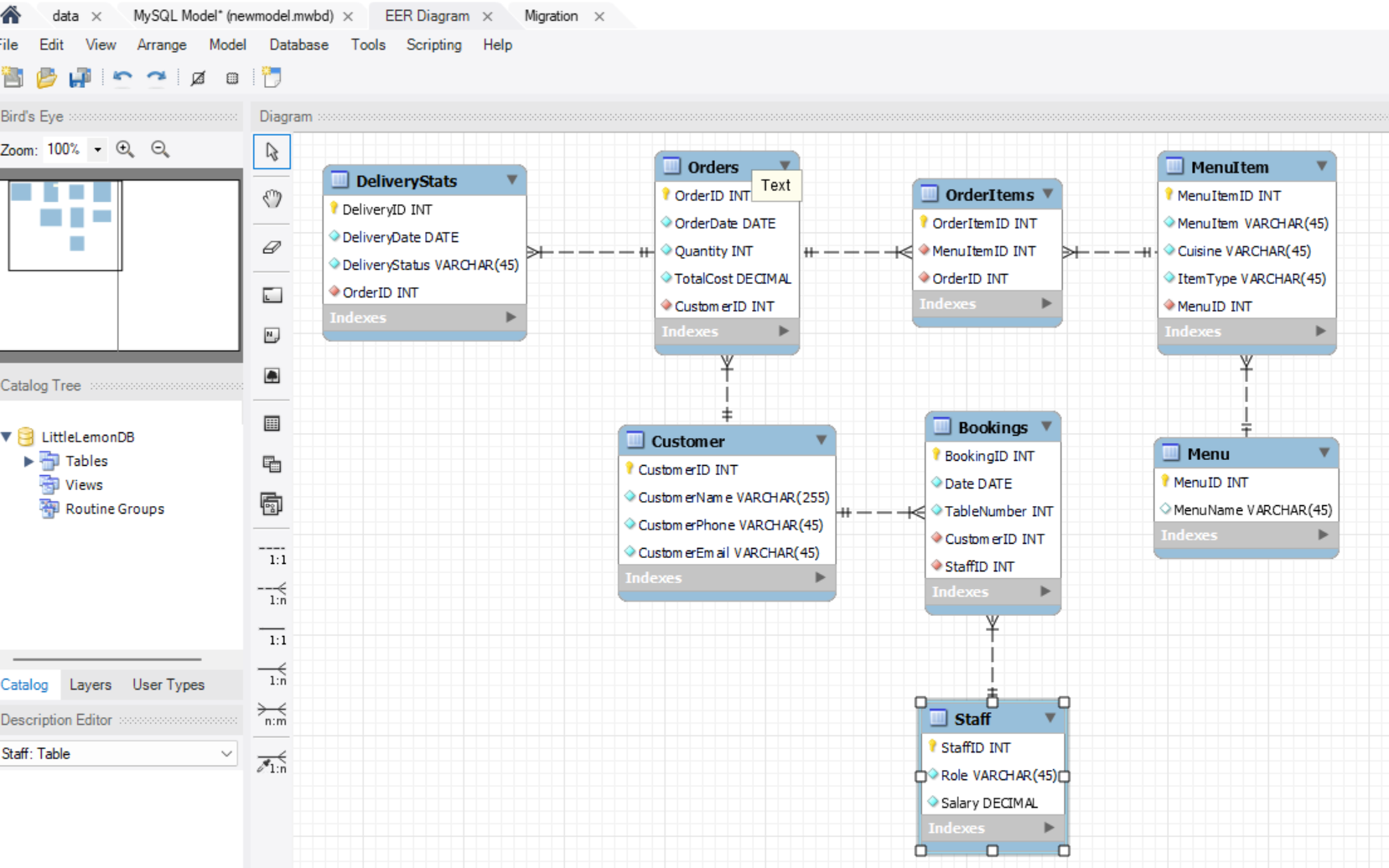
1. ER Diagram



1. WorkBench Forward Engineering:

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema LittleLemonDB

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema LittleLemonDB

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `LittleLemonDB` DEFAULT CHARACTER SET utf8 ;

USE `LittleLemonDB` ;

-- -----------------------------------------------------

-- Table `LittleLemonDB`.`Customer`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `LittleLemonDB`.`Customer` (

`CustomerID` INT NOT NULL AUTO\_INCREMENT,

`CustomerName` VARCHAR(255) NOT NULL,

`CustomerPhone` VARCHAR(45) NOT NULL,

`CustomerEmail` VARCHAR(45) NOT NULL,

PRIMARY KEY (`CustomerID`),

UNIQUE INDEX `CustomerID\_UNIQUE` (`CustomerID` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `LittleLemonDB`.`Staff`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `LittleLemonDB`.`Staff` (

`StaffID` INT NOT NULL AUTO\_INCREMENT,

`Role` VARCHAR(45) NOT NULL,

`Salary` DECIMAL NOT NULL,

PRIMARY KEY (`StaffID`),

UNIQUE INDEX `StaffID\_UNIQUE` (`StaffID` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `LittleLemonDB`.`Bookings`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `LittleLemonDB`.`Bookings` (

`BookingID` INT NOT NULL AUTO\_INCREMENT,

`Date` DATE NOT NULL,

`TableNumber` INT NOT NULL,

`CustomerID` INT NOT NULL,

`StaffID` INT NOT NULL,

PRIMARY KEY (`BookingID`),

UNIQUE INDEX `BookingID\_UNIQUE` (`BookingID` ASC) VISIBLE,

INDEX `customer\_booking\_fk\_idx` (`CustomerID` ASC) VISIBLE,

INDEX `staff\_fk\_idx` (`StaffID` ASC) VISIBLE,

CONSTRAINT `customer\_booking\_fk`

FOREIGN KEY (`CustomerID`)

REFERENCES `LittleLemonDB`.`Customer` (`CustomerID`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `staff\_fk`

FOREIGN KEY (`StaffID`)

REFERENCES `LittleLemonDB`.`Staff` (`StaffID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `LittleLemonDB`.`Orders`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `LittleLemonDB`.`Orders` (

`OrderID` INT NOT NULL AUTO\_INCREMENT,

`OrderDate` DATE NOT NULL,

`Quantity` INT NOT NULL,

`TotalCost` DECIMAL NOT NULL,

`CustomerID` INT NOT NULL,

PRIMARY KEY (`OrderID`),

UNIQUE INDEX `OrderID\_UNIQUE` (`OrderID` ASC) VISIBLE,

INDEX `customer\_fk\_idx` (`CustomerID` ASC) VISIBLE,

CONSTRAINT `customer\_fk`

FOREIGN KEY (`CustomerID`)

REFERENCES `LittleLemonDB`.`Customer` (`CustomerID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `LittleLemonDB`.`DeliveryStats`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `LittleLemonDB`.`DeliveryStats` (

`DeliveryID` INT NOT NULL AUTO\_INCREMENT,

`DeliveryDate` DATE NOT NULL,

`DeliveryStatus` VARCHAR(45) NOT NULL,

`OrderID` INT NOT NULL,

PRIMARY KEY (`DeliveryID`),

UNIQUE INDEX `DeliveryID\_UNIQUE` (`DeliveryID` ASC) VISIBLE,

INDEX `delivery\_order\_fk\_idx` (`OrderID` ASC) VISIBLE,

CONSTRAINT `delivery\_order\_fk`

FOREIGN KEY (`OrderID`)

REFERENCES `LittleLemonDB`.`Orders` (`OrderID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `LittleLemonDB`.`Menu`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `LittleLemonDB`.`Menu` (

`MenuID` INT NOT NULL AUTO\_INCREMENT,

`MenuName` VARCHAR(45) NULL,

PRIMARY KEY (`MenuID`),

UNIQUE INDEX `MenuID\_UNIQUE` (`MenuID` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `LittleLemonDB`.`MenuItem`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `LittleLemonDB`.`MenuItem` (

`MenuItemID` INT NOT NULL AUTO\_INCREMENT,

`MenuItem` VARCHAR(45) NOT NULL,

`Cuisine` VARCHAR(45) NOT NULL,

`ItemType` VARCHAR(45) NOT NULL,

`MenuID` INT NOT NULL,

PRIMARY KEY (`MenuItemID`),

UNIQUE INDEX `MenuID\_UNIQUE` (`MenuItemID` ASC) VISIBLE,

INDEX `menuitem\_menu\_fk\_idx` (`MenuID` ASC) VISIBLE,

CONSTRAINT `menuitem\_menu\_fk`

FOREIGN KEY (`MenuID`)

REFERENCES `LittleLemonDB`.`Menu` (`MenuID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `LittleLemonDB`.`OrderItems`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `LittleLemonDB`.`OrderItems` (

`OrderItemID` INT NOT NULL,

`MenuItemID` INT NOT NULL,

`OrderID` INT NOT NULL,

PRIMARY KEY (`OrderItemID`),

INDEX `orders\_orderitem\_fk\_idx` (`OrderID` ASC) VISIBLE,

INDEX `orderitems\_menuitem\_fk\_idx` (`MenuItemID` ASC) VISIBLE,

CONSTRAINT `orders\_orderitem\_fk`

FOREIGN KEY (`OrderID`)

REFERENCES `LittleLemonDB`.`Orders` (`OrderID`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `orderitems\_menuitem\_fk`

FOREIGN KEY (`MenuItemID`)

REFERENCES `LittleLemonDB`.`MenuItem` (`MenuItemID`)

ON DELETE CASCADE

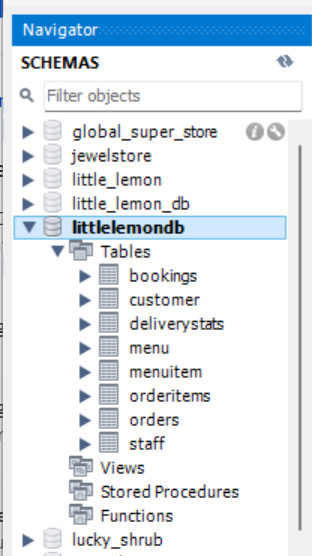
ON UPDATE CASCADE)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;



1. Populating Data:

INSERT INTO Customer

VALUES

(1, 'Vanessa McCarthy', 757536378, 'vm@mangatagallo.com'),

(2, 'Marcos Romero', 757536379, 'mr@mangatagallo.com'),

(3, 'Hiroki Yamane', 757536376, 'hy@mangatagallo.com'),

(4, 'Anna Iversen', 757536375, 'ai@mangatagallo.com'),

(5, 'Diana Pinto', 757536374, 'dp@mangatagallo.com'),

(6, 'Matthew Anders', 5614836453, 'goodmat@gmail.com'),

(7, 'Kassandra Saint', 5613604241, 'kassandrastripper@gmail.com'),

(8, 'Dmytro Salko', 5614609421, 'dm.salko@gmail.com');

INSERT INTO orders

VALUES

(1, '2022-10-15', 1, 100, 1),

(2, '2022-10-16', 2, 150, 2),

(3, '2022-10-17', 1, 200, 3),

(4, '2022-10-17', 1, 150, 4),

(5, '2022-10-18', 3, 230, 5),

(6, '2023-01-15', 2, 180, 6),

(7, '2023-01-16', 3, 310, 7),

(8, '2023-02-17', 1, 90, 8),

(9, '2023-02-17', 2, 120, 2),

(10, '2023-02-18', 4, 212, 7);

INSERT INTO DeliveryStats

VALUES

(1, '2022-10-17', 'Done', 1),

(2, '2022-10-20', 'Done', 2),

(3, '2022-10-22', 'Done', 3),

(4, '2022-10-25', 'Done', 4),

(5, '2022-10-27', 'Done', 5),

(6, '2022-11-30', 'In Progress', 6),

(7, '2023-01-15', 'In Cue', 7),

(8, '2023-12-24', 'In Progress', 8),

(9, '2023-12-24', 'Done', 9),

(10, '2023-12-24', 'In Cue', 10);

INSERT INTO Staff

VALUES

(1, 'Waiter', 42000),

(2, 'Assistant', 32000),

(3, 'Cook', 50100),

(4, 'Waiter', 36000),

(5, 'Manager', 73000);

INSERT INTO Bookings

VALUES

(1, '2023-12-04', 2, 7, 1),

(2, '2023-12-04', 1, 8, 2),

(3, '2023-12-04', 3, 5, 1),

(4, '2023-12-05', 4, 1, 3),

(5, '2023-12-05', 1, 2, 2),

(6, '2023-12-06', 1, 4, 4),

(7, '2023-12-06', 2, 3, 1),

(8, '2023-12-06', 4, 6, 2);

INSERT INTO Menu

VALUES

(1, 'Breakfast'),

(2, 'Lunch'),

(3, 'Dinner');

INSERT INTO MenuItem

VALUES

(1, 'Olives', 'Italian','Starters', 3),

(2, 'Flatbread', 'Italian', 'Starters', 2),

(3, 'Minestrone', 'Italian', 'Starters', 2),

(4, 'Tomato bread', 'American', 'Starters', 1),

(5, 'Falafel', 'Turkish', 'Starters', 3),

(6, 'Hummus', 'Kosher', 'Starters', 1),

(7, 'Greek salad', 'Greek', 'Main Courses', 2),

(8, 'Bean soup', 'Mexican', 'Main Courses', 2),

(9, 'Pizza', 'Italian', 'Main Courses', 3),

(10, 'Greek yoghurt', 'Greek','Desserts', 1),

(11, 'Ice cream', 'Turkish', 'Desserts', 2),

(12, 'Cheesecake', 'Italian', 'Desserts', 3),

(13, 'Athens White wine', 'Greek', 'Drinks', 3),

(14, 'Corfu Red Wine', 'French', 'Drinks', 3),

(15, 'Turkish Coffee', 'Italian', 'Drinks', 1),

(16, 'Turkish Delight', 'Turkish', 'Drinks', 1),

(17, 'Kabasa', 'Turkish', 'Main Courses', 2),

(18, 'Borscht', 'Ukrainian', 'Main Course', 2);

INSERT INTO OrderItems

VALUES

(1, 18, 10),

(2, 15, 10),

(3, 4, 10),

(4, 17, 1),

(5, 14, 1),

(6, 13, 2),

(7, 1, 3),

(8, 11, 3),

(9, 4, 3),

(10, 10, 4),

(11, 8, 5),

(12, 9, 5),

(13, 16, 6),

(14, 12, 6),

(15, 4, 6),

(16, 7, 7),

(17, 6, 8),

(18, 3, 8),

(19, 2, 9),

(20, 15, 9),

(21, 14, 9);

1. Creating View OrderStats:

SHOW DATABASES;

USE littlelemondb;

DROP VIEW IF EXISTS OrderStats;

CREATE VIEW OrderStats AS SELECT OrderID, Quantity, TotalCost FROM Orders WHERE Quantity >= 2;

SELECT \* FROM OrderStats;

1. Joining Multiple Tables:

SELECT Customer.CustomerID, Customer.CustomerName, Orders.OrderID,

Orders.TotalCost, MenuItem.MenuItem, MenuItem.Cuisine, MenuItem.ItemType

FROM Customer JOIN Orders JOIN MenuItem JOIN OrderItems

ON (Customer.CustomerID = Orders.CustomerID

AND Orders.OrderID = OrderItems.OrderID

AND OrderItems.MenuItemID = MenuItem.MenuItemID)

WHERE Orders.TotalCost > 150

ORDER BY Orders.TotalCost ASC;

1. Subqueries:

SELECT MenuItem FROM MenuItem JOIN OrderItems ON MenuItem.MenuItemID = OrderItems.MenuItemID WHERE OrderID = ANY

(SELECT OrderID FROM Orders WHERE Quantity >= 2);

1. Creating Procedure GetMaximumQuantity:

CREATE PROCEDURE GetMaxQuantity() SELECT MAX(Quantity) FROM Orders;

CALL GetMaxQuantity();

1. Creating Prepared Statement GetOrderDetails:

PREPARE GetOrderDetails FROM 'SELECT OrderID, Quantity, TotalCost FROM Orders WHERE OrderID = ?';

SET @id = 1;

EXECUTE GetOrderDetails USING @id;

1. Creating Stored Procedure CancelOrder:

DROP PROCEDURE IF EXISTS CancelOrder;

DELIMITER //

CREATE PROCEDURE CancelOrder(ID INT)

BEGIN

DELETE FROM Orders WHERE OrderID = ID;

SELECT CONCAT('Order ', ID, ' is cancelled') AS 'Confirmation';

END //

DELIMITER ;

CALL CancelOrder(12);

1. Adding Data to Bookings Table:

INSERT INTO Bookings

VALUES

(9, '2022-10-10', 5, 1, 2),

(10, '2022-11-12', 3, 3, 1),

(11, '2022-10-11', 2, 2, 3),

(12, '2022-10-13', 2, 1, 5);

1. Creating Stored Procedure CheckBooking:

DROP PROCEDURE IF EXISTS CheckBooking;

DELIMITER //

CREATE PROCEDURE CheckBooking(IN dt DATE, IN tble INT)

BEGIN

IF dt = (SELECT Date FROM Bookings WHERE Date = dt AND TableNumber = tble) THEN SELECT CONCAT('Table #', tble, ' is booked for ', dt) AS 'Booking Status';

END IF;

END //

DELIMITER ;

CALL CheckBooking('2022-10-10', 5);

1. Creating Stored Procedure AddValidBooking With the Transaction:

DROP PROCEDURE IF EXISTS AddValidBooking;

DELIMITER //

CREATE PROCEDURE AddValidBooking(dt DATE, tble INT)

BEGIN

DECLARE id INT;

START TRANSACTION;

SELECT (MAX(BookingID) + 1) FROM Bookings INTO id;

IF dt = (SELECT Date FROM Bookings WHERE Date = dt AND TableNumber = tble)

THEN SELECT CONCAT('Table #', tble, ' is already booked for ', dt, ' - booking cancelled!') AS 'Booking Status';

ROLLBACK;

ELSE

INSERT INTO Bookings

VALUES

(id, dt, tble, 1, 1);

SELECT CONCAT('Table #', tble, ' was booked for ', dt) AS 'Booking Status';

END IF;

END //

DELIMITER ;

CALL AddValidBooking('2024-12-04', 2);

1. Creating Stored Procedure AddBooking:

DROP PROCEDURE IF EXISTS AddBooking;

DELIMITER //

CREATE PROCEDURE AddBooking(id INT, dt DATE, tble INT, c\_id INT, s\_id INT)

BEGIN

INSERT INTO Bookings

VALUES

(id, dt, tble, c\_id, s\_id);

SELECT CONCAT('New Booking # ', id , ' was addded') AS 'Confirmation';

END //

DELIMITER ;

CALL AddBooking(9, '2022-10-10', 3, 4, 2);

1. Creating Stored Procedure UpdateBooking:

DROP PROCEDURE IF EXISTS UpdateBooking;

DELIMITER //

CREATE PROCEDURE UpdateBooking(id INT, dt DATE)

BEGIN

UPDATE Bookings SET Date = dt WHERE BookingID = id;

SELECT CONCAT('Booking # ', id , ' was updated') AS 'Confirmation';

END //

DELIMITER ;

CALL UpdateBooking(9, '2022-12-17');

1. Creating Stored Procedure CancelBooking:

DROP PROCEDURE IF EXISTS CancelBooking;

DELIMITER //

CREATE PROCEDURE CancelBooking(id INT)

BEGIN

DELETE FROM Bookings WHERE BookingID = id;

SELECT CONCAT('Booking #', id , ' cancelled') AS 'Confirmation';

END //

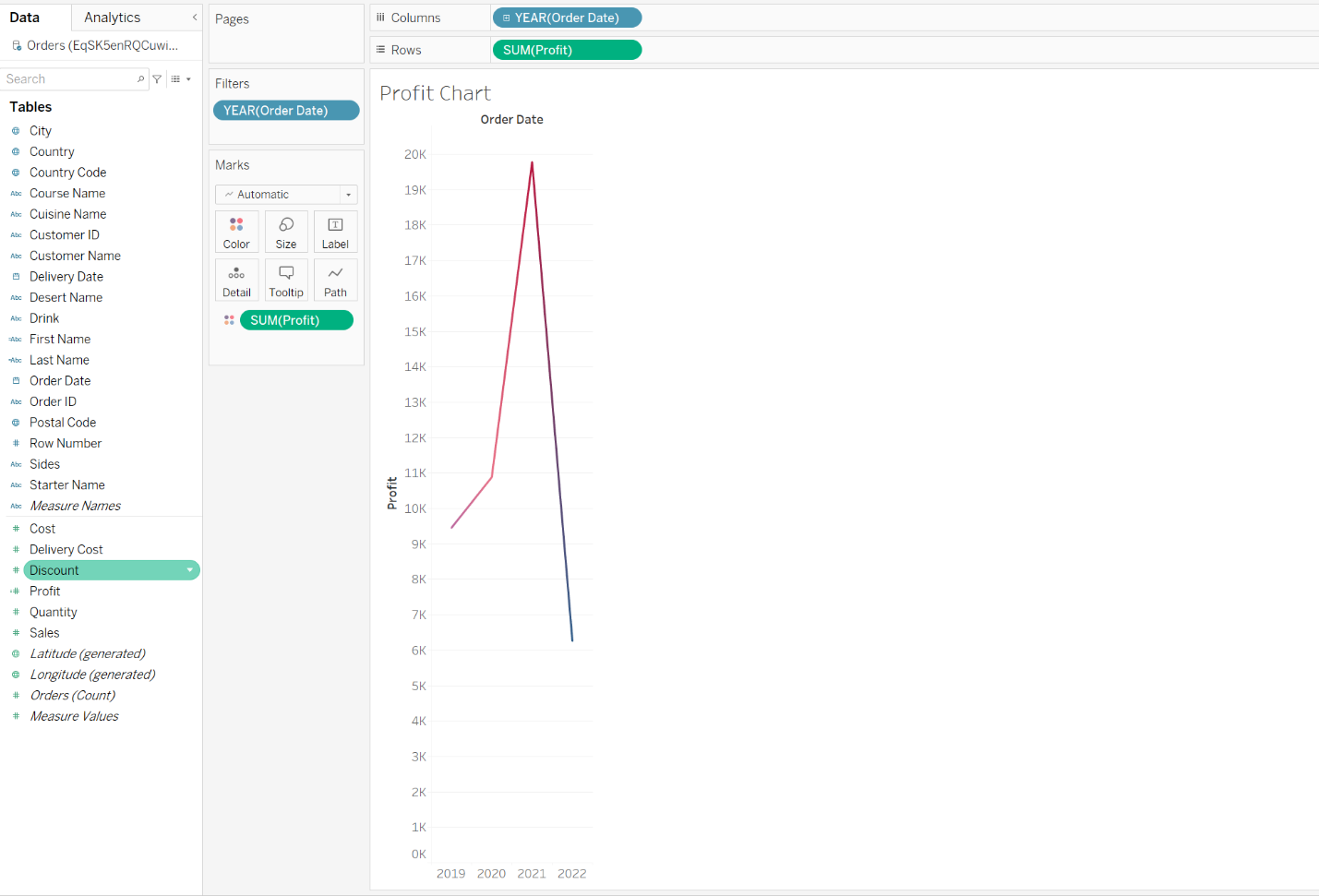
DELIMITER ;

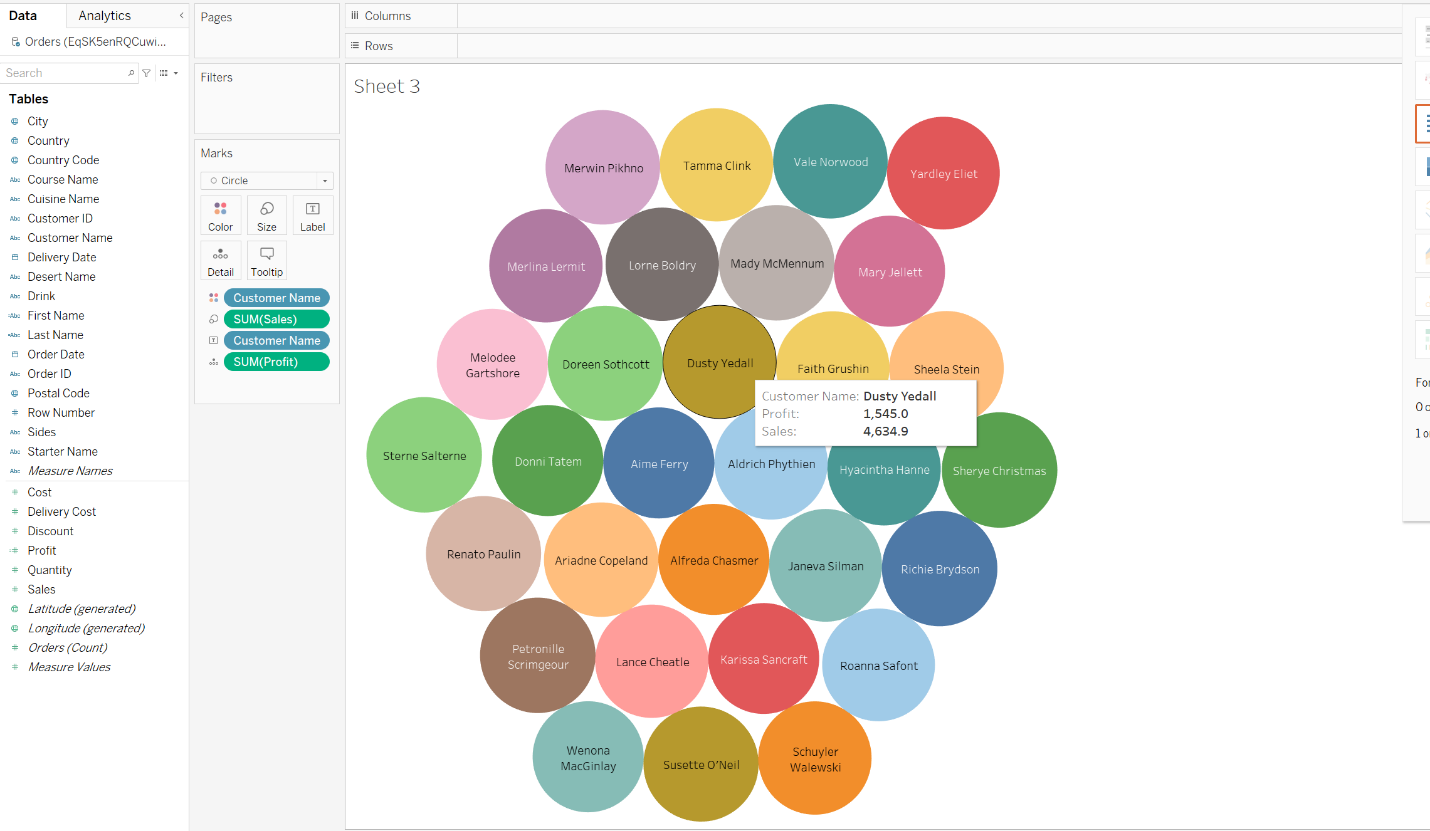
CALL CancelBooking(9);

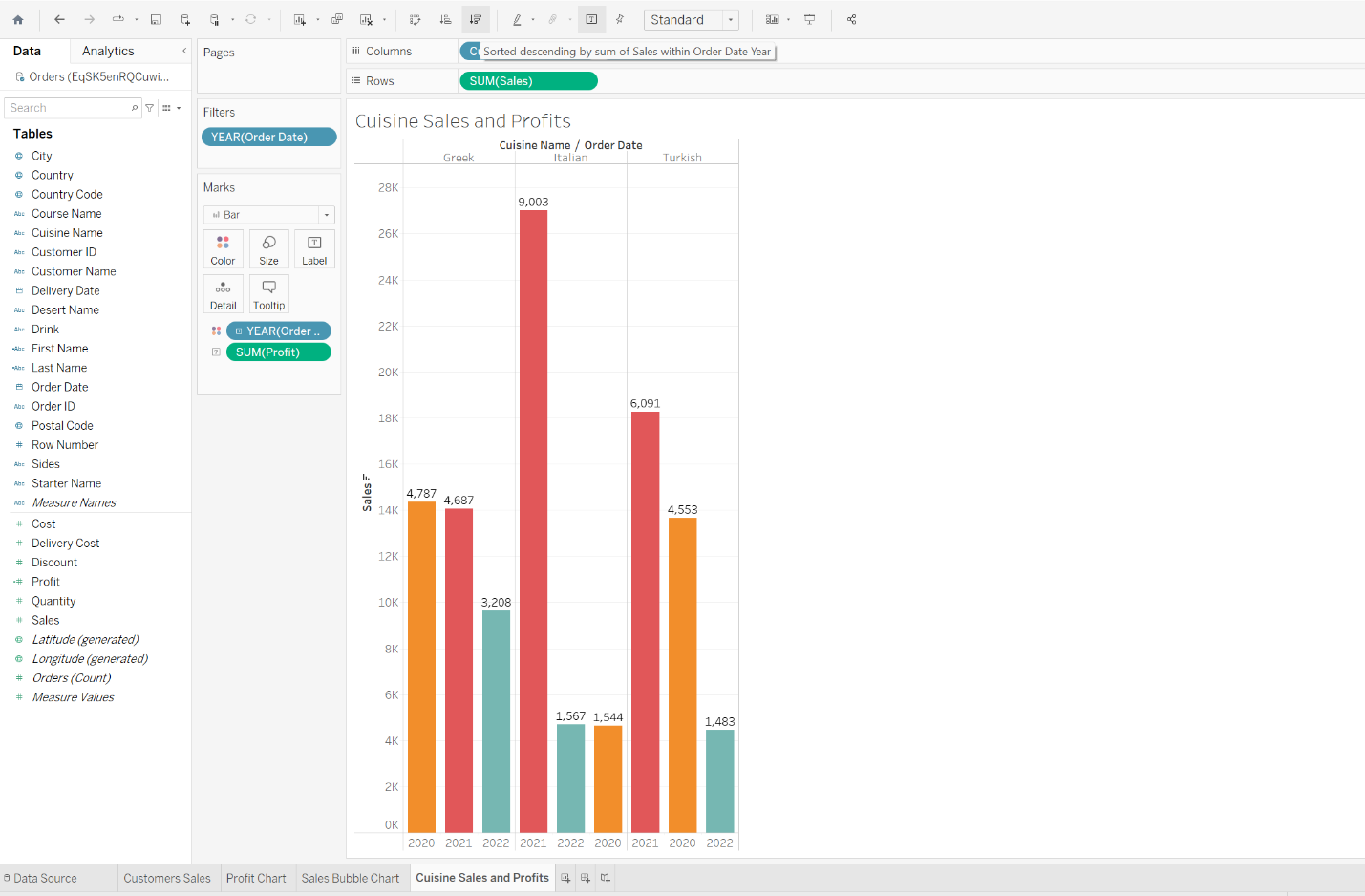
1. Tableau:

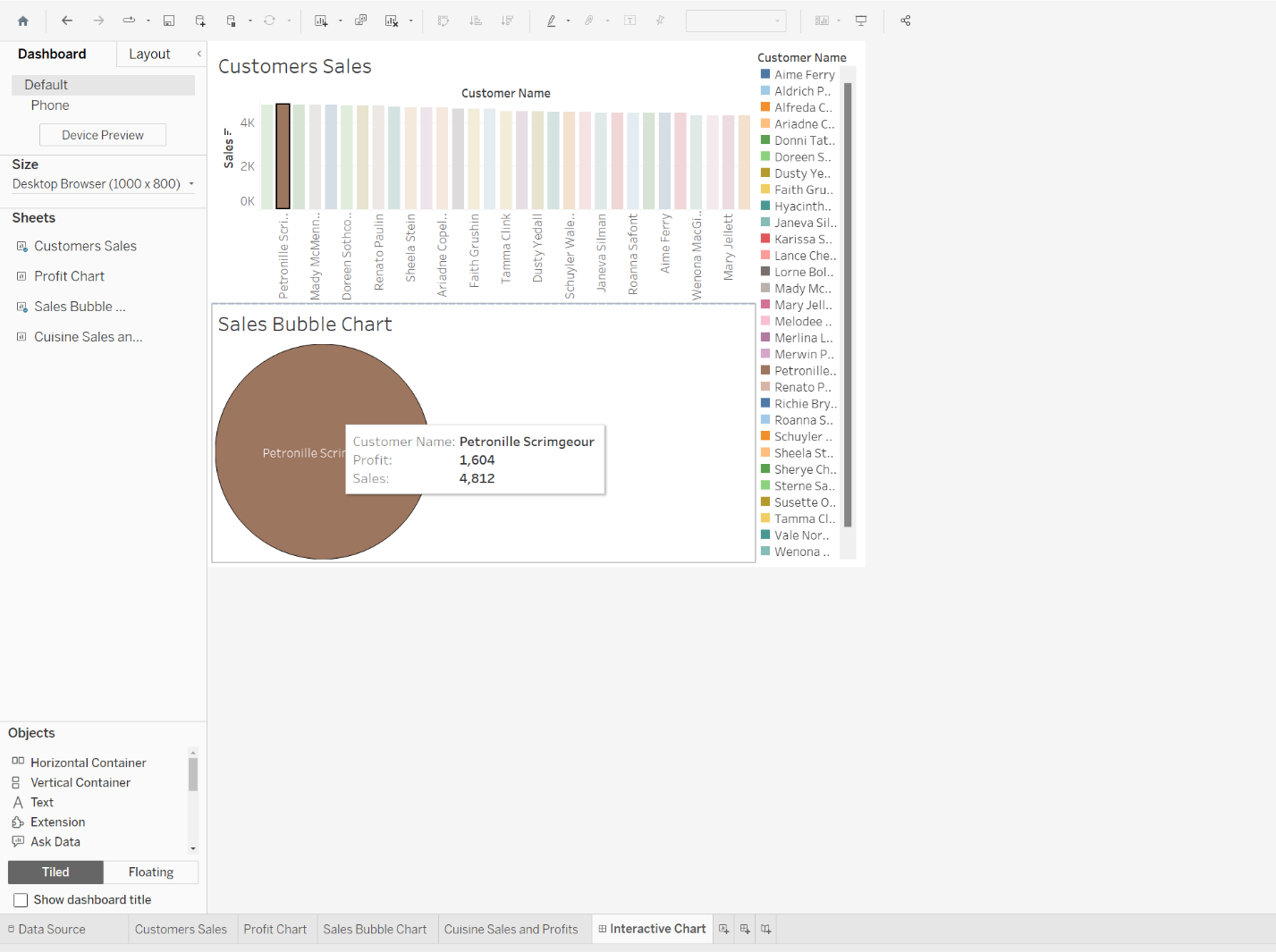
A screenshot of a computer

Description automatically generated









1. Cursor:

