Magdalena Skarbińska  
Sebastian Misztal  
Piotr Magiera

**Project of a Restaurant Database**

Table of contents

1. [Functionality 3](#_Toc94927661)

[I. Individual Clients 3](#_Toc94927662)

[II. Corporate Clients 3](#_Toc94927663)

[III. Staff 3](#_Toc94927664)

[IV. Restaurant Chef 3](#_Toc94927665)

1. [Database diagram 4](#_Toc94927666)
2. [Tables and check constraints 5](#_Toc94927667)

[I. Client 5](#_Toc94927668)

[II. Client\_Discounts 5](#_Toc94927669)

[III. Company 6](#_Toc94927670)

[IV. Discount\_Parameters 6](#_Toc94927671)

[V. Individual 7](#_Toc94927672)

[VI. Invoices 7](#_Toc94927673)

[VII. Menu 8](#_Toc94927674)

[VIII. Names 8](#_Toc94927675)

[IX. Orders 8](#_Toc94927676)

[X. Product\_On\_Menu 9](#_Toc94927677)

[XI. Products 9](#_Toc94927678)

[XII. Reservation\_Parameters 9](#_Toc94927679)

[XIII. Reservations 10](#_Toc94927680)

[XIV. Reserved\_Tables 10](#_Toc94927681)

[XV. Tables 11](#_Toc94927682)

[XVI. Relations between tables 11](#_Toc94927683)

1. [Views 12](#_Toc94927684)

[CurrentMenu 12](#_Toc94927685)

[OrdersForToday 12](#_Toc94927686)

1. [Procedures 13](#_Toc94927687)

[addDishToProducts 13](#_Toc94927688)

[AddOrderToReservation 13](#_Toc94927689)

[AddPersonToReservation 14](#_Toc94927690)

[addProductToMenu 14](#_Toc94927691)

[AddReservationToInvoice 15](#_Toc94927692)

[AddTableToReservation 15](#_Toc94927693)

[cancelReservation 16](#_Toc94927694)

[ChangeReservationStatus 17](#_Toc94927695)

[createCompanyClient 17](#_Toc94927696)

[createIndividualClient 18](#_Toc94927697)

[CreateInvoice 19](#_Toc94927698)

[createNewClient 20](#_Toc94927699)

[createReservation 20](#_Toc94927700)

[editDatesOfMenuSet 21](#_Toc94927701)

[EditProductAmountInReservation 21](#_Toc94927702)

[endReservation 22](#_Toc94927703)

[OrdersForTodaySortedByDate 22](#_Toc94927704)

[SelectSumOfAllProductsBetween 22](#_Toc94927705)

[ShowAllFreeTablesBetweenDates 22](#_Toc94927706)

1. [Functions 23](#_Toc94927707)

[generateReportMoney 23](#_Toc94927708)

[generateReportProducts 23](#_Toc94927709)

[getFreeTablesBetweenDates 23](#_Toc94927710)

[menuUntil 24](#_Toc94927711)

[CountOfAllFreeTablesBetweenDates 24](#_Toc94927712)

[getActualStatus 24](#_Toc94927713)

[getAmountOfFreeTables 24](#_Toc94927714)

[getFullReservationCost 25](#_Toc94927715)

[getHighestClientID 25](#_Toc94927716)

[getHighestEmployeeID 25](#_Toc94927717)

[getHighestIndividual 25](#_Toc94927718)

[getHighestMenuID 26](#_Toc94927719)

[getHighestReservationID 26](#_Toc94927720)

1. [Triggers 26](#_Toc94927721)

[TrgCancelReservation 26](#_Toc94927722)

1. [Indexes 27](#_Toc94927723)

[Reservation\_Status\_Index 27](#_Toc94927724)

[Reserved\_Tables\_Table 27](#_Toc94927725)

[Reserved\_Tables\_Reservation 27](#_Toc94927726)

1. [Roles 27](#_Toc94927727)

[Manager 27](#_Toc94927728)

[Customer 27](#_Toc94927729)

[Staff 27](#_Toc94927730)

# Functionality

### Individual Clients

* 1. placing an order online or on the spot
  2. access to menu
  3. making reservation (for at least 2 people)
  4. table reservation under certain circumstances
  5. confirming an order
  6. request for an invoice
  7. an overview of the order history

### Corporate Clients

* 1. placing an order online or on the spot
  2. access to menu
  3. table reservation (per company or per employee)
  4. confirming an order
  5. request for an invoice
  6. an overview of the order history

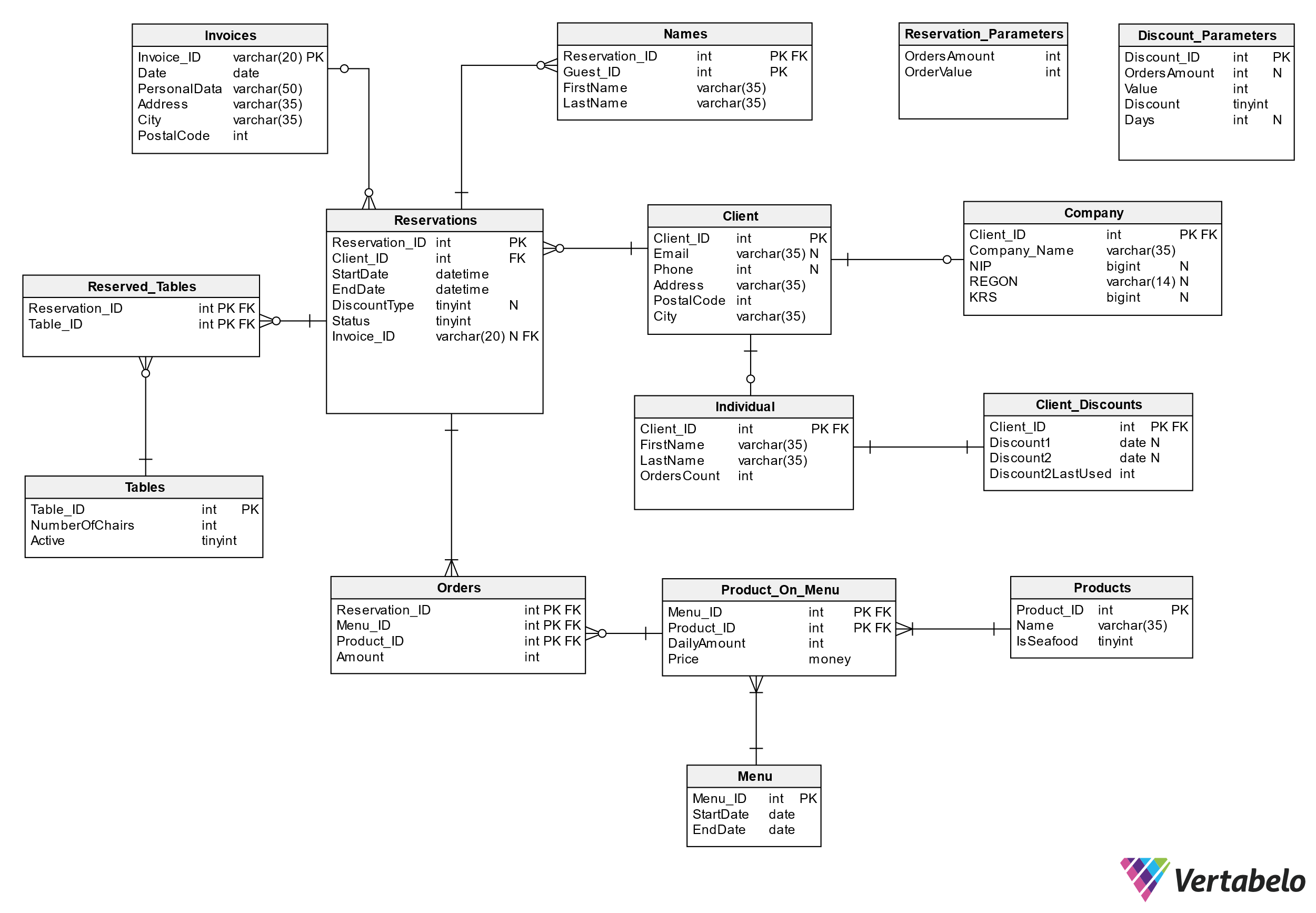
### Staff

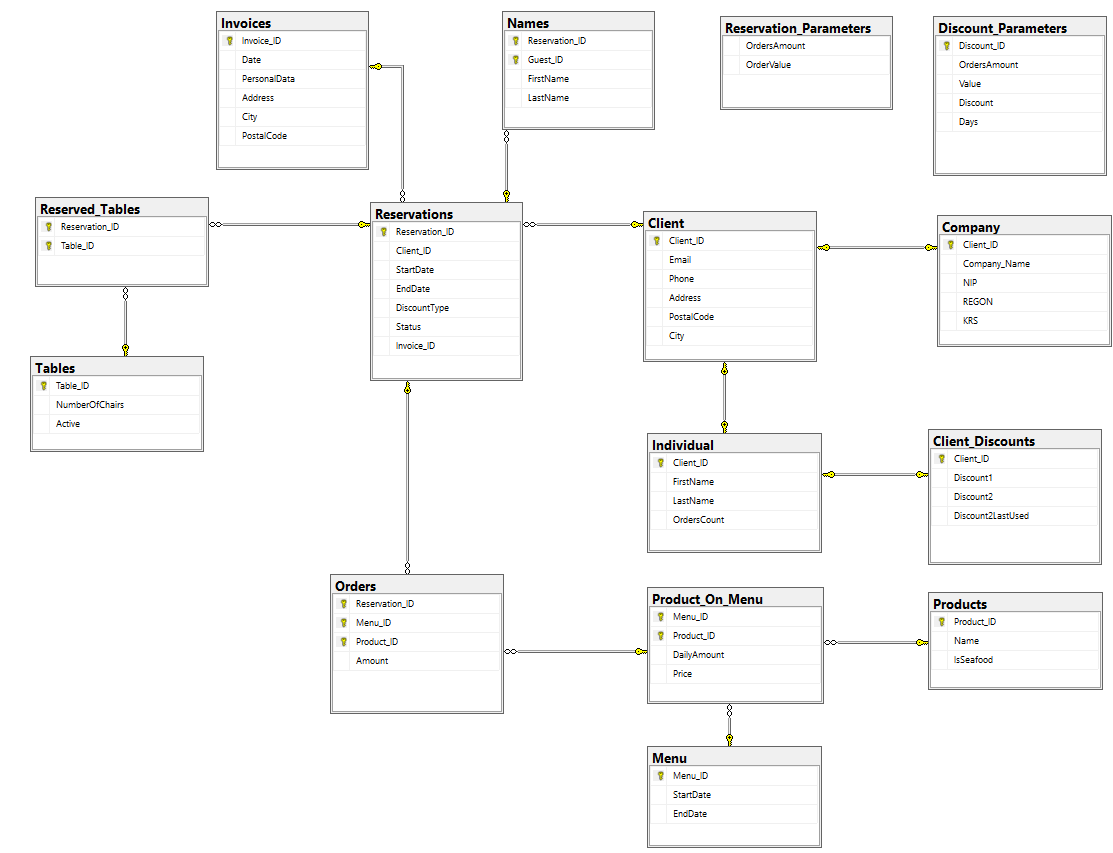
* 1. issuing an invoice for single reservations
  2. issuing a collective invoice
  3. acceptation of a reservation and assigning tables to it
  4. verifying reservation status (payment)
  5. taking orders in person

### Restaurant Chef

* 1. choosing menu
  2. generating monthly reports – income, average order value, number of reservations, amount of products sold (for each product)
  3. updating data

# Database diagram

****



# Tables and check constraints

### Client

Table with information about clients using the system.

**Client\_ID** – unique client identification number.

**Email** – email address.

**Phone** – phone number.

**Address** – street and house number.

**PostalCode** – postal code.

**City** – city of residence.

1. **CREATE** **TABLE** Client (
2. Client\_ID **int**  NOT NULL,
3. Email **varchar**(35)  NULL,
4. Phone **int**  NULL,
5. Address **varchar**(35)  NOT NULL,
6. PostalCode **int**  NOT NULL,
7. City **varchar**(35)  NOT NULL,
8. **CONSTRAINT** Email\_check **CHECK** (Email like '%[a-zA-Z0-9][@][a-zA-Z0-9]%[.][a-zA-Z0-9]%'),
9. **CONSTRAINT** Phone\_check **CHECK** (CAST(Phone **as** nvarchar) like  '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
10. **CONSTRAINT** PostalCode\_check **CHECK** (PostalCode like '[0-9][0-9][0-9][0-9][0-9]'),
11. **CONSTRAINT** Address\_check **CHECK** (Address not like '%[^a-zA-Z0-9. ]%'),
12. **CONSTRAINT** City\_check **CHECK** (City not like '%[^a-zA-Z ]%'),
13. **CONSTRAINT** Client\_pk **PRIMARY** **KEY**  (Client\_ID)
14. );

### Client\_Discounts

Table with information about discounts available for clients.

**Client\_ID** – unique client identification number.

**Discount1** – date the first discount is available from (assigned only once).

**Discount2** – date the second discount is available from (available to use for certain number of days – look up table named Discount\_Parameters).

**Discount2LastUsed** – unique reservation identification number the second discount was last used on.

1. **CREATE** **TABLE** Client\_Discounts (
2. Client\_ID **int**  NOT NULL,
3. Discount1 **date**  NULL,
4. Discount2 **date**  NULL,
5. Discount2LastUsed **int**  NOT NULL,
6. **CONSTRAINT** Discount2LastUsed\_Check **CHECK** (Discount2LastUsed >= 0),
7. **CONSTRAINT** Client\_Discounts\_pk **PRIMARY** **KEY**  (Client\_ID)
8. );

### Company

Table with information about corporate clients.

**Client\_ID** – unique corporate client identification number.

**Company\_Name** – company name.

**NIP, REGON, KRS** – Polish identification numbers (needed for invoices).

1. **CREATE** **TABLE** Company (
2. Client\_ID **int**  NOT NULL,
3. Company\_Name **varchar**(35)  NOT NULL,
4. NIP **bigint** NULL,
5. REGON **varchar**(14) NULL,
6. KRS **bigint** NULL,
7. **CONSTRAINT** NIP\_check **CHECK** (CAST(NIP **as** nvarchar) like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
8. **CONSTRAINT** REGON\_check **CHECK** (REGON like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
9. **CONSTRAINT** KRS\_check **CHECK** (CAST(KRS **as** nvarchar)  like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
10. **CONSTRAINT** Company\_pk **PRIMARY** **KEY**  (Client\_ID)
11. );

### Discount\_Parameters

Table with information about discount details.

**Discount\_ID** – unique discount identification number.

**OrdersAmount** – minimal number of reservations needed to obtain the discount.

**Value** – minimal value of reservation required for it to be accounted in OrdersAmount (0 in case of discount no. 1).

**Discount** – percentage value of the discount.

**Days** – number of days the discount is available for (NULL in case of infinite duration time).

1. **CREATE** **TABLE** Discount\_Parameters (
2. Discount\_ID **int**  NOT NULL,
3. OrdersAmount **int**  NULL,
4. Value **int**  NOT NULL,
5. Discount tinyint  NOT NULL,
6. Days **int**  NULL,
7. **CONSTRAINT** OrdersAmount\_Check2 **CHECK** (OrdersAmount > 0),
8. **CONSTRAINT** Value\_Check **CHECK** (Value > 0),
9. **CONSTRAINT** Discount\_Check **CHECK** (Discount > 0 and Discount <= 100),
10. **CONSTRAINT** Days\_Check **CHECK** (Days > 0),
11. **CONSTRAINT** Discount\_Parameters\_pk **PRIMARY** **KEY**  (Discount\_ID)
12. );

### Individual

Table with information about individual clients.

**Client\_ID** – unique client identification number.

**FirstName** – client’s first name.

**LastName** – client’s last name.

**OrdersCount** – number of reservations made by the client (used to determine if online reservation is possible).

1. **CREATE** **TABLE** Individual (
2. Client\_ID **int**  NOT NULL,
3. FirstName **varchar**(35)  NOT NULL,
4. LastName **varchar**(35)  NOT NULL,
5. OrdersCount **int**  NOT NULL,
6. **CONSTRAINT** OrdersCount\_Check **CHECK** (OrdersCount >= 0),
7. **CONSTRAINT** Individual\_pk **PRIMARY** **KEY**  (Client\_ID)
8. );

### Invoices

Table with information about issued invoices.

**Invoice\_ID** – unique invoice identification number.

**Date** – invoice issue date.

**PersonalData** – personal data on the invoice.

**Address** – street and house number.

**PostalCode** – postal code.

**City** – city of residence.

1. **CREATE** **TABLE** Invoices (
2. Invoice\_ID **varchar**(20)  NOT NULL,
3. **Date** **date**  NOT NULL,
4. PersonalData **varchar**(50)  NOT NULL,
5. Address **varchar**(35)  NOT NULL,
6. City **varchar**(35)  NOT NULL,
7. PostalCode **int**  NOT NULL,
8. **CONSTRAINT** Invoice\_ID\_Check **CHECK** (Invoice\_ID not like '%[^0-9/]%'),
9. **CONSTRAINT** Postal\_Code\_Check\_2 **CHECK** (PostalCode like '[0-9][0-9][0-9][0-9][0-9]'),
10. **CONSTRAINT** Address\_Check\_2 **CHECK** (Address not like '%[^a-zA-Z0-9. ]%'),
11. **CONSTRAINT** City\_Check\_2 **CHECK** (City not like '%[^a-zA-Z ]%'),
12. **CONSTRAINT** Invoices\_pk **PRIMARY** **KEY**  (Invoice\_ID)
13. );

### Menu

Table with information about menus.

**Menu\_ID** – unique menu identification number.

**StartDate** – date the menu starts being valid on (inclusive).

**EndDate** – date the menu stops being valid on (inclusive – still valid on EndDate).

1. **CREATE** **TABLE** Menu (
2. Menu\_ID **int**  NOT NULL,
3. StartDate **date**  NOT NULL,
4. EndDate **date**  NOT NULL,
5. **CONSTRAINT** Date\_Check **CHECK** (StartDate <= EndDate),
6. **CONSTRAINT** Menu\_pk **PRIMARY** **KEY**  (Menu\_ID)
7. );

### Names

Table with information about personal data of employees corporate clients made reservation for.

**First\_Name** – employee’s first name.

**Last\_Name** – employee’s last name.

1. **CREATE** **TABLE** Names (
2. Reservation\_ID **int**  NOT NULL,
3. Guest\_ID **int**  NOT NULL,
4. FirstName **varchar**(35)  NOT NULL,
5. LastName **varchar**(35)  NOT NULL,
6. **CONSTRAINT** Names\_pk **PRIMARY** **KEY**  (Guest\_ID,Reservation\_ID)
7. );

### Orders

Table with information about orders (note: one reservation may have many orders).

**Reservation\_ID** – unique reservation identification number.

**Menu\_ID** – unique menu identification number (menu that was valid when reservation was made).

**Product\_ID** – unique ordered product identification number.

**Amount** – amount of ordered product.

1. **CREATE** **TABLE** Orders (
2. Reservation\_ID **int**  NOT NULL,
3. Menu\_ID **int**  NOT NULL,
4. Product\_ID **int**  NOT NULL,
5. Amount **int**  NOT NULL,
6. **CONSTRAINT** Amount\_check **CHECK** (Amount > 0),
7. **CONSTRAINT** Orders\_pk **PRIMARY** **KEY**  (Reservation\_ID,Menu\_ID,Product\_ID)
8. );

### Product\_On\_Menu

Table with information about products on given menu.

**Menu\_ID** – unique menu identification number.

**Product\_ID** – unique product identification number.

**DailyAmount** – daily limit of the product (cannot sell more than that per day).

**Price** – price of the product in the given menu.

1. **CREATE** **TABLE** Product\_On\_Menu (
2. Menu\_ID **int**  NOT NULL,
3. Product\_ID **int**  NOT NULL,
4. DailyAmount **int**  NOT NULL,
5. Price money  NOT NULL,
6. **CONSTRAINT** DailyAmount\_Check **CHECK** (DailyAmount > 0),
7. **CONSTRAINT** Price\_Check **CHECK** (Price > 0),
8. **CONSTRAINT** Product\_On\_Menu\_pk **PRIMARY** **KEY**  (Menu\_ID,Product\_ID)
9. );

### Products

Table with information about product details.

**Product\_ID** – product identification number.

**Name** – product name.

**IsSeafood** – information about product category (if seafood equals 0 else 1).

1. **CREATE** **TABLE** Products (
2. Product\_ID **int IDENTITY**  NOT NULL,
3. **Name** **varchar**(35)  NOT NULL,
4. IsSeafood tinyint  NOT NULL,
5. **CONSTRAINT** Name\_Check **CHECK** (**Name** not like '%[^a-zA-Z ]%'),
6. **CONSTRAINT** IsSeafood\_Check **CHECK** (IsSeafood in (0, 1)),
7. **CONSTRAINT** Products\_pk **PRIMARY** **KEY**  (Product\_ID)
8. );

### Reservation\_Parameters

Table with information about conditions to make online reservation.

**OrdersAmount** – number of reservations required.

**OrderValue** – minimum value of reservations described above.

1. **CREATE** **TABLE** Reservation\_Parameters (
2. OrdersAmount **int**  NOT NULL,
3. OrderValue **int**  NOT NULL,
4. **CONSTRAINT** OrdersAmount\_Check **CHECK** (OrdersAmount > 0),
5. **CONSTRAINT** OrderValue\_Check **CHECK** (OrderValue > 0)
6. );

### Reservations

Table with information about clients’ reservations.

**Reservation\_ID** – unique reservation identification number.

**Client\_ID** – unique client identification number.

**StartDate** – datetime of the beginning of the reservation.

**EndDate** – datetime of the end of the reservation.

**DiscountType** – number of discount applied to the reservation.

**Status** – reservation status where: 0 - unpaid, takeaway , 1 - paid, takeaway, 2 - unpaid, on the spot, 3 - paid, on the spot, 4 - cancelled, refunded, 5 - cancelled, not refunded, 6 – paid, finished.

1. **CREATE** **TABLE** Reservations (
2. Reservation\_ID **int**  NOT NULL,
3. Client\_ID **int**  NOT NULL,
4. StartDate datetime  NOT NULL,
5. EndDate datetime  NOT NULL,
6. DiscountType tinyint  NULL,
7. Status tinyint  NOT NULL,
8. Invoice\_ID varchar(20) NULL,
9. **CONSTRAINT** Date\_Check2 **CHECK** (StartDate <= EndDate),
10. **CONSTRAINT** DiscountType\_Check **CHECK** (DiscountType in (NULL, 0, 1)),
11. **CONSTRAINT** Status\_Check **CHECK** (Status in (0, 1, 2, 3, 4, 5, 6)),
12. **CONSTRAINT** Reservations\_pk **PRIMARY** **KEY**  (Reservation\_ID)
13. );

### Reserved\_Tables

Table with information about reserved tables.

**Reservation\_ID** – unique reservation identification number the table is assigned to.

**Table\_ID** – unique table identification number.

1. **CREATE** **TABLE** Reserved\_Tables (
2. Reservation\_ID **int**  NOT NULL,
3. Table\_ID **int**  NOT NULL,
4. **CONSTRAINT** Reserved\_Tables\_pk **PRIMARY** **KEY**  (Reservation\_ID,Table\_ID)
5. );

### Tables

Table with information about tables in restaurant.

**Table\_ID** – unique table identification number.

**NumberOfChairs** – number of chairs assigned to the table.

**Active** – information about table usability (could be sold, broken etc.).

1. **CREATE** **TABLE** Tables (
2. Table\_ID **int**  NOT NULL,
3. NumberOfChairs **int**  NOT NULL,
4. Active tinyint  NOT NULL,
5. **CONSTRAINT** Active\_check **CHECK** (Active in (0, 1)),
6. **CONSTRAINT** Chairs\_check **CHECK** (NumberOfChairs > 0),
7. **CONSTRAINT** Tables\_pk **PRIMARY** **KEY**  (Table\_ID)
8. );

### Relations between tables

1. -- foreign keys
2. -- Reference: Client\_Discounts\_Clients (table: Client\_Discounts)
3. **ALTER** **TABLE** Client\_Discounts **ADD** **CONSTRAINT** Client\_Discounts\_Clients
4. **FOREIGN** **KEY** (Client\_ID)
5. **REFERENCES** Individual (Client\_ID);
7. -- Reference: Company\_Client (table: Company)
8. **ALTER** **TABLE** Company **ADD** **CONSTRAINT** Company\_Client
9. **FOREIGN** **KEY** (Client\_ID)
10. **REFERENCES** Client (Client\_ID);
12. -- Reference: Individual\_Client (table: Individual)
13. **ALTER** **TABLE** Individual **ADD** **CONSTRAINT** Individual\_Client
14. **FOREIGN** **KEY** (Client\_ID)
15. **REFERENCES** Client (Client\_ID);
17. -- Reference: Menu\_Menu\_date (table: Product\_On\_Menu)
18. **ALTER** **TABLE** Product\_On\_Menu **ADD** **CONSTRAINT** Menu\_Menu\_date
19. **FOREIGN** **KEY** (Menu\_ID)
20. **REFERENCES** Menu (Menu\_ID);
22. -- Reference: Orders\_Product\_On\_Menu (table: Orders)
23. **ALTER** **TABLE** Orders **ADD** **CONSTRAINT** Orders\_Product\_On\_Menu
24. **FOREIGN** **KEY** (Menu\_ID,Product\_ID)
25. **REFERENCES** Product\_On\_Menu (Menu\_ID,Product\_ID);
27. -- Reference: Orders\_Reservations (table: Orders)
28. **ALTER** **TABLE** Orders **ADD** **CONSTRAINT** Orders\_Reservations
29. **FOREIGN** **KEY** (Reservation\_ID)
30. **REFERENCES** Reservations (Reservation\_ID);
32. -- Reference: Product\_On\_Menu\_Products (table: Product\_On\_Menu)
33. **ALTER** **TABLE** Product\_On\_Menu **ADD** **CONSTRAINT** Product\_On\_Menu\_Products
34. **FOREIGN** **KEY** (Product\_ID)
35. **REFERENCES** Products (Product\_ID);
37. -- Reference: Reservations\_Client (table: Reservations)
38. **ALTER** **TABLE** Reservations **ADD** **CONSTRAINT** Reservations\_Client
39. **FOREIGN** **KEY** (Client\_ID)
40. **REFERENCES** Client (Client\_ID);
42. -- Reference: Reservations\_Invoices (table: Reservations)
43. **ALTER** **TABLE** Reservations **ADD** **CONSTRAINT** Reservations\_Invoices
44. **FOREIGN** **KEY** (Invoice\_ID)
45. **REFERENCES** Invoices (Invoice\_ID);
47. -- Reference: Reservations\_Names (table: Names)
48. **ALTER** **TABLE** Names **ADD** **CONSTRAINT** Reservations\_Names
49. **FOREIGN** **KEY** (Reservation\_ID)
50. **REFERENCES** Reservations (Reservation\_ID);
52. -- Reference: Reserved\_Tables\_Reservations (table: Reserved\_Tables)
53. **ALTER** **TABLE** Reserved\_Tables **ADD** **CONSTRAINT** Reserved\_Tables\_Reservations
54. **FOREIGN** **KEY** (Reservation\_ID)
55. **REFERENCES** Reservations (Reservation\_ID);
57. -- Reference: Reserved\_Tables\_Tables (table: Reserved\_Tables)
58. **ALTER** **TABLE** Reserved\_Tables **ADD** **CONSTRAINT** Reserved\_Tables\_Tables
59. **FOREIGN** **KEY** (Table\_ID)
60. **REFERENCES** Tables (Table\_ID);

# Views

### CurrentMenu

1. **CREATE VIEW** [dbo].[CurrentMenu] **as**
2. **SELECT** POM.Product\_ID, POM.Menu\_ID, M.StartDate, M.EndDate, POM.Price
3. **FROM**   dbo.Menu **AS** M
4. **INNER** JOIN dbo.Product\_On\_Menu **AS** POM **ON** M.Menu\_ID = POM.Menu\_ID
5. **WHERE** (CAST(M.EndDate **AS** **DATE**) >= CAST(GETDATE() **AS** **DATE**))

### OrdersForToday

1. **CREATE VIEW** [dbo].[OrdersForToday] **as**
2. **select** p.**Name**, o.Amount, r.StartDate
3. **from** Orders o
4. join Products p **on** p.Product\_ID = o.Product\_ID
5. left join Reservations r **on** r.Reservation\_ID = o.Reservation\_ID
6. **where** r.Status in(0,2)
7. and CAST( r.StartDate **AS** **DATE**) = CAST( GETDATE() **AS** **DATE**);

# Procedures

**Note from authors:**

We are aware that using identity property on ID of most tables would have simplified our code and would have increased efficiency (no need of using functions like getHighest(…)), although due to the tight deadline we decided to implement it only in Products tables.

### addDishToProducts

**IDENTITY used**

1. **CREATE** **PROCEDURE** [dbo].[addDishToProducts]
2. @**Name** **AS** **varchar**(35),
3. @isSeaFood **AS** tinyint
4. **AS**
5. **BEGIN**
6. **BEGIN** TRY
7. IF EXISTS
8. (
9. **SELECT** \* **FROM** Products
10. **WHERE** @**Name** = **Name**
11. )
12. **BEGIN**
13. ;THROW 52000, 'Product with this name is already in database.',1
14. **END**
16. **INSERT** **INTO** Products(**Name**, IsSeafood)
17. **VALUES** (@**Name**, @isSeaFood)
18. **END** TRY
19. **BEGIN** CATCH
20. **DECLARE** @errorMsg nvarchar(2048)
21. = 'Cannot add dish to products. Error: ' + ERROR\_MESSAGE();
22. THROW 52000, @errorMSg, 1;
23. **END** CATCH
24. **END**;

### AddOrderToReservation

1. **CREATE** **PROCEDURE** [dbo].[AddOrderToReservation]
2. @Reservation\_ID **int**,
3. @Product\_ID **int**,
4. @Amount **int**
5. **as**
6. **begin**
7. **set** nocount **on**
8. **begin** try
9. **insert** **into** Orders
10. (
11. Reservation\_ID,
12. Menu\_ID,
13. Product\_ID,
14. Amount
15. )
16. **values**
17. (
18. @Reservation\_ID,
19. (**select** pom.Menu\_ID **from** Product\_On\_Menu pom
20. join Menu m **on** m.Menu\_ID = pom.Menu\_ID
21. **where** (**select** StartDate **from** Reservations **where** Reservation\_ID = @Reservation\_ID)
22. between m.StartDate and m.EndDate),
23. @Product\_ID,
24. @Amount
25. )
26. **end** try
27. **begin** catch
28. **declare** @errorMsg nvarchar(2048)
29. = 'Cannot add order to reservation. Error message: '
30. + ERROR\_MESSAGE();
31. ;throw 52000, @errorMsg, 1
32. **end** catch
33. **end**

### AddPersonToReservation

1. **CREATE** **PROCEDURE** [dbo].[AddPersonToReservation]
2. @Reservation\_ID **int**,
3. @FirstName **varchar**(35),
4. @Lastname **varchar**(35)
5. **as**
6. **begin**
7. **set** nocount **on**
8. **begin** try
9. **insert** **into** Names
10. (
11. Reservation\_ID,
12. FirstName,
13. LastName
14. )
15. **values**
16. (
17. @Reservation\_ID,
18. @FirstName,
19. @Lastname
20. )
21. **end** try
22. **begin** catch
23. **declare** @errorMsg nvarchar(2048)
24. = 'Cannot add person to reservation. Error message: '
25. + ERROR\_MESSAGE();
26. ;throw 52000, @errorMsg, 1
27. **end** catch
28. **end**

### addProductToMenu

1. **CREATE** **PROCEDURE** [dbo].[addProductToMenu]
2. @ProductID **AS** **int**,
3. @DailyAmount **AS** **int**,
4. @Price **AS** money,
5. @StartDate **AS** **date**,
6. @EndDate **AS** **date**,
7. @MenuID **AS** **int**
8. **AS**
9. **BEGIN**
10. **BEGIN** TRY
12. IF EXISTS
13. (
14. **SELECT** \* **FROM** Product\_On\_Menu
15. JOIN Menu m **on** m.Menu\_ID = Product\_On\_Menu.Menu\_ID
16. **where** StartDate = @StartDate and Product\_ID = @ProductID
17. )
18. **BEGIN**
19. ;THROW 52000, 'Product with this id already exists in menu with this start date.',1
20. **END**
22. IF NOT(@StartDate < @EndDate)
23. **RETURN** 1;
25. **INSERT** **INTO** Menu(Menu\_ID, StartDate, EndDate)
26. **VALUES** (@MenuID , @StartDate, @EndDate)
28. **INSERT** **INTO** Product\_On\_Menu(Menu\_ID, Product\_ID, DailyAmount, Price)
29. **VALUES** (@MenuID , @ProductID, @DailyAmount, @Price)
31. **END** TRY
32. **BEGIN** CATCH
33. **DECLARE** @errorMsg nvarchar(2048)
34. = 'Cannot add dish to menu. Error: ' + ERROR\_MESSAGE();
35. THROW 52000, @errorMSg, 1;
36. **END** CATCH
38. **END**

### AddReservationToInvoice

1. **CREATE** **PROCEDURE** [dbo].[AddReservationToInvoice]
2. @Reservation\_ID **as** **int**,
3. @Invoice\_ID **as** **int**
4. **as**
5. **begin**
6. **begin** try
7. if exists
8. (**select** \* **from** Reservations **where** Reservation\_ID=@Reservation\_ID and Invoice\_ID **is** not null)
9. **begin**
10. ; throw 52000, 'reservation was already added to invoice',1
11. **end**
12. **update** Reservations
13. **set** Invoice\_ID = @Invoice\_ID
14. **WHERE** Reservation\_ID = @Reservation\_ID
15. **end** try
17. **begin** catch
18. **DECLARE** @errorMsg nvarchar(2048)='Cannot asign reservation to invoice. Error: ' + ERROR\_MESSAGE();
19. THROW 52000, @errorMsg, 1;
20. **END** CATCH
21. **end**

### AddTableToReservation

1. **CREATE** **PROCEDURE** [dbo].[AddTableToReservation]
2. @Reservation\_ID **int**,
3. @Table\_ID **int**
4. **as**
5. **begin**
6. **set** nocount **on**
7. **begin** try
8. if not exists
9. (
10. **select** \* **from** Reservations
11. **where** Reservation\_ID = @Reservation\_ID
12. )
13. **begin**
14. ;throw 52000, 'Reservation does not exist.', 1
15. **end**
17. if not exists
18. (
19. **select** \* **from** Tables
20. **where** Table\_ID = @Table\_ID
21. )
22. **begin**
23. ;throw 52000, 'Table does not exist or is not active.', 1
24. **end**
25. **DECLARE** @StartDate DATETIME;
26. **DECLARE** @EndDate DATETIME;
27. **SELECT** @StartDate=R2.StartDate, @EndDate=R2.EndDate **FROM** Reservations R2 **WHERE** R2.Reservation\_ID=@Reservation\_ID;
28. if exists
29. (
30. **select** \* **from** Reserved\_Tables rt
31. JOIN Reservations R2 **on** rt.Reservation\_ID = R2.Reservation\_ID
32. **where** ((R2.StartDate <= @StartDate
33. AND R2.EndDate >= @StartDate)
34. OR
35. (R2.StartDate <= @EndDate
36. AND R2.EndDate >= @EndDate)
37. OR
38. (R2.StartDate >= @StartDate
39. AND R2.EndDate <= @EndDate))
40. AND R2.Reservation\_ID <> @Reservation\_ID
41. AND rt.Table\_ID = @Table\_ID
42. )
43. **begin**
44. ;throw 52000, 'Table is not available at that time.', 1
45. **end**
47. **insert** **into** Reserved\_Tables
48. (
49. Reservation\_ID,
50. Table\_ID
51. )
52. **values**
53. (
54. @Reservation\_ID,
55. @Table\_ID
56. )
57. **end** try
58. **begin** catch
59. **declare** @errorMsg nvarchar(2048)
60. = 'Cannot add table to reservation. Error message: '
61. + ERROR\_MESSAGE();
62. ;throw 52000, @errorMsg, 1
63. **end** catch
64. **end**

### cancelReservation

1. **CREATE PROCEDURE** [dbo].[cancelReservation]
2. @ReservationID **AS** **int**
3. **AS**
4. **BEGIN**
5. **DECLARE** @Status **int**;
6. **SET** @Status = [dbo].getActualStatus(@ReservationID)
7. IF (@Status IN(0,2))
8. **UPDATE** Reservations
9. **SET** Status = 5
10. **WHERE** Reservation\_ID = @ReservationID
11. **ELSE**
12. IF (@Status IN (1,3))
13. **UPDATE** Reservations
14. **SET** Status = 4
15. **WHERE** Reservation\_ID = @ReservationID
16. **END**

### ChangeReservationStatus

1. **CREATE** **PROCEDURE** [dbo].[ChangeReservationStatus]
2. @Reservation\_ID **int**,
3. @Status tinyint
4. **as**
5. **begin**
6. **begin** try
7. if not exists
8. (
9. **select** \* **from** Reservations
10. **where** Reservation\_ID = @Reservation\_ID
11. )
12. **begin**
13. ;throw 52000, 'Reservation does not exist.', 1
14. **end**
16. **update** Reservations
17. **set** Status = @Status
18. **where** Reservation\_ID = @Reservation\_ID
19. **end** try
20. **begin** catch
21. **declare** @errorMsg nvarchar(2048)
22. = 'Cannot change reservation status. Error message: '
23. + ERROR\_MESSAGE();
24. ;throw 52000, @errorMsg, 1
25. **end** catch
26. **end**

### createCompanyClient

1. **CREATE PROCEDURE** [dbo].[createCompanyClient]
2. @Email **AS** **varchar**(35) = NULL,
3. @Phone **AS** **int** = NULL,
4. @Address **AS** **varchar**(35),
5. @PostalCode **AS** **int**,
6. @City **AS** **varchar**(35),
8. @CompanyName **AS** **varchar**(35),
9. @NIP **AS** **bigint** = NULL,
10. @REGON **AS** **varchar**(14) = NULL,
11. @KRS **AS** **bigint** = NULL
12. **AS**
13. **BEGIN** **TRANSACTION**
15. **BEGIN** TRY
16. **DECLARE** @ClientID **int**;
17. **SET** @ClientID = [dbo].getHighestClientID()
18. **EXEC** [dbo].createNewClient @Email, @Phone, @Address, @PostalCode, @City

21. IF EXISTS
22. (
23. **select** \* **from** Company
24. **where** @NIP = NIP
25. )
26. AND @NIP != NULL
27. **BEGIN**
28. **ROLLBACK** **TRANSACTION**
29. ;THROW 52000, 'Nip already in database.',1
30. **END**
32. IF EXISTS
33. (
34. **select** \* **from** Company
35. **where** @REGON = REGON
36. )
37. AND @REGON != NULL
38. **BEGIN**
39. **ROLLBACK** **TRANSACTION**
40. ;THROW 52000, 'regon already in database.',1
41. **END**
43. IF EXISTS
44. (
45. **select** \* **from** Company
46. **where** @KRS = KRS
47. )
48. AND @KRS != NULL
49. **BEGIN**
50. **ROLLBACK** **TRANSACTION**
51. ;THROW 52000, 'krs already in database.',1
52. **END**
54. **INSERT** **INTO** Company(Client\_ID, Company\_Name, NIP, REGON, KRS)
55. **VALUES** (@ClientID + 1, @CompanyName, @NIP, @REGON, @KRS)
57. **END** TRY
58. **BEGIN** CATCH
59. **DECLARE** @errorMsg nvarchar(2048)
60. = 'Cannot add company client. Error: ' + ERROR\_MESSAGE();
61. **ROLLBACK** **TRANSACTION**;
62. THROW 52000, @errorMSg, 1;
63. **END** CATCH
65. **COMMIT** **TRANSACTION**

### createIndividualClient

1. **CREATE PROCEDURE** [dbo].[createIndividualClient]
2. @Email **AS** **varchar**(35) = NULL,
3. @Phone **AS** **int** = NULL,
4. @Address **AS** **varchar**(35),
5. @PostalCode **AS** **int**,
6. @City **AS** **varchar**(35),
8. @FirstName **AS** **varchar**(35),
9. @LastName **AS** **varchar**(35)
10. **AS**
11. **BEGIN** **TRANSACTION**
12. **BEGIN** TRY
13. **DECLARE** @ClientID **int**;
14. **SET** @ClientID = [dbo].getHighestClientID();
15. **EXEC** [dbo].[createNewClient] @Email, @Phone, @Address, @PostalCode, @City
17. **INSERT** **INTO** Individual(Client\_ID, FirstName, LastName, OrdersCount)
18. **VALUES** (@ClientID + 1, @FirstName, @LastName, 0)
20. **INSERT** **INTO** Client\_Discounts(Client\_ID, Discount1, Discount2, Discount2LastUsed)
21. **VALUES** (@ClientID+1, NULL, NULL, 0)
22. **END** TRY
24. **BEGIN** CATCH
25. **DECLARE** @errorMsg nvarchar(2048)
26. = 'Cannot add individual client. Error: ' + ERROR\_MESSAGE();
27. **ROLLBACK** **TRANSACTION**;
28. THROW 52000, @errorMSg, 1;
29. **END** CATCH
30. **COMMIT** **TRANSACTION**;

### CreateInvoice

1. **CREATE** **PROCEDURE** [dbo].[CreateInvoice]
2. @CreationDate **as** **date**,
3. @Invoice\_ID **as** **varchar**(20),
4. @PersonalData **as** **varchar**(50),
5. @Address **as** **varchar**(35),
6. @City **as** **varchar**(35),
7. @PostalCode **as** **int**
8. **as**
9. **begin** **transaction**
11. **begin** try
12. **insert** **into** Invoices
13. (
14. Invoice\_ID,
15. **Date**,
16. PersonalData,
17. Address,
18. City,
19. PostalCode
20. )
21. **values**
22. (
23. @Invoice\_ID,
24. @CreationDate,
25. @PersonalData,
26. @Address,
27. @City,
28. @PostalCode
29. )
30. **end** try
32. **begin** catch
33. **declare** @errorMsg nvarchar(2048) = 'Cannot create invoice. Error message: '+ERROR\_MESSAGE();
34. **rollback** **transaction**;
35. ;throw 52000, @errorMsg, 1;
36. **end** catch
38. **commit** **transaction**

### createNewClient

1. **CREATE** **PROCEDURE** [dbo].[createNewClient]
2. @Email **AS** **varchar**(35) = NULL,
3. @Phone **AS** **int** = NULL,
4. @Address **AS** **varchar**(35),
5. @PostalCode **AS** **int**,
6. @City **AS** **varchar**(35)
7. **AS**
8. **BEGIN** **TRANSACTION**
9. **BEGIN** TRY
10. IF EXISTS
11. (
12. **SELECT** \* **FROM** Client
13. **WHERE** Email = @Email
14. )
15. **BEGIN**
16. **ROLLBACK** **TRANSACTION**;
17. ;THROW 52000, 'Email already exists.',1
18. **END**
20. IF EXISTS
21. (
22. **SELECT** \* **FROM** Client
23. **WHERE** Phone = @Phone
24. )
25. **BEGIN**
26. **ROLLBACK** **TRANSACTION**;
27. ;THROW 52000, 'Phone already exists.',1
28. **END**

31. **DECLARE** @ClientID **int**;
32. **SET** @ClientID = [dbo].getHighestClientID();
33. **INSERT** **INTO** Client(Client\_ID, Email, Phone, Address, PostalCode, City)
34. **VALUES**(@ClientID+1, @Email, @Phone, @Address, @PostalCode, @City);
36. **END** TRY
37. **BEGIN** CATCH
38. **DECLARE** @errorMsg nvarchar(2048)
39. = 'Cannot add client. Error: ' + ERROR\_MESSAGE();
40. **ROLLBACK** **TRANSACTION**;
41. THROW 52000, @errorMSg, 1;
42. **END** CATCH
43. **COMMIT** **TRANSACTION**

### createReservation

1. **CREATE** **PROCEDURE** [dbo].[createReservation]
2. @ClientID **AS** **int**,
3. @StartDate **AS** datetime,
4. @EndDate **AS** datetime,
5. @DiscountType **AS** tinyint,
6. @Status **AS** tinyint
7. **AS**
8. **BEGIN** **TRANSACTION**
10. **BEGIN** TRY
11. **DECLARE** @ReservationID **int**;
12. **SET** @ReservationID = [dbo].getHighestReservationID()
13. **INSERT** **INTO** Reservations(Reservation\_ID, Client\_ID, StartDate, EndDate, DiscountType, Status)
14. **VALUES** (@ReservationID + 1, @ClientID, @StartDate, @EndDate, @DiscountType, @Status)
15. **END** TRY
17. **BEGIN** CATCH
18. **DECLARE** @errorMsg nvarchar(2048)
19. = 'Cannot add reservation . Error: ' + ERROR\_MESSAGE();
20. **ROLLBACK** **TRANSACTION**;
21. THROW 52000, @errorMSg, 1;
23. **END** CATCH
25. **COMMIT** **TRANSACTION**

### editDatesOfMenuSet

1. **CREATE** **PROCEDURE** [dbo].[editDatesOfMenuSet]
2. @MenuID **AS** **int**,
3. @StartDate **AS** **date**,
4. @EndDate **AS** **date**
5. **AS**
6. **BEGIN**
7. IF (@MenuID <= [dbo].getHighestMenuID() AND @StartDate < @EndDate)
8. **UPDATE** Menu
9. **SET** StartDate = @StartDate, EndDate = @EndDate
10. **WHERE** Menu\_ID = @MenuID
11. **END**

### EditProductAmountInReservation

1. **CREATE** **PROCEDURE** [dbo].[EditProductAmountInReservation]
2. @Reservation\_ID **int**,
3. @Product\_ID **int**,
4. @Amount **int**
5. **as**
6. **begin**
7. **begin** try
8. if not exists
9. (
10. **select** \* **from** Reservations
11. **where** Reservation\_ID = @Reservation\_ID
12. )
13. **begin**
14. ;throw 52000, 'Reservation does not exist.', 1
15. **end**
16. if not exists
17. (
18. **select** \* **from** Orders o
19. **where** o.Reservation\_ID = @Reservation\_ID
20. )
21. **begin**
22. ;throw 52000, 'Order does not exist.', 1
23. **end**
24. **update** Orders
25. **set** Amount = @Amount
26. **where** Reservation\_ID = @Reservation\_ID and Product\_ID = @Product\_ID
27. **end** try
28. **begin** catch
29. **declare** @errorMsg nvarchar(2048)
30. = 'Cannot change order amount. Error message: '
31. + ERROR\_MESSAGE();
32. ;throw 52000, @errorMsg, 1
33. **end** catch
34. **end**

### endReservation

1. **CREATE** **PROCEDURE** [dbo].[endReservation]
2. @ReservationID **AS** **int**
3. **AS**
4. **BEGIN**
5. **BEGIN** TRY
6. IF NOT EXISTS
7. (
8. **SELECT** \* **FROM** Reservations
9. **WHERE** @ReservationID = Reservation\_ID
10. )
11. **BEGiN**
12. ;THROW 52000, 'No reservation with given reservation id',1
13. **END**
15. **UPDATE** Reservations
16. **SET** Status = 6
17. **WHERE** Reservation\_ID = @ReservationID
18. **END** TRY
20. **BEGIN** CATCH
21. **DECLARE** @errorMsg nvarchar(2048)
22. ='Cannot end reservation. Error: ' + ERROR\_MESSAGE();
23. THROW 52000, @errorMsg, 1;
24. **END** CATCH
26. **END**

### OrdersForTodaySortedByDate

1. **CREATE** **PROCEDURE** [dbo].[OrdersForTodaySortedByDate] **as**
2. **select** p.**Name**, o.Amount, r.StartDate
3. **from** Orders o
4. join Products p **on** p.Product\_ID = o.Product\_ID
5. left join Reservations r **on** r.Reservation\_ID = o.Reservation\_ID
6. **where** r.Status in(0,2)
7. and CAST( r.StartDate **AS** **DATE**) = CAST( GETDATE() **AS** **DATE**)
8. **order** **by** r.StartDate **asc**

### SelectSumOfAllProductsBetween

1. **CREATE** **PROCEDURE** [dbo].[SelectSumOfAllProductsBetween] @StartDate **Date**,
2. @EndDate **Date**
3. **AS**
4. **SELECT** P.**Name**, SUM(O.Amount) **AS** ilosc, SUM(O.Amount \* PoM.Price)
5. **FROM** Products P
6. JOIN Product\_On\_Menu PoM **ON** Pom.Product\_ID = P.Product\_ID
7. JOIN Orders O **ON** O.Product\_ID = PoM.Product\_ID AND O.Menu\_ID = PoM.Menu\_ID
8. JOIN Reservations R **ON** R.Reservation\_ID = O.Reservation\_ID
9. **WHERE** CAST(R.StartDate **AS** **DATE**) >= @StartDate
10. AND CAST(R.EndDate **AS** **DATE**) <= @EndDate
11. **GROUP** **BY** P.**Name**

### ShowAllFreeTablesBetweenDates

1. **CREATE** **PROCEDURE** [dbo].[ShowAllFreeTablesBetweenDates]
2. @InputStartDate datetime,
3. @InputEndDate datetime
4. **as**
5. **select** t.Table\_ID, t.NumberOfChairs
6. **from** Tables t
7. join Reserved\_Tables rt **on** rt.Table\_ID = t.Table\_ID
8. join Reservations r **on** r.Reservation\_ID = rt.Reservation\_ID
9. **where** t.Active=1 and r.EndDate <= @InputStartDate and r.StartDate >= @InputEndDate;

# Functions

### generateReportMoney

1. **CREATE** **FUNCTION** [dbo].[generateReportMoney](
2. @StartDate **AS** **date**
3. )
4. **RETURNS** **table**
5. **AS**
6. **return**(
7. **SELECT** SUM(O.Amount\*POM.Price) income, AVG(O.Amount\*POM.Price) avg\_order\_value, COUNT(R.Reservation\_ID) order\_count
8. **FROM** Reservations R
9. JOIN Orders O **on** O.Reservation\_ID = R.Reservation\_ID
10. JOIN Product\_On\_Menu POM **on** POM.Product\_ID = O.Product\_ID and POM.Menu\_ID = O.Menu\_ID
11. **WHERE** R.Status = 6 and CAST(R.EndDate **as** **date**) >= @StartDate and CAST(R.EndDate **as** **date**) <= dateadd(month, 1, @StartDate))

### generateReportProducts

1. **CREATE** **FUNCTION** [dbo].[generateReportProducts](
2. @StartDate **AS** **date**
3. )
4. **RETURNS** **table**
5. **AS**
6. **return**(
7. **SELECT** P.**Name** id\_of\_product, SUM(O.Amount) **as** units\_sold
8. **FROM** Reservations R
9. JOIN Orders O **on** O.Reservation\_ID = R.Reservation\_ID
10. JOIN Product\_On\_Menu POM **on** POM.Product\_ID = O.Product\_ID and POM.Menu\_ID = O.Menu\_ID
11. JOIN Products P **on** P.Product\_ID = POM.Product\_ID
12. **WHERE** R.Status = 6 and CAST(R.EndDate **as** **date**) >= @StartDate and CAST(R.EndDate **as** **date**) <= dateadd(month, 1, @StartDate)
13. **GROUP** **BY** P.Product\_ID, P.**Name**
14. )

### getFreeTablesBetweenDates

1. **CREATE FUNCTION** [dbo].[getFreeTablesBetweenDates](
2. @StartDate **AS** datetime,
3. @EndDate **AS** datetime
4. )
5. **RETURNS** **TABLE**
6. **AS**
7. **return**(
8. **SELECT** T.Table\_ID
9. **FROM** Tables T
10. **WHERE** T.Active = 1
12. **EXCEPT** (
13. **SELECT** **DISTINCT** T.Table\_ID
14. **FROM** Tables T
15. JOIN Reserved\_Tables RT **on** T.Table\_ID = RT.Table\_ID
16. JOIN Reservations R2 **on** RT.Reservation\_ID = R2.Reservation\_ID
17. **WHERE** (R2.StartDate<= @StartDate AND R2.EndDate >= @StartDate)
18. OR (R2.StartDate <= @EndDate AND R2.EndDate >= @EndDate)
19. )
20. )

### menuUntil

1. **CREATE FUNCTION** [dbo].[menuUntil](@EndDate **DATE**)
2. **RETURNS** **TABLE**
3. **AS**
4. **return**(**SELECT** c.Product\_ID, c.Menu\_ID, c.StartDate, c.EndDate
5. **FROM** currentMenu c
6. **WHERE** c.StartDate <= @EndDate)

### CountOfAllFreeTablesBetweenDates

1. **CREATE** **FUNCTION** [dbo].[CountOfAllFreeTablesBetweenDates]
2. (@InputStartDate datetime,
3. @InputEndDate datetime)
4. **returns** tinyint
5. **as**
6. **begin**
7. **return**
8. (**select** count(\*)
9. **from** Tables t
10. join Reserved\_Tables rt **on** rt.Table\_ID = t.Table\_ID
11. join Reservations r **on** r.Reservation\_ID = rt.Reservation\_ID
12. **where** t.Active=1 and r.EndDate <= @InputStartDate and r.StartDate >= @InputEndDate);
13. **end**

### getActualStatus

1. **CREATE** **FUNCTION** [dbo].[getActualStatus](
2. @ReservationID **AS** **int**
3. )
4. **RETURNS** **int**
5. **AS**
6. **BEGIN**
7. **return** (
8. **SELECT** R.Status
9. **FROM** Reservations R
10. **WHERE** R.Reservation\_ID = @ReservationID
11. )
12. **END**

### getAmountOfFreeTables

1. **CREATE FUNCTION** [dbo].[getAmountOfFreeTables](
2. @StartDate **AS** datetime,
3. @EndDate **AS** datetime,
4. @MinimalSize **AS** **int**
5. )
6. **RETURNS** **int**
7. **AS**
8. **BEGIN**
9. **return**(
10. **SELECT** COUNT(\*)
11. **FROM** [dbo].getFreeTablesBetweenDates(@StartDate, @EndDate) FT
12. JOIN Tables T **ON** T.Table\_ID = FT.Table\_ID
13. **WHERE** T.NumberOfChairs >= @MinimalSize
14. )
15. **END**

### getFullReservationCost

1. **CREATE FUNCTION** [dbo].[getFullReservationCost](@ReservationID **int**)
2. **RETURNS** **INT**
3. **AS**
4. **BEGIN**
5. **return**(**SELECT** SUM(o.Amount \* POM.Price) **FROM** Orders o
6. JOIN Product\_On\_Menu POM **ON** POM.Menu\_ID = o.Menu\_ID AND POM.Product\_ID = o.Product\_ID
7. **WHERE** o.Reservation\_ID = @ReservationID)
8. **END**

### getHighestClientID

1. **CREATE** **FUNCTION** [dbo].[getHighestClientID] ()
2. **RETURNS** **int**
3. **AS**
4. **BEGIN**
5. **return** (
6. **SELECT** **TOP** 1 C.Client\_ID
7. **FROM** Client C
8. **ORDER** **BY** C.Client\_ID **DESC**
9. )
10. **END**

### getHighestEmployeeID

1. **CREATE** **FUNCTION** [dbo].[getHighestEmployeeID]()
2. **RETURNS** **int**
3. **AS**
4. **BEGIN**
5. **return** (
6. **SELECT** **TOP** 1 N.Guest\_ID
7. **FROM** Names N
8. **ORDER** **BY** N.Guest\_ID **DESC**
9. )
10. **END**

### getHighestIndividual

1. **CREATE** **FUNCTION** [dbo].[getHighestIndividual] ()
2. **RETURNS** **int**
3. **AS**
4. **BEGIN**
5. **return** (
6. **SELECT** **TOP** 1 C.Client\_ID
7. **FROM** Individual C
8. **ORDER** **BY** C.Client\_ID **DESC**
9. )
10. **END**

### getHighestMenuID

1. **CREATE** **FUNCTION** [dbo].[getHighestMenuID]()
2. **RETURNS** **int**
3. **AS**
4. **BEGIN**
5. **return** (
6. **SELECT** **TOP** 1 PoM.Menu\_ID
7. **FROM** Product\_On\_Menu PoM
8. **ORDER** **BY** PoM.Menu\_ID **DESC**
9. )
10. **END**

### getHighestReservationID

1. **CREATE** **FUNCTION** [dbo].[getHighestReservationID]()
2. **RETURNS** **int**
3. **AS**
4. **BEGIN**
5. **return** (
6. **SELECT** **TOP** 1 R.Reservation\_ID
7. **FROM** Reservations R
8. **ORDER** **BY** R.Reservation\_ID **DESC**
9. )
10. **END**

# Triggers

### TrgCancelReservation

**Trigger incrementing order count for reservations meeting conditions for first discount.**

1. **CREATE** **TRIGGER** [dbo].[trgCancelReservation] **ON** [dbo].[Reservations]
2. **AFTER** **UPDATE**
3. **AS**
4. **BEGIN**
5. **DECLARE** @ClientID **int**;
6. **DECLARE** @ReservationID **int**;
7. **DECLARE** @Status tinyint;
9. IF ((**SELECT** COUNT(\*) **FROM** inserted) = 0)
10. THROW 51000, 'empty\_Update', 1;
12. IF ((**SELECT** COUNT(\*) **FROM** inserted) > 0)
13. **BEGIN**
14. **SELECT** @ClientID = i.Client\_ID, @ReservationID = i.Reservation\_ID, @Status = i.Status **FROM** inserted i;
15. IF((**SELECT** COUNT(\*) **FROM** [dbo].[Individual]) > 0
16. AND @Status = 6
17. AND [dbo].getFullReservationCost(@ReservationID) > (**SELECT** **TOP** 1 d.Value **FROM** [dbo].[Discount\_Parameters] d))
18. **UPDATE** Individual
19. **SET** OrdersCount += 1
20. **WHERE** Client\_ID = @ClientID;
21. **END**
22. **END**

# Indexes

### Reservation\_Status\_Index

1. **CREATE** **INDEX** Reservation\_Status\_Index **on** Reservations (Status);

### Reserved\_Tables\_Table

1. **CREATE** **INDEX** Reserved\_Tables\_Table **on** Reserved\_Tables(Table\_ID);

### Reserved\_Tables\_Reservation

1. **CREATE** **INDEX** Reserved\_Tables\_Reservation **on** Reserved\_Tables(Reservation\_ID);

# Roles

### Manager

1. **CREATE** ROLE manager
3. **GRANT** **EXECUTE** **to** manager
4. **GRANT** **SELECT**, **INSERT**, **UPDATE**, **DELETE**, **ALTER** **to** manager

### Customer

1. **CREATE** ROLE customer
2. **GRANT** **EXECUTE** **ON** [dbo].[AddPersonToReservation] **to** customer
3. **GRANT** **EXECUTE** **ON** [dbo].[cancelReservation]**to** customer
4. **GRANT** **EXECUTE** **ON** [dbo].[createCompanyClient]**to** customer
5. **GRANT** **EXECUTE** **ON** [dbo].[createIndividualClient]**to** customer
6. **GRANT** **SELECT** **ON** dbo.CurrentMenu **to** customer

### Staff

1. **CREATE** ROLE staff
2. **GRANT** **EXECUTE** **ON** [dbo].[addDishToProducts] **to** Staff
3. **GRANT** **EXECUTE** **ON** [dbo].[AddOrderToReservation] **to** staff
4. **GRANT** **EXECUTE** **ON** [dbo].[AddPersonToReservation] **to** staff
5. **GRANT** **EXECUTE** **ON** [dbo].[addProductToMenu]**to** staff
6. **GRANT** **EXECUTE** **ON** [dbo].[AddReservationToInvoice]**to** staff
7. **GRANT** **EXECUTE** **ON** [dbo].[AddTableToReservation]**to** staff
8. **GRANT** **EXECUTE** **ON** [dbo].[cancelReservation]**to** staff
9. **GRANT** **EXECUTE** **ON** [dbo].[ChangeReservationStatus]**to** staff
10. **GRANT** **EXECUTE** **ON** [dbo].[createCompanyClient]**to** staff
11. **GRANT** **EXECUTE** **ON** [dbo].[createIndividualClient]**to** staff
12. **GRANT** **EXECUTE** **ON** [dbo].[CreateInvoice]**to** staff
13. **GRANT** **EXECUTE** **ON** [dbo].[createNewClient]**to** staff
14. **GRANT** **EXECUTE** **ON** [dbo].[createReservation]**to** staff
15. **GRANT** **EXECUTE** **ON** [dbo].[editDatesOfMenuSet]**to** staff
16. **GRANT** **EXECUTE** **ON** [dbo].[EditProductAmountInReservation]**to** staff
17. **GRANT** **EXECUTE** **ON** [dbo].[endReservation]**to** staff
18. **GRANT** **EXECUTE** **ON** [dbo].[OrdersForTodaySortedByDate]**to** staff
19. **GRANT** **EXECUTE** **ON** [dbo].[SelectSumOfAllProductsBetween]**to** staff
20. **GRANT** **EXECUTE** **ON** [dbo].[ShowAllFreeTablesBetweenDates]**to** staff

23. **GRANT** **SELECT** **ON** dbo.CurrentMenu **to** staff
24. **GRANT** **SELECT** **ON** dbo.OrdersForToday **to** staff
26. **GRANT** **SELECT** **ON** [dbo].[getFreeTablesBetweenDates] **to** staff
27. **GRANT** **SELECT** **ON** [dbo].[menuUntil] **to** staff
29. **GRANT** **EXECUTE** **ON** [dbo].[CountOfAllFreeTablesBetweenDates] **to** staff
30. **GRANT** **EXECUTE** **ON** [dbo].[getActualStatus] **to** staff
31. **GRANT** **EXECUTE** **ON** [dbo].[getAmountOfFreeTables] **to** staff
32. **GRANT** **EXECUTE** **ON** [dbo].[getFullReservationCost] **to** staff