Dieu Hoa Nguyen

Option A: Database Design and Technical Report
Topic: Waterloo restaurants' customer ratings on the online food ordering application Doordash

Chris Brogly
CS338
University of Waterloo

I. Overview:

The Doordash database contains data related to customers' ratings on orders made on the online food ordering application Doordash during the time from January 1, 2020 to June 30, 2020. There is a total of 252 restaurants as recorded on their official website by August 11. Many restaurants are not registered with Doordash anymore, thus not being able to search on the website, which explains a decrease in number of restaurants compared to June 13, 2020 when the Project Proposal was due. This database will also update on whether the restaurants are newly added, open or closed, the locations that these places ship to, and most importantly the current customer ratings for both the shippers' services and the quality of the restaurants as a whole and on each order specifically. The ratings will be useful in finding the n-top restaurants due to customer preferences, as well as reviewing the performance of these restaurants and shippers on this application. The intended audience for the usage of this database is employees at Doordash, who needs to store data on some sources and use data properly within the permission of their roles.

The database should be applicable for both Basic SQL (since data types stored are mostly types that are used frequently with Basic SQL statements) and Advanced SQL (tables/columns are complicated enough to use Advanced SQL statements to generate).

There are 8 tables as stated in the Project Proposal: 1 table for the contact information and public ratings of all restaurants appearing on Doordash up to the date of August 11, 2020 called "Restaurants", 1 zip code table tracking all zip codes that restaurants can ship to within the region of Waterloo called "ZipCode", and 1 table called "Category" which lists all the categories of food that Doordash labels for Waterloo restaurants. Data for these three tables (except for the "PhoneNo" column in the "Restaurants" table were taken from a data generating tool) are collected manually from the official Doordash website: https://www.doordash.com/en-US. I couldn't cite the direct link to the list of all Waterloo restaurants on Doordash since it will always direct to the main website. I entered the keyword "Waterloo" when they ask for my address in the main page and got to the destination site. The "ZipCode" and "Category" tables were also taken manually from this page. Other tables include: 1 table with all information of shippers registered

with Doordash called “Shippers”, 1 table for all promotion and discount in the application called “Promotion”, 1 table of all orders from customers of the Waterloo region in the first 6 months of 2020, and 1 final table including all ratings that customers made for restaurants order, shipping order, and dishes order. These data, together with the “PhoneNo” column in the “Restaurants” table, were taken from data generating tools Spawner and Generatedata.com. Therefore, all data in these tables are made up and not the correct ratings on the application. Spawner was used to generate real numbers on specific ratings for the “OrderRating” column in the “Ratings” table. All other tables and columns were generated through Generatedata.com since it is able to generate more diverse data types.

The “Restaurants” table does not include the columns “Website” and “Shipping” as in the Project Proposal. There are no websites listed on the official Doordash webpage, and all the data generating tools that I tried for this data types do not generate the website type. The “Shipping” column was created because I assumed that there are shippers working for Doordash and shippers that restaurants own separately, but since it does not relate to other tables, so I did not include this column anymore. The “Category” has one more column “Type”, which is useful in labelling the categories even though it contributes to the database not being in 3NF. The Ratings for Restaurants and Shipping in the “Ratings” table are unique to their specific rating, while the Ratings as in the “Res_Rate” and “Ship_Rate” of the “Restaurants” and “Shippers” tables are their accumulated ratings from even before January 1, 2020.

II. Usage Documentation

1. Application to Basic SQL:

a, Creating and deleting tables:

- Create the Orders table:

```
CREATE TABLE "Orders" (
        "OrderID"    INTEGER NOT NULL UNIQUE,
        "ResID"      TEXT,
        "ShipID"     INTEGER,
        "ZipID"      TEXT,
        "Date"TEXT,
        "PayID"      INTEGER,
        "PromoID"    INTEGER,
        PRIMARY KEY("OrderID"),
        FOREIGN KEY("ResID") REFERENCES "Restaurants" ("ResID"),
        FOREIGN KEY("PromoID") REFERENCES "Promotion" ("PromoID"),
        FOREIGN KEY("ShipID") REFERENCES "Shippers" ("ShipID"),
        FOREIGN KEY("ZipID") REFERENCES "ZipCode" ("ZipID"),
        FOREIGN KEY("PayID") REFERENCES "Payment" ("PayID")
);
```

b, The INSERT Statement:

- Insert a payment method to the Payment table:

```
INSERT INTO Payment VALUES (906, "Free Meal Code", “Due to a cancel order in the past”);
```

- Insert a new restaurant to the Restaurants table:

```
INSERT INTO Restaurants VALUES("#0253","Hem Hakka’s Restaurant", "1-519-650-3883","13 King St N", "Hakka", 4.6, "Open");
```

c, The SELECT Statement:

- All the restaurants with a rating of 5.0:

```
select * from Restaurants where Res_Rate = 5.0;
```

- Shippers that drives a Honda:

select * from Shippers where Vehicle = "Honda";

- Shippers that ship to area with the zip code starting with “n2l”:

select Name from Shippers where ShipID in (select ShipID from Orders where ZipID in (select ZipID from ZipCode where ZipCode like "n2l%"));

- The average ratings of shippers that drive a Mazda:

select avg(ShipRating) from Ratings where ShipID in (select ShipID from Shippers where Vehicle = "Mazda");

d, The UPDATE statement:

- Change the ratings for all orders with the restaurant Chef on Call to be 4.0:

update Ratings set OrderRating = 5.0 where ResID in
(select ResID from Restaurants where Name = "Chef on Call");

- Change the promotion of all orders from “Royal Pizza” to be monthly promotion:

update Orders set PromoID = (select PromoID from Promotion where Type = "Individual Restaurants promotion") where ResID in (select ResID from Restaurants where Name = "Royal Pizza");

2. Application to Advanced SQL:

- 5 restaurants that have the highest average rating for the first half of 2020:

select round(avg(Ratings.ResRating),2) as Best_Rate, Restaurants.Name
from Ratings inner join Restaurants on Ratings.ResID = Restaurants.ResID
group by Ratings.ResID order by Best_Rate DESC limit 5;

- See the average overall ratings for all shippers to each area during the first 6 months of 2020:

select avg(Shippers.Ship_Rate), ZipCode.ZipCode from (Shippers inner join Orders on Shippers.ShipID = Orders.ShipID) inner join ZipCode on Orders.ZipID = ZipCode.ZipID group by Orders.ZipID;

3. Views:

- Creating a view of a comparison of the overall ratings of all restaurants that have orders in the first half of 2020 to their average ratings in the first half of 2020:

Create view Comparison as select Restaurants.Name, round(avg(Ratings.ResRating), 1) as New_Rating, Restaurants.Res_Rate from Ratings inner join Restaurants on Ratings.ResID = Restaurants.ResID group by Ratings.ResID;

III. Normalization:

The database is in 2NF, since there is only one primary key for each table and thus no functional dependencies are be partially dependent on any of the primary keys. However, the database is not in 3NF as seen in the normalization below:

Let Primary Key be PK, and Functional Dependencies as FD:

1. Category table:
   PK: CatID
   FD1: CatID -> Name
   FD2: Name -> Type
   Thus, the table is in 2NF but not in 3NF due to transitive dependency from FD1 to FD2.
2. Payment table:
   PK: PayID
   FD1: PayID -> Method
   FD2: Method -> Notes
   Thus, the table is in 2NF but not in 3NF due to transitive dependency from FD1 to FD2.
3. Promotion table:
   PK: PromoID
   FD1: PromoID -> Details
   FD2: Details -> Type
   Thus, the table is in 2NF but not in 3NF due to transitive dependency from FD1 to FD2.
4. ZipCode table:
   PK: ZipID
   FD: ZipID-> ZipCode
   The table is in 3NF since there are no transitive dependency.
5. Shippers table:
   PK: ShipID
   FD1: ShipID -> Name

FD2: ShipID -> PhoneNo

FD3: Name -> PhoneNo

Thus, the table is in 2NF but not in 3NF due to transitive dependency from FD2 to FD3.

6. Restaurants table:

   PK: ResID

   FD1: ResID -> Name

   FD2: Name -> PhoneNo

   Thus, the table is in 2NF but not in 3NF due to transitive dependency from FD1 to FD2.

7. Orders table:

   PK: Order ID

   FD: OrderID -> {ResID, ShipID, ZipID, Date, PayID, PromoID}

   The table is in 3NF because there are no transitive dependencies.

8. Ratings table:

   PK: RateID

   FD1: RateID -> OrderID

   FD2: RateID -> ResID

   FD3: RateID -> ShipID

   FD4: OrderID -> OrderRating

   FD5: ResID -> ResRating

   FD6: ShipID -> ShipRating

   Thus, the table is in 2NF but not in 3NF due to transitive dependency from FD1 to FD4, FD2 to FD5, and FD3 to FD6.

References:

Data generating tools used during the building of the database:

1. Spawner:

Link to download the application: https://sourceforge.net/projects/spawner/

This is an open source data generating tool that requires downloading to the local machine to be able to use. The license for the tool is GNU General Public License version 2.0 (GPLv2) as stated in the same URL to download this application.

2. Generatedata.com:

This is an open source data generating tool with the manual to use the online script (as at https://generatedata.com) or to download to the local machine can be found here: https://github.com/benkeen/generatedata

The license of the tool is GPL 3, which can also be found in the manual link above.