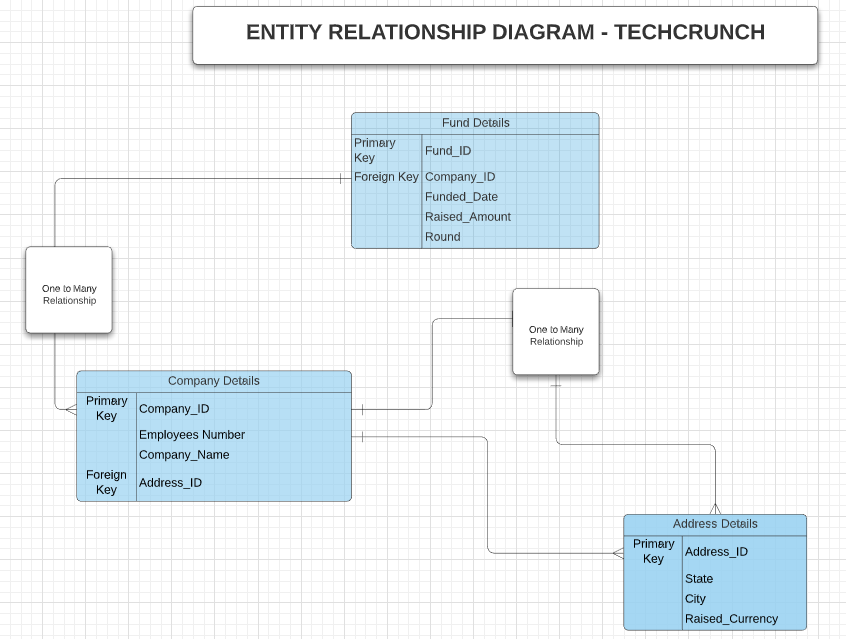
In order for a table to satisfy the 2NF criteria, it has to be 1NF and the table should not have partial dependencies, which means that none of the attributes in the table should have partial dependencies on each other. As we can see in the ‘techcrunch’ dataset, the number of employees are important for the companies in raising funds. Therefore, there exists a partial dependency amongst columns, “numEmps” and “raisedAmt”. Although, the table is in 1NF form, but there exists a partial dependency. Hence, the table does not satisfy the 2NF normalization form. In order to eliminate this dependency, we can create another table with “fund\_id” or introduce a unique “Company\_Identifier” as the primary key. This will help in reducing the partial dependency.

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

A table is said to be in 3NF, if it is in 2NF form and there are no transitive dependencies. As we can interpret, our table does not satisfy the 2NF form and there exist some transitive dependencies. Hence, the table does not satisfy the 3NF criteria. The company records are not related to the “fund\_id”, because the records have funds as per the company name. We can see that, “fund\_id” is a unique identifier, but does not have relation with the raised amount for every company.

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.)**

The Entity Relationship Diagram shows the relationships between various entities and their respective attributes. The diagram shows my proposed entity relationship diagram for the dataset, ‘techcrunch’. As we can see, there are three tables, namely, Fund Details, Company Details and Address Details. The Fund Details table has a one to many relationship with Company Details table. Whereas, there exists a one to many relationship between Company Details and Address Details tables. Also, we the “Fund\_ID” would be considered as the primary key and “Company\_ID”, as a foreign key to connect the tables.



*Fig.1: ER Diagram for ‘Techcrunch’ Dataset*

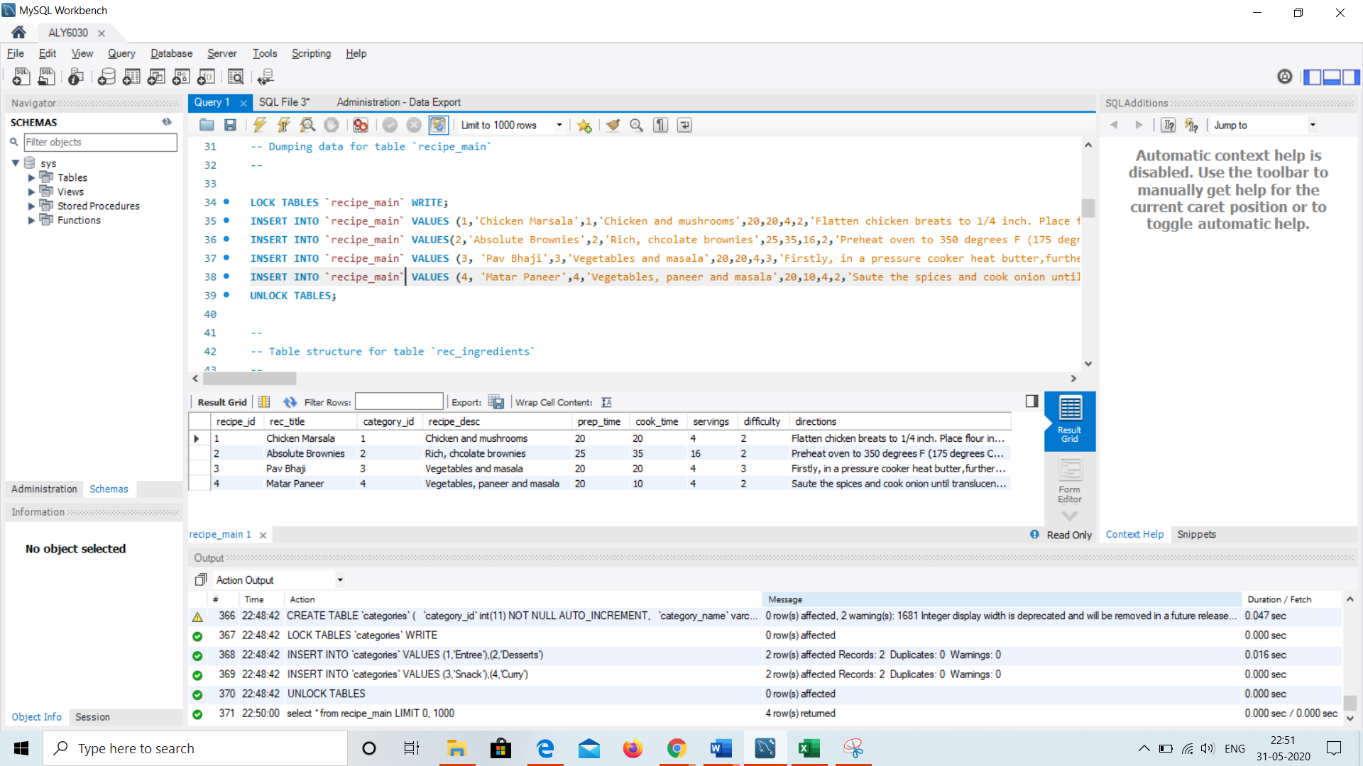
**Problem 2: SQL Queries**

The Structured Query Language is a very effective language used for communicating with the database for performing manipulations on the data for faster data retrieval.

1. **Once you run the sql code you’ll notice that each table has data populated for two recipes, Chicken Marsala and Absolute Brownies. Use the INSERT INTO statement to insert new information about two (2) completely new recipes of your choosing into the database.**

The two recipes for ‘Pav Bhaji’ and ‘Matar Paneer’ have been inserted into the database, ‘recipes’ using the ‘INSERT INTO’ command.

I have inserted the data for the 2 recipes in the tables, namely, ‘recipe\_main’, ‘rec\_ingredients’, ’ingredients’ and ‘categories’.

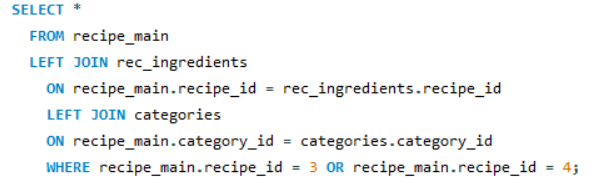


*Fig.2: Insertion of two recipes in the table*

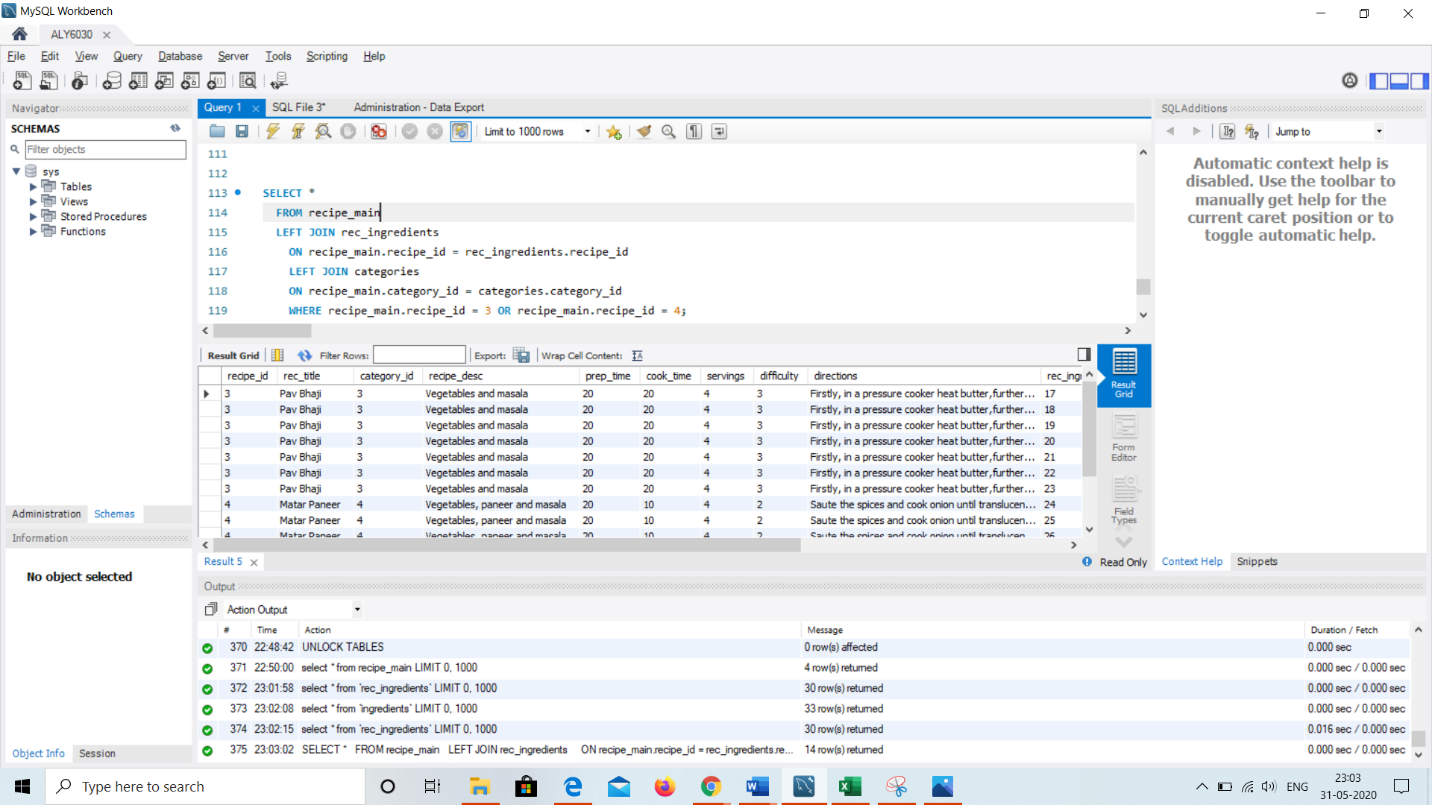
1. **Write only one SQL query that returns all information on only the two new recipes you inserted from all the tables you created in step 1 above. Your query should show all relevant information from all four of the tables from step 1. Don’t worry that the output table may duplicate rows, for example the recipe\_main table will duplicate rows for each of the ingredients you enter which is ok. The idea is to get you comfortable with joins.**

We have used the LEFT JOIN on ‘rec\_ingredients’ and ‘categories’ tables for joining the two tables as per the ‘recipe\_id’ and ‘category\_id’ respectively. Further we have used the WHERE clause for displaying the recently added recipes.

The below output shows all the details of the two newly added recipes.



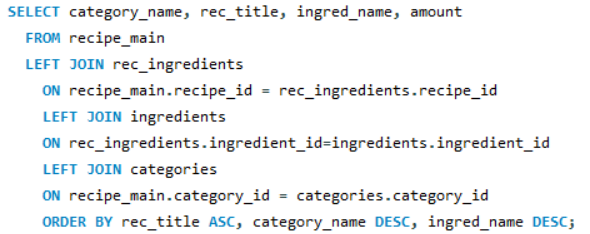
*Fig.3: SQL Code*



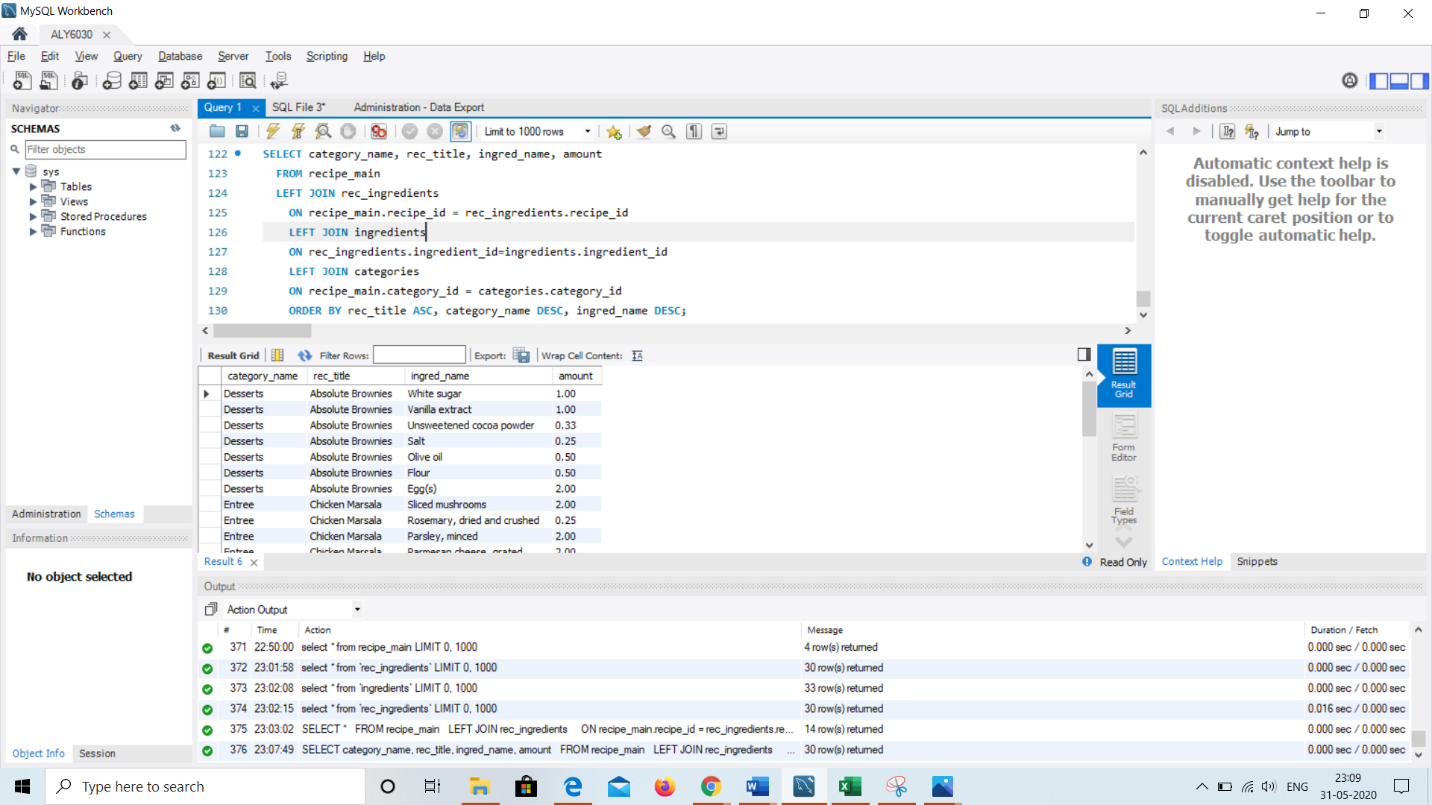
*Fig.4: Using LEFT JOIN for displaying all the records for the newly added recipes*

1. **Write a SELECT query that identifies the recipe name, category name, and ingredient name, and ingredient amount. No other variables should be included.Your output should be sorted first by descending category name, then by ascending recipe name, followed by descending ingredient name. Recall that string variables are sorted alphabetically (ascending only) when you check your query results.**

As we can see, we have used LEFT JOIN on ‘rec\_ingredients’, ‘categories’ and ‘ingredients’ tables for joining the tables as per the ‘recipe\_id’, ‘ingredient\_id’ and ‘category\_id’. Further, we have used the ORDER BY clause for sorting the recipe\_name, category\_name and ‘ingredient name’ in the ascending and descending order.



*Fig.5: SQL Code*



*Fig.6: Using LEFT JOIN to display the sorted results*

**Conclusion**

We have learnt the importance of normalization in optimization of data for faster data retrieval. Moreover, the practical experience of working on SQL queries has helped in understanding the communication with database for data retrieval using various DSQL commands.

**References**

[1] Comeau A. (2015). MySQL Explained: Your Step-by-Step Guide

[2] Köhler, H., & Link, S. (2018). SQL schema design: Foundations, normal forms, and normalization. Information Systems, 76(C), 88-113.