**Project Title:** Restaurant Loyalty Database

**Team Names:** Jeffrey Liu, Javier Ramirez, Joshua Parada

**Abstract**

Loyalty programs have been entering the market for many types of industry varying from fast food, gas company, online merchandise, and more. Our database will be developed to manage and maintain data for a loyalty program for a restaurant. This project will create a database system that can keep track of employee and customer information to provide better services through the restaurant’s loyalty program. The database will allow customers to set up a loyalty account to allow them access to the loyalty and reward program which can be managed by employees. It also offers information on menu items, order details, payment details, and reservation services. To put it simply, this database system offers restaurants the ability to provide incentives for their customers to revisit because of their loyalty program while allowing access to certain data to improve work efficiency and promote customer service.

**Mission Statement**

The goal of this project is to provide a system that is able to efficiently maintain the data created and used to support the loyalty program of a restaurant, manage their customer and employee information, and promote easier access to this data.

**Mission Objectives**

To maintain (enter, update, and delete) data on Employee.

To maintain (enter, update, and delete) data on Customers.

To maintain (enter, update, and delete) data on Loyalty.

To maintain (enter, update, and delete) data on Rewards.

To maintain (enter, update, and delete) data on Orders.

To maintain (enter, update, and delete) data on Payment.

To maintain (enter, update, and delete) data on Menu.

To maintain (enter, update, and delete) data on Reservation.

To maintain (enter, update, and delete) data on TableServe.

To perform searches on Employee.

To perform searches on Customers.

To perform searches on Loyalty.

To perform searches on Rewards.

To perform searches on Order.

To perform searches on Payment.

To perform searches on Menu.

To perform searches on Reservation.

To perform searches on TableServ.

To report on Employee.

To report on Customers.

To report on Loyalty.

To report on Rewards.

To report on Order.

To report on Payment.

To report on Menu.

To report on Reservation.

To report on TableServ.

**MAJOR USER VIEWS**

|  | **Manager** | **Waiter** | **Customer** |
| --- | --- | --- | --- |
| EmployeeId | x | x |  |
| EmployeeName | x | x |  |
| Position | x |  |  |
| Salary | x |  |  |
| EmployeeAddress | x | x |  |
| EmployeePhoneNum | x | x |  |
| EmployeeEmail | x | x |  |
| CustomerId | x | x | x |
| CustomerName | x | x | x |
| CustomerPhoneNum | x | x | x |
| CustomerEmail | x | x | x |
| DateOfBirth | x | x | x |
| JoinDate | x | x | x |
| TotalPoints | x | x | x |
| RewardId | x | x | x |
| PointsRequired | x | x | x |
| MenuId | x | x | x |
| PriceDeducted | x | x | x |
| OrderId | x | x |  |
| OrderDate | x | x |  |
| OrderTime | x | x |  |
| PaymentId | x | x | x |
| BalanceDue | x | x | x |
| Gratuity | x | x | x |
| Total | x | x | x |
| PointsEarned | x | x | x |
| MenuId | x | x |  |
| MenuName | x | x | x |
| Cost | x | x | x |
| TableNumber | x | x |  |
| ReserveDate | x | x | x |
| ReserveTime | x | x | x |
| NumOfPeople | x | x | x |
| Seats | x | x |  |
| Available | x | x |  |

| Data | **Access Type** | **Owner** | **Manager** | **Worker** | **Customer** |
| --- | --- | --- | --- | --- | --- |
| All Employees | Maintain |  |  |  |  |
| Query | x |  |  |  |
| Report | x |  |  |  |
| Single Employee | Maintain |  | x |  |  |
| Query |  | x | x |  |
| Report |  | x | x |  |
| All Tables | Maintain |  |  |  |  |
| Query | x |  |  |  |
| Report | x |  |  |  |
| Single Table | Maintain |  | x |  |  |
| Query |  | x | x |  |
| Report |  | x | x |  |
| All Services | Maintain |  |  |  |  |
| Query | x |  |  |  |
| Report | x |  |  |  |
| Single Service | Maintain |  | x | x |  |
| Query |  | x | x |  |
| Report |  | x | x |  |
| All Reservations | Maintain |  |  |  |  |
| Query | x |  |  |  |
| Report | x |  |  |  |
| Single Reservation | Maintain |  | x | x |  |
| Query |  | x | x |  |
| Report |  | x | x |  |
| All Customers | Maintain |  |  |  |  |
| Query | x |  |  |  |
| Report | x |  |  |  |
| Single Customer | Maintain |  | x |  | x |
| Query |  | x | x | x |
| Report |  | x | x |  |
| All Redemptions | Maintain |  |  |  |  |
| Query | x |  |  |  |
| Report | x |  |  |  |
| Single Redemption | Maintain |  | x |  |  |
| Query |  | x | x | x |
| Report |  | x | x |  |
| All Rewards | Maintain |  |  |  |  |
| Query | x |  |  |  |
| Report | x |  |  |  |
| Single Reward | Maintain |  | x |  |  |
| Query |  | x | x | x |
| Report |  | x | x |  |
| All Menus | Maintain |  |  |  |  |
| Query | x |  |  |  |
| Report | x |  |  |  |
| Single Menu | Maintain |  | x |  |  |
| Query |  | x | x |  |
| Report |  | x | x |  |
| All Menu\_Orders | Maintain |  |  |  |  |
| Query |  |  |  |  |
| Report |  |  |  |  |
| Single Menu\_Order | Maintain |  | x |  |  |
| Query |  | x | x |  |
| Report |  | x | x |  |
| All Orders | Maintain |  |  |  |  |
| Query | x |  |  |  |
| Report | x |  |  |  |
| Single Order | Maintain |  | x |  |  |
| Query |  | x | x | x |
| Report |  | x | x |  |



**ER DRAFT:**

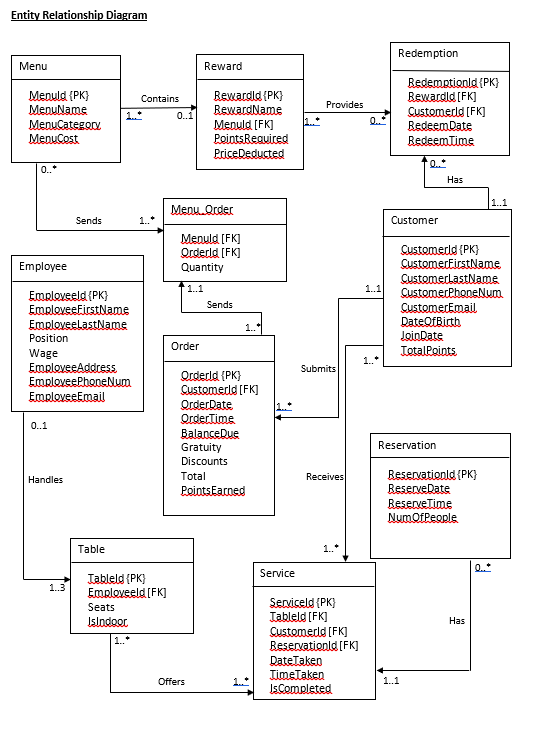










****

**Entity Descriptions**

**Employee**

The Employee entity provides information for all the staff that works for the restaurant. It has a primary key called EmployeeId and other attributes that contain personal information. EmployeeId will be an INT that increments by 1 starting from 100 and it can also be referenced as a foreign key in an enumerated table called Table\_List where an employee manages 1 to 3 tables. EmployeeFirstName, EmployeeLastName, and Position will be in VARCHAR allowing only alphabetical characters and hyphen (for names only). Wage will be DECIMAL type that represents hourly pay and can only contain 2 digits after the decimal. EmployeeAddress will be VARCHAR that follows the format 12345 Street City, State Zip-code. EmployeePhoneNum will be INT that is constrained to 10 digits exactly. EmployeeEmail will be VARCHAR of the format {email}@{email\_domain}.{domain} in which case {email} is the user’s email containing alphanumeric characters, {email\_domain} is the email domain being used that is strictly alphabetical, and {domain} is the web domain such as .com, .edu, etc that is also strictly alphabetical.

**TableServ**

The TableServ entity provides listed information of tables in the restaurant and is enumerated. The primary key for the table is TableId and is an INT type that increments starting from 1 until the maximum table the restaurant has. It references the Service table as a foreign key to indicate which table the service is taking place. EmployeeId is a foreign key that indicates which employee is responsible for this table. The Seats attribute is an INT that indicates how many seats this table can accommodate and will range from 2 to 8. The IsIndoor attribute is a BIT type in which 0 and 1 would represent false and true, respectively. This is used in conjunction with reservation if the party prefers indoor or outdoor.

**Service**

The Service entity provides information of the customers being handled in conjunction with the table (and sometimes reservation) entities. ServiceId will be the primary key of INT type that increments starting from 4001. It will also contain three foreign keys, TableId, CustomerId, and ReservationId to mark which tables are occupied, optionally by a loyalty customer and by reservation. Values for the two latter foreign key attributes can be NULL if the customer has not registered for the loyalty program and the customer did not make a reservation to get service for the table occupied. ReservationId would be listed as optional and can take a NULL value if the table isn’t reserved by a loyalty customer. DateTaken and TimeTaken will be VARCHAR type of the formats ##-##-#### and ##:## (24-hour clock), respectively, whereas ‘#’ represents numerical values only. The attribute IsCompleted is a BIT type where 1 represents the service is completed, indicating that the table is available for later reservation. On the front-end side, there would be rules where customers can only be seated or reserve a table only if the most recent instance of a certain TableId has IsCompleted marked as 1 to indicate it is free.

**Reservation**

The Reservation entity provides information of the reservations the customers could make, assigning them a specific table at the restaurant with time and date. It has a primary key called ReservationId that is INT type and increments starting from 301. In order to identify the customer who made the reservation, a join query must be called with Service where the ReservationId and CustomerId acts as the foreign keys for Service. There is also a ReserveDate and ReserveTime that follows the same rules as DateTaken and TimeTaken from the Service entity table mentioned previously. Finally, there is NumOfPeople attribute that is of an INT type that counts the number of people. It will accept a value between 2 to 8 to reflect upon the number of seats available for a table mentioned in TableServ entity above. If a customer has more than 8, another instance of a ReservationId pointing to the same CustomerId would be created instead.

**Customer**

The Customer entity provides information of the customer that signed up for the loyalty program for the restaurant. It includes personal information similar to the Employee entity with the exception that it does not contain position, wage, and address, but it does include DateOfBirth, JoinDate, and TotalPoints used for redemption purposes (refer to Redemption entity). The data type and format for most of the Customer attributes follow the one mentioned in Employee (EmployeeId for CustomerId, EmployeeFirstName for CustomerFirstName, so on and so forth). CustomerId is the primary key for this table and acts as the foreign key for the Service table, Orders table, and Redemption table. DateOfBirth and JoinDate will have the DATE data type and are used for the front-end side of the Rewards entity. Customers will be offered rewards on their birthdays and on anniversaries of their join date. TotalPoints is an INT type that is at least 0 representing the number of points the customer has accumulated. These points are earned based on the amount they have spent for an order (refer to Orders entity) and can be spent on rewards (refer to Reward entity).

**Redemption**

The Redemption entity provides information of the rewards that the customers have redeemed. Employees will use the information from here, joined by the information with their respective reward from the Reward table, to reduce prices on the customers’ order listed as Discounts in Orders Table. The primary key is RedemptionId which is INT type that increments starting from 00000001 (8-digit format). There are also two foreign keys, RewardId and CustomerId, to indicate what reward a customer has redeemed. RedeemDate and RedeemTime has the DATE type and TIME type, respectively. The time would exclude the seconds from the TIME data type.

**Reward**

The Reward entity provides information about the list of rewards that the customers can redeem. It describes the menu item that is to be discounted and how much is discounted. The primary key is RewardId and is incremented as INT type starting from 1. RewardName is an attribute of VARCHAR type that briefly describes the reward. MenuId is a foreign key from the Menu entity so that it can mark the available discounts offered for certain menu items. PointsRequired is an INT type that has a value of at least 1 and indicates how many points the customer needs to have to be able to redeem this reward. This will subtract the TotalPoints for that respective customer. PriceDeducted is a DECIMAL type with 2 digits after the decimal place and is used as Discounts for the Orders entity.

**Menu**

The Menu entity provides information on the list of food and beverages the restaurant offers. It has the primary key MenuId which is an INT type that increments starting from 1. It acts as a foreign key in both Reward and Menu\_Order entities to indicate what item the said entity tables are referencing to. MenuName and MenuCategory are VARCHAR values that describe the food or beverage and are strictly alphanumerical including white-space characters. Examples for MenuCategory would be something like appetizer, chicken entrée, desserts, alcoholic beverages, etc. Finally, we have MenuCost that is a DECIMAL type with 2 digits after the decimal place, which simply shows the price of the item.

**Menu\_Order**

The Menu\_Order entity is an associative table that links the details between Menu and Orders table. The purpose of this table is to be able to list all the menu items for an order so that the total price could be calculated later in the Orders entity. It contains 2 foreign keys, MenuId and OrderId, and an attribute called Quantity that is an INT type of at least 1. The foreign keys here can be used as a candidate key as no two items on this table will have both the exact same MenuId and OrderId. OrderId cannot be used as a superkey because this table can contain different MenuId pointing to the same OrderId and MenuId cannot be a superkey because the same MenuId can point to different OrderId.

**Orders**

The Orders entity provides information of the order and payments the customer must provide. Its primary key is OrderId which is an INT type that increments starting from 1 and is used as a foreign key for Menu\_Order with reasons stated for that entity above. CustomerId is a foreign key to show who the order is assigned to. OrderDate and OrderTime follow the data type and format mentioned previously for dates and times. BalanceDue is DECIMAL type (2-digits after decimal) to show the initial balance due before gratuity and/or discounts are applied. Following the same data type and format, Gratuity, Discounts, and Total are attributes indicating how much the initial cost from BalanceDue could be added or subtracted to get the grand total (Total as attribute). Finally, PointsEarned is an INT type that is of at least 1 and is reflective based on BalanceDue. This should not consider gratuity and discounts applied and is strictly based on what the original cost was to show how much points can be earned by the customer which is added to the TotalPoints in the Customer entity.

**Relational Model – Table Attributes**

***Legends: Primary Key – {PK}, Foreign Key – [FK], Candidate Key – [CK]***

**Employee**

| **Attribute** | **Data Type** | **Domain** | **Constraints** | **Default** |
| --- | --- | --- | --- | --- |
| **EmployeeId** | **INT** |  | **{PK}** | **AUTOINCREMENT** |
| **EmployeeFirstName** | **VARCHAR(30)** | **alphabets, hyphens** | **NOT NULL** |  |
| **EmployeeLastName** | **VARCHAR(30)** | **alphabets, hyphens** | **NOT NULL** |  |
| **Position** | **VARCHAR(30)** | **alphabets, spaces, hyphens** | **NOT NULL** |  |
| **Wage** | **DECIMAL** |  | **NOT NULL** |  |
| **EmployeeAddress** | **VARCHAR(100)** | **alphanumeric, spaces, hyphens** | **NOT NULL** |  |
| **EmployeePhoneNum** | **VARCHAR(10)** | **numeric** | **NOT NULL** |  |
| **EmployeeEmail** | **VARCHAR(100)** | **alphanumeric, symbols (only dot, underscore, and hyphen)** | **CHECK(‘%@%.%’),**  **NOT NULL** |  |

**TableServ**

| **Attribute** | **Data Type** | **Domain** | **Constraints** | **Default** |
| --- | --- | --- | --- | --- |
| **TableId** | **TINYINT** |  | **{PK}** | **AUTOINCREMENT** |
| **EmployeeId** | **INT** |  | **[FK] in Employee** |  |
| **Seats** | **TINYINT** | **2 to 8** | **NOT NULL** |  |
| **IsIndoor** | **BIT** |  | **NOT NULL** |  |

**Service**

| **Attribute** | **Data Type** | **Domain** | **Constraints** | **Default** |
| --- | --- | --- | --- | --- |
| **ServiceId** | **INT** |  | **{PK}** | **AUTOINCREMENT** |
| **TableId** | **TINYINT** |  | **[FK] in TableServ, NOT NULL** |  |
| **CustomerId** | **INT** |  | **[FK] in Customer** |  |
| **ReservationId** | **INT** |  | **[FK] in Reservation** |  |
| **DateTaken** | **DATE** |  | **NOT NULL** |  |
| **TimeTaken** | **TIME** | **09:00 to 21:30** | **CHECK (‘##:##’)** |  |
| **IsCompleted** | **BIT** |  | **NOT NULL** |  |

**Reservation**

| **Attribute** | **Data Type** | **Domain** | **Constraints** | **Default** |
| --- | --- | --- | --- | --- |
| **ReservationId** | **INT** |  | **{PK}** | **AUTOINCREMENT** |
| **ReserveDate** | **DATE** |  | **NOT NULL** |  |
| **ReserveTime** | **TIME** | **09:00 to 21:30** | **CHECK (‘##:##’), NOT NULL** |  |
| **NumOfPeople** | **TINYINT** | **2 to 16** | **NOT NULL** |  |

**Customer**

| **Attribute** | **Data Type** | **Domain** | **Constraints** | **Default** |
| --- | --- | --- | --- | --- |
| **CustomerId** | **INT** |  | **{PK}** | **AUTOINCREMENT** |
| **CustomerFirstName** | **VARCHAR(30)** | **alphabets, hyphens** | **NOT NULL** |  |
| **CustomerLastName** | **VARCHAR(30)** | **alphabets, hyphens** | **NOT NULL** |  |
| **CustomerPhoneNum** | **VARCHAR(10)** | **numeric** | **NOT NULL** |  |
| **CustomerEmail** | **VARCHAR(100)** | **alphanumeric, symbols (only dot, underscore, and hyphen), 1 @ symbol** | **CHECK(‘%@%.%’), NOT NULL** |  |
| **DateOfBirth** | **DATE** |  | **NOT NULL** |  |
| **JoinDate** | **DATE** |  | **NOT NULL** |  |
| **TotalPoints** | **SMALLINT** |  | **NOT NULL** | **0** |

**Redemption**

| **Attribute** | **Data Type** | **Domain** | **Constraints** | **Default** |
| --- | --- | --- | --- | --- |
| **RedemptionId** | **INT** |  | **{PK}** |  |
| **RewardId** | **TINYINT** |  | **[FK] in Reward, NOT NULL** |  |
| **CustomerId** | **INT** |  | **[FK] in Customer, NOT NULL** |  |
| **RedeemDate** | **DATE** |  | **NOT NULL** |  |
| **RedeemTime** | **TIME** | **09:00 to 21:30** | **CHECK (‘##:##’), NOT NULL** |  |

**Reward**

| **Attribute** | **Data Type** | **Domain** | **Constraints** | **Default** |
| --- | --- | --- | --- | --- |
| **RewardId** | **TINYINT** |  | **{PK}** |  |
| **RewardName** | **VARCHAR(50)** |  | **NOT NULL** |  |
| **MenuId** | **TINYINT** |  | **[FK] in Menu, NOT NULL** |  |
| **PointsRequired** | **SMALLINT** |  | **NOT NULL** |  |
| **PriceDeducted** | **SMALLINT** |  | **NOT NULL** |  |
| **Available** | **BIT** |  | **NOT NULL** | **0** |

**Menu**

| **Attribute** | **Data Type** | **Domain** | **Constraints** | **Default** |
| --- | --- | --- | --- | --- |
| **MenuId** | **TINYINT** |  | **{PK}** | **AUTOINCREMENT** |
| **MenuName** | **VARCHAR(30)** | **alphabets and spaces** | **NOT NULL** |  |
| **MenuCategory** | **VARCHAR(20)** | **alphabets and spaces** | **NOT NULL** |  |
| **MenuCost** | **DECIMAL** |  | **NOT NULL** |  |

**Menu\_Order**

| **Attribute** | **Data Type** | **Domain** | **Constraints** | **Default** |
| --- | --- | --- | --- | --- |
| **MenuId** | **TINYINT** |  | **[FK] in Menu, [CK] with OrderId, NOT NULL** |  |
| **OrderId** | **INT** |  | **[FK] in Orders, [CK] with MenuId, NOT NULL** |  |
| **Quantity** | **TINYINT** |  | **NOT NULL** | **1** |

**Orders**

| **Attribute** | **Data Type** | **Domain** | **Constraints** | **Default** |
| --- | --- | --- | --- | --- |
| **OrderId** | **INT** |  | **{PK}** | **AUTOINCREMENT** |
| **CustomerId** | **INT** |  | **[FK] in Customer, NOT NULL** |  |
| **OrderDate** | **DATE** |  | **NOT NULL** |  |
| **OrderTime** | **TIME** | **09:00 to 21:30** | **CHECK('##:##')** |  |
| **BalanceDue** | **DECIMAL** |  | **NOT NULL** |  |
| **Gratuity** | **DECIMAL** |  | **NOT NULL** |  |
| **Discounts** | **DECIMAL** |  | **NOT NULL** |  |
| **Total** | **DECIMAL** |  | **NOT NULL** |  |
| **PointsEarned** | **SMALLINT** |  | **NOT NULL** |  |

**Use Cases**

**Actor: Manager**

*Entity: Employee*

Perform aggregate query for Employee

1. Manager will click “Query” button
2. Prompt manager to enter Employee Name or Employee ID
3. List of possible employees will populate
4. Manager will select employee if found

Enter a new Employee to Restaurant

1. Manager will click “Add New Employee” button
2. Prompt manager to input employee information
3. Manager will click “Save Entry” button

**INSERT INTO** employee

**VALUES** ( EmployeeID, EmployeeName, Position, Salary, Address Phone, Email)

Update an Employee from Restaurant

1. Manager will click “Edit Employee” button
2. Prompt manager to enter Employee ID or Employee Name
3. Prompt manager to select employee information to make changes to
4. Manager will click “Save Entry” button

**UPDATE FROM** employee

**SET** ( EmployeeID = “”, EmployeeName= “”, Position= “”, Salary= “”, Address Phone = “”, Email = “” )

Remove an Employee from Restaurant

1. Manager will click “Remove Employee” button
2. Prompt manager to enter Employee ID or Employee Name
3. Manager will click “Delete Entry” button
4. Manager will click “Confirm Deletion” button

**DELETE FROM** employee

**WHERE** employee\_id = The id that wants to be removed/deleted

*Entity: Menu*

Perform aggregate query for Menu

1. Manager will click “Query” button
2. Prompt manager to enter Menu Name
3. List of possible menus will populate
4. Manager will select menu if found

Enter a new Menu entry

1. Manager will click “Add New Menu” button
2. Prompt manager to input new menu information : Menu Name, Menu category, and Menu Cost
3. Manager will click “Save Entry” button.

**INSERT INTO** menu

**VALUES** (Menu ID, Menu Name, Menu Category, Menu cost)

Update a Menu entry

1. Manager will click “Edit Menu” button
2. Prompt manager to enter Menu ID, Menu Name, Menu category, and Menu Cost
3. Prompt manager to select menu information to make changes to
4. Manager will click “Save Entry” button

**UPDATE FROM** menu

Delete a Menu entry

1. Manager will click “Delete Menu” button
2. Prompt manager to choose to delete Menu ID / Menu name
3. Manager will click “Delete Entry” button
4. Manager will click “Confirm Deletion” button

**DELETE FROM** menu

**WHERE** menu\_id = The id that wants to be removed/deleted

*Entity: Menu\_Order*

Perform aggregate query for Menu\_order

1. Manager will click “Query” button
2. Prompt manager to enter Menu\_id or Order\_id
3. List of possible menus will populate
4. Manager will select menu\_order if found

Enter a new Menu\_Order entry

1. Manager will click “Add New Menu Order” button
2. Manager will add : Menu id, Order id, and Quantity for a new menu order
3. Manager will click “Save Entry” button.

**INSERT INTO** menuorder

**VALUES** (MenuID, OrderID, Quantity)s

Update a Menu order entry

1. Manager will click “Edit Menu\_order” button
2. Prompt manager to update: Menu id, Order id, and Quantity for an existing menu order
3. Display updated menu\_order information.
4. Manager will click “Save Entry” button

**UPDATE FROM** menuorder

**SET** MenuID = “”, OrderID = “”, Quantity = “”

Delete a Menu order entry

1. Manager will click “Delete Menu” button
2. Prompt manager to enter Menu id / Order id and chose what he wishes to delete.
3. Manager will click “Delete Entry” button
4. Manager will click “Confirm Deletion” button

**DELETE FROM** menuorder

**WHERE** MenuID = “”

*Entity: Redemption*

Perform aggregate query for Redemption

1. Manager will click “Query” button
2. Prompt manager to enter Redemption ID or Customer ID
3. List of possible Redemptions will populate
4. Manager will select redemption if found

Enter a new Redemption entry

1. Manager will click “Add New Redeemable” button
2. Prompt manager to input Redemption information
3. Manager will click “Save Redemption Entry” button.

**INSERT INTO** Redemption

**VALUES** (RedemptionID , RewardID , CustomerID , RedeemDate , RedeemTime )

Update a Redemption entry

1. Manager will click “Edit Redemption” button
2. Prompt manager to enter Redemption ID or Customer ID
3. Prompt manager to select Redemption information to make changes to.
4. Manager will click “Save Redemption Entry” button.

**UPDATE FROM** Redemption

**SET** (RedemptionID = “” , RewardID = “” , CustomerID = “”, RedeemDate = “”, RedeemTime = “” )

Delete a Redemption entry

1. Manager will click “Delete Redemption” button
2. Prompt manager to enter Redemption ID or Customer ID
3. Manager will click “Delete Redemption Entry” button
4. Manager will click “Confirm Deletion” button.

**DELETE FROM** Redemption

**WHERE** redemption\_id = “”

*Entity: Reward*

Perform aggregate query for Reward

1. Manager will click “Query” button
2. Prompt manager to enter Reward Name or Reward ID
3. List of possible Rewards will populate
4. Manager will select reward if found

Enter a new Reward entry

1. Manager will click “Add New Reward” button
2. Prompt manager to input reward information
3. Manager will click “Save Entry” button.

**INSERT INTO** reward

**VALUES** (RewardID, RewardName, MenuID, PointsRequired, PointsDeducted)

Update a Reward entry

1. Manager will click “Edit Reward” button
2. Prompt manager to enter Reward ID or Reward Name
3. Prompt manager to select menu information to make changes to
4. Manager will click “Save Entry” button

**UPDATE FROM** reward

**SET** RewardID = “”, RewardName = “”, MenuID = “”, PointsRequired = “”, PointsDeducted = “”

Delete a Reward entry

1. Manager will click “Delete Reward” button
2. Prompt manager to enter Reward ID or Reward Name
3. Manager will click “Delete Entry” button
4. Manager will click “Confirm Deletion” button

**DELETE FROM** reward

**WHERE** RewardID = “”

**Actor: Waiter**

*Entity: Customer*

Perform aggregate query for Customer

1. Manager will click “Query” button
2. Prompt manager to enter Customer ID
3. List of possible customers will populate
4. Manager will select customer if found

Enter a new Customer entry

1. Waiter will click “Add New Customer” button
2. Prompt waiter to input customer information
3. Waiter will click “Save Entry” button.

**INSERT INTO** customer

**VALUES** (CustomerID, CustomerName, CustomerPhoneNum, CustomerEmail, DateOfBirth, JoinDate, TotalPoints)

Update a Customer entry

1. Waiter will click “Edit Customer” button
2. Prompt waiter to enter Customer ID
3. Prompt waiter to select order information to make changes to
4. Waiter will click “Save Entry” button

**UPDATE FROM** customer

**SET** CustomerID = “”, CustomerName, CustomerPhoneNum = “”, CustomerEmail = “”, DateOfBirth = “”, JoinDate = “”, TotalPoints = “”

Delete a Customer entry

1. Waiter will click “Delete Customer” button
2. Prompt waiter to enter Customer ID
3. Waiter will click “Delete Entry” button
4. Waiter will click “Confirm Deletion” button

**DELETE FROM** customer

**WHERE** CustomerID = “”

*Entity: TableServ*

Perform aggregate query for TableServ

1. Manager will select “Query” button
2. Prompt manager to enter Table Number
3. List of possible tables will populate
4. Manager will select table if found

Enter a new TableServ entry

1. Waiter will click “Add New Table” button
2. Prompt waiter to input Table information
3. Waiter will click “Save Entry” button.

**INSERT INTO** TableServ

**VALUES** (TableID, TableNum, ReservationID, DateTaken, TimeTaken, IsOccupied)

Update a TableServ entry

1. Waiter will click “Edit Table” button
2. Prompt waiter to enter Table Number
3. Prompt waiter to select order information to make changes to
4. Manager will click “Save Entry” button

**UPDATE FROM** TableServ

**SET** TableID = “”, TableNum = “”, ReservationID = “”, DateTaken = “”, TimeTaken = “”, IsOccupied = “”

Delete a TableServ entry

1. Waiter will click “Delete Table” button
2. Prompt waiter to enter Table Number
3. Waiter will click “Delete Entry” button
4. Waiter will click “Confirm Deletion” button

**DELETE FROM** TableServ

**WHERE** TableID = “”

*Entity: Reservation*

Perform aggregate query for Reservation

1. Manager will select “Query” button
2. Prompt manager to enter Reservation ID
3. List of possible reservations will populate
4. Manager will select reservation if found

Enter a new Reservation entry

1. Waiter will click “Add New Reservation” button
2. Prompt waiter to input Reservation information
3. Waiter will click “Save Entry” button.

**INSERT INTO** reservation

**VALUES** (ReservationID, CustomerID, ReserveDate, RerserveTime, NumOfPeople)

Update a Reservation entry

1. Waiter will click “Edit Reservation” button
2. Prompt waiter to enter Reservation ID
3. Prompt waiter to select reservation information to make changes to
4. Waiter will click “Save Entry” button

**UPDATE FROM** reservation

**SET** ReservationID = “”, CustomerID = “”, ReserveDate = “”, ReserveTime = “”, NumOfPeople = ‘’

Delete a Reservation entry

1. Waiter will click “Delete Reservation” button
2. Prompt waiter to enter Reservation ID
3. Waiter will click “Delete Entry” button
4. Waiter will click “Confirm Deletion” button

**DELETE FROM** reservation

**WHERE** ReservationID = “”

**Actor: Manager & Waiter**

*Entity: Orders*

Perform aggregate query for Orders

1. Manager will click “Query” button
2. Prompt manager to enter Order ID
3. List of possible orders will populate
4. Manager will select order if found

Enter a new Orders entry

1. Actor will click “Add New Order” button
2. Prompt actor to input order information and enter Customer ID
3. Actor will calculate Balance due and will be displayed for a specific order
4. Actor will click “Save Entry” button.

**INSERT INTO** Orders

**VALUES** (OrderID, CustomerID, OrderDate, OrderTime, BalanceDue, Gratuity, Total, PointsEarned)

Update an Orders entry

1. Actor will click “Edit Order” button
2. Prompt actor to enter Order ID or Customer ID
3. Prompt actor to change order info or the total balance due.
4. Actor will click “Save Entry” button

**UPDATE FROM** Orders

**SET** OrderID = “”, CustomerID = “”, OrderDate = “”, OrderTime = “”, BalanceDue = “”, Gratuity = “”, Total = “”, PointsEarned = “”

Delete an Orders entry

1. Actor will click “Delete Order” button
2. Prompt actor to enter Order ID and display its contents.
3. Actor will click “Delete Order” button
4. Actor will click “Confirm Deletion” button

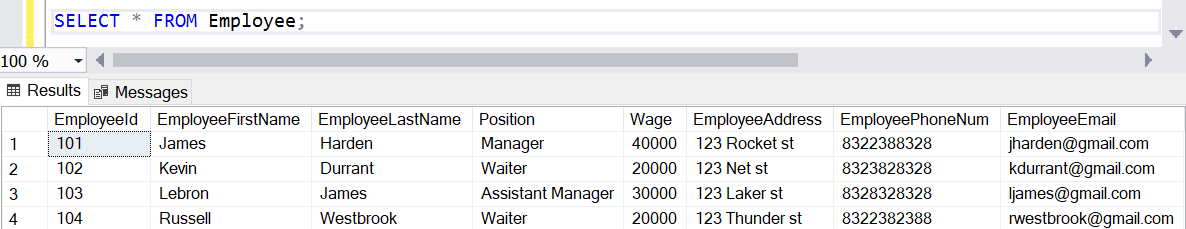
**DELETE FROM** Orders

**WHERE** OrderID = “”

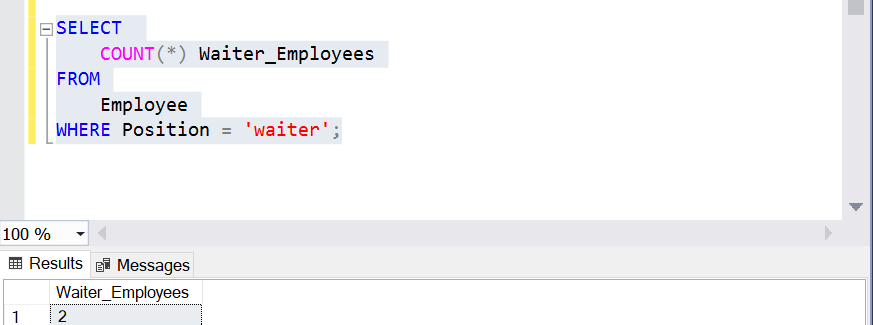
**Test Plan and Records**

**Employee Table**

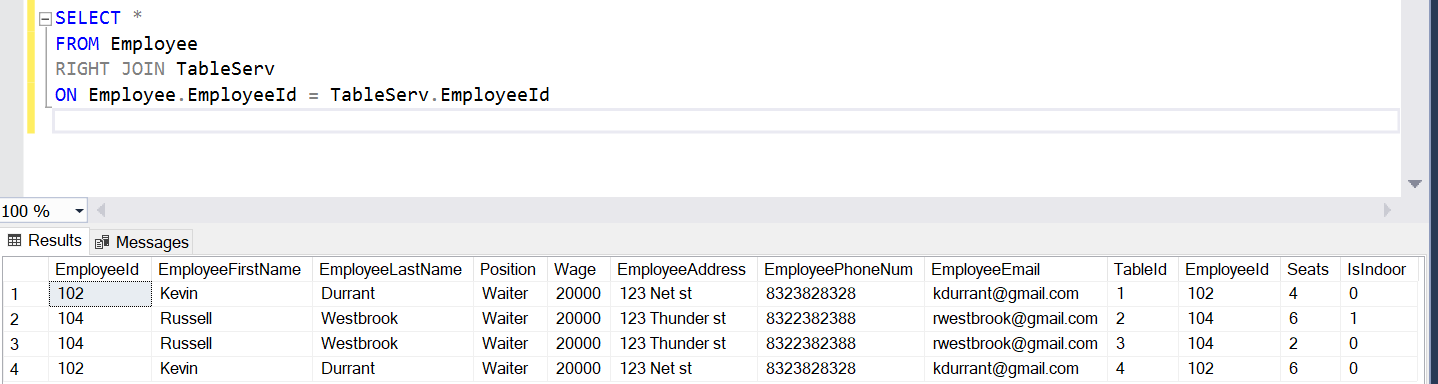
All Data:

****

Aggregate Function:

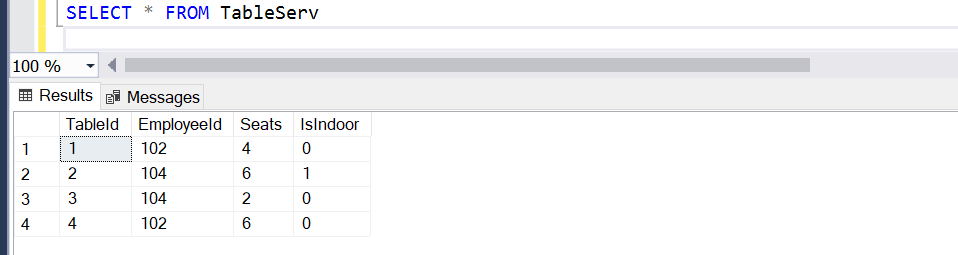


Joint Query with TableServ Entity:

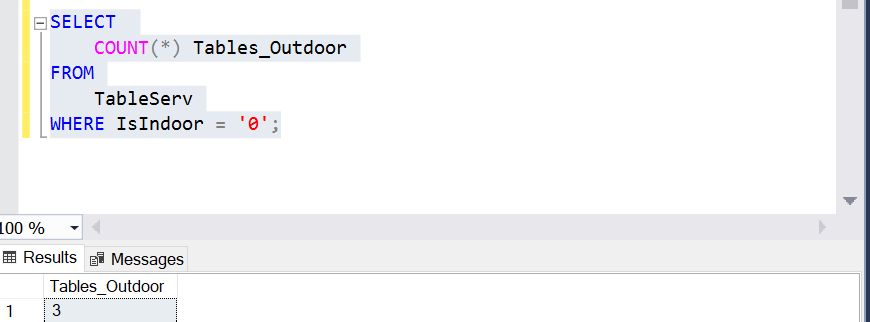


**TableServ Table**

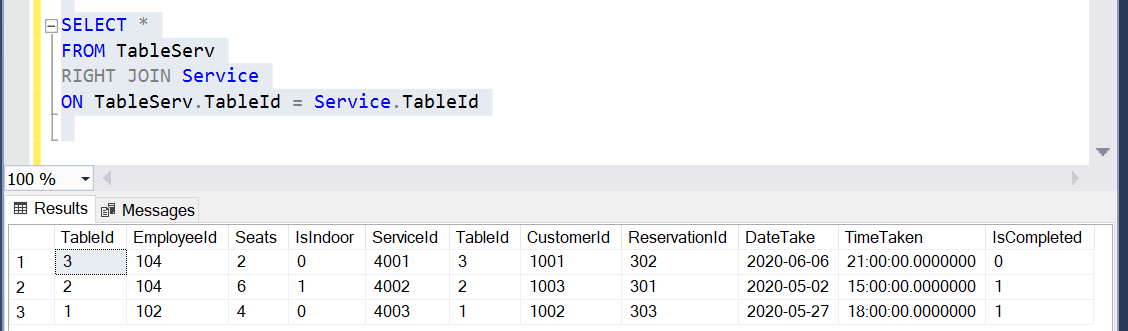
All Data:

****

Aggregate Function:

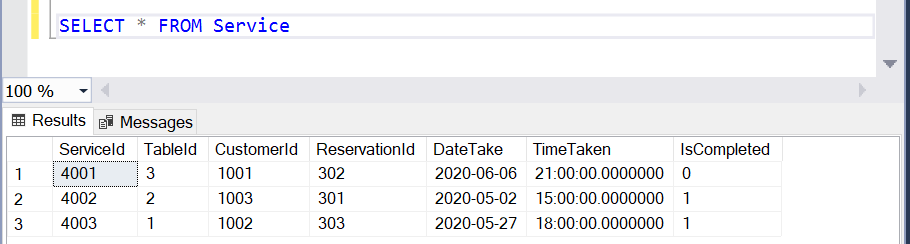


Joint Query with Service Entity:

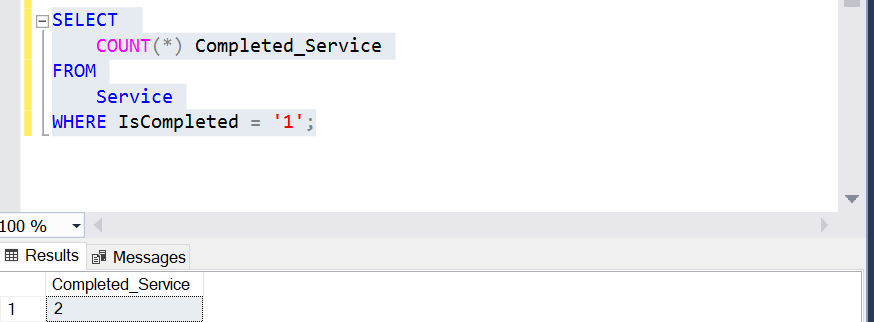


**Service Table**

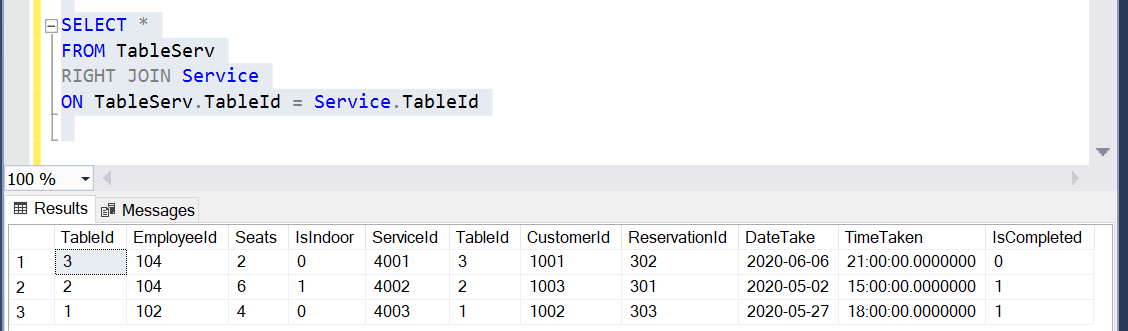
All Data:

****

Aggregate Function:

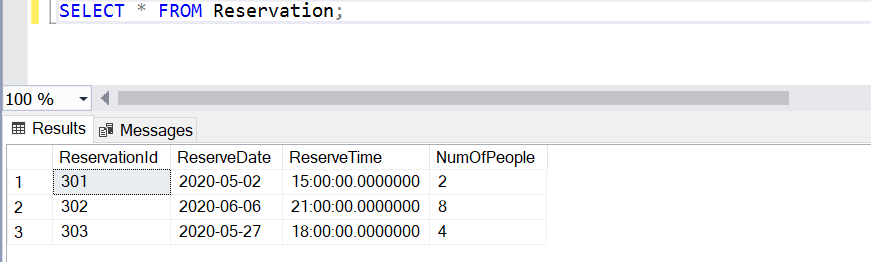
****

Joint Query with TableServ Entity:

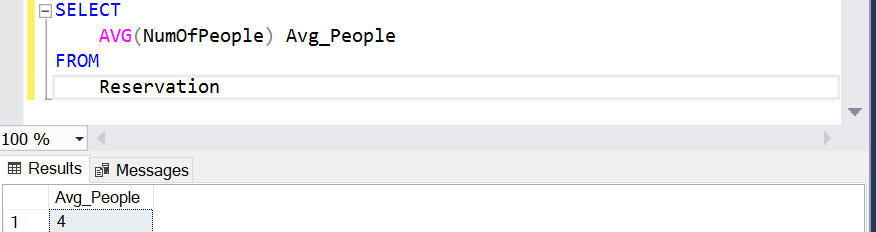


**Reservation Table**

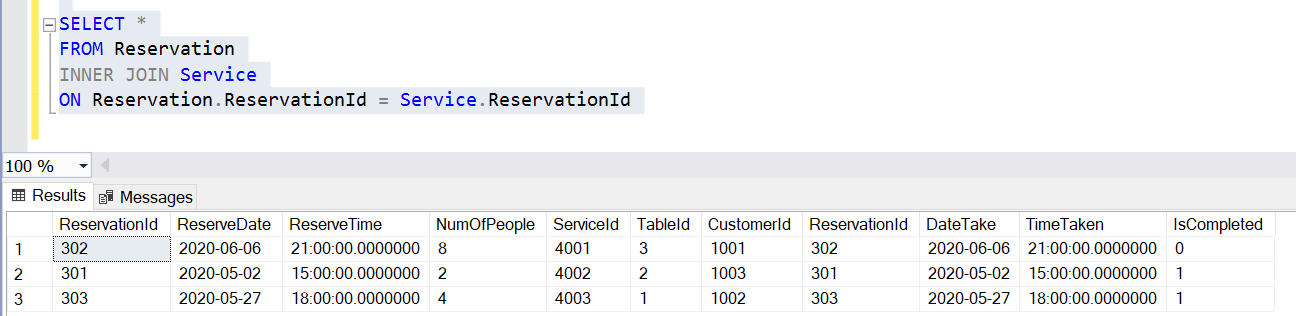
All Data:

****

Aggregate Function:

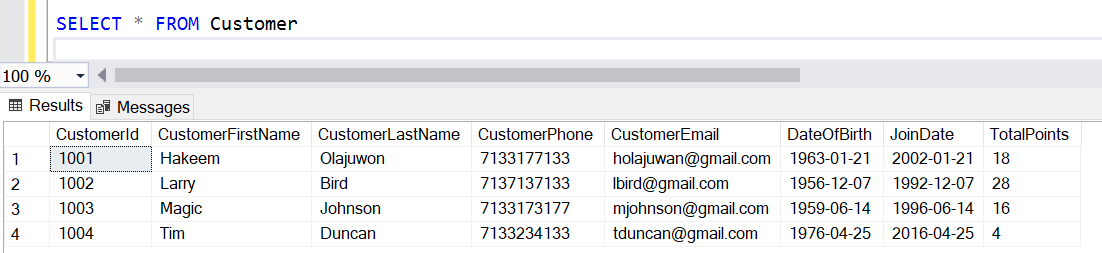


Joint Query with Service Entity:

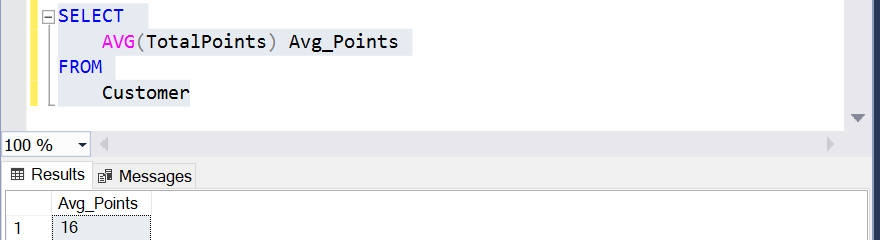


**Customer Table**

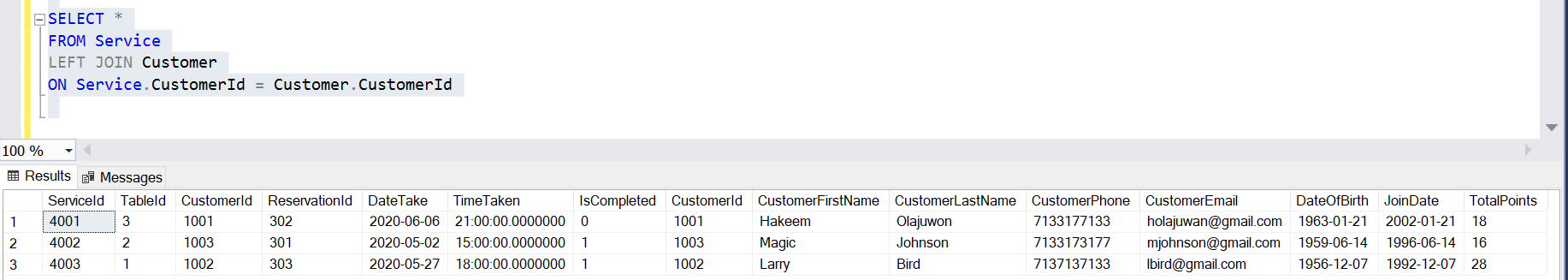
All Data:

****

Aggregate Function:

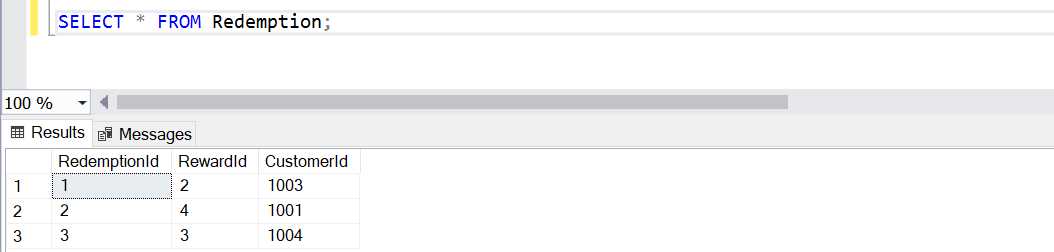


Joint Query with Service Entity:

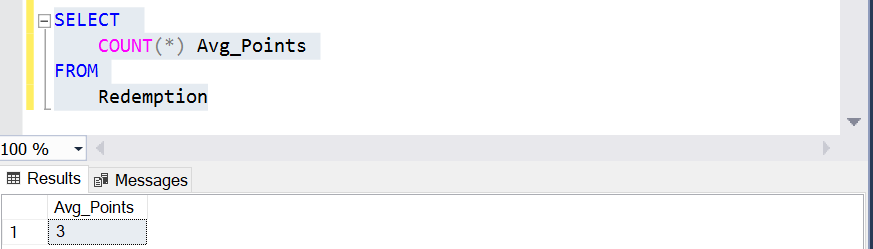


**Redemption Table**

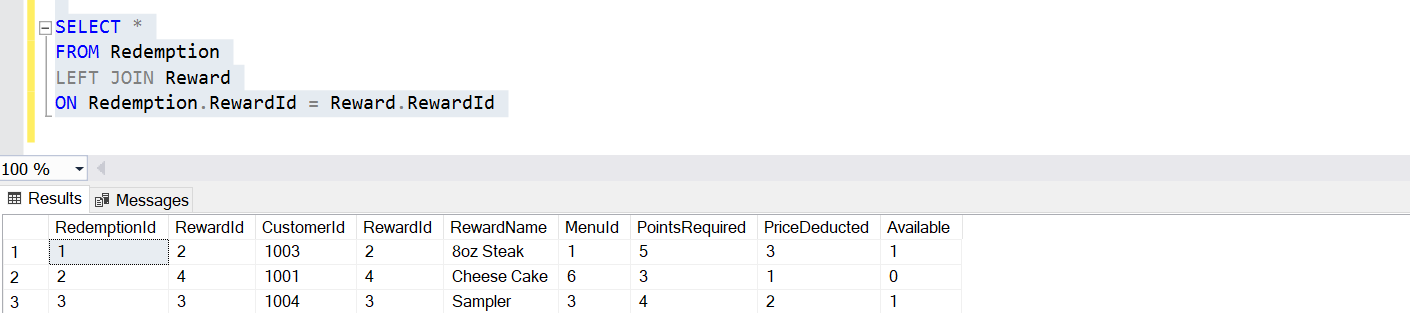
All Data:

****

Aggregate Function:

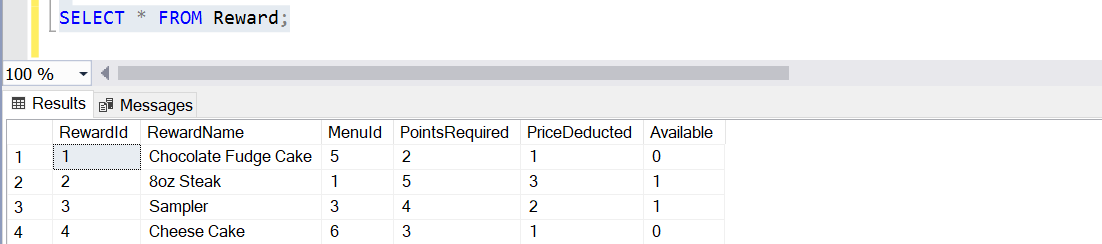
****

Joint Query with Reward Entity:

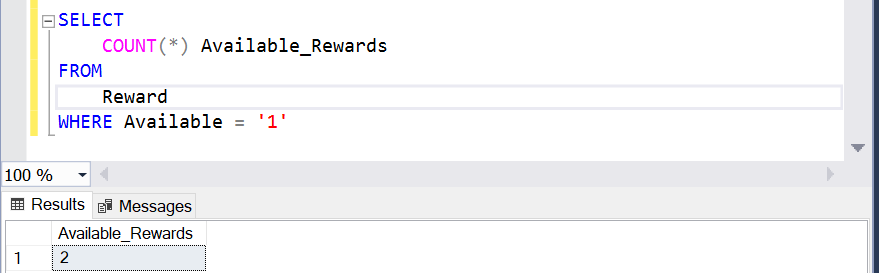


**Reward Table**

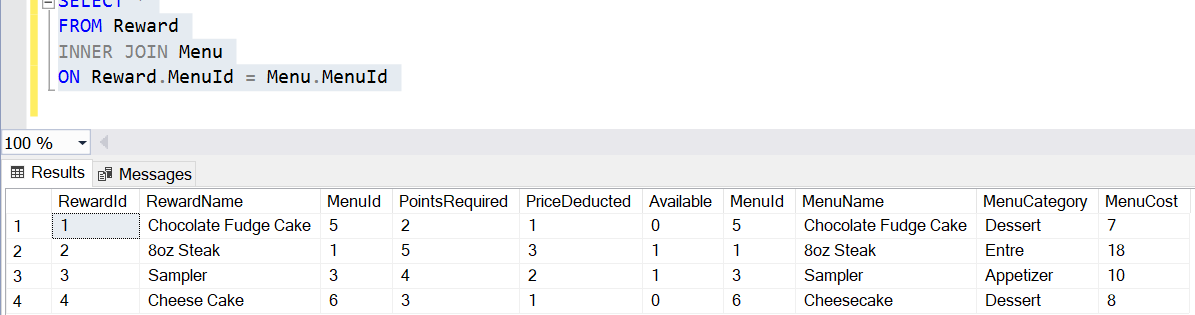
All Data:

****

Aggregate Function:

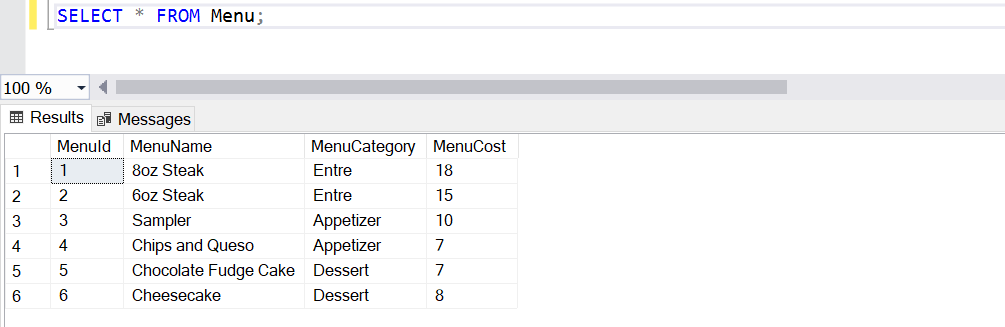
****

Joint Query with Menu Entity:

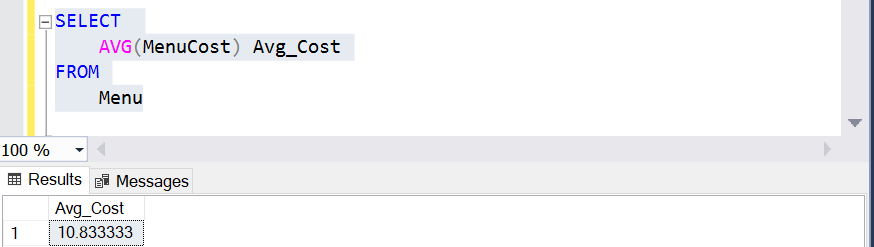


**Menu Table**

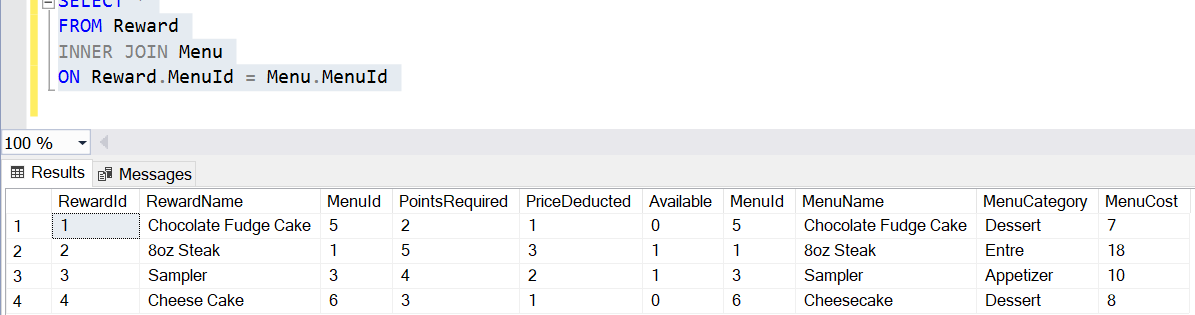
All Data:

****

Aggregate Function:

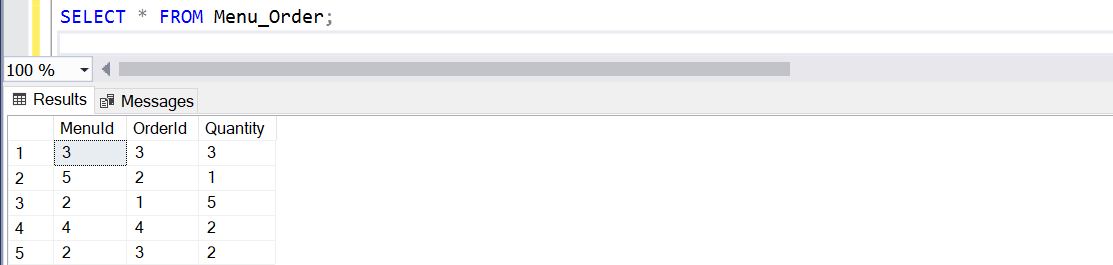


Joint Query with Reward Entity:

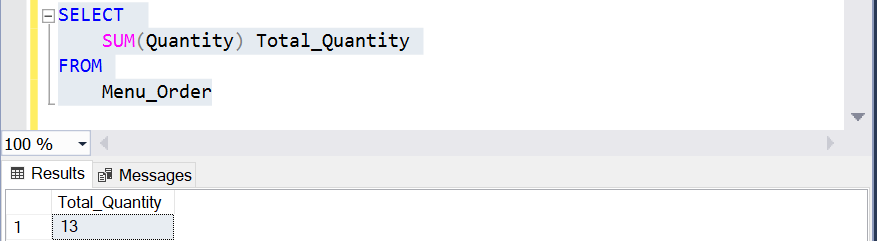


**Menu\_Order Table**

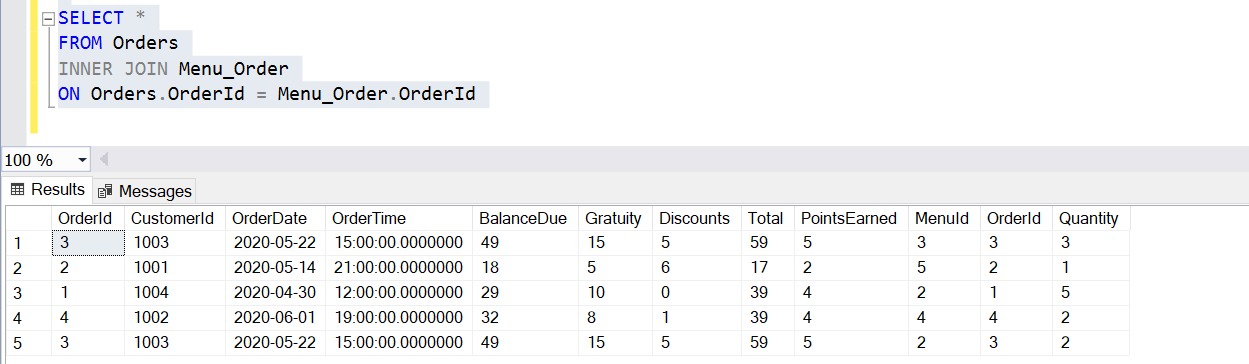
All Data:

****

Aggregate Function:

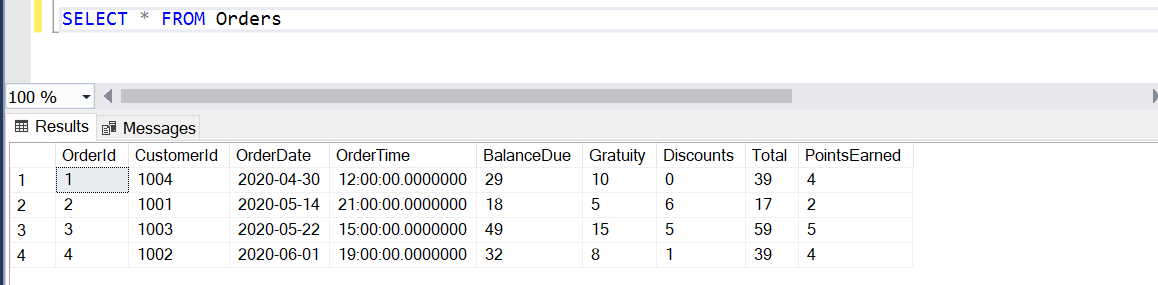


Joint Query with Orders Entity:

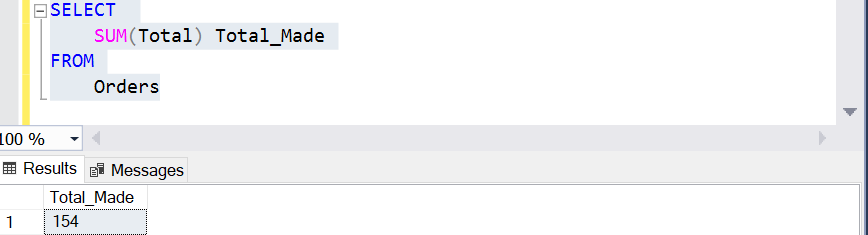


**Orders Table**

All Data:

****

Aggregate Function:



Joint Query with Menu\_Order Entity:

