DATABASE SPECIFICATIONS

Next-Gen Restaurant Application

Raditya Fahritama
rkf5230@psu.edu

School of Graduate Professional Studies
Information Science Department
IN SC 521 - Introduction to Database Concepts

Fall II, 2021

DOCUMENT CONTROL

Work carried out by:

Name	Email Address	Other
Raditya Fahritama	**rkf5230@psu.edu**	radityafahritama@gmail.com

Revision Sheet

Release No.	Date	Revision Description

DATABASE SPECIFICATIONS

TABLE OF CONTENTS

MILESTONE 1: DATA REQUIREMENTS

System Name or Title

Next-Gen Restaurant Application

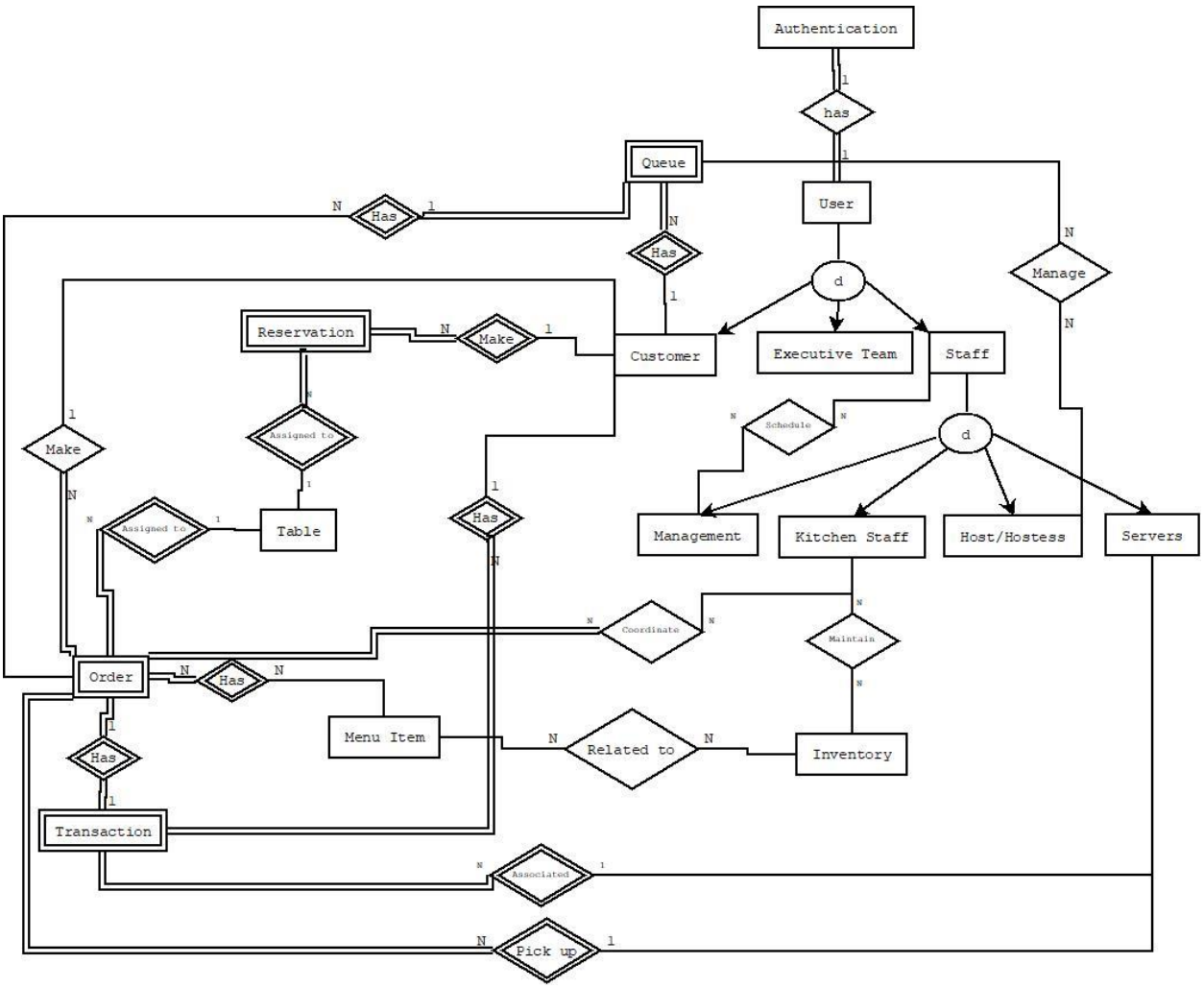
Core requirements

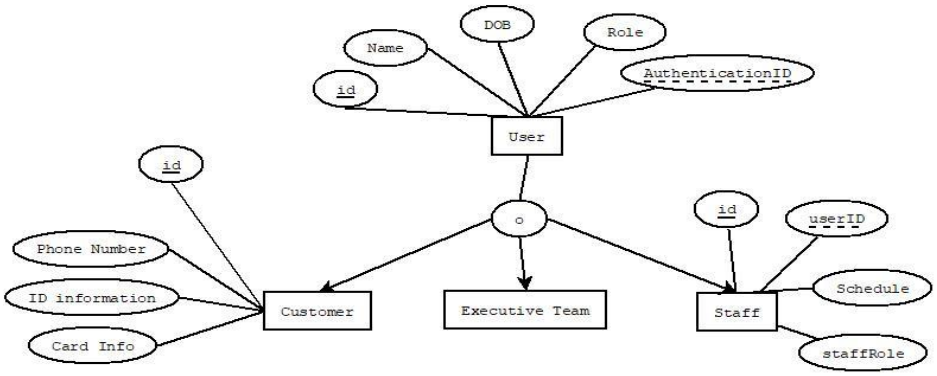
No.	Requirement	Referenced page in SRS	Referenced Section in SRS	Referenced Paragraph in Section
1	The system should store information about the customers	3	1.2	1
2	The system should store information about the order of every customer	3	1.2	2
3	The system should store information about reservation that the customer made	5 12	2.1 3.5.6	3 1
4	The system should store information of the items on the menu.	19 4	8.1.1 1.2	1 2
5	The system should store information about the restaurant's table	39 54	8.1.20 9.5	1 1
6	The system should store information about Restaurant's staffs	5	2.1	3
7	The system should store information of the user of the application.	6	2.3	1
8	The details of the customers: phone number, ID information, credit card info.	5 10 58	2.2 3.5.3 9.9	1 2 1
9	The details of the Order: Customer ID, table assigned ID, Menu Item ID, ServerID, time of customer visit.	5 9	2.2 3.5	1 1
10	The details of the reservation: receipts related to the reservation, reservation details, wait queue, Table Assigned ID, CustomerID	12	3.5.6	1
11	The details of items on the menu: Item name, item Category, item popularity, item stock	19 5	8.1.1 2.1	1 3

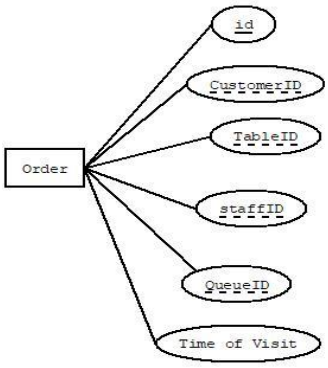
12	Item will have label whether they contain alcohol or not	8	3.1	1
13	Order Payment use three forms of payment: cash, credit card, gift card	9	3.5.1	9
14	The detail of restaurant's table: table number, table seat type, capacity, table shape, table shape size, Occupancy	39 54	8.1.20 9.5	1 1
15	Table seat is divided to bar and casual dining	39	8.1.20	1
16	The detail about staff: staff schedule	5	2.1	3
17	The detail of User: user age, user role, authentication, user DOB, user Name	6	2.3	1
18	The user roles: Executive team, Restaurant Management, Servers, Host/Hostess, Kitchen Staff, Customer	6	2.3	2
19	The Database should store information about kitchen inventory	7	2.3	2
20	The detail of inventory: Ingredients.	7	2.3	2
21	The database should store information about transaction receipts	5	2.2	4
22	The detail of transaction: Order ID, Payment type. Total. Gratuities, ServerID	5 9	2.2 3.5	4 1
23.	The database should store information about Wait queue	10	3.5.3	1
24	The detail of Wait Queue: Customer Name, Party Size, Customer Number	10	3.5.3	2

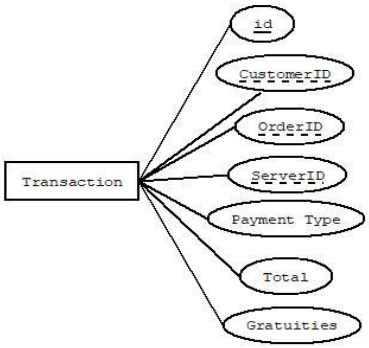
MILESTONE 2: CONCEPTUAL DESIGN

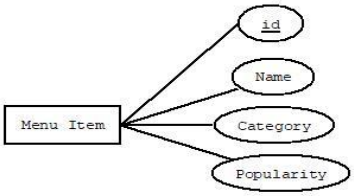
Diagram

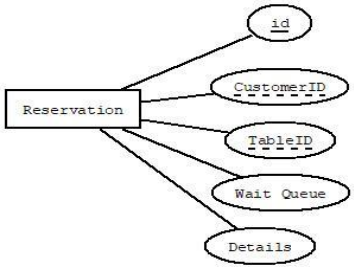


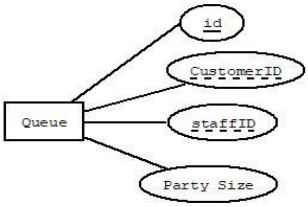

DOB
Role
Name
AuthenticationID
id
User
id
o
id
userID
Phone Number
ID information
Card Info
Customer
Executive Team
Staff
Schedule
staffRole

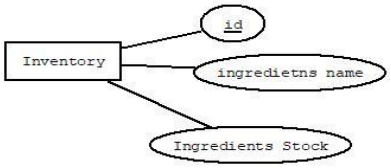

id
CustomerID
TableID
Order
staffID
QueueID
Time of Visit

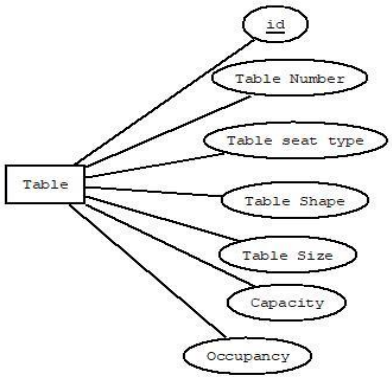

id
CustomerID
OrderID
Transaction
ServerID
Payment Type
Total
Gratuities

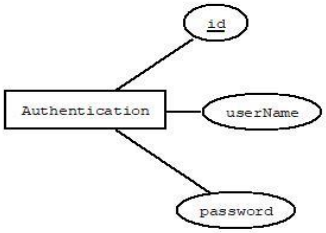

id
Name
Menu Item
Category
Popularity


id
CustomerID
Reservation
TableID
Wait Queue
Details


id
CustomerID
Queue
staffID
Party Size


id
Inventory
ingredietns name
Ingredients Stock


id
Table Number
Table seat type
Table
Table Shape
Table Size
Capacity
Occupancy

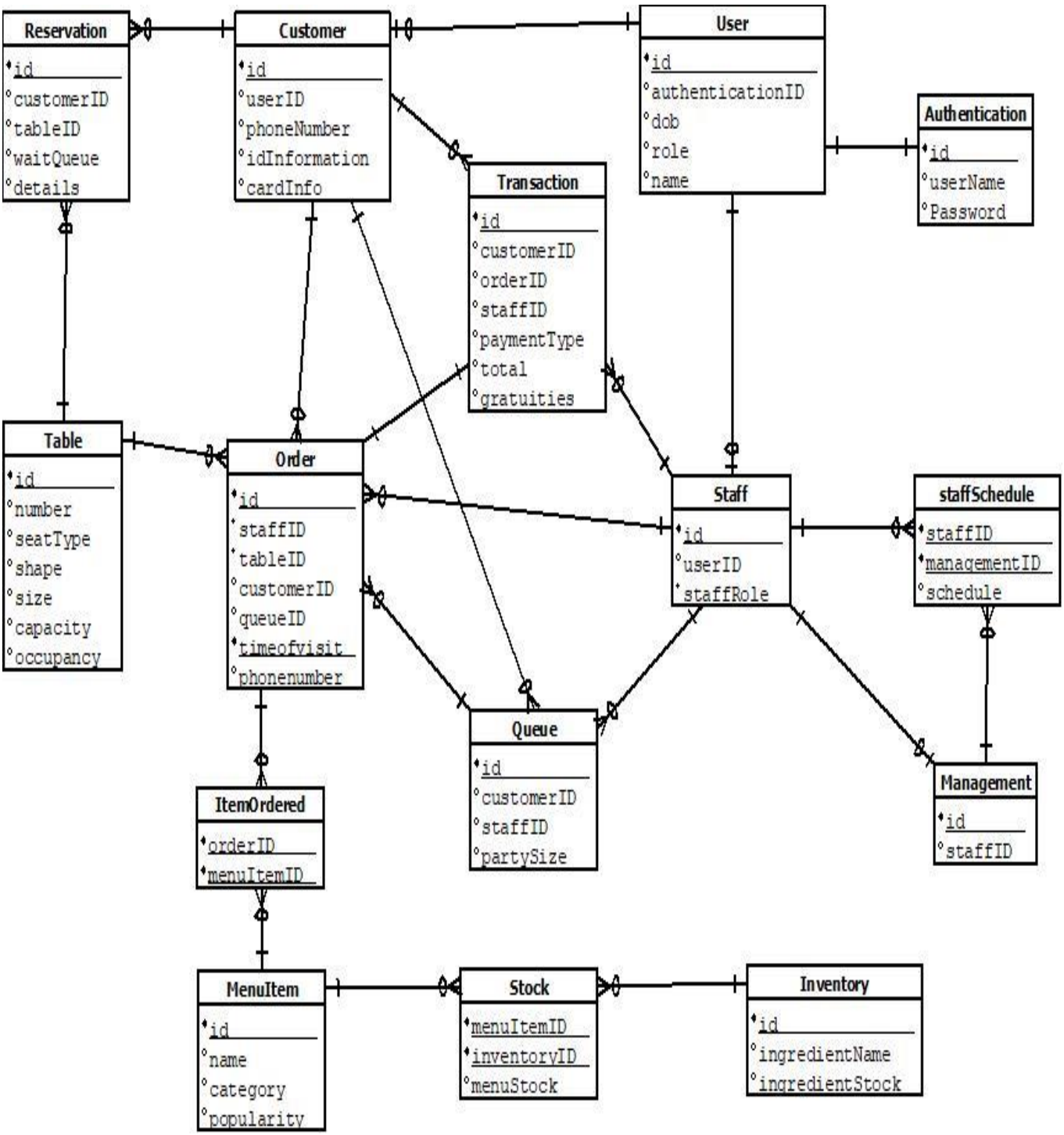

id
Authentication
userName
password

Assumptions and Constraints

-Age is derived from DOB.
-Customer's phone number is multivalued.
-Order contains ID from Customer, Table, Menu Item, Server.
-Transaction contains ID from Order, Server.
-Reservation contains ID from Customer, Table.
-Queue contains ID from Order, Customer

MILESTONE 3: LOGICAL DESIGN

Entity Relationship Diagram



Entity name: User

Attributes:

userID, name, DOB, role, authenticationDetails,

Functional dependencies:

userID -> name, DOB, role, authenticationDetails

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	userID		Name, DOB, role, authenticationDetails

Attribute closures (if any):
userID+ = userID, name, DOB, role, authenticationDetails

(userID) is a super key

Unique keys: the key for this table is/are
userID

Entity name: Table

Attributes:

tableID, number, seatType, shape, size, capacity, occupancy

Functional dependencies:

tableID -> number, seatType, shape, size, capacity, occupancy

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	tableID		number, seatType, shape, size, capacity, occupancy

Attribute closures (if any):
tableID + = number, seatType, shape, size, capacity, occupancy
(tableID) is a super key

Unique keys: the key for this table is/are
tableID

Entity name: Staff

Attributes:

staffID, schedule

Functional dependencies:

staffID -> schedule

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	staffID		schedule

Attribute closures (if any):
staffID+ = staffID, schedule
(staffID) is a super key

Unique keys: the key for this table is/are
staffID

Entity name: Inventory

Attributes:

inventoryID, ingredientsName, ingredientsStock

Functional dependencies:

inventoryID -> ingredientsName, ingredientsStock

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	inventoryID		ingredientsName, ingredientsStock

Attribute closures (if any):
inventoryID+ = inventoryID, ingredientsName, ingredientsStock
(inventoryID) is a super key

Unique keys: the key for this table is/are
inventoryID

Entity name: MenuItem

Attributes:

menuItemID, name, category, popularity, stock

Functional dependencies:

menuItemID -> name, category, popularity, stock

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	menuItemID		name, category, popularity, stock

Attribute closures (if any):
menuItemID+ = menuItemID, name, category, popularity, stock
(menuItemID) is a super key

Unique keys: the key for this table is/are
menuItemID

Entity name: Staff

Attributes:

staffID, schedule

Functional dependencies:

staffID -> schedule

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	staffID		schedule

Attribute closures (if any):
staffID+ = staffID, schedule
(staffID) is a super key

Unique keys: the key for this table is/are
staffID

Entity name: Order

Attributes:

orderID, customerID, tableID, menuItemID, serverID, queueID, timeOfVisit

Functional dependencies:

orderID -> customerID, tableID, menuItemID, serverID, queueID, timeOfVisit

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	orderID		customerID, tableID, menuItemID, serverID, queueID, timeOfVisit

Attribute closures (if any):
orderID+ = orderID, customerID, tableID, menuItemID, serverID, queueID, timeOfVisit
(orderID) is a super key

Unique keys: the key for this table is/are
orderID

Entity name: Queue

Attributes:

queueID, customerID, customerName, partySize, customerNumber

Functional dependencies:

queueID -> customerID, customerName, partySize, customerNumber

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	queueID		customerID, customerName, partySize, customerNumber

Attribute closures (if any):
queueID+ = queueID, customerID, customerName, partySize, customerNumber
(queueID) is a super key

Unique keys: the key for this table is/are
queueID

Entity name: Transaction

Attributes:

transactionID, orderID, serverID, paymentType, total, gratuities

Functional dependencies:

transactionID -> orderID, serverID, paymentType, total, gratuities

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	transactionID		orderID, serverID, paymentType, total, gratuities

Attribute closures (if any):
transactionID+ = transactionID, orderID, serverID, paymentType, total, gratuities
(transactionID) is a super key

Unique keys: the key for this table is/are
transactionID

Entity name: Reservation

Attributes:

reservationID,

Functional dependencies:

reservationID -> customerID, tableID, waitQueue, details

Attributes not in FD	Attributes on the left	Attributes on both sides	Attributes on the right side
	reservationID		customerID, tableID, waitQueue, details

Attribute closures (if any):
reservationID+ = reservationID, customerID, tableID, waitQueue, details
(reservationID) is a super key

Unique keys: the key for this table is/are
reservationID

Assumptions and Constraints

-Order contains ID from Customer, Table, Menu Item, Server.
-Transaction contains ID from Order, Server.
-Reservation contains ID from Customer, Table.
-Queue contains ID from Order, Customer

MILESTONE 4: NORMALIZATION AND

MILESTONE 5: PHYSICAL DESIGN

Assumptions and Constraints

-Order contains ID from Customer, Table, Menu Item, Server.
-Transaction contains ID from Order, Server.
-Reservation contains ID from Customer, Table.
-Queue contains ID from Order, Customer

Naming Conventions

Discuss the naming standards and conventions that you have used for table creation.

Tables

	Name of the table	***User***			
	Description	User of the application			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	userID	Id of user	Integer	Between 1 and 999999	
	authenticationID	Id of related authentication detail	Integer	Between 1 and 999999	
	dob	Date of birth of user	date	1/1/1999	Must be bigger than 1/1/1900
	role	Role of the user	Varchar(50)	CUSTOMER	3 types of role: customer, executive team, staff
	Name	name of the user	Varchar(50)	Andrew	
	Functional Dependencies and Keys				
	Functional dependencies	userID -> authenticationID userID -> dob userID -> role userID -> name authenticationID -> username, password			
	Candidate keys	userID, authenticationID			
	Normalization				
	1NF	**Yes**	All cells contain a unique value		
	2NF	**Yes**	The key of the table is a single attribute		
	3NF	**Yes**	All the non-key attributes depend only on a key		

	BCNF	**Yes**	All the attributes depend only on a key
	Physical Design		
	Primary Key	userID	
	Foreign Keys	authenticationID	
	SQL Code	CREATE TABLE P_USER(USERID INTEGER, AUTHENTICATIONID INTEGER, USERDOB DATE, USERROLE VARCHAR(50) NOT NULL, USERNAME VARCHAR(50) NOT NULL, CONSTRAINT P_USER PRIMARY KEY (USERID), FOREIGN KEY (AUTHENTICATIONID) REFERENCES "P_AUTHENTICATION"(AUTHENTICATIONID), CHECK (USERROLE = 'CUSTOMER' OR USERROLE = 'EXECUTIVE TEAM' OR USERROLE = 'STAFF'), CHECK (USERDOB > TO_DATE('1900/1/1', 'YYYY/MM/DD')));	
	Count of records in the table	**6**	

...

	Name of the table	***Authentication***			
	Description	Contains username and password of user			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	authenticationID	Id of authentication	integer	Between 1 and 999999	
	username	Username of user	Varchar(20)	Asdfg1236	Cannot be null
	password	Password of user	Varchar(20)	Testpassword7	Cannot be null
	Functional Dependencies and Keys				
	Functional dependencies	authenticationID -> username authenticationID -> password			
	Candidate keys	authenticationID			
	Normalization				
	1NF	**Yes**	All cells contain a unique value		
	2NF	**Yes**	The key of the table is a single attribute		
	3NF	**Yes**	All the non-key attributes depend only on a key		
	BCNF	**Yes**	All the attributes depend only on a key		
	Physical Design				
	Primary Key	authenticationID			
	Foreign Keys	**-**			

SQL Code	CREATE TABLE P_AUTHENTICATION(AUTHENTICATIONID INTEGER, USERNAME VARCHAR(50) NOT NULL, PASSWORD VARCHAR(50) NOT NULL, CONSTRAINT P_AUTHENCTICATION PRIMARY KEY(AUTHENTICATIONID));
Count of records in the table	**6**

	Name of the table	***Customer***			
	Description	The customer of the restaurant. Also the user of the application			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	customerID	Id of the customer	integer	Between 1 and 999999	
	userID	Id of related user and customer	integer	Between 1 and 999999	
	phoneNumber	Number of the customer	Char(10)	5451231234	
	idInformation	Id number of the customer	Char(12)	090909090808 0808	
	cardInfo	Card number of the customer	Char(12)	010102020303 0404	
	Functional Dependencies and Keys				
	Functional dependencies	customerID -> userID customerID -> phoneNumber customerID -> idInformation customerID -> cardInfo userID -> authenticationID, dob, role, name authenticationID -> username, password			
	Candidate keys	customerID, userID, authenticationID			
	Normalization				
	1NF	**Yes**	All cells contain a unique value		
	2NF	**Yes**	The key of the table is a single attribute		

	3NF	**Yes**	All the non-key attributes depend only on a key
	BCNF	**Yes**	All the attributes depend only on a key
	Physical Design		
	Primary Key	customerID	
	Foreign Keys	userID	
	SQL Code	CREATE TABLE P_CUSTOMER(CUSTOMERID INTEGER, USERID INTEGER, PHONENUMBER CHAR(10), IDINFORMATION CHAR(12), CARDINFO CHAR(12), CONSTRAINT P_CUSTOMER PRIMARY KEY (CUSTOMERID), FOREIGN KEY (USERID) REFERENCES "P_USER"(USERID), CHECK(PHONENUMBER NOT LIKE '%[^0-9]%'), CHECK(IDINFORMATION NOT LIKE '%[^0-9]%'), CHECK(CARDINFO NOT LIKE '%[^0-9]%'));	
	Count of records in the table	**2**	

	Name of the table	***Staff***			
	Description	The staff of the restaurant. Also the user of the application.			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	staffID	Id of staff	integer	Between 1 and 999999	
	userID	Id of the related user and staff	integer	Between 1 and 999999	
	staffRole	Role of the staff	Varchar(20)	Host	4 types of roles: manager, kitchen staff, host/hostess, servers
	Functional Dependencies and Keys				
	Functional dependencies	staffID -> userID staffID -> staffRole userID -> authenticationID, dob, role, name authenticationID -> username, password			
	Candidate keys	staffID, userID, authenticationID			
	Normalization				
	1NF	**Yes**	All cells contain a unique value		
	2NF	**Yes**	The key of the table is a single attribute		
	3NF	**Yes**	All the non-key attributes depend only on a key		
	BCNF	**Yes**	All the attributes depend only on a key		

	Physical Design	
	Primary Key	staffID
	Foreign Keys	userID
	SQL Code	CREATE TABLE P_STAFF(STAFFID INTEGER, USERID INTEGER, STAFFROLE VARCHAR(20), CONSTRAINT P_STAFF PRIMARY KEY(STAFFID), FOREIGN KEY (USERID) REFERENCES "P_USER"(USERID), CHECK(STAFFROLE = 'MANAGER' OR STAFFROLE = 'KITCHEN STAFF' OR STAFFROLE = 'HOST/HOSTESS' OR STAFFROLE = 'SERVER'));
	Count of records in the table	**4**

	Name of the table	***Queue***			
	Description	Queue of the restaurant. Customer has to be assigned to queue before entering restaurant.			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	queueID	Id of the queue	integer	Between 1 and 999999	
	customerID	Id of related customer and queue	integer	Between 1 and 999999	
	staffID	Id of related staff and queue	integer	Between 1 and 999999	
	partySize	How much people in one queue	integer	Between 1 and 100	Cannot be null
	Functional Dependencies and Keys				
	Functional dependencies	queueID-> customerID queueID -> staffID queueID -> partySize customerID-> userID, phoneNumber,idInformation,cardInfo staffID->userID, staffRole userID -> authenticationID, dob, role, name			
	Candidate keys	queueID, customerID, staffID, userID			
	Normalization				
	1NF	**Yes**	All cells contain a unique value		
	2NF	**Yes**	The key of the table is a single attribute		
	3NF	**Yes**	All the non-key attributes depend only on a key		
	BCNF	**Yes**	All the attributes depend only on a key		
	Physical Design				
	Primary Key	queueID			
	Foreign Keys	customerID, staffID			

SQL Code	CREATE TABLE P_QUEUE(QUEUEID INTEGER, CUSTOMERID INTEGER, STAFFID INTEGER, PARTYSIZE INTEGER NOT NULL, CONSTRAINT P_QUEUE PRIMARY KEY(QUEUEID), FOREIGN KEY (CUSTOMERID) REFERENCES "P_CUSTOMER"(CUSTOMERID), FOREIGN KEY (STAFFID) REFERENCES "P_STAFF"(STAFFID));
Count of records in the table	**2**

	Name of the table	***Reservation***			
	Description	Reservation to the restaurant. Reservation that is made by customer will be recorded here.			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	reservationID	Id of the reservation	integer	Between 1 and 999999	
	customerID	Id of the related customer and reservation	integer	Between 1 and 999999	
	tableID	Id of related table and reservation	integer	Between 1 and 999999	
	waitQueue	Queue position of the reservation	integer	Between 1 and 100	Cannot be null
	details	Reservation details	Varchar(50)	Reservation of mr.john	
	Functional Dependencies and Keys				
	Functional dependencies	reservationID-> customerID reservationID-> tableID reservationID-> waitQueue reservationID-> details customerID-> userID, phoneNumber,idInformation,cardInfo tableID->number, seatType, shape, size, capacity, occupancy userID -> authenticationID, dob, role, name			
	Candidate keys	**reservationID, customerID, tableID, userID**			
	Normalization				
	1NF	**Yes**	All cells contain a unique value		
	2NF	**Yes**	The key of the table is a single attribute		
	3NF	**Yes**	All the non-key attributes depend only on a key		
	BCNF	**Yes**	All the attributes depend only on a key		
	Physical Design				
	Primary Key	reservationID			
	Foreign Keys	customerID, tableID			

SQL Code	CREATE TABLE P_RESERVATION(RESERVATIONID INTEGER, CUSTOMERID INTEGER, TABLEID INTEGER, WAITQUEUE INTEGER NOT NULL, DETAILS VARCHAR(50), CONSTRAINT P_RESERVATION PRIMARY KEY (RESERVATIONID), FOREIGN KEY (CUSTOMERID) REFERENCES "P_CUSTOMER"(CUSTOMERID), FOREIGN KEY (TABLEID) REFERENCES "P_TABLE"(TABLEID));
Count of records in the table	**2**

	Name of the table	***Table***			
	Description	Record of tables of the restaurant. Provides details of the tables.			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	tableID	Id of the table	integer	Between 1 and 999999	
	Number	Number of the table	Char(2)	Numeric from 01 to 99	
	seatType	Seat type of the table	char(6)	CASUAL	2 types. Casual dining and Bar.
	Shape	The table shape	Varchar(10)	ROUND	2 types. Square and round.
	Size	Size of the table	Varchar(10)	LARGE	3 types. Large, medium, small.
	Capacity	Capacity that the table can handle	integer	Between 1 and 99	
	Occupancy	Table occupancy status	Char(1)	N	‘O’ for occupied. ‘N’ for not occupied.
	Functional Dependencies and Keys				
	Functional dependencies	tableID->number tableID->seatType tableID->shape tableID->size tableID->capacity tableID->occupancy			
	Candidate keys	tableID			
	Normalization				
	1NF	**Yes**	All cells contain a unique value		
	2NF	**Yes**	The key of the table is a single attribute		

	3NF	**Yes**	All the non-key attributes depend only on a key
	BCNF	**Yes**	All the attributes depend only on a key
	Physical Design		
	Primary Key	tableID	
	Foreign Keys	-	
	SQL Code	CREATE TABLE P_TABLE(TABLEID INTEGER, TABLENUMBER CHAR(2), SEATTYPE CHAR(6), SHAPE VARCHAR(10), "SIZE" VARCHAR(10), "CAPACITY" INTEGER, OCCUPANCY CHAR(1), CONSTRAINT P_TABLE PRIMARY KEY (TABLEID), CHECK(TABLENUMBER NOT LIKE '%[^0-9]%'), CHECK(SEATTYPE = 'CASUAL' OR SEATTYPE = 'BAR'), CHECK(SHAPE = 'ROUND' OR SHAPE = 'SQUARE'), CHECK("SIZE" = 'LARGE' OR "SIZE" = 'MEDIUM' OR "SIZE" = 'SMALL'), CHECK(OCCUPANCY = 'N' OR OCCUPANCY = 'O'));	
	Count of records in the table	**2**	

	Name of the table	***Inventory***			
	Description	Record of the kitchen inventory. All of the ingredients are recorded here.			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	inventoryID	Id of the inventory	Integer	Between 1 and 999999	
	ingredientsName	Name of the ingredients	Varchar(50)	Tomato	
	ingredientsStock	Stock of the ingredients	integer	Between 1 and 9999	
	Functional Dependencies and Keys				
	Functional dependencies	inventoryID->ingredientsName inventoryID-> ingredientsStock			
	Candidate keys	inventoryID			
	Normalization				
	1NF	**Yes**	All cells contain a unique value		
	2NF	**Yes**	The key of the table is a single attribute		
	3NF	**Yes**	All the non-key attributes depend only on a key		
	BCNF	**Yes**	All the attributes depend only on a key		
	Physical Design				

	Primary Key	inventoryID
	Foreign Keys	**-**
	SQL Code	CREATE TABLE P_INVENTORY(INVENTORYID INTEGER, INGREDIENTSNAME VARCHAR(50), INGREDIENTSSTOCK INTEGER, CONSTRAINT P_INVENTORY PRIMARY KEY(INVENTORYID));
	Count of records in the table	**2**

	Name of the table	***MenuItem***			
	Description	The item of the menu that customer can order.			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	menuitemID	Id of the menuitem	Integer	Between 1 and 999999	
	Name	Name of the menu item	Varchar(50)	Fetuccine Alfredo	
	Category	Category of the menu item	Char(20)	ENTREE	Can be appetizer, entrée, dessert, beverages, etc.
	Popularity	Popularity rating of item	integer	Between 0 and 100	
	Functional Dependencies and Keys				
	Functional dependencies	menuitemID-> name menuitemID->category menuitemID->popularity			
	Candidate keys	menuitemID			
	Normalization				
	1NF	**Yes**	All cells contain a unique value		
	2NF	**Yes**	The key of the table is a single attribute		
	3NF	**Yes**	All the non-key attributes depend only on a key		
	BCNF	**Yes**	All the attributes depend only on a key		
	Physical Design				
	Primary Key	menuitemID			
	Foreign Keys	**-**			

SQL Code	CREATE TABLE P_MENUITEM(MENUITEMID INTEGER, MENUITEMNAME VARCHAR(50), MENUCATEGORY VARCHAR(50), MENUPOPULARITY INTEGER, CONSTRAINT P_MENUITEM PRIMARY KEY(MENUITEMID), CHECK(MENUCATEGORY = 'APPETIZER' OR MENUCATEGORY = 'ENTREE' OR MENUCATEGORY = 'DESSERT' OR MENUCATEGORY = 'BEVERAGES'), CHECK(MENUPOPULARITY >= 0 AND MENUPOPULARITY <= 100));
Count of records in the table	**2**

	Name of the table	***Order***			
	Description	Order list of the customer.			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	orderID	Id of the order	Integer	Between 1 and 999999	
	staffID	Id of related staff and order	Integer	Between 1 and 999999	
	tableID	Id of related table and order	Integer	Between 1 and 999999	
	queueID	Id of related queue and order	Integer	Between 1 and 999999	
	customerID	Id of related customer and order	Integer	Between 1 and 999999	
	Timeofvisit	Time of the order made	Char(20)	01/01/2021 12:00:00	Cannot be null
	PhoneNumber	Phone Number of customer	Char(10)	8459613457	
	Functional Dependencies and Keys				
	Functional dependencies	orderID -> staffID orderID -> tableID orderID -> queueID orderID -> customerID orderID -> timeofvisit, phonenumber staffID->userID, staffRole tableID->number, seatType, shape, size, capacity, occupancy customerID-> userID, phoneNumber,idInformation,cardInfo queueID-> customerID, staffID, partysize Timeofvisit -> orderID			
	Candidate keys	orderID, staffID, tableID, customerID, queueID, Timeofvisit			
	Normalization				
	1NF	**Yes**	All cells contain a unique value		

	2NF	**Yes**	The key of the table is a single attribute
	3NF	**Yes**	All the non-key attributes depend only on a key
	BCNF	**Yes**	All the attributes depend only on a key
	Physical Design		
	Primary Key	OrderID, TimeofVisit	
	Foreign Keys	staffID, tableID, customerID, queueID	
	SQL Code	CREATE TABLE P_ORDER(ORDERID INTEGER, STAFFID INTEGER, TABLEID INTEGER, QUEUEID INTEGER, CUSTOMERID INTEGER, TIMEOFVISIT CHAR(20), PHONENUMBER CHAR(10), CONSTRAINT P_ORDER PRIMARY KEY(ORDERID, TIMEOFVISIT), FOREIGN KEY (STAFFID) REFERENCES "P_STAFF"(STAFFID), FOREIGN KEY (TABLEID) REFERENCES "P_TABLE"(TABLEID), FOREIGN KEY (QUEUEID) REFERENCES "P_QUEUE"(QUEUEID), FOREIGN KEY (CUSTOMERID) REFERENCES "P_CUSTOMER"(CUSTOMERID), CHECK(PHONENUMBER NOT LIKE '%[^0-9]%'));	
	Count of records in the table	**2**.	

	Name of the table	***Transaction***			
	Description	Transaction record of every order made by customer			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	transactionID	Id of the transaction	Integer	Between 1 and 999999	
	customerID	Id of the related customer and transaction	Integer	Between 1 and 999999	
	orderID	Id of the related order and transaction	Integer	Between 1 and 999999	
	staffID	Id of related staff and transaction	Integer	Between 1 and 999999	
	paymentType	Type of payment method	Char(2)	CS	‘CS’ for cash, ‘CC’ for credit card, ‘GC’ for gift card
	Total	Total of the check	integer	50	Cannot be null
	Gratuities	Gratuities amount from the check	integer	10	Cannot be null

	Functional Dependencies and Keys		
	Functional dependencies	transactionID-> customerID transactionID-> orderID transactionID->staffID transactionID-> paymentType transactionID->total transactionID->gratuities customerID-> userID, phoneNumber,idInformation,cardInfo staffID->userID, staffRole orderID-> staffID, tableID, customerID, queueID, timeofVisit	
	Candidate keys	transactionID, customerID, staffID, orderID	
	Normalization		
	1NF	**Yes**	All cells contain a unique value
	2NF	**Yes**	The key of the table is a single attribute
	3NF	**Yes**	All the non-key attributes depend only on a key
	BCNF	**Yes**	All the attributes depend only on a key
	Physical Design		
	Primary Key	transactionID	
	Foreign Keys	customerID, staffID, orderID	
	SQL Code	CREATE TABLE P_TRANSACTION(TRANSACTIONID INTEGER, CUSTOMERID INTEGER, ORDERID INTEGER, STAFFID INTEGER, PAYMENTTYPE CHAR(2), TOTAL INTEGER, GRATUITIES INTEGER, TIMEOFVISIT CHAR(20), CONSTRAINT P_TRANSACTION PRIMARY KEY(TRANSACTIONID), FOREIGN KEY (CUSTOMERID) REFERENCES "P_CUSTOMER"(CUSTOMERID), FOREIGN KEY (ORDERID, TIMEOFVISIT) REFERENCES "P_ORDER"(ORDERID, TIMEOFVISIT), FOREIGN KEY (STAFFID) REFERENCES "P_STAFF"(STAFFID), CHECK(PAYMENTTYPE = 'CS' OR PAYMENTTYPE = 'CC' OR PAYMENTTYPE = 'GC'));	
	Count of records in the table	**2**	

	Name of the table	***ItemOrdered***			
	Description	Ordered menu items on a single order			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	OrderID	Id of Order	Integer	Between 1 and 999999	
	MenuItemID	Id of Menu Item	Integer	Between 1 and 999999	
	TimeOfVisit	Time of visit of the customer	Char(20)	1/1/1999	Generally left blank
	Functional Dependencies and Keys				
	Functional dependencies	orderID -> MenuItemID menuItemID -> OrderID			
	Candidate keys	orderID, menuItemID			
	Normalization				
	1NF	**Yes**	All cells contain a unique value		
	2NF	**Yes**	The key of the table is a single attribute		
	3NF	**Yes**	All the non-key attributes depend only on a key		
	BCNF	**Yes**	All the attributes depend only on a key		
	Physical Design				
	Primary Key	orderID, menuItemID			
	Foreign Keys	orderID, menuItemID			
	SQL Code	CREATE TABLE P_ITEMORDERED(ORDERID INTEGER, MENUITEMID INTEGER, TIMEOFVISIT CHAR(20), CONSTRAINT P_ITEMORDERED PRIMARY KEY(ORDERID, MENUITEMID), FOREIGN KEY (ORDERID,TIMEOFVISIT) REFERENCES "P_ORDER"(ORDERID,TIMEOFVISIT), FOREIGN KEY (MENUITEMID) REFERENCES "P_MENUITEM"(MENUITEMID));			
	Count of records in the table	**2**			

	Name of the table	***Stock***			
	Description	Record the available stock for a menu item based on the availability of ingredients			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	menuitemID	Id of menuitem	Integer	Between 1 and	

				999999	
	inventoryID	Id of the ingredients	Integer	Between 1 and 999999	
	menustock	Available stock number of menut	integer	Between 1 and 999999	
	Functional Dependencies and Keys				
	Functional dependencies	Menuitemid, menuinventoryid -> meuStock			
	Candidate keys	Menuitemid, menuinventoryid			
	Normalization				
	1NF	**Yes**	All cells contain a unique value		
	2NF	**Yes**	The key of the table is a single attribute		
	3NF	**Yes**	All the non-key attributes depend only on a key		
	BCNF	**Yes**	All the attributes depend only on a key		
	Physical Design				
	Primary Key	Menuitemid, menuinventoryid			
	Foreign Keys	Menuitemid, menuinventoryid			
	SQL Code	CREATE TABLE P_STOCK(MENUITEMID INTEGER, INVENTORYID INTEGER, MENUSTOCK INTEGER, CONSTRAINT P_STOCK PRIMARY KEY(MENUITEMID, INVENTORYID), FOREIGN KEY (MENUITEMID) REFERENCES "P_MENUITEM"(MENUITEMID), FOREIGN KEY (INVENTORYID) REFERENCES "P_INVENTORY"(INVENTORYID));			
	Count of records in the table	**2**			

	Name of the table	***Management***			
	Description	Record of the managers of the restaurant			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	managementID	Id of Manager	Integer	Between 1 and 999999	
	StaffID	Id of related Staff info	Integer	Between 1 and 999999	
	Functional Dependencies and Keys				
	Functional dependencies	ManagementID->StaffID			
	Candidate keys	managementID			
	Normalization				
	1NF	**Yes**	All cells contain a unique value		

	2NF	**Yes**	The key of the table is a single attribute
	3NF	**Yes**	All the non-key attributes depend only on a key
	BCNF	**Yes**	All the attributes depend only on a key
	Physical Design		
	Primary Key	managementID	
	Foreign Keys	StaffID	
	SQL Code	CREATE TABLE P_MANAGEMENT(MANAGEMENTID INTEGER, STAFFID INTEGER, CONSTRAINT P_MANAGEMENT PRIMARY KEY(MANAGEMENTID), FOREIGN KEY (STAFFID) REFERENCES "P_STAFF"(STAFFID));	
	Count of records in the table	**2**	

	Name of the table	***Schedule***			
	Description	Working schedule of every staff			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	StaffID	Id of Staff	Integer	Between 1 and 999999	
	ManagementID	Id of manager who created the schedule	Integer	Between 1 and 999999	
	Schedule	Workday of staff	Varchar(10)	'MONDAY'	Must be a day in a week
	Functional Dependencies and Keys				
	Functional dependencies	staffID, ManagementID -> schedule			
	Candidate keys	staffID, ManagementID			
	Normalization				
	1NF	**Yes**	All cells contain a unique value		
	2NF	**Yes**	The key of the table is a single attribute		
	3NF	**Yes**	All the non-key attributes depend only on a key		
	BCNF	**Yes**	All the attributes depend only on a key		
	Physical Design				
	Primary Key	staffID, ManagementID			
	Foreign Keys	staffID, ManagementID			

	SQL Code	CREATE TABLE P_STAFFSCHEDULE(STAFFID INTEGER, MANAGEMENTID INTEGER, SCHEDULE VARCHAR(10), CONSTRAINT P_STAFFSCHEDULE PRIMARY KEY(STAFFID, MANAGEMENTID), FOREIGN KEY (STAFFID) REFERENCES "P_STAFF"(STAFFID), FOREIGN KEY (MANAGEMENTID) REFERENCES "P_MANAGEMENT"(MANAGEMENTID), CHECK(SCHEDULE = 'MONDAY' OR SCHEDULE = 'TUESDAY' OR SCHEDULE = 'WEDNESDAY' OR SCHEDULE = 'THURSDAY' OR SCHEDULE = 'FRIDAY' OR SCHEDULE = 'SATURDAY' OR SCHEDULE = 'SUNDAY'));
	Count of records in the table	**2**

MILESTONE 6: SQL QUERIES AND

Note: Please make sure you add/have 25 records in each table, on average.

Query 1	
English version	Return all staff who is old enough to handle alcoholic menu
Source for the query need in the SRS document	SRS PAGE 8, SECTION 3.1.2
SQL sentence	SELECT "USERNAME", (EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM USERDOB)) AS AGE FROM P_USER PU, P_STAFF PS WHERE PS.USERID = PU.USERID AND STAFFROLE = 'SERVER' AND (EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM USERDOB)) > 21 ORDER BY AGE;
Example of returned rows (cropped screen caption)	

	USERNAME	AGE
1	ILEZRA	22
2	MATTHEW PECK	25
3	PORORO	32
4	LEO ARABIA	45
5	BRANDO GEMA	52

Query 2	
English version	Return total of gratuities that a staff got.
Source for the query need in the SRS document	-
SQL sentence	SELECT PU."USERNAME", SUM(GRATUITIES) AS TOTAL_GRATUITIES FROM P_TRANSACTION PT, P_STAFF PS, P_USER PU WHERE PT.STAFFID = PS.STAFFID AND PS.USERID = PU.USERID AND PU."USERNAME" = 'BRANDO GEMA' GROUP BY PU."USERNAME";
Example of returned rows (cropped screen caption)	

	USERNAME	TOTAL_GRATUITIES
1	BRANDO GEMA	42

Query 3	
English version	Return all staffs who have done at least 2 orders
Source for the query need in the SRS document	-
SQL sentence	SELECT PU."USERNAME", COUNT(*) AS TOTAL_ORDER FROM P_STAFF PS, P_USER PU, P_ORDER PO WHERE PO.STAFFID = PS.STAFFID AND PS.USERID = PU.USERID GROUP BY PU."USERNAME" HAVING COUNT(*) >= 2;
Example of returned rows (cropped screen caption)	

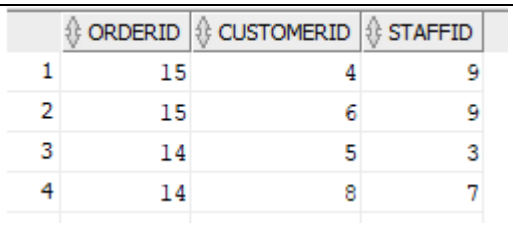
	USERNAME	TOTAL_ORDER
1	BRANDO GEMA	4
2	BAMBANG GENTOLET	3
3	KAREN WADAW	2
4	PORORO	2

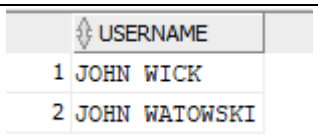
…

Query 4	
English version	Return the oldest staff that has schedule on Monday.
Source for the query need in the SRS document	-
SQL sentence	SELECT PU."USERNAME", EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM PU.USERDOB) AS AGE FROM P_USER PU, P_STAFF PS WHERE PS.USERID = PU.USERID AND PU.USERDOB IN (SELECT MIN(PU.USERDOB) FROM P_STAFF PS, P_STAFFSCHEDULE PSS, P_USER PU WHERE PS.STAFFID = PSS.STAFFID AND PS.USERID = PU.USERID AND PSS.SCHEDULE = 'MONDAY');
Example of returned rows (cropped screen caption)	

	USERNAME	AGE
1	KZARKA	66

Query 5	
English version	Return union between order and transaction. Used to check whether there is an error or not in the order.
Source for the query need in the SRS document	-
SQL sentence	SELECT ORDERID, CUSTOMERID, STAFFID FROM P_ORDER UNION SELECT ORDERID, CUSTOMERID, STAFFID FROM P_TRANSACTION ORDER BY ORDERID DESC;

Example of returned rows (cropped screen caption)	 There are errors here where 1 order is reffered to two different customerid.

Query 6	
English version	Return all customers whose name starts with J
Source for the query need in the SRS document	-
SQL sentence	SELECT PU."USERNAME" FROM P_USER PU WHERE "USERNAME" LIKE 'J%' AND USERROLE = 'CUSTOMER'
Example of returned rows (cropped screen caption)	

Query 7	
English version	Return all ingredients that stock is less than or equal to 100.
Source for the query need in the SRS document	-
SQL sentence	SELECT PI.INGREDIENTSNAME, PI.INGREDIENTSSTOCK FROM P_INVENTORY PI WHERE INGREDIENTSSTOCK <= 100 ORDER BY INGREDIENTSSTOCK;

Example of returned rows (cropped screen caption)	

	INGREDIENTSNAME	INGREDIENTSSTOCK
1	HONEY	10
2	SALMON	20
3	PAPRIKA	30
4	ASPARAGUS	50
5	RICE	50
6	MAYONAISE	50
7	PASTA	50
8	ONION	60
9	ALMOND	70
10	TOMATO	100

Query 8	
English version	Return all dessert that a customer ordered based on customer's name.
Source for the query need in the SRS document	-
SQL sentence	SELECT PM.MENUITEMNAME FROM P_USER PU, P_CUSTOMER PC, P_ORDER PO, P_ITEMORDERED PIO, P_MENUITEM PM WHERE PU.USERID = PC.USERID AND PO.CUSTOMERID = PC.CUSTOMERID AND PO.ORDERID = PIO.ORDERID AND PIO.MENUITEMID = PM.MENUITEMID AND PU."USERNAME" = 'ANDREW GARFIELD' AND PM.MENUCATEGORY = 'DESSERT';
Example of returned rows (cropped screen caption)	

	MENUITEMNAME
1	TIRAMISU

Query 9	
English version	Return staff that got the least gratuities
Source for the query need in the SRS document	-

SQL sentence	SELECT PU."USERNAME", SUM(GRATUITIES) AS TOTAL_GRATUITIES FROM P_TRANSACTION PT, P_STAFF PS, P_USER PU WHERE PT.STAFFID = PS.STAFFID AND PS.USERID = PU.USERID GROUP BY PU."USERNAME" ORDER BY TOTAL_GRATUITIES FETCH FIRST 1 ROWS ONLY;
Example of returned rows (cropped screen caption)	

	USERNAME	TOTAL_GRATUITIES
1	PORORO	5