|  |
| --- |
| Dummy  Zomato  Database  SQL Project Report (2021-2022) |
| A project by:  3506\_Aishani Anavkar  3507\_Om Gaikwad  3508\_Shikha Karmokar  3509\_ Sandesh Kotwal  3510\_Parth Kadam |
| |  |  |  | | --- | --- | --- | | SQL |  | Sem III | |

Contents

|  |  |
| --- | --- |
| **S.NO** | **TITLE** |
| 1. | Introduction |
| 2. | Purpose/Objective |
| 3. | Security & Encryption |
| 4. | Normalization |
| 5. | Detailed Expansion |
| 6. | E-R Diagram |
| 7. | Relationships between tables |
| 8. | Summary |
| 9. | Conclusion |

Introduction

Zomato is an Indian multinational restaurant aggregator and food delivery company founded by Deepinder Goyal, Pankaj Chaddah, and Gunjan Patidar in 2008. This network caters to millions of users every month. It currently operates in 23 countries. Zomato lets users search restaurants, get recommendations, add reviews, photos and such. They've recently released this directory of all restaurants & eateries currently under its purview.

**Description:**

Zomato provides information, menus and user-reviews of restaurants as well as food delivery options from partner restaurants in select cities. As of 2019, the service is available in 24 countries and in more than 10,000 cities. Zomato was founded as Foodiebay in 2008, and was renamed Zomato on 18 January 2010 as Zomato Media Pvt. Ltd. In 2011, Zomato expanded across India to Delhi NCR, Mumbai, Bangalore, Chennai, Pune and Kolkata. In 2012, the company expanded operations internationally in several countries, including the United Arab Emirates, Sri Lanka, Qatar, the United Kingdom, the Philippines, and South Africa. In 2013, Zomato was launched in New Zealand, Turkey, Brazil and Indonesia, with its website and apps available in Turkish, Portuguese, Indonesian and English languages. In April 2014, Zomato launched its services in Portugal, which was followed by launches in Canada, Lebanon and Ireland in 2015. In 2019, Zomato acquired Seattle-based food portal Urban spoon, which led to the firm's entry into the United States and Australia. This U.S.-expansion brought Zomato into direct competition with similar models such as Yelp and Foursquare. Zomato had also made a name for itself for its prowess in digital marketing.

Zomato provides products such as [table management](https://www.zomato.com/book) and [online ordering](https://www.zomato.com/business/order) for businesses along with [advertising and other merchant tools](https://www.zomato.com/business/apps). It is built to help restaurants thrive in the online marketspace. Connecting customers to restaurants is Zomato’s main goal. They work to ensure customers have the best experience at any restaurant found on their platform.



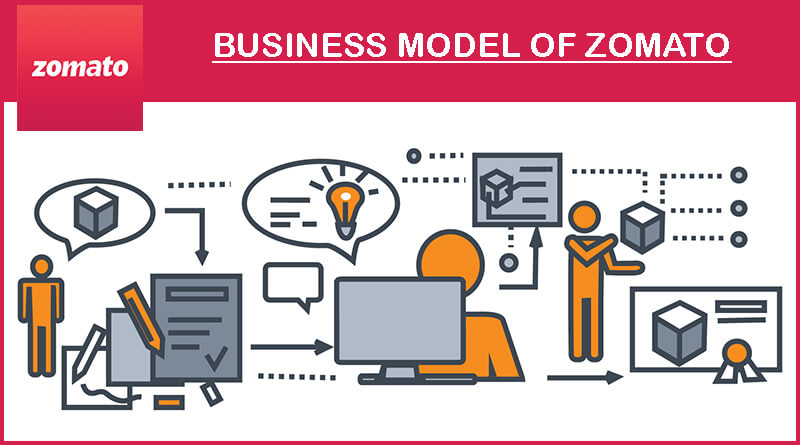
Purpose/Objective:

The purpose of this study and of creating this database is to get a clear idea of how big brand business like Zomato stay organized and keep information easily accessible, so that it can be used. But it isn’t a magic solution to all your data concerns.

This study helps us understand how Zomato organizes and extracts information from a database so that it is useable whenever necessary. We get a clearer idea of how a database system can help organize data, extract it, move it, and use it.

In this study we will see how database can:

* Reduce data redundancy
* Reduce updating errors and increased consistency
* Greater data integrity and independence from applications programs
* Improve data access to users through use of host and query languages
* Improve data security
* Reduce data entry, storage, and retrieval costs



Security & Encryption

Encrypting an entire database should be done with caution since it can result in a serious performance impact. It is therefore wise to encrypt only individual fields or tables. Encrypting data-at-rest protects the data from physical theft of hard drives or unauthorized file storage access. The purpose of data encryption is to protect digital data confidentiality as it is stored on computer systems and transmitted using the internet or other computer networks.

**CODE:**

import java.security.MessageDigest;

import oracle.sql.\*;

public class sha

{

public static oracle.sql.RAW get\_digest( String p\_string, int p\_bits ) throws Exception

{

MessageDigest v\_md = MessageDigest.getInstance( "SHA-" + p\_bits );

byte[] v\_digest;

v\_digest = v\_md.digest( p\_string.getBytes( "UTF-8" ) );

return RAW.newRAW(v\_digest);

}

}

create or replace FUNCTION "SHA" (p\_string IN VARCHAR2, p\_bits IN NUMBER)

RETURN RAW

AS

LANGUAGE JAVA

NAME 'sha.get\_digest( java.lang.String, int ) return oracle.sql.RAW';

select \* from users;

BEGIN

for Rec in (select \*

from users)

LOOP

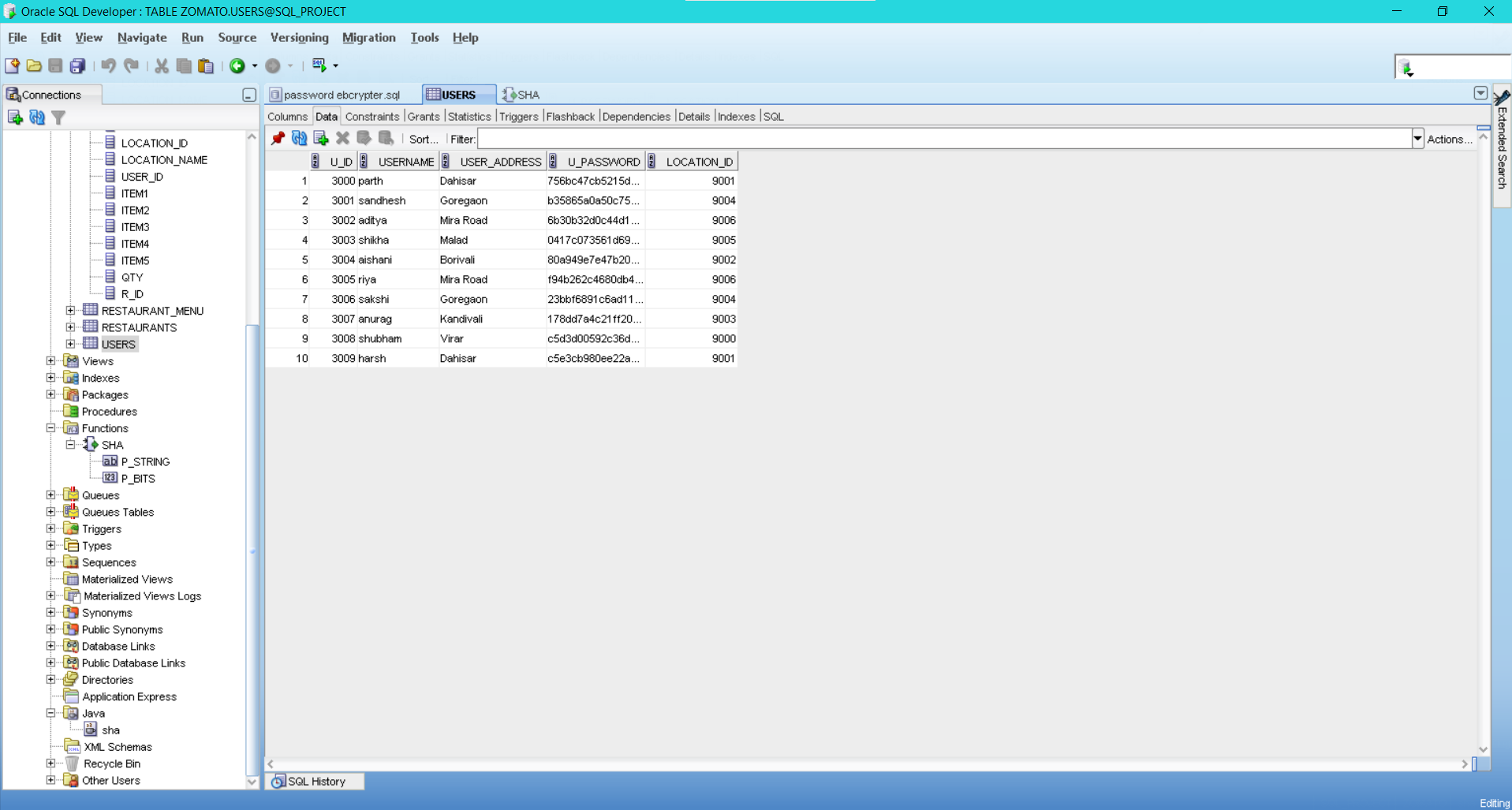
update users

set u\_password = LOWER(sha (u\_password, 256))

WHERE u\_id = Rec.u\_id;

END LOOP ;

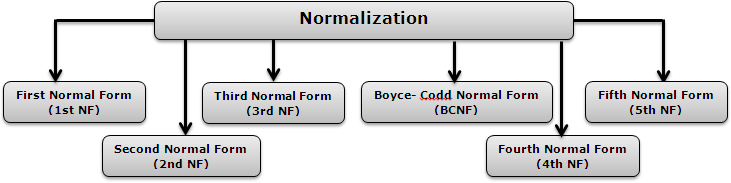
end;



Normalization

Normalization is the process of organizing data into a related table; it also eliminates redundancy and increases the integrity which improves performance of the query. To normalize a database, we divide the database into tables and establish relationships between the tables.

### Normalization Avoids

* **Duplication of Data**
* **Insert Anomaly**
* **Delete Anomaly**
* **Update Anomaly**

## **First Normal Form (1st NF)**

* The table cells must be of a single value.
* Eliminate repeating groups in individual tables.
* Create a separate table for each set of related data.
* Identify each set of related data with a primary key.

|  |  |  |  |
| --- | --- | --- | --- |
| Order | Customer | Delivery Boy | Total |
| 1 | Rishabh | Manish | 134.23 |
| 2 | Preeti | Rohan | 521.24 |
| 3 | Rishabh | Manish | 1042.42 |
| 4 | Rishabh | Manish | 928.53 |

## **Second Normal Form (2nd NF)**

* Remove Partial Dependencies.
* Functional Dependency: The value of one attribute in a table is determined entirely by the value of another.
* Partial Dependency: A type of functional dependency where an attribute is functionally dependent on only part of the primary key (primary key must be a composite key).
* Create a separate table with the functionally dependent data and the part of the key on which it depends. The tables created at this step will usually contain descriptions of resources.

|  |  |
| --- | --- |
| Customer | Delivery Boy |
| Rishabh | Manish |
| Preeti | Rohan |

|  |  |  |
| --- | --- | --- |
| Order | Customer | Total |
| 1 | Rishabh | 134.23 |
| 2 | Preeti | 521.24 |
| 3 | Rishabh | 1042.42 |
| 4 | Rishabh | 928.53 |

## **Third Normal Form (3rd NF)**

* Remove transitive dependencies.
* Transitive Dependency A type of functional dependency where an attribute is functionally dependent on an attribute other than the primary key. Thus its value is only indirectly determined by the primary key.
* Create a separate table containing the attribute and the fields that are functionally dependent on it. The tables created at this step will usually contain descriptions of either resources or agents. Keep a copy of the key attribute in the original file.

|  |  |  |  |
| --- | --- | --- | --- |
| Restaurant | City | State | ZIP |
| ABC Ltd. | Mumbai | MH | 10169 |
| XYZ Ltd. | Noida | UP | 33196 |
| ASD Ltd. | Chennai | TN | 21046 |

The above table is not in the 3NF.

|  |  |
| --- | --- |
| Restaurant | ZIP |
| ABC Ltd. | 10169 |
| XYZ Ltd. | 33196 |
| ASD Ltd. | 21046 |

|  |  |  |
| --- | --- | --- |
| City | State | ZIP |
| Mumbai | MH | 10169 |
| Noida | UP | 33196 |
| Chennai | TN | 21046 |

## **Fourth Normal Form (4th NF)**

* Has no multiple sets of multi-valued dependencies. In other words, 4NF states that no entity can have more than a single one-to-many relationship within an entity if the one-to-many attributes are independent of each other.
* Fourth Normal Form applies to situations involving many-to-many relationships.

|  |  |  |
| --- | --- | --- |
| Order | Restaurant | Delivery Boy |
| 1 | ABC | Manish |
| 1 | ABC | Raj |
| 1 | XYZ | Manish |
| 1 | XYZ | Raj |
| 2 | XYZ | Manish |
| 2 | XYZ | Suresh |
| 2 | PQR | Manish |
| 2 | PQR | Suresh |

|  |  |
| --- | --- |
| Order | Restaurant |
| 1 | ABC |
| 1 | XYZ |
| 2 | XYZ |
| 2 | PQR |
| Order | Delivery Boy |
| 1 | Manish |
| 1 | Manish |
| 2 | Raj |
| 2 | Suresh |

## **Fifth Normal Form (5th NF)**

* A relation that has a join dependency cannot be decomposed by a projection into other relations without spurious results
* A relation is in 5NF when its information content cannot be reconstructed from several smaller relations i.e. from relations having fewer attributes than the original relation

|  |  |  |
| --- | --- | --- |
| Customer | Restaurant | Item |
| Shalley | ABC | Fries |
| Mary | ABC | Pizza |
| Shalley | XYZ | Burrito |
| Mary | XYZ | Extra mayo |
| Shalley | XYZ | Chicken Puff |

Customer-Restaurant

|  |  |
| --- | --- |
| Customer | Restaurant |
| Shalley | ABC |
| Mary | ABC |
| Shalley | XYZ |
| Mary | XYZ |
| Shalley | XYZ |

Restaurant-Item

|  |  |
| --- | --- |
| Restaurant | Item |
| ABC | Fries |
| ABC | Pizza |
| XYZ | Burrito |
| XYZ | Extra mayo |
| XYZ | Chicken Puff |

Customer-Item

|  |  |
| --- | --- |
| Customer | Item |
| Shalley | Fries |
| Mary | Pizza |
| Shalley | Burrito |
| Mary | Extra mayo |
| Shalley | Chicken Puff |

Implementation

Let's see the basic procedure for how we created the database as you see. The whole plot

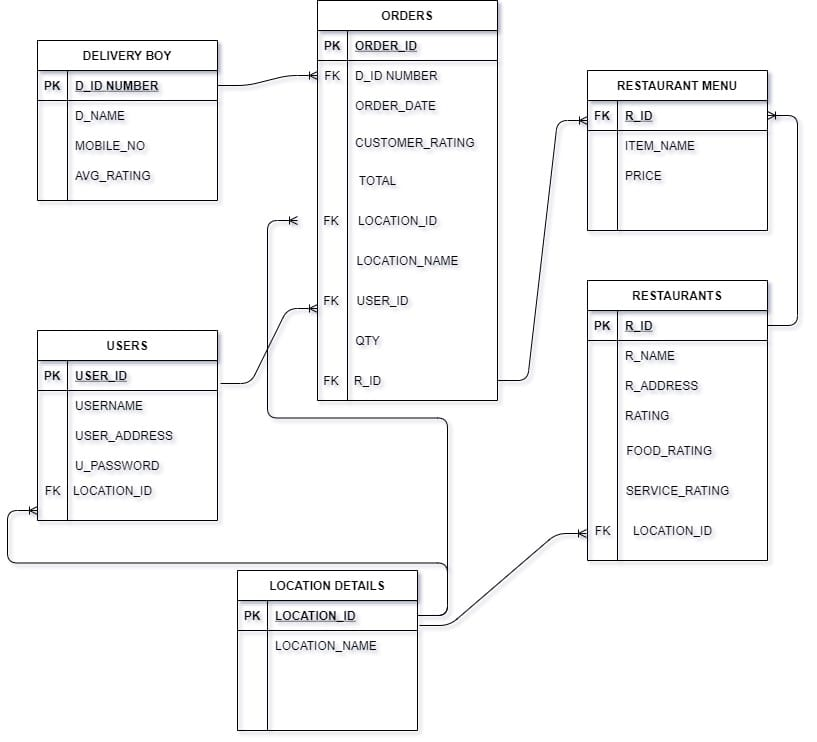
revolves around creating different tables according to your needs and data available and linking them to each other using constraints.

* Our database consists of 6 tables and for creating those tables, we used the "CREATE TABLE" function.
* Now suppose you have a column named Order ID in "ORDERS" table and also in some other table. We can link these tables using the "CONSTRAINT" function which allows us to set the same column from the first table as a "PRIMARY KEY" and that from second table as "FOREIGN KEY" or vice versa as per our requirements.
* Finally, after creating all the tables we used the "INSERT" function to insert the values and complete our database.

Detailed Expansion

This was just a vague idea of how the database was created. Let's dive deep into those tables and see why we created each and every variable in detail.

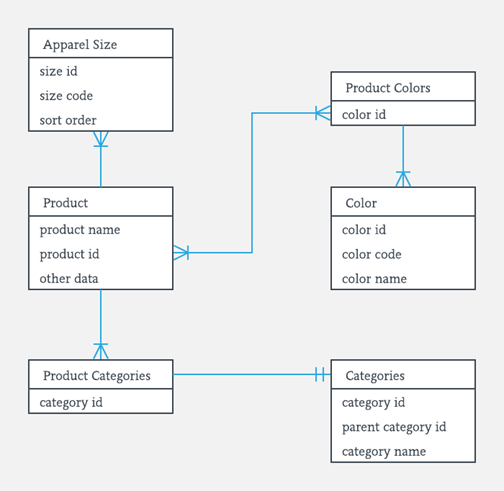
* First of is the table named DELIVERY BOY. The ID number is set to be the primary key so that we can identify each row of that table uniquely. The other three variable include the common details of any Valet who would deliver you your parcel.
* The second table ORDERS table which has 10 variables. The ORDER\_ID is the primary key so that no orders are mixed up. We also have the Name of the valet who will deliver the parcel as the foreign key. The other foreign keys are LOCATION\_ID, USER\_ and R\_ID.
* The third table is RESTAURANT MENU table. It also has the ID for each restaurant as a foreign key which refers to the Restaurant’s table and also has a third variable PRICE which reflects the price for each item available in different restaurants.
* The fourth table is USERS which contains all the data of our customers. The variables used are USER\_ID being the primary key for unique identification, USERNAME, USER\_ADDRESS, U\_PASSWORD and LOCATION\_ID being the foreign key refers to the Location Details table.
* The RESTAURANTS table has 7 variables out of which R\_ID being the primary key and LOCATION\_ID being the foreign key.
* The last table is the LOCATION DETAILS table which has only 2 variables out of which one is Primary key which is the LOCATION\_ID. This tables holds the address for Restaurants as well as for users.



E-R Diagram

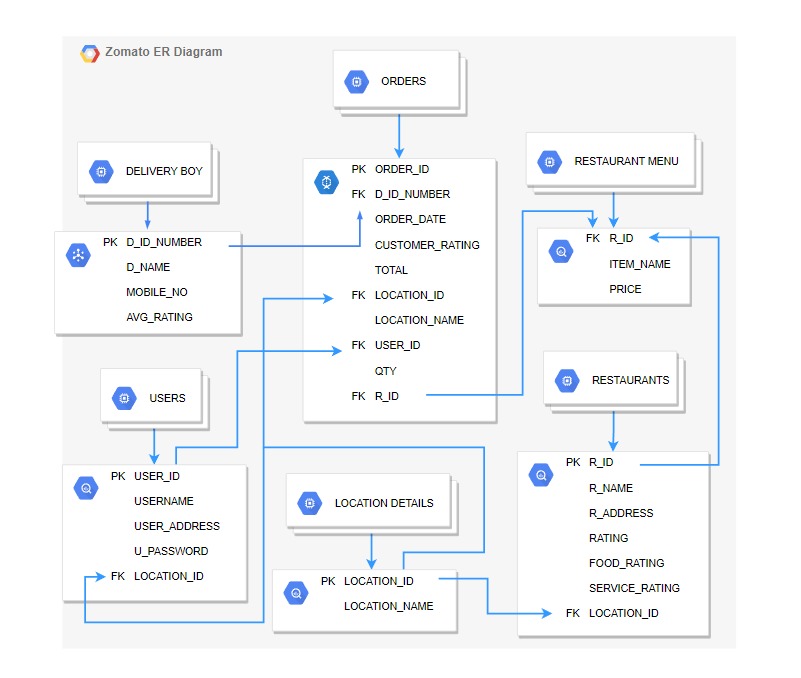
An entity–relationship model (or ER model) describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between [entities](https://en.wiktionary.org/wiki/entity) (instances of those entity types).

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation that depicts relationships among people, objects, places, concepts or events within an information technology (IT) system.



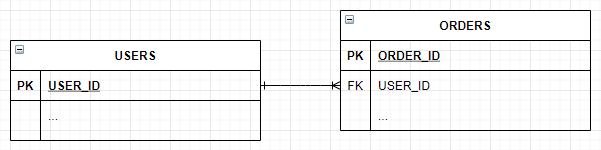
A basic E-R Diagram example for an apparel store

Zomato Dummy Database E-R Diagram



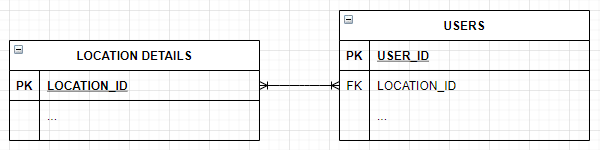
Relationships between attributes

A relationship type represents the association between entity types. For example, ‘Enrolled in’ is a relationship type that exists between entity type Student and Course.

****

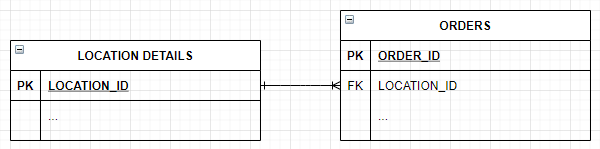
One-To-Many relation between USERS and ORDERS

ORDERS takes the USER\_ID fk from USERS in order to trace back a unique order to a unique user. One user can have multiple orders.



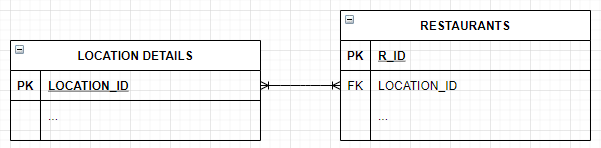
Many-To-Many relation between LOCATIONS DETAILS and USERS

USERS takes the LOCATION\_ID fk from LOCATION DETAILS where LOCATION\_ID is the unique ID created for every unique address. This is done in order for the address to be accessible to the end user and be available with the order but not be accessible to an administrator. One address can have multiple users and vice-versa.

****

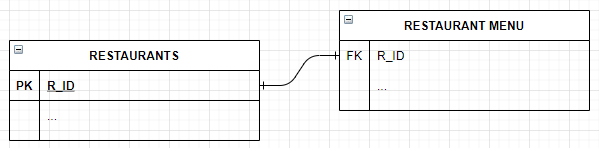
One-To-Many relation between LOCATION DETAILS and ORDERS

As mentioned in the previous relation, orders have to be available with the location ID since a delivery boy needs to know where to deliver the order. Hence, ORDERS takes LOCATION\_ID from LOCATION DETAILS. One location can have many orders.



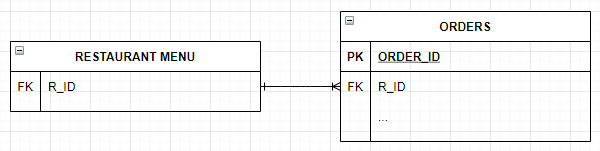
Many-To-Many Relation between LOCATION DETAILS

Similarly, restaurants also need to identify the users’ address in order to deliver the order. One location can order from many restaurants and a restaurant can have orders from many locations.



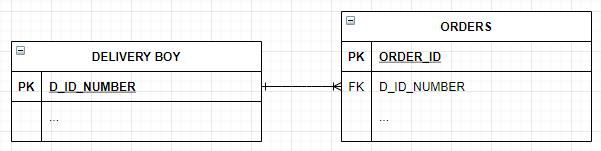
One-To-One Relation between RESTAURANTS and RESTAURANT MENU

RESTAURANT MENU takes the R\_ID from RESTAURANTS in order for a menu to uniquely identify towards a restaurant as that restaurant’s menu. One restaurant usually has one menu.



One-To-Many relation between RESTAURANT MENU and ORDERS

ORDER takes R\_ID fk from RESTAURANT MENU since an order has to placed from within a restaurant menu belonging to a restaurant. Many orders can be placed from one restaurant menu.



One-To-Many relation between DELIVERY BOY and ORDERS

ORDERS takes D\_ID\_NUMBER fk from DELIVERY BOY since orders are allocated a delivery boy for them to be delivered. One delivery boy can take multiple orders.

Queries

**Ranking of delivery boys according to customer rating**

* SELECT d\_id, AVG(customer\_rating) FROM orders

group by d\_id;

Could be used to review a delivery boy’s rating in order to ensure a good delivery beforehand

**Write a SQL query to find the list of Zomato users who made more than 5 orders in total**

* SELECT user\_id, count(\*) FROM orders GROUP BY user\_id HAVING count(\*) >5;

SELECT \* FROM users WHERE user\_id = 2;

Could be used by an administrator to reward and incentivize activity on a large scale by providing active users offers etc.

**Write a query to get the details of orders and delivery boys who have delivered in Kandivali**

* SELECT \*

FROM delivery\_boy, orders

WHERE location\_name='Kandivali';

Could be used to analyse a certain areas activity by a user to estimate service or by an admin to allocate resources.

Summary

ER Model is used to model the logical view of the system from data perspective which consists of these components:

An **Entity** may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.

In our case, the entities within the E-R Diagram include

* Users
* Restaurants
* Restaurant Menu
* Orders
* Delivery Boy
* Location Details

**Users**, as the name suggests are the users to the service aided by the management system. Users are what all the other entities like restaurants and orders have to be traced back to. The entity ‘USERS’ has attributes:

* User\_ID
* Username
* User\_Address
* U\_Password
* Location\_ID

**Orders** are the delivery requests put in by the users towards restaurants  
logged into the database. The entity ‘ORDERS’ has attributes:

* Order\_ID
* D\_ID\_Number
* Order\_Date
* Customer\_Rating
* Total
* Location\_ID
* Location\_Name
* User\_ID
* Qty
* R\_ID

**Restaurants** are the entities that receive each unique order to be satisfied towards the users. The entity ‘RESTAURANTS’ has attributes:

* R\_ID
* R\_Name
* R\_Address
* Rating
* Food\_Rating
* Service\_Rating
* Location\_ID

**Restaurant Menu** is the entity containing the unique menu for each corresponding unique restaurants. The entity ‘RESTAURANT MENU’  
has attributes:

* R\_ID
* Item\_Name
* Price

**Delivery Boy** is the entity that gets allotted the orders to be fulfilled from the restaurant to the user. The entity ‘DELIVERY BOY’ has attributes:

* D\_ID\_Number
* D\_Name
* Mobile\_No
* Avg\_Rating

**Location Details** is the entity utilized to retain privacy of the user by automatically producing an ID for the users’ location along with the address itself so as to not be accessible to any administrators. The entity ‘LOCATION DETAILS’ has attributes:

* Location\_ID
* Location\_Name

Conclusion:

We have seen how a database works in detail. Now time to summarise the report with an example about how Zomato works with their original database. Zomato originally uses MySql database for its database management. It has an efficient system for DBMS and storing data in ay cloud based easy to use pos system. Thus, making the data safe and easily accessible. The data stored is then used for various functions like:

1. INVENTORY MANAGEMENT
2. MENU MANAGEMENT
3. DELIVERY MANAGEMENT
4. PAYMENTS
5. CUSTOMER RELATIONSHIP MANAGEMENT.
6. ETC

Here we showed you a sample model of Zomato's database with some dummy data in it.

I hope u liked our project

THANK YOU!!