BUILD: Project I / Design & Implement a Relational Database

Project proposal:

We're going to build a restaurant review management system that lets users submit reviews for a restaurant and lets the restaurant/user manage these reviews.
The application will support:

- CRUD operations of the reviewers
- CRUD operations of the restaurants
- CRUD operations of the reviews
- Query/filter reviews based on cost, service, parking, waiting time, and overall rating
- Query/filter restaurants based on name, food type, working hours, accepted payment methods, Facilities
- Query/filter users based on food preferences and available payment methods

Requirements of the problem:

In today's age **restaurants** lean on services like yelp to help them advertise their **restaurant**. These services try to summarize the **customer's** experience by allowing them to *rate and review* the restaurants. Restaurants use the services to build credibility by *asking* customers for positive reviews when they feel that the customer is happy. The Project consists of the following parts: The ability to add reviews, customer and restaurants.

The customer should be able to *create* an **account** by providing the following details: Information on if they are a **smoker**, if they are **heavy drinkers**, the **budget**, the **ambiance** the customer prefers, the preferred **dress code**, what **cuisine** they like and what **payment methods** they have available.

After creating an account, a customer can *write* reviews for an already registered restaurant. A restaurant has the following information provided: The **address**, **city, state, country**, **price range**, the **cuisine** they are serving. The restaurant also provides certain **facilities** like **parking space** available, **ambience, seating area** and **services** like **alcohol, dress code, smoking area.** They have a set of available **payment methods** and **working hours.**

When *creating* a review, the customer *provides* a **rating** of 0-5 for the **cost**, **food**, **service**, **parking**, **waiting time**, and if they want a written **review**.

Nouns in bold and verbs in Italic

Nouns:

Restaurants

- Name
- Address
- Price Range
- Cuisine

Facilities

- Ambience
- Seating Area
- Parking

Services

- Alcohol
- dress code
- smoking area
- payment methods
- working hours

Reviews

- Cost
- Food
- Service
- Parking
- waiting time

Account/customer

- smoker
- heavy drinkers
- Budget
- Ambiance
- dress code
- Cuisine
- payment methods
- name

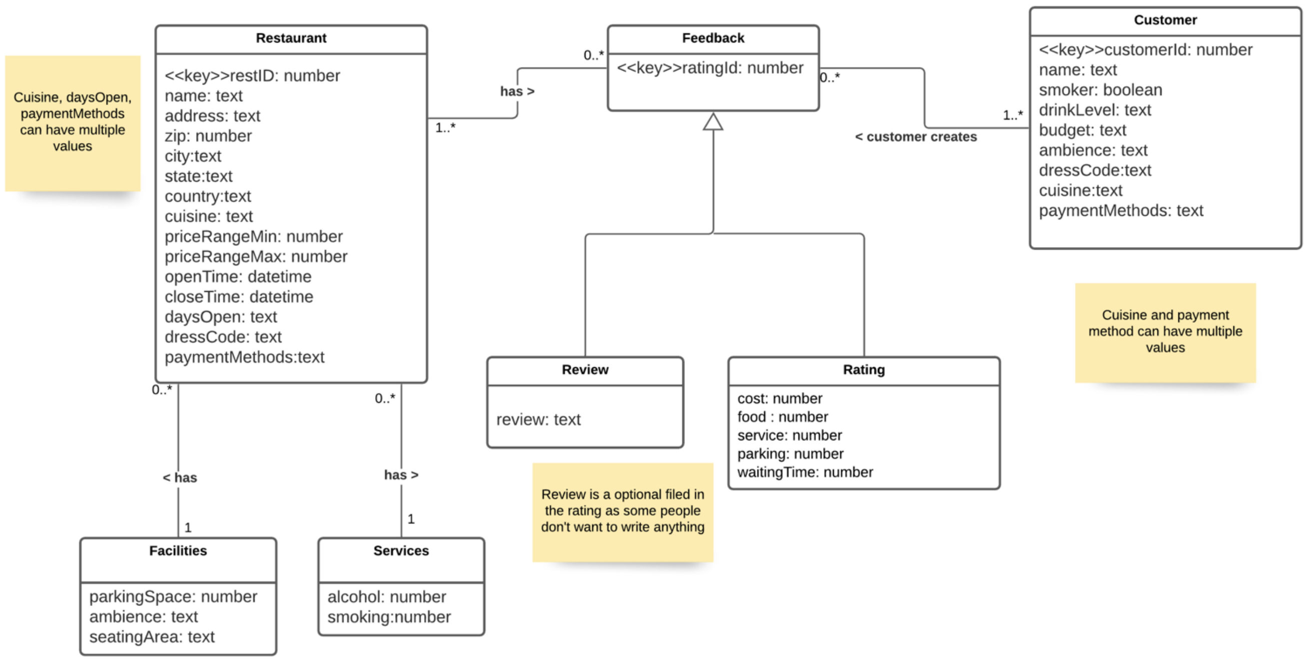
Verbs:

- Asking – restaurants ask customers to review them
- Creates - customer creates reviews on restaurant
- Provides – restaurants provide certain facilities and services

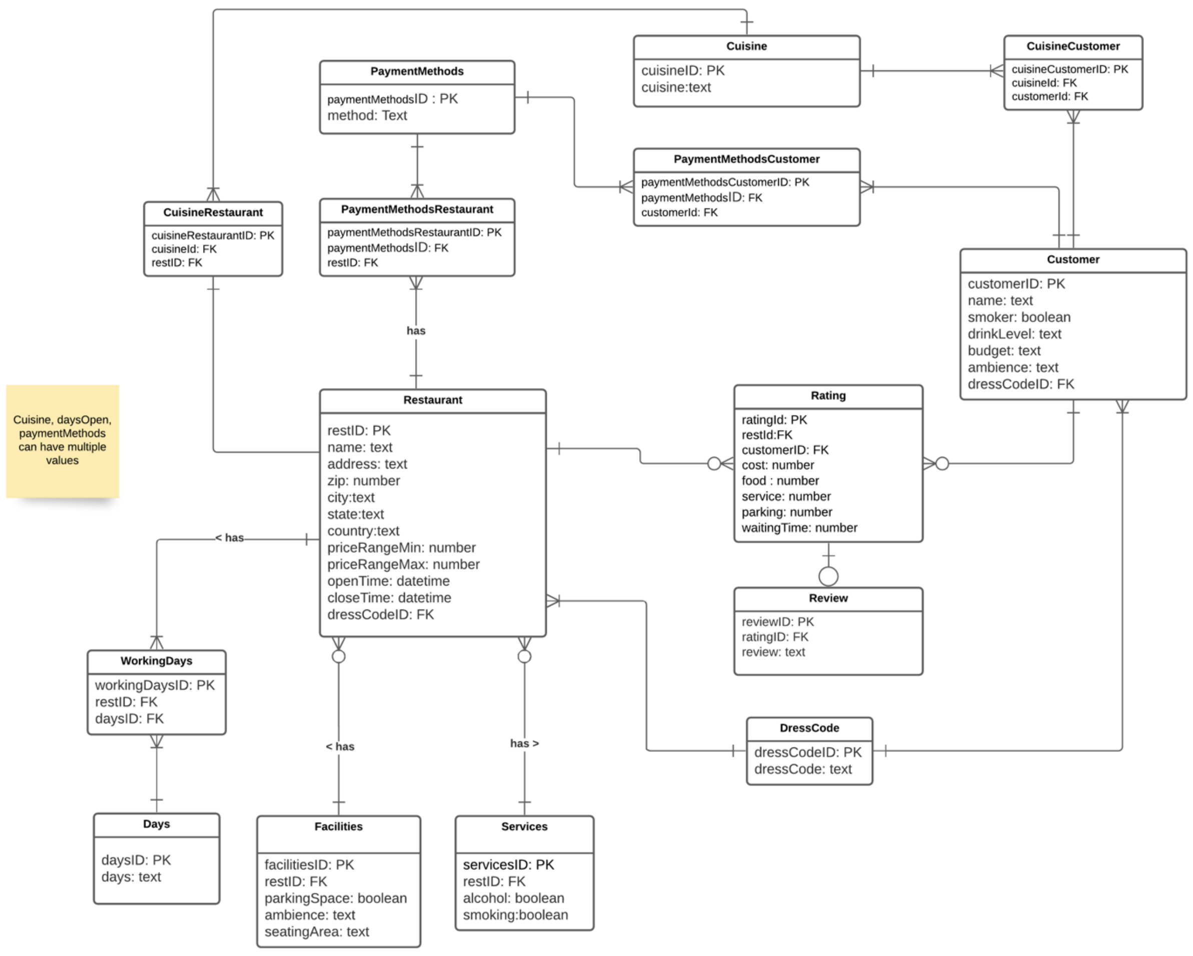
Business Rules:

1. Facilities can be offered by many restaurants while each restaurant provides a set of facilities.
2. Services can be offered by many restaurants while each restaurant provides a set of services.
3. A restaurant can get none to many feedbacks while feedback has 1 or more restaurants
4. Customer can create 0 or more feedbacks while feedbacks have 1 or more customers
5. Cuisine, Days Open and Payment methods can have multiple values
6. There are about 19 cuisines in total to choose from
7. There are 5 total dress code types
8. There are 5 different payment methods
9. 7 working days

Conceptual model:



Logical model:



https://lucid.app/documents/view/dfef3e2d-917a-4e03-b69d-3cdab87b2dab

1. Many-to-Many relation between Restaurant and feedback & customer and feedback has been broken down.
2. Multivalued attributes like Cuisine, Working Days, Payment Methods have been eliminated
3. Dress code can have a certain range of 5 values like Informal, formal, casual, etc and hence is separated.

Relational scheme:

1. Restaurant(restID,dressCodeID,name,address, zip, city, state, country, priceRangeMin, priceRangeMax, openTime, closeTime)
2. WorkingDays(workingDaysID, restID, daysID)
3. Days(daysID, day)
4. Facilities(facilitiesID, restID, parkingSpace, ambience, seatingArea)
5. Services(restID, servicesID , alcohol,smoking)
6. PaymentMethodsRestaurant(paymentMethodsRestaurantID, paymentMethodsID, restID)
7. CuisineRestaurant(cuisineCustomerID, cuisineId, restID)
8. DressCode(dressCodeID, dressCode)
9. Cuisine(cuisineID, cuisine)
10. Rating(ratingId, cost, food, service, parking, waitingTime)
11. Review(reviewID, ratingID, review)
12. Customer(customerID name, smoker, drinkLevel, budge, ambience, dressCodeID)
13. CuisineCustomer(cuisineCustomerID, cuisineId, customerId)
14. PaymentMethodsCustomer(paymentMethodsCustomerID, paymentMethodsID, customerId)
15. PaymentMethods(paymentMethodsID, method)

Relational schema in at least BCNF

Restaurant:

restID -> dressCodeID, name, address, zip, city, state, country, priceRangeMin, priceRangeMax, openTime, closeTime

WorkingDays:

workingDaysID -> restID, daysID

Days

daysID-> day

Facilities:

facilitiesID -> RestID, parkingSpace, ambience, seatingArea

Services:

servicesID -> restID, alcohol, smoking

PaymentMethodsRestaurant:

paymentMethodsRestaurantID -> paymentMethodsID, restID

CuisineRestaurant:

cuisineCustomerID -> cuisineId, restID

DressCode:

dressCodeID -> dressCode

Cuisine:

cuisineID -> cuisine

Rating:

ratingId -> cost, food, service, parking, waitingTime

Review:

reviewID -> ratingID, review

Customer:

customerID -> name, smoker, drinkLevel, budge, ambience, dressCodeID

CuisineCustomer:

cuisineCustomerID -> cuisineId, customerId

PaymentMethodsCustomer:

paymentMethodsCustomerID -> paymentMethodsID, customerId

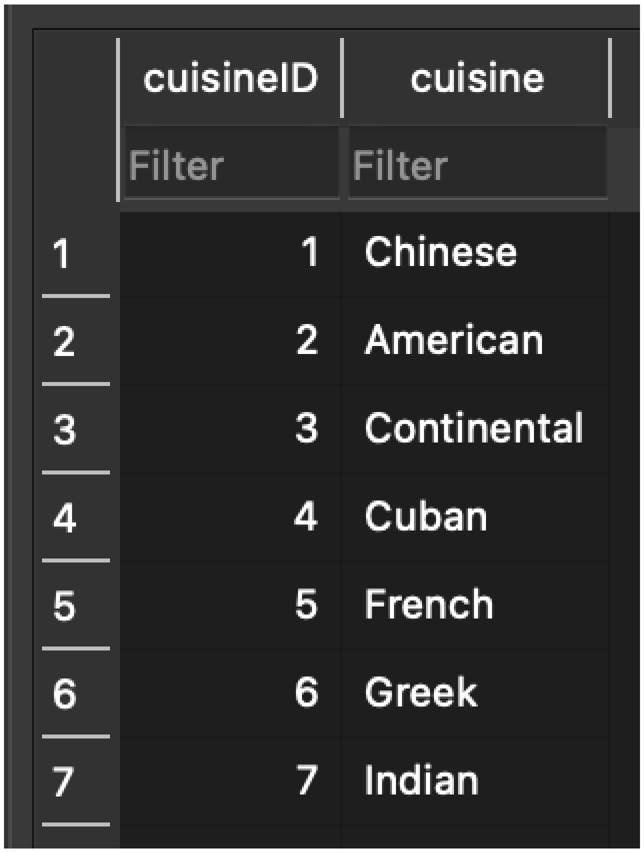
PaymentMethods:

paymentMethodsID -> method

SQL data definition statements

Cuisine:

```
CREATE TABLE "Cuisine" (
"cuisineID"       INTEGER NOT NULL UNIQUE,
"cuisine"         TEXT NOT NULL,
PRIMARY KEY("cuisineID" AUTOINCREMENT)
)
```

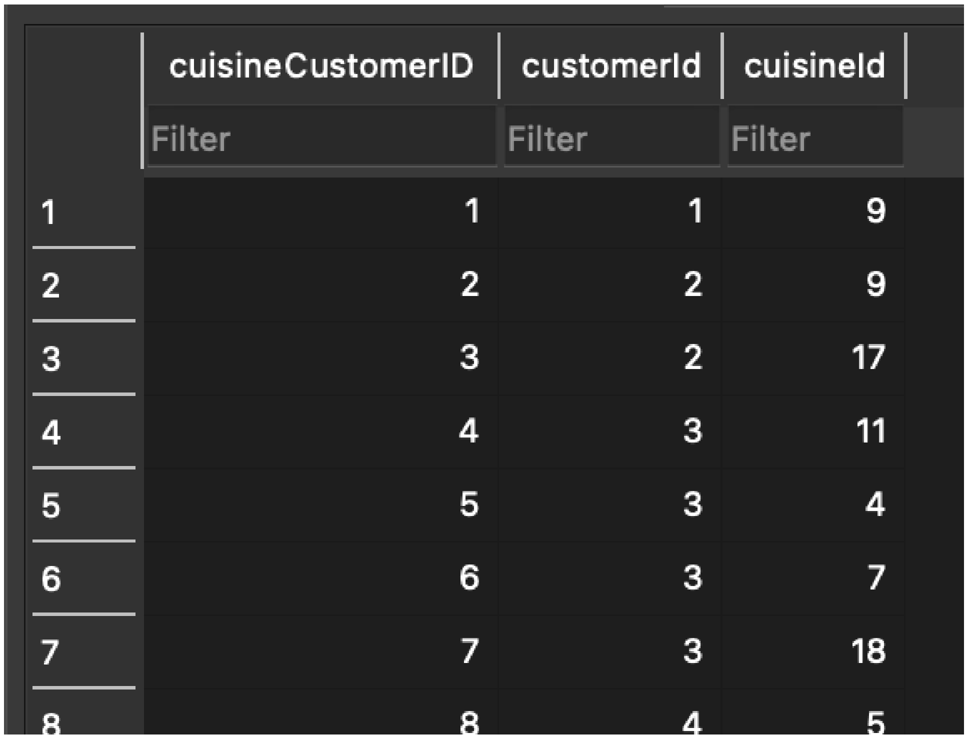


	cuisineID	cuisine
1	1	Chinese
2	2	American
3	3	Continental
4	4	Cuban
5	5	French
6	6	Greek
7	7	Indian

CuisineCustomer:

```
CREATE TABLE "CuisineCustomer" (
"cuisineCustomerID"     INTEGER NOT NULL UNIQUE,
"customerId"    INTEGER,
"cuisineId"       INTEGER NOT NULL,
PRIMARY KEY("cuisineCustomerID" AUTOINCREMENT),
FOREIGN KEY("customerId") REFERENCES "Customer"("customerID"),
FOREIGN KEY("cuisineId") REFERENCES "Cuisine"("cuisineID")
```

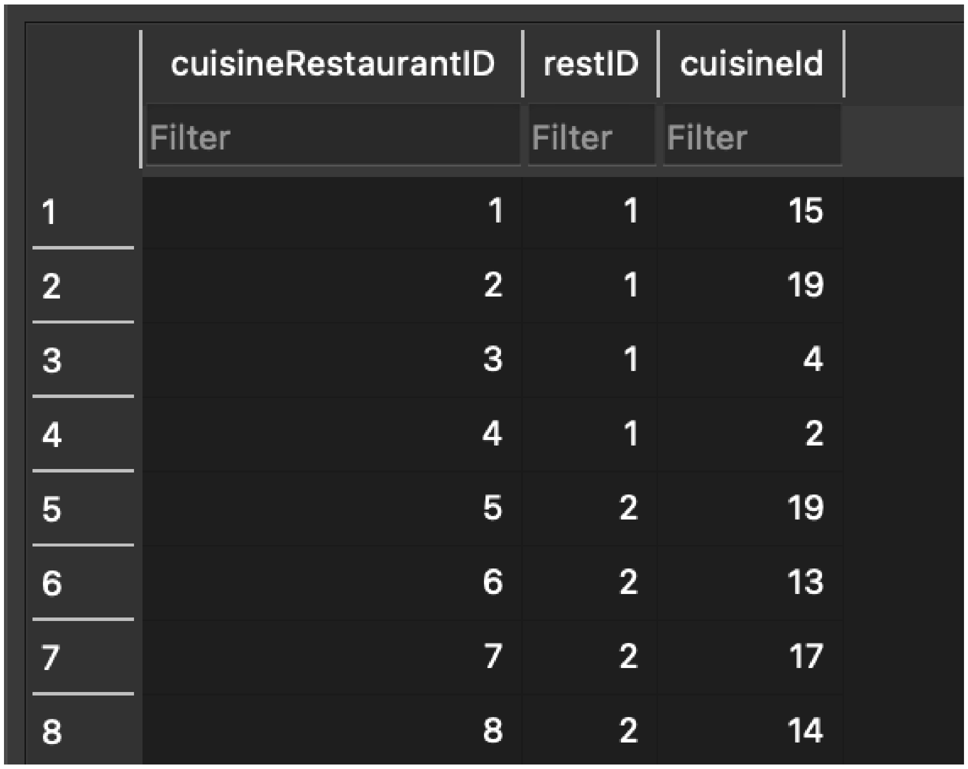
)



	cuisineCustomerID	customerId	cuisineId
	Filter	Filter	Filter
1	1	1	9
2	2	2	9
3	3	2	17
4	4	3	11
5	5	3	4
6	6	3	7
7	7	3	18
8	8	4	5

CuisineRestaurant:

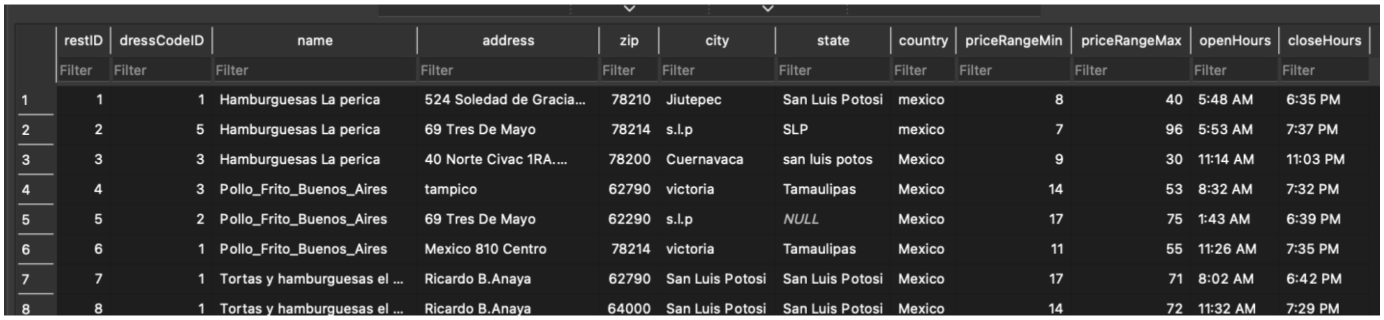
```
CREATE TABLE "CuisineRestaurant" (
"cuisineRestaurantID"    INTEGER NOT NULL UNIQUE,
"restID"INTEGER NOT NULL,
"cuisineId"         INTEGER NOT NULL,
PRIMARY KEY("cuisineRestaurantID" AUTOINCREMENT),
FOREIGN KEY("restID") REFERENCES "Restaurant"("restID"),
FOREIGN KEY("cuisineId") REFERENCES "Cuisine"("cuisineID")
)
```



	cuisineRestaurantID	restID	cuisineId
	Filter	Filter	Filter
1	1	1	15
2	2	1	19
3	3	1	4
4	4	1	2
5	5	2	19
6	6	2	13
7	7	2	17
8	8	2	14

Restaurant:

```
CREATE TABLE "Customer" (
"customerID"    INTEGER NOT NULL UNIQUE,
"name" TEXT NOT NULL,
"smoker"        INTEGER NOT NULL,
"drinkLevel"    TEXT NOT NULL,
"dressCodeID"   INTEGER NOT NULL,
"ambience"      TEXT NOT NULL,
"budget"        TEXT NOT NULL,
PRIMARY KEY("customerID" AUTOINCREMENT),
FOREIGN KEY("dressCodeID") REFERENCES "DressCode"("dressCodeID")
)
```



	restID	dressCodeID	name	address	zip	city	state	country	priceRangeMin	priceRangeMax	openHours	closeHours
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	Hamburguesas La perica	524 Soledad de Gracia...	78210	Jiutepec	San Luis Potosi	mexico	8	40	5:48 AM	6:35 PM
2	2	5	Hamburguesas La perica	69 Tres De Mayo	78214	s.l.p	SLP	mexico	7	96	5:53 AM	7:37 PM
3	3	3	Hamburguesas La perica	40 Norte Civac 1RA....	78200	Cuernavaca	san luis potos	Mexico	9	30	11:14 AM	11:03 PM
4	4	3	Pollo_Frito_Buenos_Aires	tampico	62790	victoria	Tamaulipas	Mexico	14	53	8:32 AM	7:32 PM
5	5	2	Pollo_Frito_Buenos_Aires	69 Tres De Mayo	62290	s.l.p	*NULL*	Mexico	17	75	1:43 AM	6:39 PM
6	6	1	Pollo_Frito_Buenos_Aires	Mexico 810 Centro	78214	victoria	Tamaulipas	Mexico	11	55	11:26 AM	7:35 PM
7	7	1	Tortas y hamburguesas el ...	Ricardo B.Anaya	62790	San Luis Potosi	San Luis Potosi	Mexico	17	71	8:02 AM	6:42 PM
8	8	1	Tortas y hamburguesas el ...	Ricardo B.Anaya	64000	San Luis Potosi	San Luis Potosi	Mexico	14	72	11:32 AM	7:29 PM

Days:

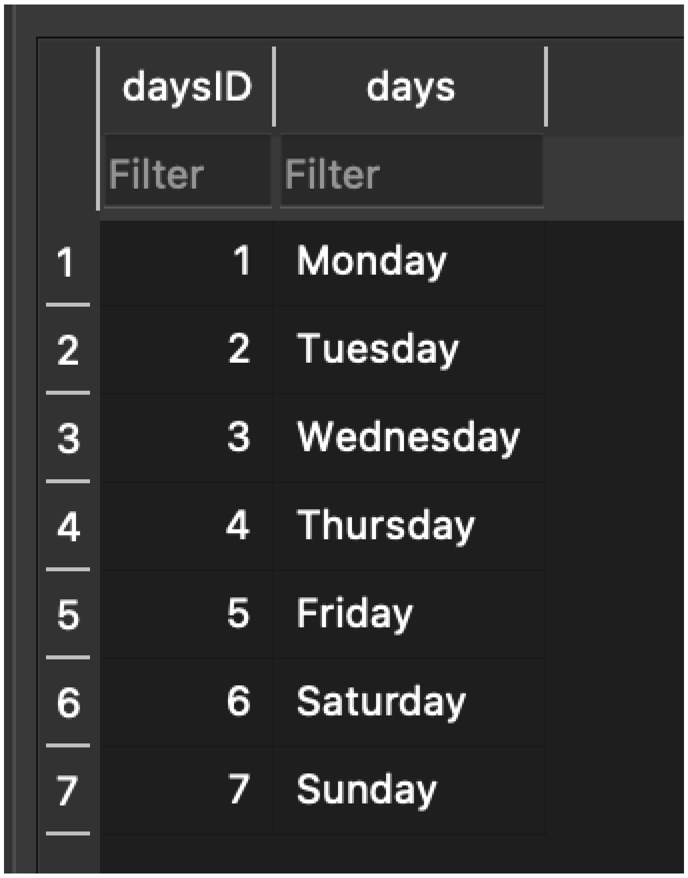
CREATE TABLE "Days" (

"daysID" INTEGER NOT NULL UNIQUE,

"days" TEXT NOT NULL,

PRIMARY KEY("daysID" AUTOINCREMENT)

)

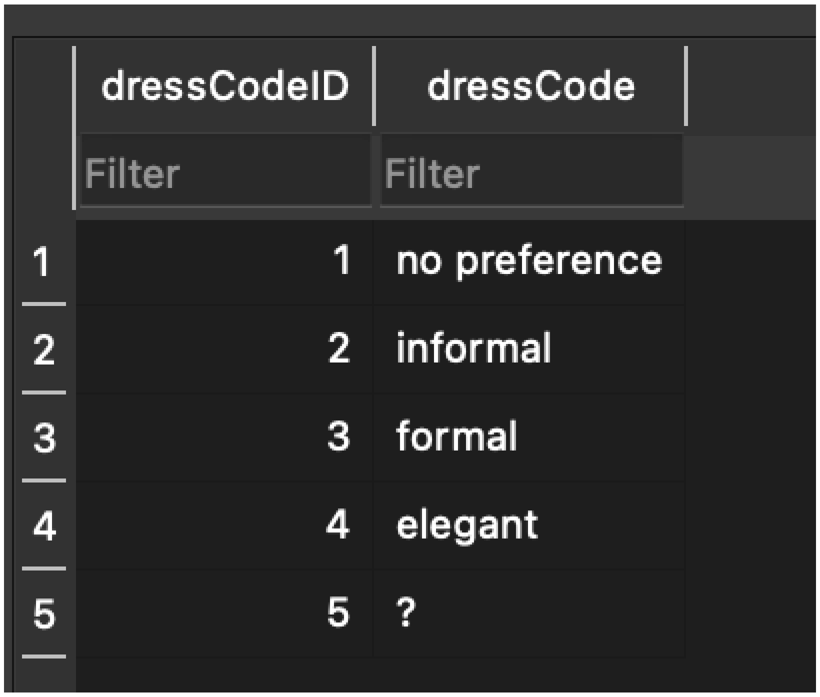


	daysID	days
	Filter	Filter
1	1	Monday
2	2	Tuesday
3	3	Wednesday
4	4	Thursday
5	5	Friday
6	6	Saturday
7	7	Sunday

DressCode:

CREATE TABLE "DressCode" (

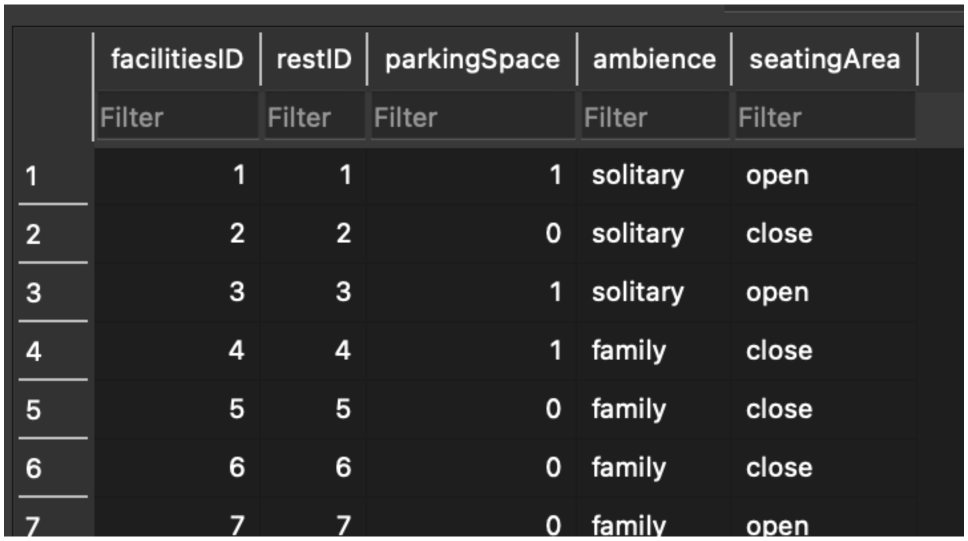
```
"dressCodeID"   INTEGER NOT NULL UNIQUE,
"dressCode"     TEXT NOT NULL,
PRIMARY KEY("dressCodeID" AUTOINCREMENT)
)
```



	dressCodeID	dressCode
1	1	no preference
2	2	informal
3	3	formal
4	4	elegant
5	5	?

Facilities:

```
CREATE TABLE "Facilities" (
"facilitiesID"    INTEGER NOT NULL UNIQUE,
"restID"INTEGER NOT NULL,
"parkingSpace"  INTEGER NOT NULL,
"ambience"      TEXT NOT NULL,
"seatingArea"   TEXT NOT NULL,
PRIMARY KEY("facilitiesID" AUTOINCREMENT),
FOREIGN KEY("restID") REFERENCES "Restaurant"("restID")
)
```



	facilitiesID	restID	parkingSpace	ambience	seatingArea
	Filter	Filter	Filter	Filter	Filter
1	1	1	1	solitary	open
2	2	2	0	solitary	close
3	3	3	1	solitary	open
4	4	4	1	family	close
5	5	5	0	family	close
6	6	6	0	family	close
7	7	7	0	family	open

PaymentsMethods:

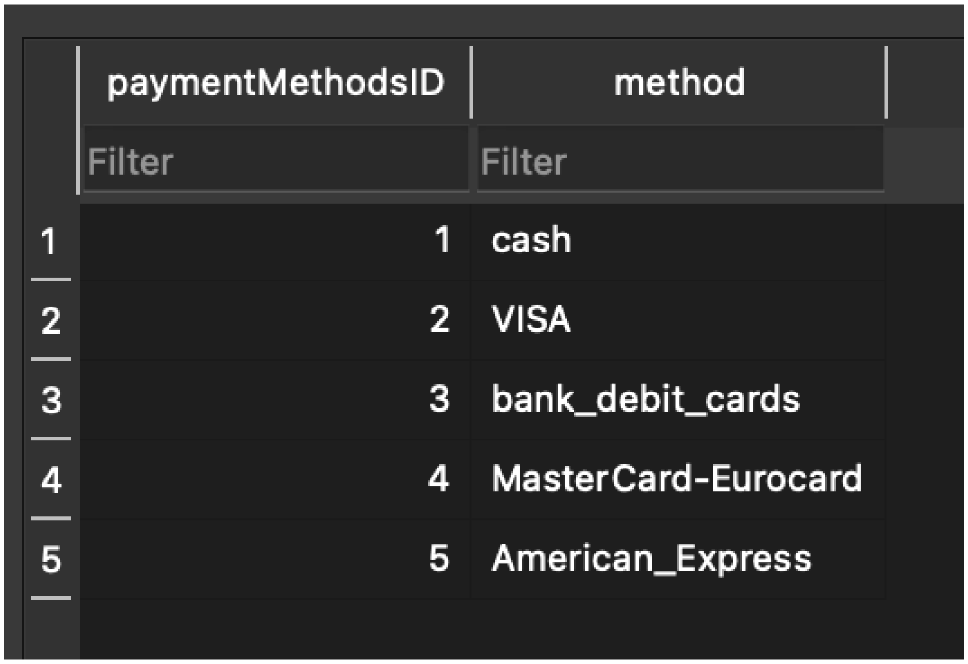
CREATE TABLE "PaymentMethods" (

"paymentMethodsID"   INTEGER NOT NULL UNIQUE,

"method"   BLOB NOT NULL,

PRIMARY KEY("paymentMethodsID" AUTOINCREMENT)

)

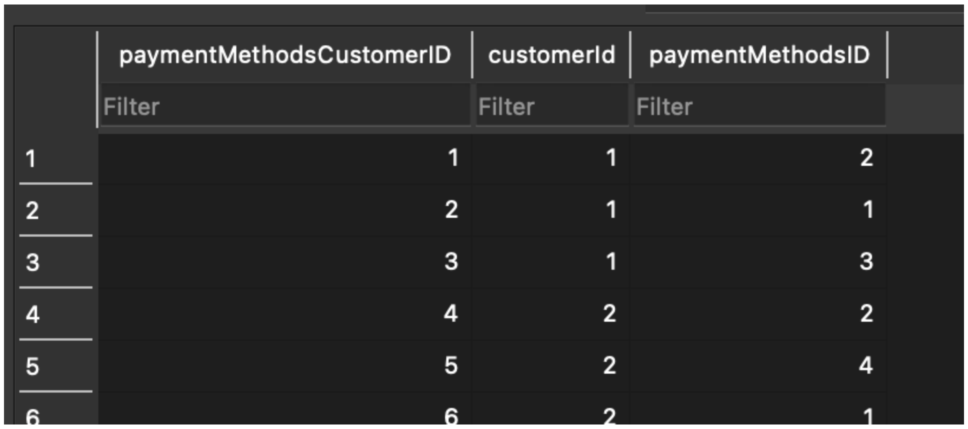


	paymentMethodsID	method
	Filter	Filter
1	1	cash
2	2	VISA
3	3	bank_debit_cards
4	4	MasterCard-Eurocard
5	5	American_Express

PaymentMethodsCustomer:

CREATE TABLE "PaymentMethodsCustomer" (

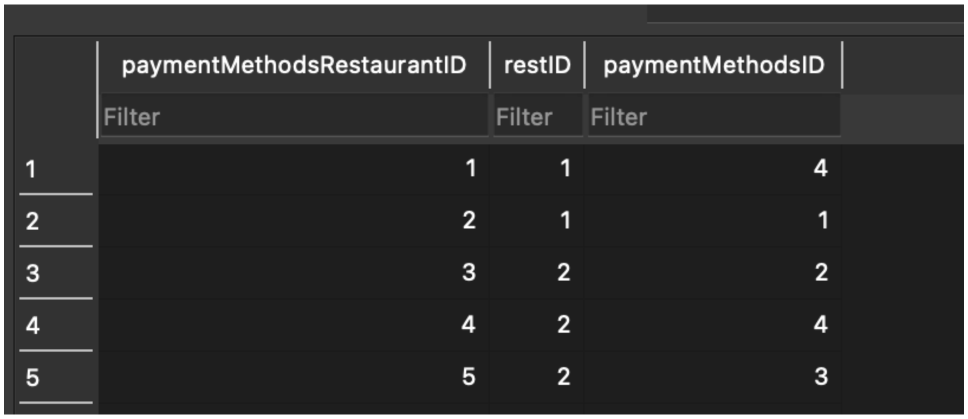
```
"paymentMethodsCustomerID" INTEGER NOT NULL UNIQUE,
"customerId"     INTEGER NOT NULL,
"paymentMethodsID"    INTEGER NOT NULL,
PRIMARY KEY("paymentMethodsCustomerID" AUTOINCREMENT),
FOREIGN KEY("paymentMethodsID") REFERENCES "PaymentMethods"("paymentMethodsID"),
FOREIGN KEY("customerId") REFERENCES "Customer"("customerID")
);
```



	paymentMethodsCustomerID	customerId	paymentMethodsID
	Filter	Filter	Filter
1	1	1	2
2	2	1	1
3	3	1	3
4	4	2	2
5	5	2	4
6	6	2	1

PaymentMethodsRestaurant:

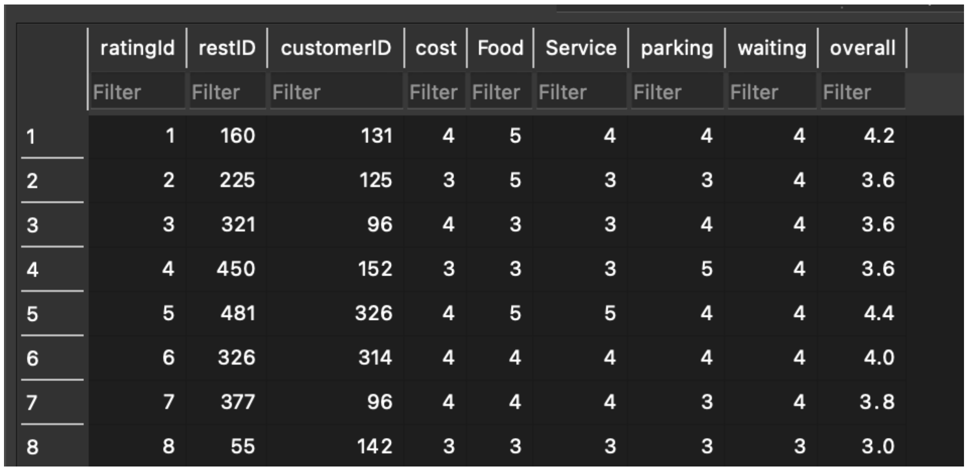
```
CREATE TABLE "PaymentMethodsRestaurant" (
"paymentMethodsRestaurantID"            INTEGER NOT NULL UNIQUE,
"restID"INTEGER NOT NULL,
"paymentMethodsID"    INTEGER NOT NULL,
PRIMARY KEY("paymentMethodsRestaurantID" AUTOINCREMENT),
FOREIGN KEY("restID") REFERENCES "Restaurant"("restID"),
FOREIGN KEY("paymentMethodsID") REFERENCES "PaymentMethods"("paymentMethodsID")
);
```



	paymentMethodsRestaurantID	restID	paymentMethodsID
	Filter	Filter	Filter
1	1	1	4
2	2	1	1
3	3	2	2
4	4	2	4
5	5	2	3

Rating:

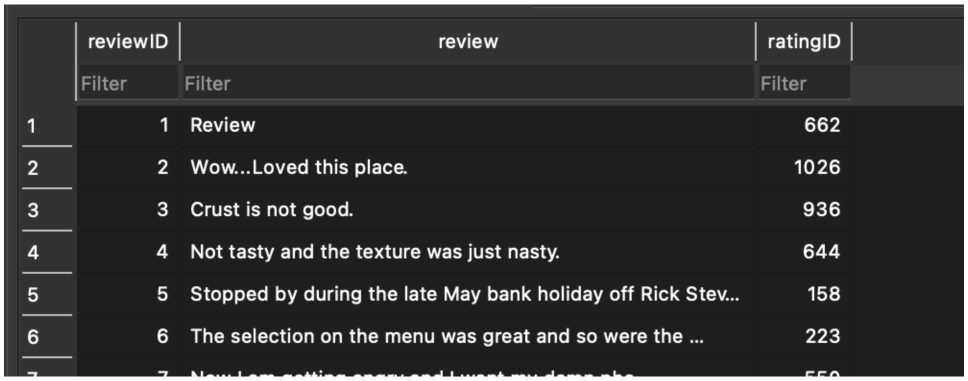
```
CREATE TABLE "Rating" (
"ratingId"      INTEGER NOT NULL UNIQUE,
"restID"INTEGER NOT NULL,
"customerID"    INTEGER NOT NULL,
"cost"  INTEGER NOT NULL,
"Food"  INTEGER NOT NULL,
"Service"       INTEGER NOT NULL,
"parking"       INTEGER NOT NULL,
"waiting"       INTEGER NOT NULL,
"overall"       REAL NOT NULL,
PRIMARY KEY("ratingId" AUTOINCREMENT),
FOREIGN KEY("restID") REFERENCES "Restaurant"("restID"),
FOREIGN KEY("customerID") REFERENCES "Customer"("customerID")
);
```



	ratingId	restID	customerID	cost	Food	Service	parking	waiting	overall
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	160	131	4	5	4	4	4	4.2
2	2	225	125	3	5	3	3	4	3.6
3	3	321	96	4	3	3	4	4	3.6
4	4	450	152	3	3	3	5	4	3.6
5	5	481	326	4	5	5	4	4	4.4
6	6	326	314	4	4	4	4	4	4.0
7	7	377	96	4	4	4	3	4	3.8
8	8	55	142	3	3	3	3	3	3.0

Review:

```
CREATE TABLE "Review" (
"reviewID"      INTEGER NOT NULL UNIQUE,
"review"        TEXT NOT NULL,
"ratingID"      INTEGER NOT NULL,
PRIMARY KEY("reviewID" AUTOINCREMENT),
FOREIGN KEY("ratingID") REFERENCES "Rating"("ratingId")
);
```



	reviewID	review	ratingID
	Filter	Filter	Filter
1	1	Review	662
2	2	Wow...Loved this place.	1026
3	3	Crust is not good.	936
4	4	Not tasty and the texture was just nasty.	644
5	5	Stopped by during the late May bank holiday off Rick Stev...	158
6	6	The selection on the menu was great and so were the ...	223
7	7	Now I am getting angry and I want my damn pho	550

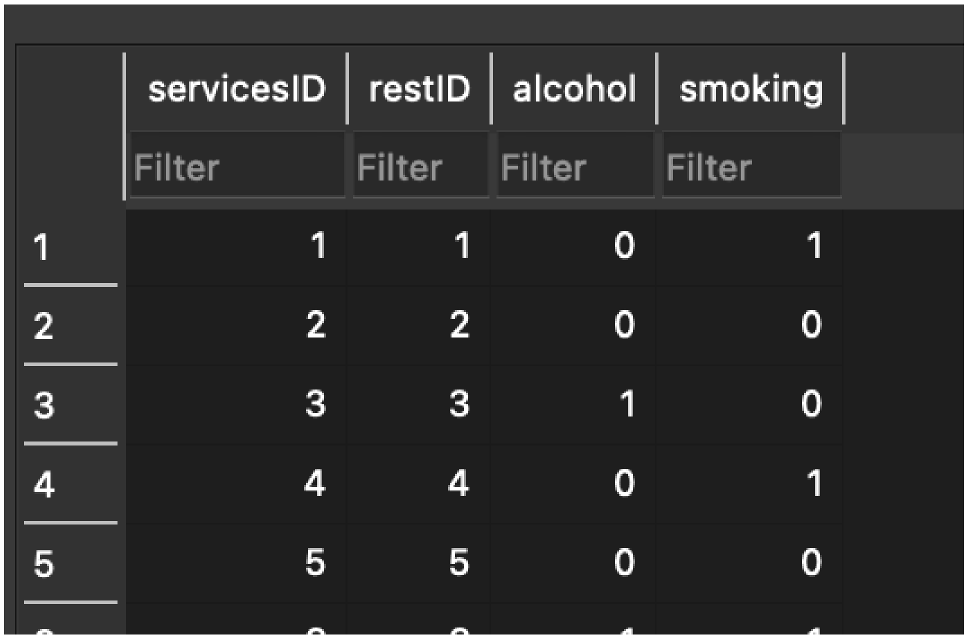
Services:

```
CREATE TABLE "Services" (
"servicesID"    INTEGER NOT NULL UNIQUE,
"restID"INTEGER NOT NULL,
"alcohol"       INTEGER NOT NULL,
"smoking"       INTEGER NOT NULL,
```

PRIMARY KEY("servicesID" AUTOINCREMENT),

FOREIGN KEY("restID") REFERENCES "Restaurant"("restID")

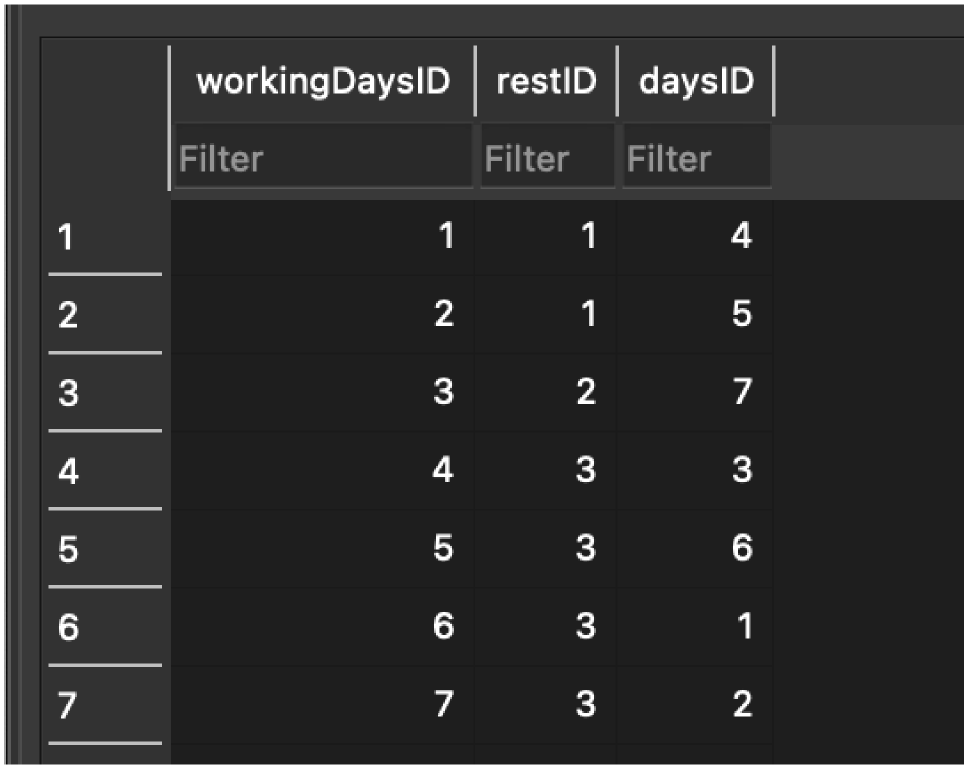
);



	servicesID	restID	alcohol	smoking
	Filter	Filter	Filter	Filter
1	1	1	0	1
2	2	2	0	0
3	3	3	1	0
4	4	4	0	1
5	5	5	0	0

WorkingDays:

```
CREATE TABLE "WorkingDays" (
"workingDaysID"          INTEGER NOT NULL UNIQUE,
"restID"INTEGER NOT NULL,
"daysID"          INTEGER NOT NULL,
PRIMARY KEY("workingDaysID" AUTOINCREMENT),
FOREIGN KEY("daysID") REFERENCES "Days"("daysID"),
FOREIGN KEY("restID") REFERENCES "Restaurant"("restID")
);
```



	workingDaysID	restID	daysID
	Filter	Filter	Filter
1	1	1	4
2	2	1	5
3	3	2	7
4	4	3	3
5	5	3	6
6	6	3	1
7	7	3	2

Define and execute at least five queries that show your database

1. **JOIN OF 3 TABLES**

```
select R.restID,R.name,R.address,D.days from Restaurant R
inner join WorkingDays W on R.restID = W.restID
inner join Days D on D.daysID = W.daysID
where D.days = "Sunday" or D.days=  "Saturday"
```

```
SELECT Customer.name, Customer.customerId, Customer.ambience,Rating.overall
from Customer
INNER JOIN Rating on Rating.customerID = Customer.customerId
WHERE Customer.ambience = 'friends'
```

2. **SUBQUERY**

```
select restID,name,address,city,state,zip from Restaurant where restID in (
select restID from CuisineRestaurant where cuisineId in(
select cuisineId from cuisine where cuisine like "%Chinese%")
)
```

```
SELECT Customer.name, Customer.customerId, Customer.ambience
from Customer WHERE Customer.customerId in (
SELECT customerId from PaymentMethodsCustomer WHERE customerId in (
SELECT customerId from PaymentMethods WHERE PaymentMethods.method like
"%VISA%")
)
```

3. **GROUP BY WITH A HAVING CLAUSE**

```
select restID,name,address,priceRangeMax from Restaurant R
group by priceRangeMax
having priceRangeMax < 100

SELECT Customer.name, Customer.customerId, Rating.overall
from Customer
INNER JOIN Rating on Rating.customerID = Customer.customerId
GROUP by Customer.customerID
HAVING Rating.overall > 4
```

4. COMPLEX SEARCH CRITERION

```
select R.restID,R.name,R.address from Restaurant R
inner join Facilities F on R.restID = F.restID
inner join Services S on R.restID = S.restID
inner join WorkingDays W on R.restID = W.restID
inner join days D on W.daysID   = D.daysID
WHERE D.days = 'Saturday'
AND  F.seatingArea = "open"
AND (S.smoking = 1 or  S.alcohol = 1)
AND  F.parkingSpace = 1

select C.customerId,C.name,C.budget , cuisine.cuisine, PaymentMethods.method
from Customer C
inner join CuisineCustomer CC on CC.customerId =C.customerId
inner join cuisine  on cuisine.cuisineId =CC.cuisineId
```

inner join PaymentMethodsCustomer PC on PC.customerId =C.customerId

inner join PaymentMethods on PaymentMethods.paymentMethodsID =PC.paymentMethodsID

WHERE PaymentMethods.method = 'VISA'

AND (cuisine.cuisine = 'Italian' or cuisine.cuisine = 'French')

AND C.budget != '?'

5. **SELECT CASE/WHEN.**

```
SELECT name, zip, restID , country, state from Restaurant

ORDER BY (CASE WHEN state IS NULL THEN country ELSE state END);


SELECT Customer.customerId , Customer.name,

CASE

WHEN Rating.overall > 4.5 THEN "good food"

WHEN Rating.overall > 4 THEN "avarage food"

WHEN Rating.overall > 3 THEN "normal food"

ELSE 'ok food'

END AS overall

from Rating

INNER JOIN Customer on Customer.customerId=Rating.customerId
```