COURSE : DATA ANALYTICS I

INSTRUCTOR: DR. KAMALAKAR KARLAPALEM

Project Phase I : Mapping Excel Sheet data to a Data Warehouse and practising OLAP queries

---

Names with Roll no. :

- Sayantan Jana (20171185)
- Aditya Khandelwal (20171117)

Dataset used :

- URL of data set :

  www.kaggle.com/datafiniti/pizza-restaurants-and-the-pizza-they-sell

- Name of data set : Pizza Restaurants and the pizza they sell

Number of pages in the report :

9

The rationale behind picking the data set :

This is a list of over 3,500 pizzas from multiple restaurants provided by Datafiniti's Business Database. The dataset includes the category, name, address, city, state, menu information, price range, and more for each pizza restaurant. It offers easy availability and accessibility being a kind of shortened data set that we took from kaggle while the full data set is available on Datafiniti's Business Database.

This data set was one of the few which offers absolute chance for aggregating, summarizing or manipulating. One could easily find out in the data set how certain attributes refers to multiple other attributes in the original data table. As for example the id for the rows helps us figure out the address, latitude, longitude of the location coordinates , country , province, while the postal code always seemed to hint at the region currency, these gave enough chances to create dimension tables out of this, bringing in uniqueness and granularity to the data warehouse. The uniqueness that we target to achieve through this also bows down to the consistency and relevance throughout the data set.

Regarding the usefulness, the data set carries the potential to discover the most and least expensive pizza places in the country, the median price of a large plain pizza across US, also most crowded locations serving pizzas and also helps find out number of restaurants serving pizza per capita.

Finally we could possibly achieve a database offering numeric data, the quantities (prices, latitudes, longitudes, postal codes) and the dimensions clubbing columns of data, so we could perform business intelligence queries , one of the significant purposes of OLAP.

Also we had taken sufficient care to clean the data , i.e., at points where entries were null, we have simply tried to omit those rows, incidentally this kaggle dataset provides empty strings in place of null where it has no entry.

The Data Warehouse Schema (construction):

The original data set taken from kaggle contained column names as 'keys' and names with dot operator, which threw errors and hence respectively they were replaced with shopidentifier and underscores.

Originally the data set contained 21 columns , but as we had earlier described it contained redundancies which we aim to reduce creating dimension tables and possibly dimension tables out of the existing ones.

The steps taken for mapping to the warehouse schema in mysql goes as :

- ➔ We started by creating database pizza_db and creating the original datatable in it on which we are supposed to act.

```
mysql> use pizza_db;
Database changed
mysql> create table pizzastations(id VARCHAR(1024),address VARCHAR(1024),categories VARCHAR(1024),city VARCHAR(1024),country VARCHAR(1024),shopidentifier
 VARCHAR(1024),latitude VARCHAR(1024),longitude VARCHAR(1024),menuPageURL VARCHAR(1024),menus_amountMax VARCHAR(1024),menus_amountMin VARCHAR(1024),menus
_currency VARCHAR(1024),menus_dateSeen VARCHAR(1024),menus_description VARCHAR(1024),menus_name VARCHAR(1024),name VARCHAR(1024),postalCode VARCHAR(1024)
,priceRangeCurrency VARCHAR(1024),priceRangeMin VARCHAR(1024),priceRangeMax VARCHAR(1024),province VARCHAR(1024));
Query OK, 0 rows affected (0.04 sec)

mysql> describe pizzastations;
+-------------------+--------------+------+-----+---------+-------+
| Field             | Type         | Null | Key | Default | Extra |
+-------------------+--------------+------+-----+---------+-------+
| id                | varchar(1024) | YES  |     | NULL    |       |
| address           | varchar(1024) | YES  |     | NULL    |       |
| categories        | varchar(1024) | YES  |     | NULL    |       |
| city              | varchar(1024) | YES  |     | NULL    |       |
| country           | varchar(1024) | YES  |     | NULL    |       |
| shopidentifier    | varchar(1024) | YES  |     | NULL    |       |
| latitude          | varchar(1024) | YES  |     | NULL    |       |
| longitude         | varchar(1024) | YES  |     | NULL    |       |
| menuPageURL       | varchar(1024) | YES  |     | NULL    |       |
| menus_amountMax   | varchar(1024) | YES  |     | NULL    |       |
| menus_amountMin   | varchar(1024) | YES  |     | NULL    |       |
| menus_currency    | varchar(1024) | YES  |     | NULL    |       |
| menus_dateSeen    | varchar(1024) | YES  |     | NULL    |       |
| menus_description | varchar(1024) | YES  |     | NULL    |       |
| menus_name        | varchar(1024) | YES  |     | NULL    |       |
| name              | varchar(1024) | YES  |     | NULL    |       |
| postalCode        | varchar(1024) | YES  |     | NULL    |       |
| priceRangeCurrency | varchar(1024) | YES  |     | NULL    |       |
| priceRangeMin     | varchar(1024) | YES  |     | NULL    |       |
| priceRangeMax     | varchar(1024) | YES  |     | NULL    |       |
| province          | varchar(1024) | YES  |     | NULL    |       |
+-------------------+--------------+------+-----+---------+-------+
21 rows in set (0.00 sec)
```

➔ We load the data from csv file into the data table this way next : LOAD data infile '/var/lib/mysql-files/pizzashops.csv' into table pizzastations columns terminated by ',' enclosed by '\"' lines terminated by '\n' IGNORE 1 ROWS;

➔ We try to clean the data at this step removing all the rows where menuPageURL or priceRangeCurrency is null. Query : DELETE FROM pizzastations where menuPageURL = null; Query : DELETE FROM pizzastations where priceRangeCurrency = null;

➔ Next we create a dimension table out of postal code and currency in the region as region_info.

```
mysql> CREATE TABLE region_info SELECT DISTINCT postalCode,priceRangeCurrency from pizzastations;
Query OK, 922 rows affected (0.07 sec)
Records: 922  Duplicates: 0  Warnings: 0

mysql> describe region_info;
+--------------------+---------------+------+-----+---------+-------+
| Field              | Type          | Null | Key | Default | Extra |
+--------------------+---------------+------+-----+---------+-------+
| postalCode         | varchar(1024) | YES  |     | NULL    |       |
| priceRangeCurrency | varchar(1024) | YES  |     | NULL    |       |
+--------------------+---------------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql> ALTER TABLE pizzastations DROP priceRangeCurrency;
Query OK, 0 rows affected (0.79 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

➔ Next we create a dimension table out of id,latitude,longitude,city,categories,address,country,shopidentifier,province into a shopaddress table.

```
mysql> CREATE TABLE shopaddress SELECT DISTINCT id,latitude,longitude,city,categories,address,country,shopidentifier,province from pizzastations;
Query OK, 989 rows affected (0.14 sec)
Records: 989  Duplicates: 0  Warnings: 0

mysql> describe shopaddress;
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| id            | varchar(1024) | YES  |     | NULL    |       |
| latitude      | varchar(1024) | YES  |     | NULL    |       |
| longitude     | varchar(1024) | YES  |     | NULL    |       |
| city          | varchar(1024) | YES  |     | NULL    |       |
| categories    | varchar(1024) | YES  |     | NULL    |       |
| address       | varchar(1024) | YES  |     | NULL    |       |
| country       | varchar(1024) | YES  |     | NULL    |       |
| shopidentifier | varchar(1024) | YES  |     | NULL    |       |
| province      | varchar(1024) | YES  |     | NULL    |       |
+---------------+--------------+------+-----+---------+-------+
9 rows in set (0.00 sec)

mysql> ALTER TABLE pizzastations DROP latitude, DROP longitude;
Query OK, 0 rows affected (0.95 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE pizzastations DROP city, DROP categories,DROP address,DROP country,DROP shopidentifier, DROP province;
Query OK, 0 rows affected (0.57 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

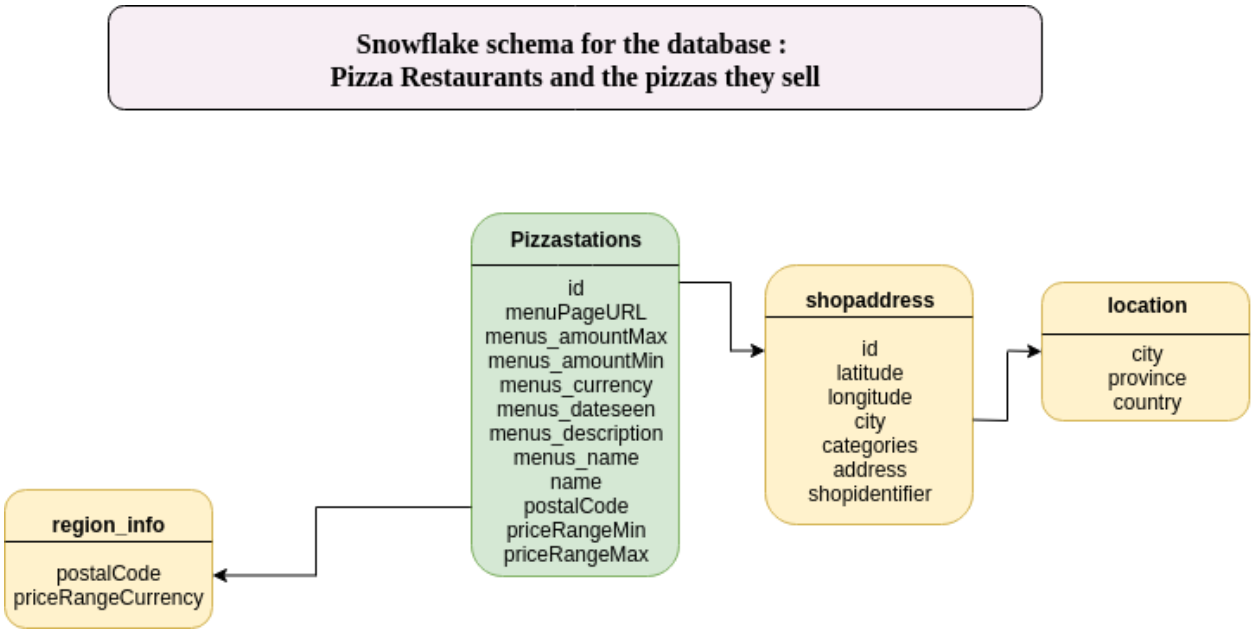
➔ We can perform citywise aggregation if we have a dimension table denoting location that gives us the country, province for a shop in a particular country. We take this dimension table from already existing fact table.

```
mysql> CREATE TABLE location select id,city,province,country from shopaddress;
Query OK, 989 rows affected (0.07 sec)
Records: 989  Duplicates: 0  Warnings: 0

mysql> describe location;
+----------+---------------+------+-----+---------+-------+
| Field    | Type          | Null | Key | Default | Extra |
+----------+---------------+------+-----+---------+-------+
| id       | varchar(1024) | YES  |     | NULL    |       |
| city     | varchar(1024) | YES  |     | NULL    |       |
| province | varchar(1024) | YES  |     | NULL    |       |
| country  | varchar(1024) | YES  |     | NULL    |       |
+----------+---------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> ALTER TABLE shopaddress DROP province,country;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that c
 'country' at line 1
mysql> ALTER TABLE shopaddress DROP province,DROP country;
Query OK, 0 rows affected (0.23 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> describe shopaddress;
+----------------+---------------+------+-----+---------+-------+
| Field          | Type          | Null | Key | Default | Extra |
+----------------+---------------+------+-----+---------+-------+
| id             | varchar(1024) | YES  |     | NULL    |       |
| latitude       | varchar(1024) | YES  |     | NULL    |       |
| longitude      | varchar(1024) | YES  |     | NULL    |       |
| city           | varchar(1024) | YES  |     | NULL    |       |
| categories     | varchar(1024) | YES  |     | NULL    |       |
| address        | varchar(1024) | YES  |     | NULL    |       |
| shopidentifier | varchar(1024) | YES  |     | NULL    |       |
+----------------+---------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

Code logs :

```
sudo mysql -uroot -p

1. create database pizza_db;
2. use pizza_db;
3. create table pizzastations(id VARCHAR(1024),address VARCHAR(1024),categories VARCHAR(1024),city
VARCHAR(1024),country VARCHAR(1024),shopidentifier VARCHAR(1024),latitude VARCHAR(1024),longitude
VARCHAR(1024),menuPageURL VARCHAR(1024),menus_amountMax VARCHAR(1024),menus_amountMin VARCHAR(1024),menus_currency
VARCHAR(1024),menus_dateSeen VARCHAR(1024),menus_description VARCHAR(1024),menus_name VARCHAR(1024),name
VARCHAR(1024),postalCode VARCHAR(1024),priceRangeCurrency VARCHAR(1024),priceRangeMin VARCHAR(1024),priceRangeMax
VARCHAR(1024),province VARCHAR(1024));
4. LOAD data infile '/var/lib/mysql-files/pizzashops.csv' into table pizzastations columns terminated by ','
enclosed by '\"' lines terminated by '\n' IGNORE 1 ROWS;
5. CREATE TABLE region_info SELECT DISTINCT postalCode,priceRangeCurrency from pizzastations;
6. describe region_info;
7. ALTER TABLE pizzastations DROP priceRangeCurrency;
8. CREATE TABLE shopaddress SELECT DISTINCT
id,latitude,longitude,city,categories,address,country,shopidentifier,province from pizzastations;
9. describe shopaddress;
10. ALTER TABLE pizzastations DROP latitude, DROP longitude;
11. ALTER TABLE pizzastations DROP city, DROP categories,DROP address,DROP country,DROP shopidentifier, DROP
province;
12. describe pizzastations;
13. CREATE TABLE location select id,city,province,country from shopaddress;
14. ALTER TABLE shopaddress DROP province,DROP country;
15. describe shopaddress;
16. describe pizzastations;
17. describe location;
18. describe region_info;
```

The Data Warehouse Pictorial Representation :



We use a snowflake schema for mapping the data warehouse where the yellow tables are the dimension tables but also they refer to some other dimension table (here shopaddress refers to location) . As we notice the dimensions normalized to some other related table we call it snowflake schema.

Database system used for loading data set :

The database preferred by us for mapping is mysql. The prime reason being our familiarity with the database system. Moreover, mysql is open-source, effective and easy to manage. The necessities of the database system involved is creation of a database, using it to create tables, loading the data from csv file to the tables, and then the segregation of table columns in order to eliminate redundancies. All of these are easily available on mysql using simple to use commands like CREATE, LOAD, ALTER, DROP, SELECT, describe , etc. Though we had not performed queries based on OLAP , we found out from online resources how OLAP queries can be easily implemented on mysql.

OLAP queries :

1. **GET all the unique shop names with the count of pizza types they sell and the sum of their menus_amountMax :**

   This is a simple query just made on the main fact table that helps analyse unique shop names with the count of pizza types and sums of the maximums of pizza menu amounts, signifying the total bill for ordering most expensive items from each shop, for a lavish party maybe!

```
mysql> select name,count(name),sum(menus_amountMax) from pizzastations group by name with ROLLUP;
```

   The output is not shown as it is one of 984 rows !

2. **Grouping by postalCodes , GET the count of pizza options available whose maximum shop amount is greater than 15 and minimum amount is less than 10 :**

   This query asks to link the main fact table with the dimension table region_info and perform a query grouping by the postal codes, carrying the significance of sorting out options based on pizza prices in a region carrying same postal code.

```
mysql> select r.postalCode, count(*) from pizzastations p,region_info r where r.postalCode = p.postalCode and p.menus_amountMax>15 and p.menus_amountMin<
10 group by r.postalCode with ROLLUP;
+------------+----------+
| postalCode | count(*) |
+------------+----------+
| 06037-1355 |        3 |
| 10018      |        5 |
| 15212      |       20 |
| 21157      |        1 |
| 27609      |        1 |
| 28792      |        2 |
| 32714      |        2 |
| 43050      |        2 |
| 43105      |        2 |
| 45895      |        2 |
| 4901       |        1 |
| 61036      |        1 |
| 61615      |        1 |
| 63090      |        1 |
| 6379       |        1 |
| 6470       |        1 |
| 67002      |        1 |
| 76063      |        2 |
| 89052      |        2 |
| 90292      |        2 |
| 94501      |       11 |
| NULL       |       64 |
+------------+----------+
22 rows in set (0.02 sec)
```

3. Grouping by province, get the province names, entries per province only where the pizza shops charge a maximum amount greater than 15 and longitudes lie less than -120 :

```
mysql> select l.province, count(*) as total
    -> from pizzastations p, shopaddress sa, location l
    -> where p.menus_amountMax>15 and p.id=sa.id and sa.longitude<-120 and sa.id=l.id
    -> group by l.province
    -> with rollup;
+------------------+-------+
| province         | total |
+------------------+-------+
| CA               |    23 |
| Hono             |     3 |
| Larimers Corner  |     2 |
| Mormon Island    |     2 |
| OR               |     3 |
| Paradise Park    |     1 |
| Strawberry Point |     8 |
| WA               |     6 |
| Wankers Corners  |     1 |
| NULL             |    49 |
+------------------+-------+
10 rows in set (0.02 sec)
```

4. Grouping by province, get the province names, entries per province only where the pizza shops charge a maximum amount less than 15 and longitudes lie greater than -120 and less than -118 and latitudes greater than 34 and less than 36 :

```
mysql> select l.province, count(*) as total
    -> from pizzastations p, shopaddress sa, location l
    -> where p.menus_amountMax < 15 and p.id=sa.id and sa.longitude > -120 and sa.longitude < -118 and sa.latitude > 34 and sa.latitude < 36 and sa.id=l.
id
    -> group by l.province with rollup;
+---------------+-------+
| province      | total |
+---------------+-------+
| Arco-plaza    |     1 |
| Bicentennial  |     2 |
| Bouquet Canyon |     2 |
| Brentwood     |     4 |
| CA            |    30 |
| La Conchita   |     3 |
| Los Feliz     |     4 |
| Wla           |     9 |
| NULL          |    55 |
+---------------+-------+
9 rows in set (0.02 sec)
```

Both of the 3rd and 4th queries are quite heavy queries linking two dimension tables with the fact table, carrying significance in locating out options based on geographical locations maybe used

while searches on a map with the luxury factor in budget and tight budget constraints in respective cases.