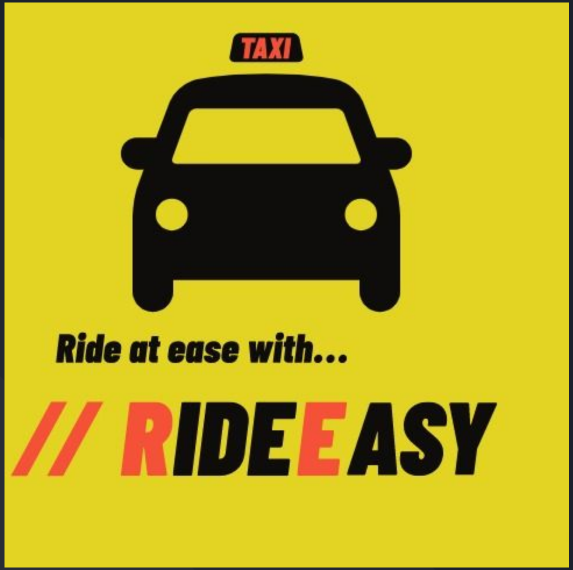


// RIDEEASY DATABASE SYSTEM

INFO 6210 - GROUP 5

// Akshay Khandelwal
// Patti Venkata Avinash Gupta
// Saurabh Ambardekar
// Sumit Malbari
// Varada Kulkarni
// Zarana Bhadricha

Why cab service?

Business Objectives

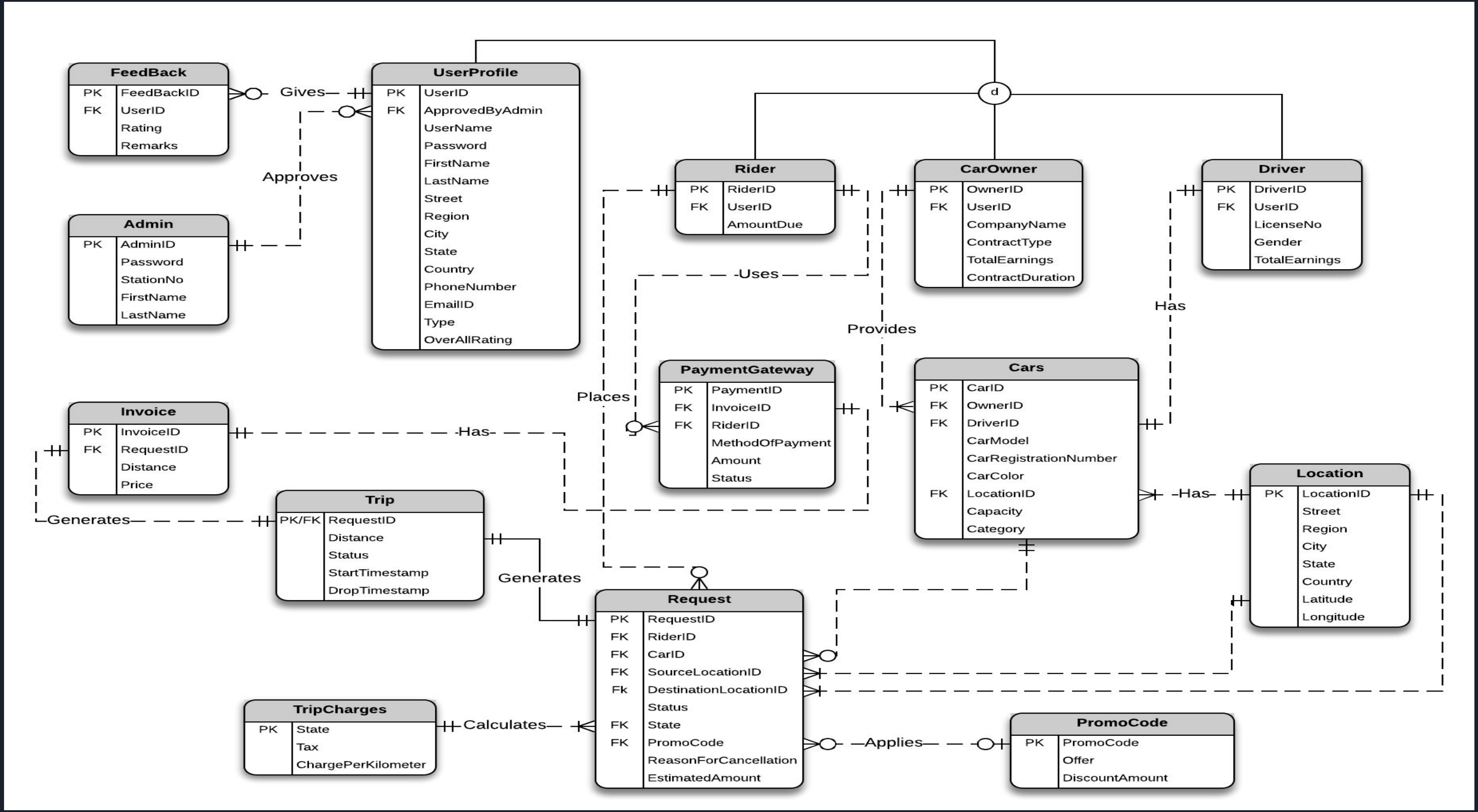
Key Components

- // Service Providers
- // Riders
- // Drivers
- // Car Owners

Ride at ease with...

// RIDEEASY

// Entity Relationship Diagram



FeedBack

	FeedBack
PK	FeedBackID
FK	UserID
	Rating
	Remarks

UserProfile

	UserProfile
PK	UserID
FK	ApprovedByAdmin
	UserName
	Password
	FirstName
	LastName
	Street
	Region
	City
	State
	Country
	PhoneNumber
	EmailID
	Type
	OverAllRating

Admin

	Admin
PK	AdminID
	Password
	StationNo
	FirstName
	LastName

Rider

	Rider
PK	RiderID
FK	UserID
	AmountDue

CarOwner

	CarOwner
PK	OwnerID
FK	UserID
	CompanyName
	ContractType
	TotalEarnings
	ContractDuration

Driver

	Driver
PK	DriverID
FK	UserID
	LicenseNo
	Gender
	TotalEarnings

PaymentGateway

	PaymentGateway
PK	PaymentID
FK	InvoiceID
FK	RiderID
	MethodOfPayment
	Amount
	Status

Cars

	Cars
PK	CarID
FK	OwnerID
FK	DriverID
	CarModel
	CarRegistrationNumber
	CarColor
FK	LocationID
	Capacity
	Category

Invoice

	Invoice
PK	InvoiceID
FK	RequestID
	Distance
	Price

Trip

	Trip
PK/FK	RequestID
	Distance
	Status
	StartTimestamp
	DropTimestamp

Location

	Location
PK	LocationID
	Street
	Region
	City
	State
	Country
	Latitude
	Longitude

Request

	Request
PK	RequestID
FK	RiderID
FK	CarID
FK	SourceLocationID
Fk	DestinationLocationID
	Status
FK	State
FK	PromoCode
	ReasonForCancellation
	EstimatedAmount

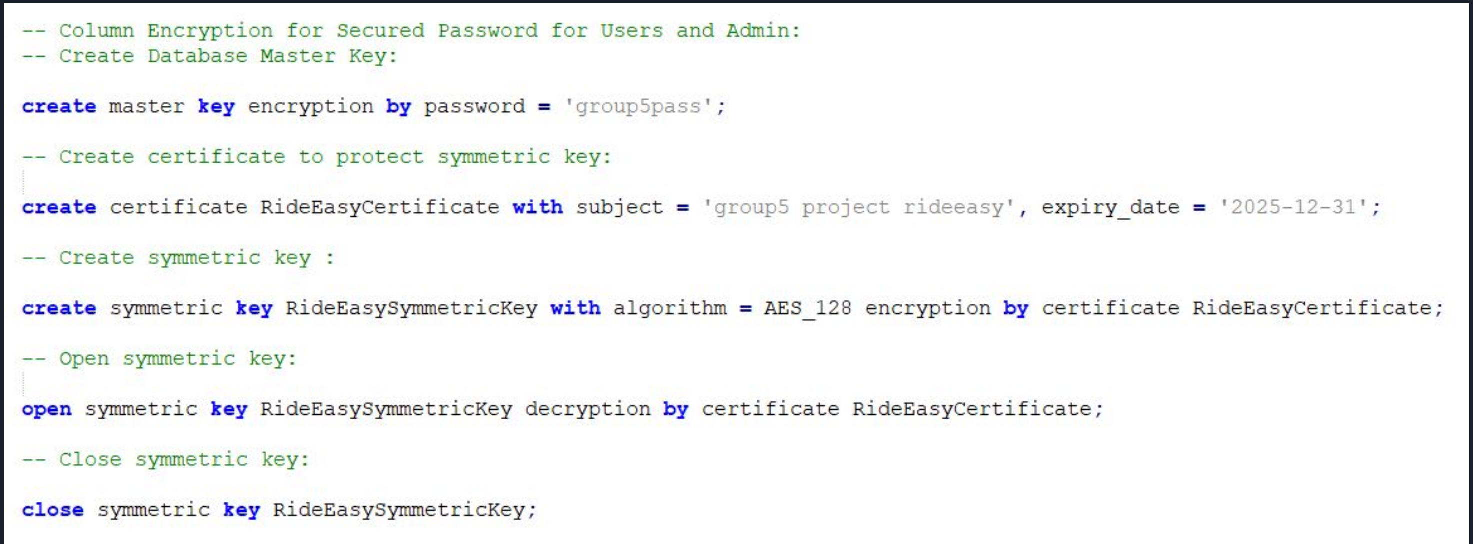
TripCharges

	TripCharges
PK	State
	Tax
	ChargePerKilometer

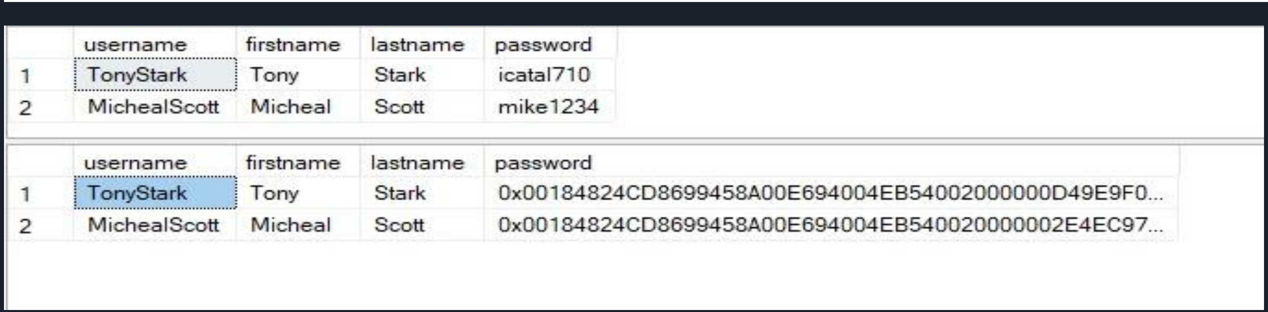
PromoCode

	PromoCode
PK	PromoCode
	Offer
	DiscountAmount

// Column Data Encryption & Decryption



```
-- Column Encryption for Secured Password for Users and Admin:
-- Create Database Master Key:

create master key encryption by password = 'group5pass';

-- Create certificate to protect symmetric key:

create certificate RideEasyCertificate with subject = 'group5 project rideeasy', expiry_date = '2025-12-31';

-- Create symmetric key :

create symmetric key RideEasySymmetricKey with algorithm = AES_128 encryption by certificate RideEasyCertificate;

-- Open symmetric key:

open symmetric key RideEasySymmetricKey decryption by certificate RideEasyCertificate;

-- Close symmetric key:

close symmetric key RideEasySymmetricKey;
```

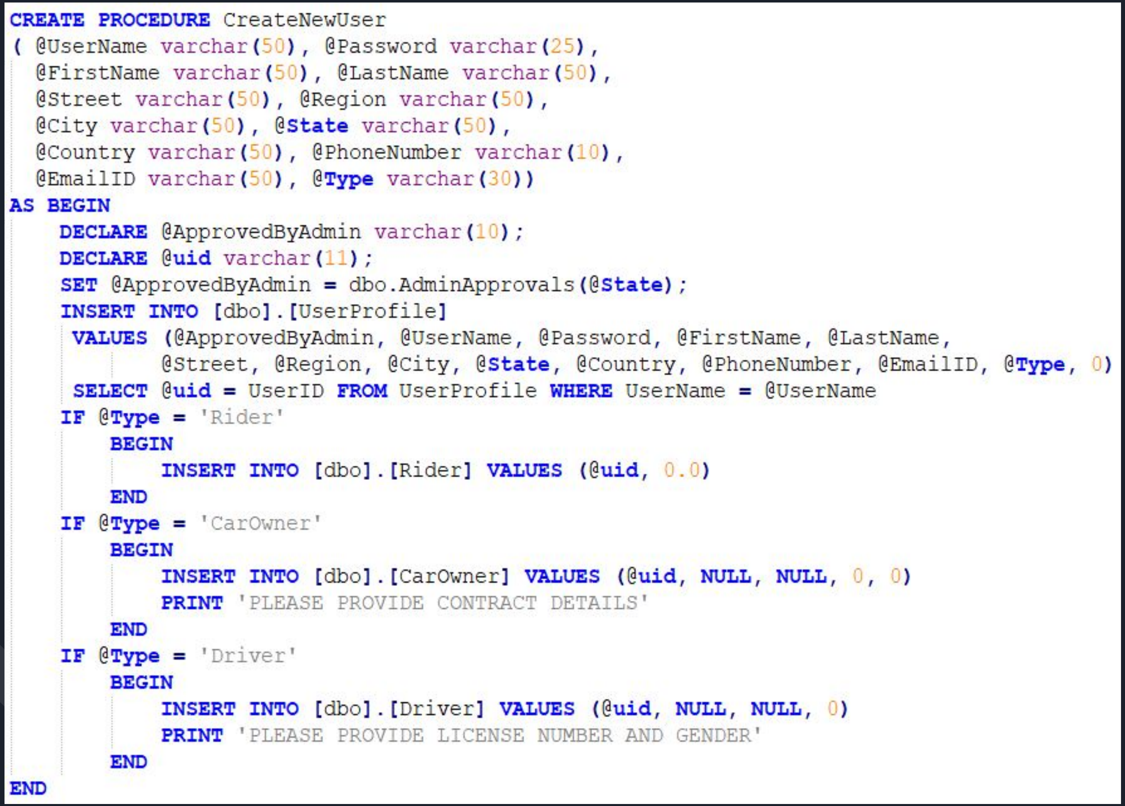


	username	firstname	lastname	password
1	TonyStark	Tony	Stark	icatal710
2	MichealScott	Micheal	Scott	mike1234

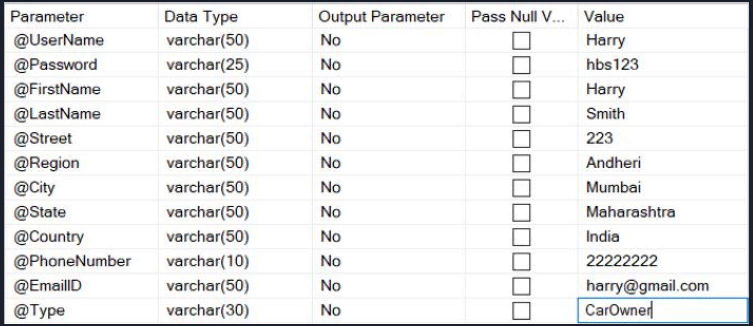
	username	firstname	lastname	password
1	TonyStark	Tony	Stark	0x00184824CD8699458A00E694004EB54002000000D49E9F0...
2	MichealScott	Micheal	Scott	0x00184824CD8699458A00E694004EB540020000002E4EC97...

// Stored Procedures

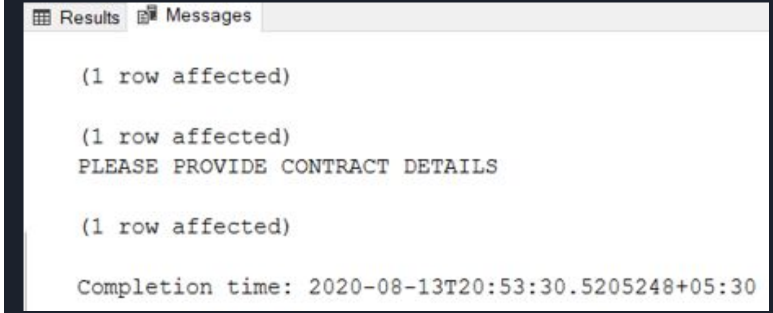
New user registration using stored procedure



```
CREATE PROCEDURE CreateNewUser
( @UserName varchar(50), @Password varchar(25),
  @FirstName varchar(50), @LastName varchar(50),
  @Street varchar(50), @Region varchar(50),
  @City varchar(50), @State varchar(50),
  @Country varchar(50), @PhoneNumber varchar(10),
  @EmailID varchar(50), @Type varchar(30))
AS BEGIN
    DECLARE @ApprovedByAdmin varchar(10);
    DECLARE @uid varchar(11);
    SET @ApprovedByAdmin = dbo.AdminApprovals(@State);
    INSERT INTO [dbo].[UserProfile]
     VALUES (@ApprovedByAdmin, @UserName, @Password, @FirstName, @LastName,
             @Street, @Region, @City, @State, @Country, @PhoneNumber, @EmailID, @Type, 0)
     SELECT @uid = UserID FROM UserProfile WHERE UserName = @UserName
    IF @Type = 'Rider'
        BEGIN
            INSERT INTO [dbo].[Rider] VALUES (@uid, 0.0)
        END
    IF @Type = 'CarOwner'
        BEGIN
            INSERT INTO [dbo].[CarOwner] VALUES (@uid, NULL, NULL, 0, 0)
            PRINT 'PLEASE PROVIDE CONTRACT DETAILS'
        END
    IF @Type = 'Driver'
        BEGIN
            INSERT INTO [dbo].[Driver] VALUES (@uid, NULL, NULL, 0)
            PRINT 'PLEASE PROVIDE LICENSE NUMBER AND GENDER'
        END
END
```



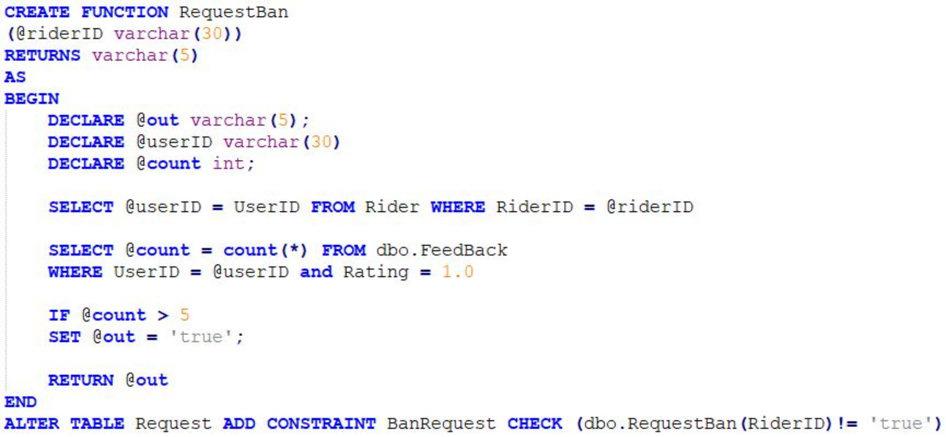
Parameter	Data Type	Output Parameter	Pass Null V...	Value
@UserName	varchar(50)	No	□	Harry
@Password	varchar(25)	No	□	hbs123
@FirstName	varchar(50)	No	□	Harry
@LastName	varchar(50)	No	□	Smith
@Street	varchar(50)	No	□	223
@Region	varchar(50)	No	□	Andheri
@City	varchar(50)	No	□	Mumbai
@State	varchar(50)	No	□	Maharashtra
@Country	varchar(50)	No	□	India
@PhoneNumber	varchar(10)	No	□	22222222
@EmailID	varchar(50)	No	□	harry@gmail.com
@Type	varchar(30)	No	□	CarOwner



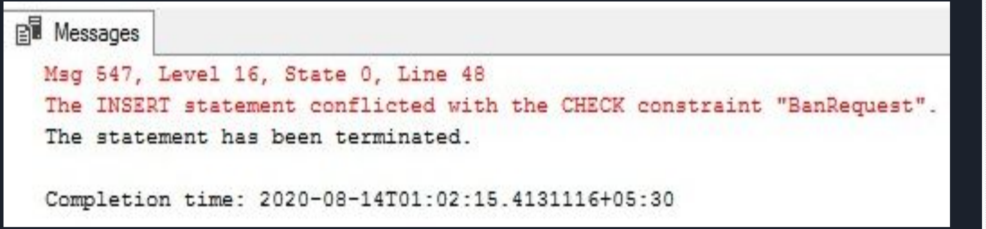
```
(1 row affected)

(1 row affected)
PLEASE PROVIDE CONTRACT DETAILS

(1 row affected)

Completion time: 2020-08-13T20:53:30.5205248+05:30
```

// Table Level Check Constraints & Triggers

Applying a table level check constraint to ban riders with a rating lower than 1 from placing a ride request

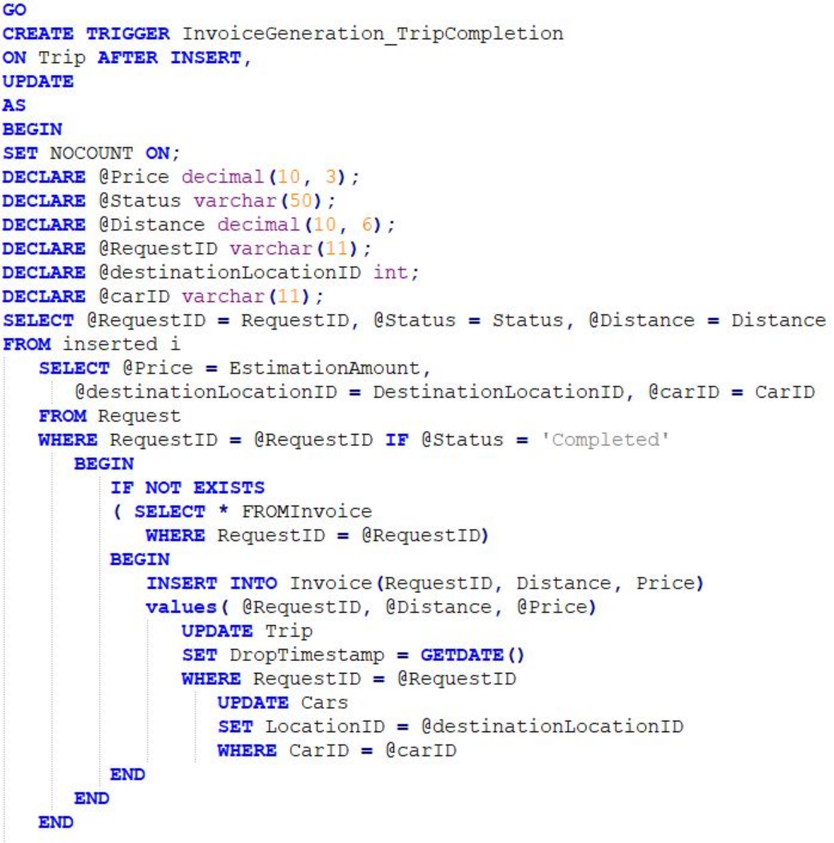


```
CREATE FUNCTION RequestBan
(@riderID varchar(30))
RETURNS varchar(5)
AS
BEGIN
    DECLARE @out varchar(5);
    DECLARE @userID varchar(30)
    DECLARE @count int;

    SELECT @userID = UserID FROM Rider WHERE RiderID = @riderID

    SELECT @count = count(*) FROM dbo.FeedBack
    WHERE UserID = @userID and Rating = 1.0

    IF @count > 5
    SET @out = 'true';

    RETURN @out
END
ALTER TABLE Request ADD CONSTRAINT BanRequest CHECK (dbo.RequestBan(RiderID)!= 'true')
```



```
Msg 547, Level 16, State 0, Line 48
The INSERT statement conflicted with the CHECK constraint "BanRequest".
The statement has been terminated.

Completion time: 2020-08-14T01:02:15.4131116+05:30
```

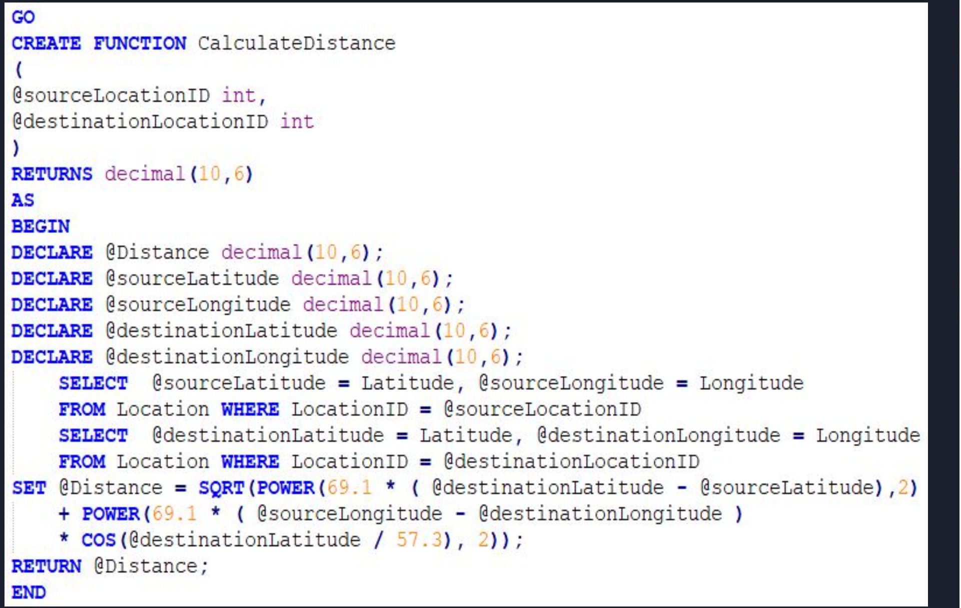
Trigger for generating invoice once a trip is completed along with updating the car location in the Cars Entity.



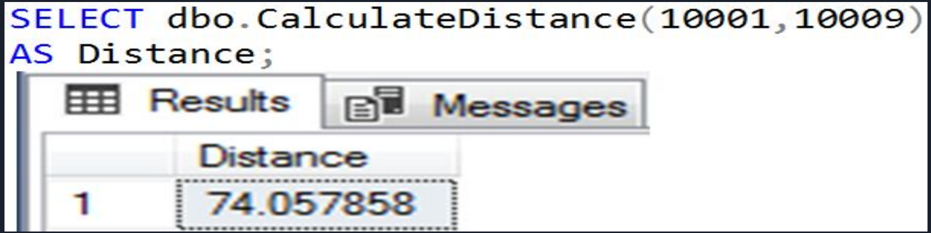
```
GO
CREATE TRIGGER InvoiceGeneration_TripCompletion
ON Trip AFTER INSERT,
UPDATE
AS
BEGIN
SET NOCOUNT ON;
DECLARE @Price decimal(10, 3);
DECLARE @Status varchar(50);
DECLARE @Distance decimal(10, 6);
DECLARE @RequestID varchar(11);
DECLARE @destinationLocationID int;
DECLARE @carID varchar(11);
SELECT @RequestID = RequestID, @Status = Status, @Distance = Distance
FROM inserted i
   SELECT @Price = EstimationAmount,
      @destinationLocationID = DestinationLocationID, @carID = CarID
   FROM Request
   WHERE RequestID = @RequestID IF @Status = 'Completed'
      BEGIN
         IF NOT EXISTS
         ( SELECT * FROMInvoice
            WHERE RequestID = @RequestID)
         BEGIN
            INSERT INTO Invoice(RequestID, Distance, Price)
            values( @RequestID, @Distance, @Price)
               UPDATE Trip
               SET DropTimestamp = GETDATE()
               WHERE RequestID = @RequestID
                  UPDATE Cars
                  SET LocationID = @destinationLocationID
                  WHERE CarID = @carID
         END
      END
   END
```

// Computed Columns based on a function

Distance Calculation based on SourceLocationID and DestinationLocationID (Latitude and Longitude)



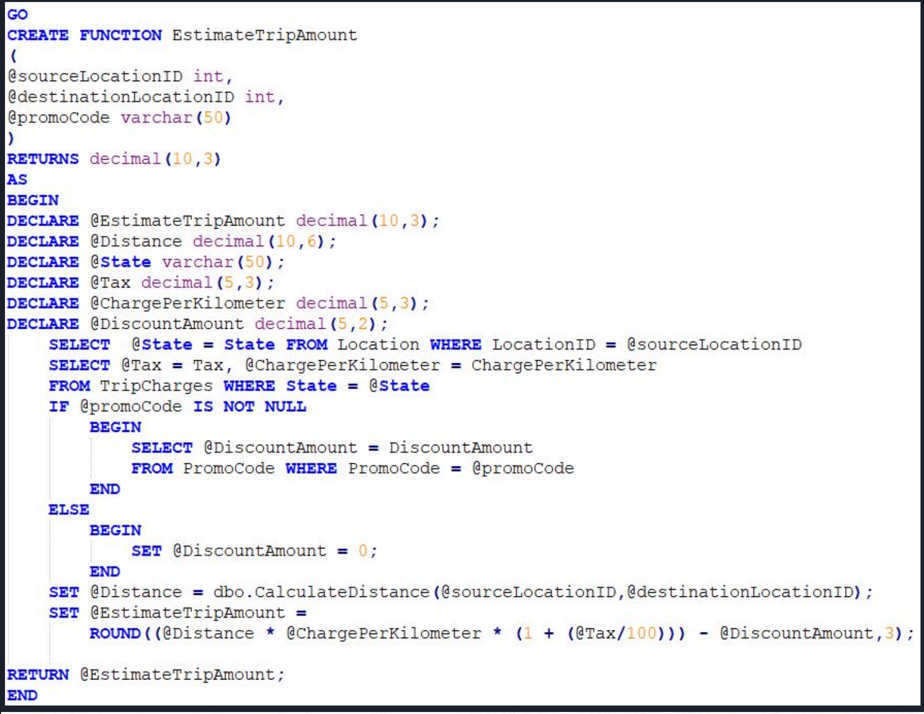
```
GO
CREATE FUNCTION CalculateDistance
(
@sourceLocationID int,
@destinationLocationID int
)
RETURNS decimal(10,6)
AS
BEGIN
DECLARE @Distance decimal(10,6);
DECLARE @sourceLatitude decimal(10,6);
DECLARE @sourceLongitude decimal(10,6);
DECLARE @destinationLatitude decimal(10,6);
DECLARE @destinationLongitude decimal(10,6);
    SELECT  @sourceLatitude = Latitude, @sourceLongitude = Longitude
    FROM Location WHERE LocationID = @sourceLocationID
    SELECT  @destinationLatitude = Latitude, @destinationLongitude = Longitude
    FROM Location WHERE LocationID = @destinationLocationID
SET @Distance = SQRT(POWER(69.1 * ( @destinationLatitude - @sourceLatitude),2)
    + POWER(69.1 * ( @sourceLongitude - @destinationLongitude )
    * COS(@destinationLatitude / 57.3), 2));
RETURN @Distance;
END
```



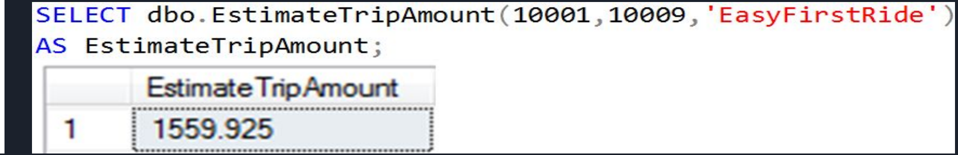
```
SELECT dbo.CalculateDistance(10001,10009)
AS Distance;
```

```
   Distance
1  74.057858
```

Calculating Estimated Trip Amount using distance and Tripcharges Entity and then storing it in Request Entity



```
GO
CREATE FUNCTION EstimateTripAmount
(
@sourceLocationID int,
@destinationLocationID int,
@promoCode varchar(50)
)
RETURNS decimal(10,3)
AS
BEGIN
DECLARE @EstimateTripAmount decimal(10,3);
DECLARE @Distance decimal(10,6);
DECLARE @State varchar(50);
DECLARE @Tax decimal(5,3);
DECLARE @ChargePerKilometer decimal(5,3);
DECLARE @DiscountAmount decimal(5,2);
    SELECT  @State = State FROM Location WHERE LocationID = @sourceLocationID
    SELECT @Tax = Tax, @ChargePerKilometer = ChargePerKilometer
    FROM TripCharges WHERE State = @State
    IF @promoCode IS NOT NULL
        BEGIN
            SELECT @DiscountAmount = DiscountAmount
            FROM PromoCode WHERE PromoCode = @promoCode
        END
    ELSE
        BEGIN
            SET @DiscountAmount = 0;
        END
    SET @Distance = dbo.CalculateDistance(@sourceLocationID,@destinationLocationID);
    SET @EstimateTripAmount =
        ROUND((@Distance * @ChargePerKilometer * (1 + (@Tax/100))) - @DiscountAmount,3);

RETURN @EstimateTripAmount;
END
```

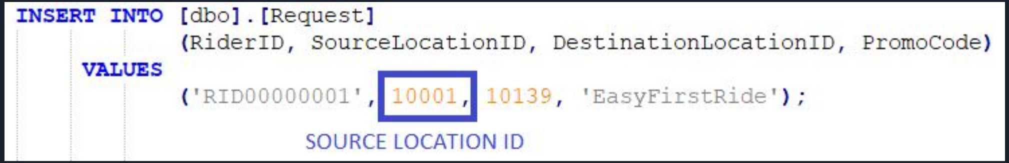


```
SELECT dbo.EstimateTripAmount(10001,10009,'EasyFirstRide')
AS EstimateTripAmount;
```

```
   EstimateTripAmount
1  1559.925
```

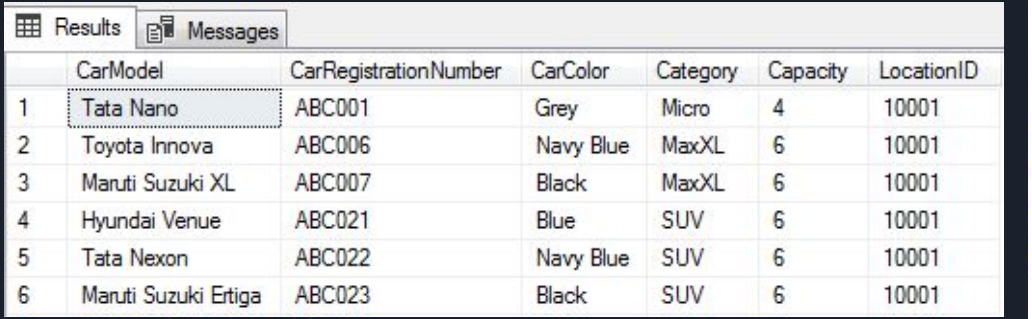
// Requesting a Ride

View to display nearby Rides when a Request is placed by the Rider

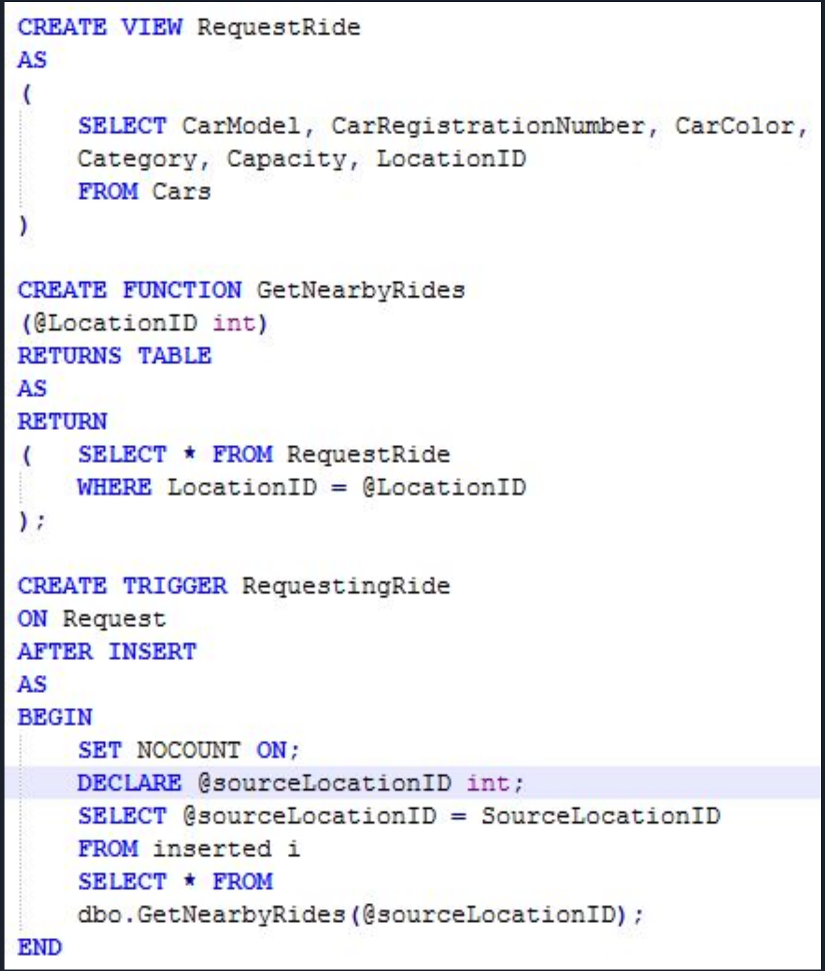


```
INSERT INTO [dbo].[Request]
           (RiderID, SourceLocationID, DestinationLocationID, PromoCode)
     VALUES
           ('RID00000001', 10001, 10139, 'EasyFirstRide');
```

SOURCE LOCATION ID



	CarModel	CarRegistrationNumber	CarColor	Category	Capacity	LocationID
1	Tata Nano	ABC001	Grey	Micro	4	10001
2	Toyota Innova	ABC006	Navy Blue	MaxXL	6	10001
3	Maruti Suzuki XL	ABC007	Black	MaxXL	6	10001
4	Hyundai Venue	ABC021	Blue	SUV	6	10001
5	Tata Nexon	ABC022	Navy Blue	SUV	6	10001
6	Maruti Suzuki Ertiga	ABC023	Black	SUV	6	10001



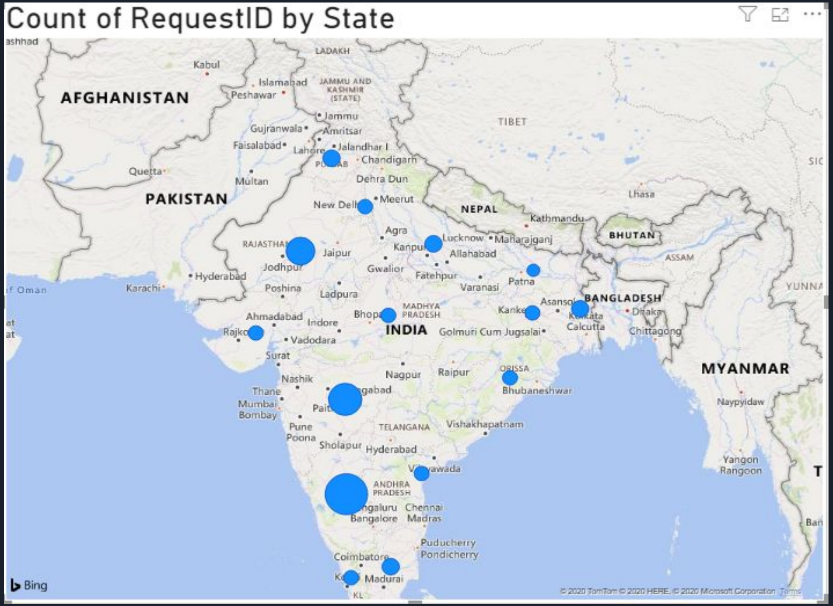
```
CREATE VIEW RequestRide
AS
(
    SELECT CarModel, CarRegistrationNumber, CarColor,
    Category, Capacity, LocationID
    FROM Cars
)

CREATE FUNCTION GetNearbyRides
(@LocationID int)
RETURNS TABLE
AS
RETURN
(   SELECT * FROM RequestRide
    WHERE LocationID = @LocationID
);

CREATE TRIGGER RequestingRide
ON Request
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @sourceLocationID int;
    SELECT @sourceLocationID = SourceLocationID
    FROM inserted i
    SELECT * FROM
    dbo.GetNearbyRides(@sourceLocationID);
END
```

TAXI

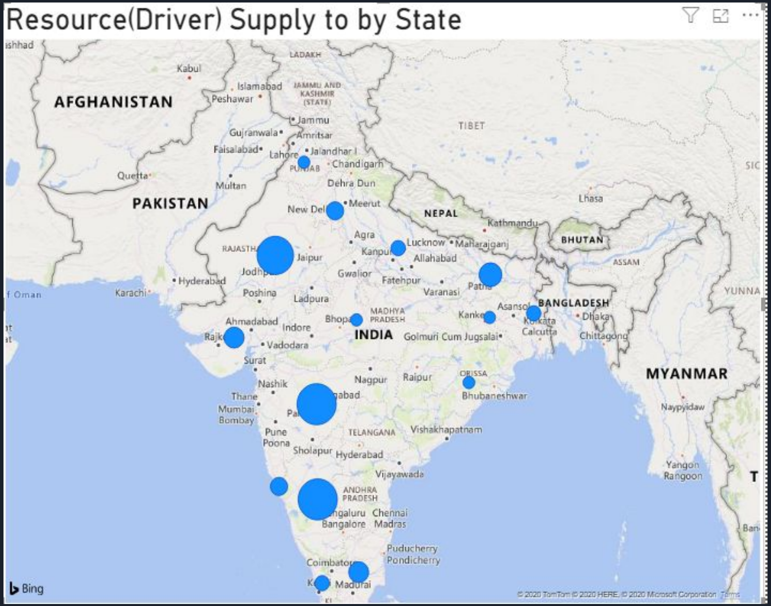
Ride at ease with...

// RIDEEASY

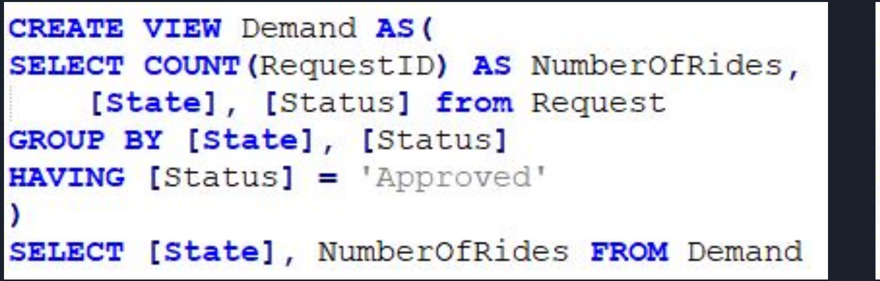
// Analytics: Demand & Supply



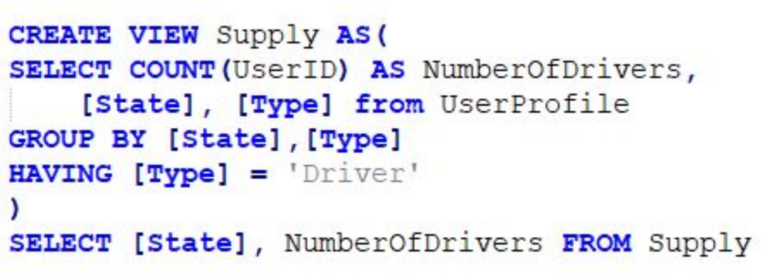
Count of RequestID by State



Resource(Driver) Supply to by State

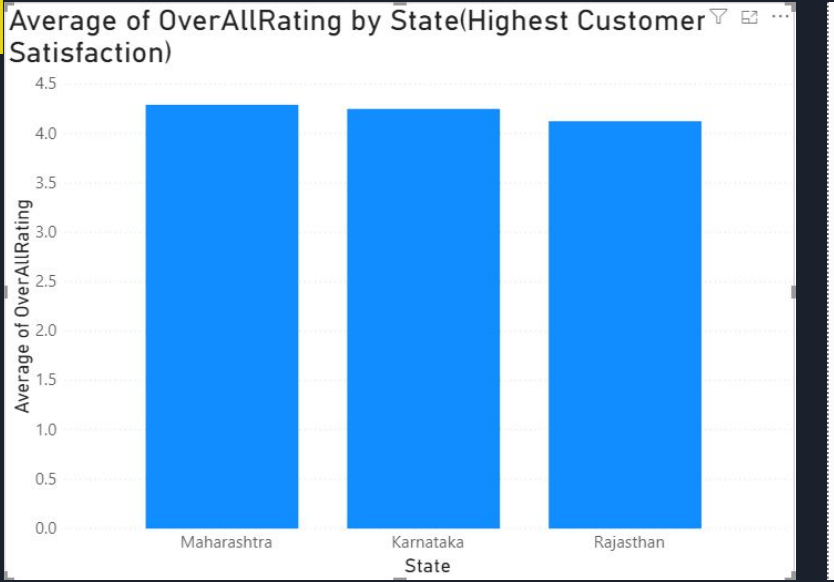


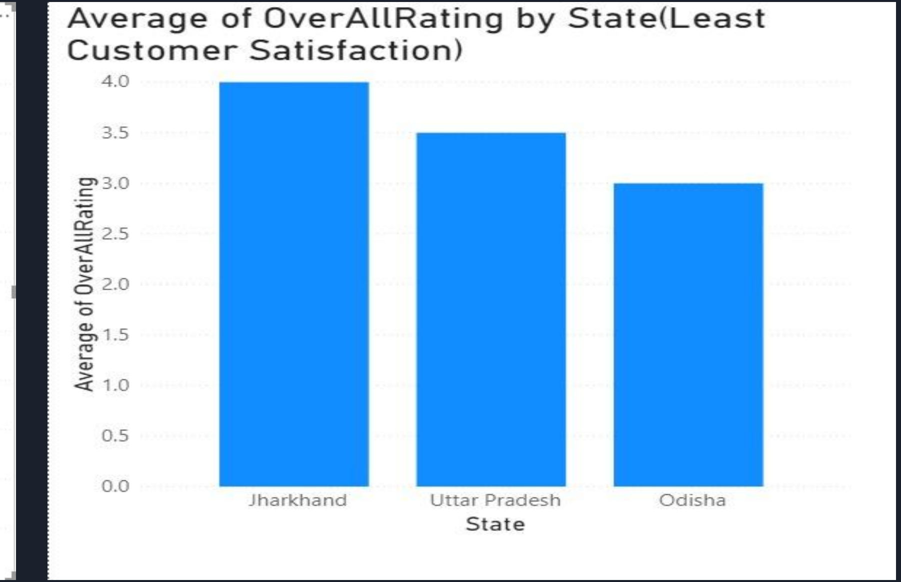
```
CREATE VIEW Demand AS(
SELECT COUNT(RequestID) AS NumberOfRides,
    [State], [Status] from Request
GROUP BY [State], [Status]
HAVING [Status] = 'Approved'
)
SELECT [State], NumberOfRides FROM Demand
```

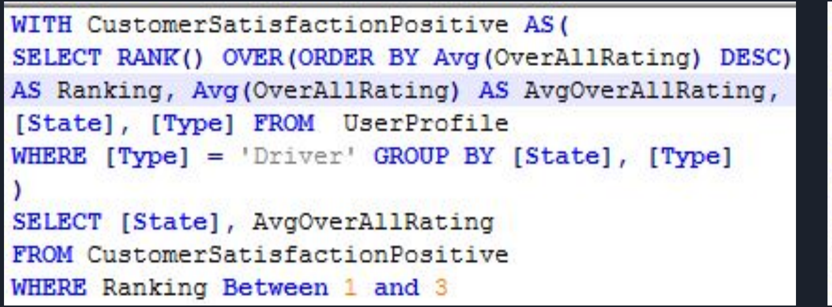


```
CREATE VIEW Supply AS(
SELECT COUNT(UserID) AS NumberOfDrivers,
    [State], [Type] from UserProfile
GROUP BY [State],[Type]
HAVING [Type] = 'Driver'
)
SELECT [State], NumberOfDrivers FROM Supply
```

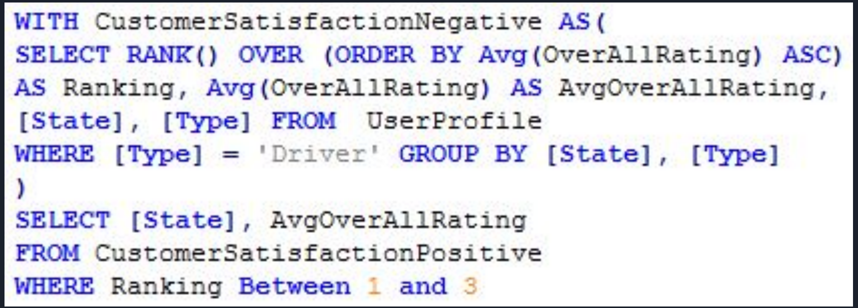
// Analytics: State Wise Customer Ratings


Average of OverAllRating by State(Highest Customer Satisfaction)


Average of OverAllRating by State(Least Customer Satisfaction)



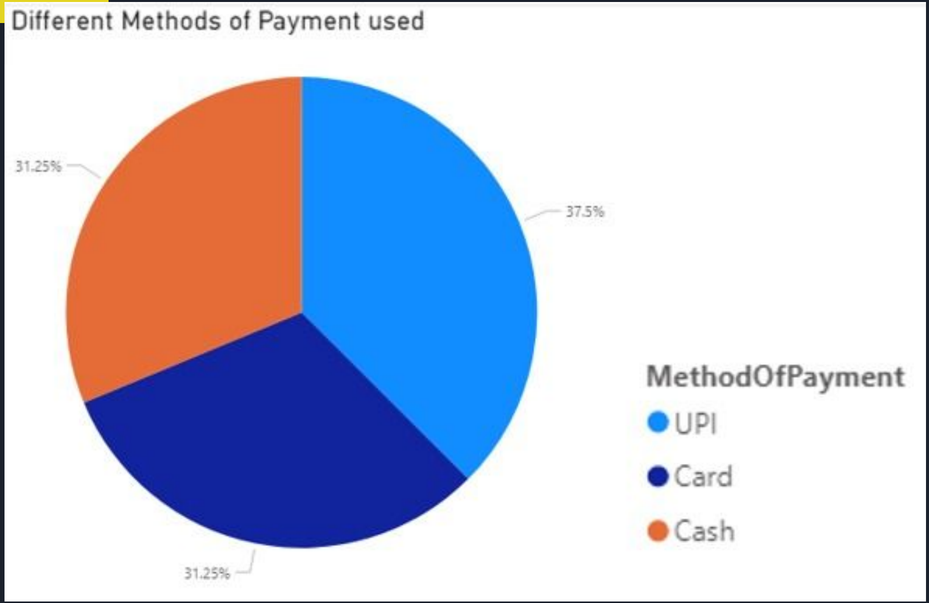
```
WITH CustomerSatisfactionPositive AS(
SELECT RANK() OVER(ORDER BY Avg(OverAllRating) DESC)
AS Ranking, Avg(OverAllRating) AS AvgOverAllRating,
[State], [Type] FROM  UserProfile
WHERE [Type] = 'Driver' GROUP BY [State], [Type]
)
SELECT [State], AvgOverAllRating
FROM CustomerSatisfactionPositive
WHERE Ranking Between 1 and 3
```

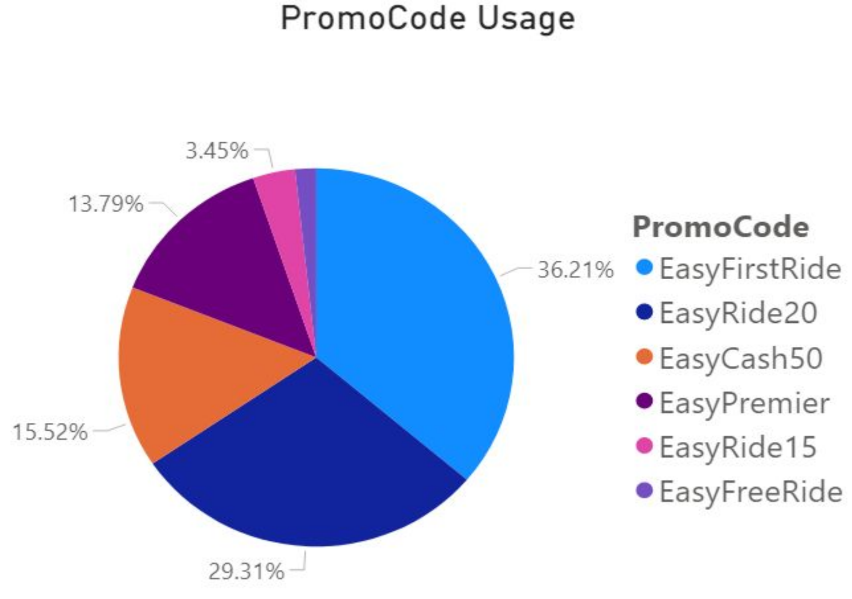


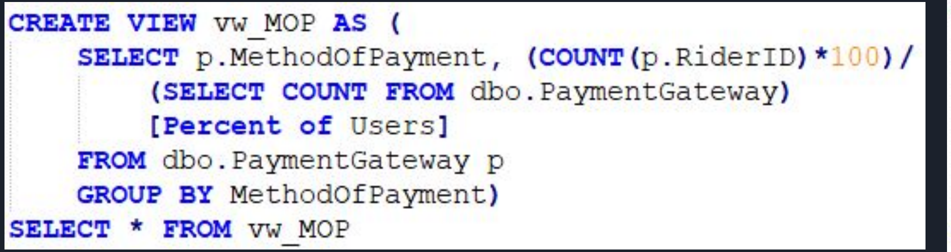
```
WITH CustomerSatisfactionNegative AS(
SELECT RANK() OVER (ORDER BY Avg(OverAllRating) ASC)
AS Ranking, Avg(OverAllRating) AS AvgOverAllRating,
[State], [Type] FROM  UserProfile
WHERE [Type] = 'Driver' GROUP BY [State], [Type]
)
SELECT [State], AvgOverAllRating
FROM CustomerSatisfactionPositive
WHERE Ranking Between 1 and 3
```

TAXI
Ride at ease with...
// RIDEEASY

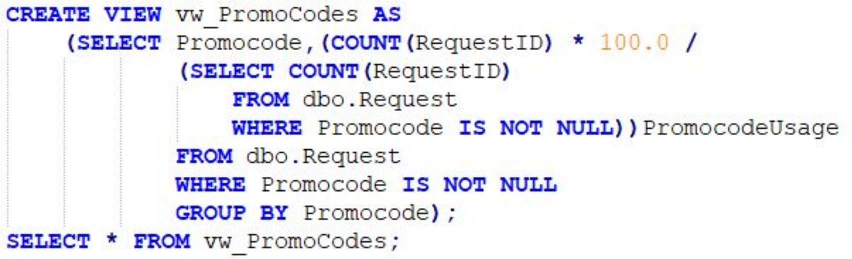
// Analytics: Methods of Payment & Promo Code


Different Methods of Payment used


PromoCode Usage



```
CREATE VIEW vw_MOP AS (
    SELECT p.MethodOfPayment, (COUNT(p.RiderID)*100)/
        (SELECT COUNT FROM dbo.PaymentGateway)
        [Percent of Users]
    FROM dbo.PaymentGateway p
    GROUP BY MethodOfPayment)
SELECT * FROM vw_MOP
```

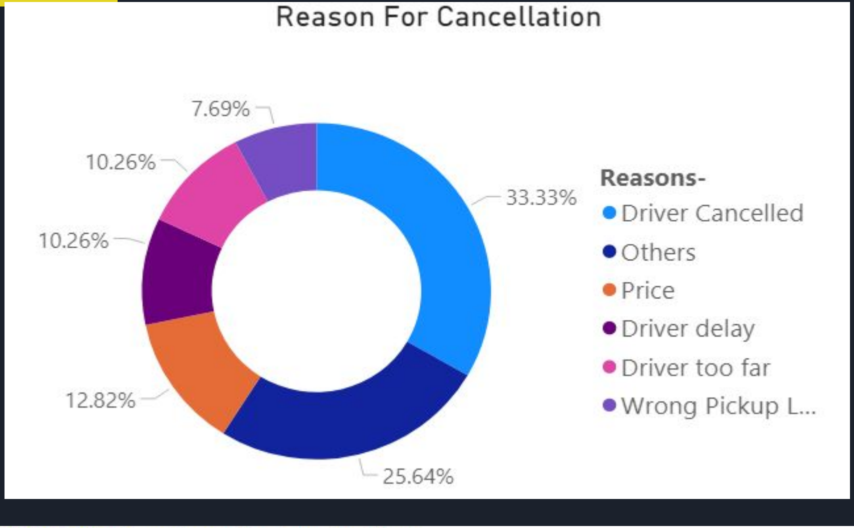


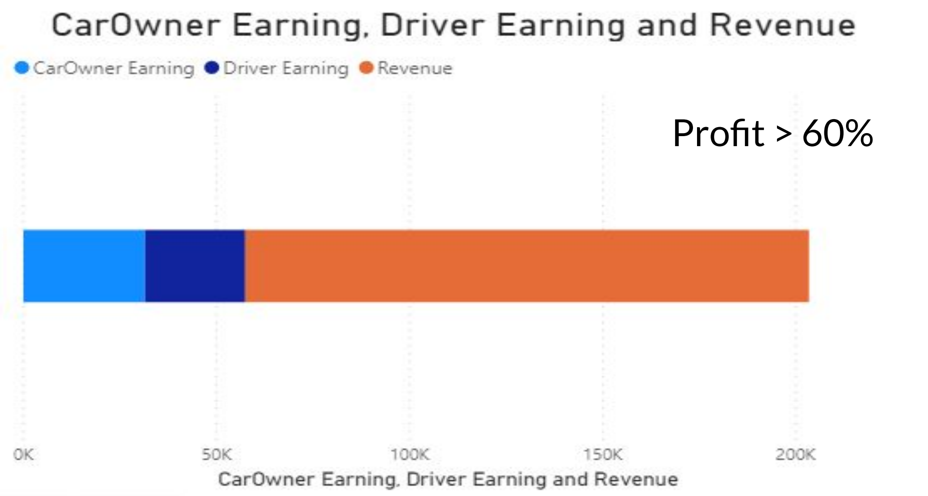
```
CREATE VIEW vw_PromoCodes AS
    (SELECT Promocode,(COUNT(RequestID) * 100.0 /
            (SELECT COUNT(RequestID)
                FROM dbo.Request
                WHERE Promocode IS NOT NULL))PromocodeUsage
            FROM dbo.Request
            WHERE Promocode IS NOT NULL
            GROUP BY Promocode);
SELECT * FROM vw_PromoCodes;
```

Ride at ease with...

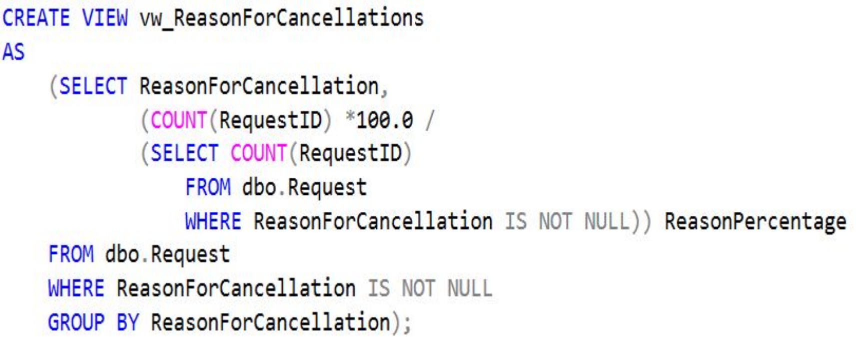
// RIDEEASY

// Analytics: Reasons for Trip Cancellation & Revenue

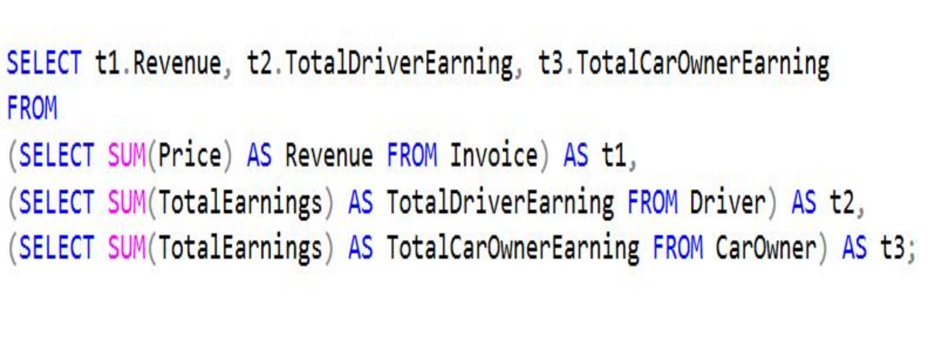




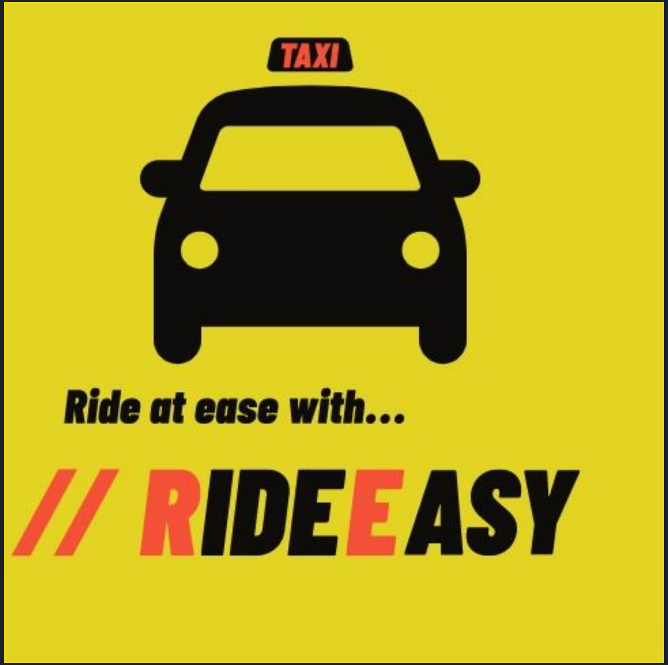
Profit > 60%



```sql
CREATE VIEW vw_ReasonForCancellations
AS
    (SELECT ReasonForCancellation,
            (COUNT(RequestID) *100.0 /
            (SELECT COUNT(RequestID)
                FROM dbo.Request
                WHERE ReasonForCancellation IS NOT NULL)) ReasonPercentage
    FROM dbo.Request
    WHERE ReasonForCancellation IS NOT NULL
    GROUP BY ReasonForCancellation);
```



```sql
SELECT t1.Revenue, t2.TotalDriverEarning, t3.TotalCarOwnerEarning
FROM
(SELECT SUM(Price) AS Revenue FROM Invoice) AS t1,
(SELECT SUM(TotalEarnings) AS TotalDriverEarning FROM Driver) AS t2,
(SELECT SUM(TotalEarnings) AS TotalCarOwnerEarning FROM CarOwner) AS t3;
```



THANK YOU
Q & A