Marmara University

Department of Computer Science and Engineering

CSE3055 Database Systems



Technical Service Database Report

150117013 - Mehmet Ali Yüksel

150117023 – Anıl Şenay

150117072 – Bilgehan Geçici

Due Date: 30/12/2020

Report

Project Description

Database project that we implemented is about managing a technical service of mobile devices and storing the repairing process information of them. This database will be used by the workers of the technical service.

A complete repairment process of one device is as following:

Step-1: A customer arrives to service and contacts a Smart Service Employee. Then customer is asked to give their personal and device's information by the employee. Registration is completed, the customer gives their device and leaves the service.

Step-2: Since device's information is registered, warranty information is also known on the system. According to warranty information device is handed in to a related technician (In Warranty Technician: only repairs the devices with valid warranty conditions and date. Out of Warranty Technician: only repairs the devices with invalid warranty conditions and date.)

Step-3: Technician detects the problem and changes the damaged parts of the device with a new one. To do this replacement first technician needs to check storage if the needed parts are available. If yes, the technician repairs the device and fills the repairment field of this specific case with the actions he/she took. Such as; changed parts, end time of the repairment, repairment degree, remark about the actions taken in details. If no, the technician creates an order and skips the current repairment until the ordered part comes.

Step-4: Repairment is completed and device is tested. The device is ready to deliver. The customer comes to take their device, contacts to accountant. Accountant states the total cost to the customer and payment (if needed) is done via several payment methods.

This is an example scenario which covers most of the steps, different scenarios also happen.

Our database handles storing information of repairing processes mentioned above. Information of each actor, device and process status are kept. Different actions are taken in different events related to this information.

Scope

What is included?

- Personal information of the actors and their performances (employees, customers) is stored.
- Since a device is our main object in this business, we keep detailed information about devices.
- Categorized case description is stored to have detailed data about different cases and to understand the problem.
- Repairment is the business, so we keep detailed information about this process.
 Technicians update the repairment information during the process, according to their actions.
- Available parts (including spare devices) to use is kept and these parts are stored in a physical storage and we also keep the information of physical place of the parts(boxNumbers).
- o Orders of needed parts are kept to track orders.
- Payment information is stored along with the process information to inform the customers during payment.
- o Inserting new actors, devices, creating new repairments, making orders and updating this information is handled.

What is excluded?

- Salary of the employees is not included because it is included in a different database along with insurance information and is not a requirement in this project.
- Queuing of customers inside the services is not included because it is handled in a different system and is not a requirement in this project. Categorized case description is stored to have detailed data about different cases and to understand the problem.
- Tracking of spare devices, which are given to customers after they leave their device in the service, is not included because it is handled on papers, physically and is not a requirement in this project. Available parts (including spare devices) to use are kept and these parts are stored in a physical storage and we also keep the information of physical place of the parts(boxNumbers).
- Device logs are not covered in details because logs of devices are taken via special application in this service and very technically detailed also is not a requirement in this project. There's only a placeholder of this feature in our project, represented as deviceLogs in Customer Device table. Payment information is stored along with the process information to inform the customers during payment.
- Inserting new actors, devices, creating new repairments, making orders and updating this information is handled.
- Testing process of the devices after repairment is not included because it is also handled in detail via device log application and not a requirement of this project.

Data and Requirement Analysis

We indicated data types, defaults, unique variables, identities, indexes and computed variables. We usually use identities at ID values for simplicity. For the default values we usually use bit data types for that. We will cover them in store procedures. For the computed values, we usually use our date variable types. We also add unique values for our table variables. We usually use unique for the data types that also can be primary key. We indexed the most frequent selected foreign keys to have a speed efficiency in queries.

We indicated our requirements based on efficiency and functionality, we used also some check constrains that also optimize the database and prevent some functionality errors. Here is the list of the necessary requirements.

- The Database project can be used by multiple employees.
- The Database project must save the information taken from the customers.
- Each employee must have a unique account on system.
- Only admins can update or change saved information after they've been saved.
- No multiple employees can serve the same customer device simultaneously in vice versa.
- If a customer has to wait more than needed time for their device to be repaired, they are provided with a spare device, excluding tablets and smart watches. This case may occur when a part needs to be changed and there aren't any available new parts in the storage.
- If an in-warranty technician detects any out-of-warranty condition on the device he/she is working on, then the technician must assign the device to a technician who repairs only out-of-warranty devices.
- Customers can demand for change of their devices unless the device meets the requirements of changing procedure.
- An employee must be over 18 years old.
- An employee's username's length must be greater than 3 and must be unique.
- Each employee has a unique e-mail.
- Repairment degree can only be 0, 1, 2 or 3. This degree represents the level of labor done on repairment and cost is determined accordingly.
- In storage quantity and boxNumber should be greater than or equal 0.

Business Process and Their Definitions

Goals

The purpose of the process is to provide a wide range database system to a technical service who needs the store information about their customers, employee, repairment process and the parts being physically stored in their building etc. This business is created in order to simplify the information storage and keep the operation of repairment work effective and quick. We can understand our success from our customer's satisfaction from the system we created.

Planning and Mapping Process

Our first strategy to achieve our goal is to understand our customer's requirements. Then split the work parts between team members and execute. After that we keep our customer informed about our process, ask them their opinions and make improvements guided by our customer if needed. Repeat this process until the work is done.

Actions and Stakeholders

All the team members have participated in creating tables, procedures, triggers and views in SQL server and testing their functionality. We applied extreme programming method.

Our customer Destek Bilişim (Kadıköy Şubesi) provided necessary information about their business process and their requirements. Also, they provided overviews and feedback to guide us in our progress.

Business Rules

- A Manager manages multiple employees.
- An employee must be over 18 years old.
- An employee's username's length must be greater than 3 and must be unique
- Each employee has a unique e-mail
- An employee may request multiple orders.
- An order has to have one type of ordered part vice versa.
- Multiple parts may be ordered.
- Exactly one storage has at least one part.
- Multiple parts may be needed to change in at most one repairment.
- At most one accountant may manage multiple payments.
- Exactly one customer may have multiple payments.
- Exactly one customer has at least one customer device.
- Exactly one customer has at least one repairment process.
- Exactly one customer device has exactly one case.
- Exactly one customer device has at least one repairment process.
- At least one employee may control multiple repairments.
- Exactly one employee provides exactly one spare device to exactly one customer.
- Exactly one repairment requires at most one payment.
- Exactly one case type has at least one case category
- Exactly one case category has at least one case specification
- An employee may have many employee availability
- A device may have many device logs
- A payment must have exactly one payment method
- Repairment degree can only be 0, 1, 2 or 3. This degree represents the level of labor done on repairment and cost is determined accordingly.
- In storage quantity and boxNumber should be greater than or equal 0.
- Exactly one case type has at least one case category.
- Exactly one case category has at least one case specification.
- An employee may have many employee availabilities.
- A device may have many device logs.
- A payment must have exactly one payment method.

Tables

We implemented 24 tables for our Technical Service (Destek Bilişim). We demonstrate the main parts of the business process of our technical service below, including all necessary tables listed with their definitions.

	EMPLOYEE									
Defi	efinition: This table stores personal and business-related information of employees.									
Che	Check Constraints: All the employees must be over 18 - username's length value must be greater than 3.									
Trig	Friggers: -									
PK	Column Name	Data Type	FK	Identity	Default Value	Unique	Index	Computed Columns		
✓	ID	tinyint		\checkmark						
	firstName	nvarchar(50)								
	lastName	nvarchar(50)								
	phoneName	nvarchar(20)								
	username	nvarchar(30)				\checkmark				
	password	nvarchar(32)								
	email	nvarchar(100)				\searrow				
	address	nvarchar(150)								
	dateOfBirth	date								
	startDate	date								
	isManager	bit			\checkmark					
	isSmartService	bit			\checkmark					
	isTechician	bit			\checkmark					
	isStorageMan	bit			\checkmark					
	isTester	bit			\checkmark					
	isAccountant	bit			\checkmark					

Check Constraints: -

deviceID

logDate

logMessage

bigint

tinyint

Trig	gers: -							
PK	Column Name	Data Type	FK	Identity	Default Value	Unique	Index	Computed Columns
<	ID	smallint		<				
	employeeID	tinyint	\					
	monday	bit			\searrow			
	tuesday	bit			\searrow			
	wednesday	bit			\searrow			
	thursday	bit			\checkmark			
	friday	bit			\checkmark			
	saturday	bit			\checkmark			
	startHour	time(0)			\checkmark			
	endHour	time(0)			\checkmark			
				DEVIC	E_LOG			
Defi	nition: This table s	tores detailed infor	mation	about cust	omer's device on	its parts.		
Che	ck Constraints: -							
Trig	gers: -							
PK	Column Name	Data Type	FK	Identity	Default Value	Unique	Index	Computed Columns
<	ID	tinyint		<				

EMPLOYEE_AVAILABILITY

Definition: This table stores the employee's available days and their start and end work hours

✓

✓

	Definition: This table represents the repairment process (during and after process). Stores the information gathered from devices, customers and the employees.								
Che	Check Constraints: -								
Trig	Triggers: It updates the numOfLateRepairment value by one when the repairmentDuration exceeds 1 hour.								
PK	Column Name	Data Type	FK	Identity	Default Value	Unique	Index	Computed Columns	
✓	ID	int		>					
	deviceID	bigint	✓				\checkmark		
	caseID	int	>						
	customerID	int	>				\checkmark		
	employeeID	tinyint	>				\		
	repairmentStartDate	datetime							
	repairmentEndDate	datetime							
	isPartWaited	bit			\checkmark				
	isInWarranty	bit			\checkmark				
	remark	nvarchar(100)							
	repairmentDuration							\checkmark	
				CAS	SE				
Defi	nition: This table keeps	descriptive inforr	nation	about cus	tomer's complai	nt taken fro	m thems	selves.	
Che	ck Constraints: -								
Trig	gers: -								
PK	Column Name	Data Type	FK	Identity	Default Value	Unique	Index	Computed Columns	
/	ID	int		>					
	caseType	tinyint	\						
	caseCategory	tinyint	/						

✓

tinyint

nvarchar(100)

caseSpecification

caseDescription

REPAIRMENT

	CASE_TYPE								
Defi	nition: This table s	stores the case typ	e.						
Che	ck Constraints: -								
Trig	Triggers: -								
PK	Column Name	Data Type	FK	Identity	Default Value	Unique	Index	Computed Columns	
/	ID	tinyint		✓					
	type	nvarchar(30)				V			
CASE_CATEGORY									
Defi	nition: This table s	tores the case cate							
	Definition: This table stores the case categories. Check Constraints: -								
Trig	gers: -								
PK	Column Name	Data Type	FK	Identity	Default Value	Unique	Index	Computed Columns	
✓	ID	tinyint		\searrow					
	category	nvarchar(30)				\			
	caseType	tinyint	>						
			CA	ASE_SPE	CIFICATION				
Defi	nition: This table s	stores the case spe	ecificati	on.					
Che	ck Constraints: -								
Trig	gers: -								
PK	Column Name	Data Type	FK	Identity	Default Value	Unique	Index	Computed Columns	
✓	ID	tinyint		\checkmark					
	specification	nvarchar(50)				>			
	caseCategory	tinyint	>						
					<u> </u>				

	CUSTOMER								
Defi	efinition: This table stores personal information about customers.								
Che	Check Constraints: -								
Trig	riggers: -								
PK	Column Name	Data Type	FK	Identity	Default Value	Unique	Index	Computed Columns	
✓	ID	tinyint		\checkmark					
	firstName	nvarchar(50)							
	lastName	nvarchar(20)							
	phoneNumber	nvarchar(20)							
					R_DEVICE				
	nition: This table storifications and condit			n the custo	mers themselves	s about thei	r devices	s, in addition to	
Che	ck Constraints: -								
Trig	gers: -	_							
PK	Column Name	Data Type	FK	Identity	Default Value	Unique	Index	Computed Columns	
✓	deviceID	bigint							
	customerID	int	\checkmark						
	model	nvarchar(20)							
	colorCode	nvarchar(10)							
	serialCode	nvarchar(15)				>			
	warrantyDueDate	date							
	warranty				\checkmark			\checkmark	
	physicalCondition	nvarchar(100)							
	proofOfPurchase	bit			\checkmark				

	ADDRESS									
Defi	Definition: This table stores the customer's detailed address.									
Che	heck Constraints: -									
Trig	gers: -									
PK	Column Name	D	ata Type	F	K	Identity	Default Value	Unique	Index	Computed Columns
✓	ID	int		[\checkmark				
	customerID	int		[✓					
	streetName	nvar	char(50)	[
	streetNumber	nvar	char(50)	[
	city	nvar	char(50)	[
Country nvarchar(50)										
zipcode nvarchar(10)										
						MANA	GER			
Defi	nition: This table st	ores tit	le of manag	ers,	for ins	stance "Br	anch Manager"	or "Smart S	Service L	eader".
Che	ck Constraints: -									
Trig	gers: -									
PK	Column Name	Dat	а Туре	FK	ld	entity	Default Value	Unique	Index	Computed Columns
✓	ID	tinyint								
	title	nvarch	nar(30)							
					SN	IART_S	ERVICE			
Defi	nition: This table st	ores av	ailability ar	d bu	sines	s-related i	nformation of we	orkers in da	ays and I	nours.
Che	ck Constraints: -									
Trig	gers: -									
PK	PK Column Name Data Type FK Identity Default Value Unique Index Computed Columns									
✓										
	customerHappines	ss	tinyint				$\overline{\vee}$			
	RRR		decimal(5	2)						
	numOfLateRepair	ment	tinyint				\checkmark			

	TECHNICIAN								
	Definition: This table stores types of technicians and business-related information. There are 2 main type of technicians, hese are "In-Warranty Repairer" and "Out-of-Warranty Repairer".								
Che	ck Constraints: -								
Trig	gers: -								
PK	Column Name		Data Type	FK	Identity	Default Value	Unique	Index	Computed Columns
☑ ID			tinyint						
isInWarranty		tinyint			$\overline{\checkmark}$				
RRR			decimal(5,2)			\checkmark			
numOfLateRepairment			tinyint			\checkmark			
					ORDE	R			
Defi	nition: This table sto	res ke	eps track of the	e ordei	rs.				
Che	ck Constraints: -								
Trig	Triggers: -								
PK	Column Name	D	ata Type	FK	Identity	Default Value	Unique	Index	Computed Columns
\checkmark	orderID	int			\checkmark				
	totalCost	smal	Imoney	<u> </u>		\checkmark			
	employeeID	tinyir	nt						
	orderDate								\checkmark
	isConfirmed	bit				\checkmark			
				OF	RDERED	_PART			
Defi	nition: This table sto	res qu	uantity of ordere	ed part	S.				
Che	ck Constraints: qua	ntity v	alue must be g	reater	than 0.				
Trig	Triggers: It updates order's price when the technician makes the order.								
PK	Column Name		Data Type	FK	Identity	Default Value	Unique	Index	Computed Columns
/	orderID		int	\checkmark					
✓	partID		bigint	$\overline{\ }$					
	quantity		smallint			✓			

				PAI	RT			
Defi	Definition: This table stores detailed information of parts.							
Che	ck Constraints: -							
Trig	gers: -							
PK	Column Name	Data Type	FK	Identity	Default Value	Unique	Index	Computed Columns
✓	ID	bigint						
	partName	nvarchar(50)						
	partModel	nvarchar(50)						
	partColor	nvarchar(10)						
	price	smallmoney			\searrow			
			PAR	TS_NEE	D_CHANGE			
Defi	nition: This table co	nnects the PARTS	table a	nd REPAIF	RMENT table.			
Che	ck Constraints: -							
Trig	gers: -							
PK	Column Name	Data Type	FK	Identity	Default Value	Unique	Index	Computed Columns
✓	repairmentID	int	\checkmark					
✓	partID	bigint	\checkmark					
				PAYN	IENT			
	nition: This table sto		ery simila	ar to a rece	eipt. In addition, i	t has inform	nation ab	out the repairment
Che	ck Constraints: -							
Trig	gers: -							
PK	Column Name	Data Type	FK	Identity	Default Value	Unique	Index	Computed Columns
✓	ID	int						
	repairmentID	int	\checkmark					
	accountantID	tinyint	\checkmark					
	totalCost	smallmoney						
	paymentMethod	tinyint	✓					
	date							abla

Definition: This table stores all the payment methods.

Che	ck Constraints: -							
Trig	gers: -							
PK	Column Name	Data Type	FK	Identity	Default Value	Unique	Index	Computed Columns
✓	ID	int		\checkmark				
	paymentMethod	nvarchar(50)				\checkmark		
		S	MART	_SERVIC	E_REPAIRME	NT		
Defi	nition: This table re	presents the repa	irments	which can	be done quickly a	and without	a need o	of technical assistance.
Che	ck Constraints: -							
Trig	gers: -							
PK	Column Name	Data Type	FK	Identity	Default Value	Unique	Index	Computed Columns
✓	ID	int						
	solutionType	tinyint	\checkmark					
			TEC	HNICAL_	REPAIRMENT			
Defi	nition: This table re	presents the repa	irments	which can	only be done via	technical o	peration.	
Che	ck Constraints: rep	pairmentDegree va	alue sh	ould be 0, 1	1, 2 or 3.			
Trig	gers: -							
PK	Column Name	Data Type	FK	Identity	Default Value	Unique	Index	Computed Columns
✓	ID	int						
	repairmentDegree	tinyint						

PAYMENT_METHOD

Detii	nition: This table st	ores the SMART_	SEKVI	CE_REPA	IRMENTS all solut	ion types.		
Che	ck Constraints: -							
Trig	gers: -							
PK	Column Name	Data Type	FK	Identity	Default Value	Unique	Index	Computed Columns
✓	ID	tinyint						
	solutionType	nvarchar(50)				\checkmark		
				STO	RAGE			
	nition: This table re ed parts.	presents a real sto	orage e	exists in the	e technical service	building. K	eeps the	information about
Che	ck Constraints: qu	antity and boxNur	nber sl	nould be g	eater than or equa	ıl 0.		
Trig	gers: -							
PK	Column Name	e Data Ty _l	oe l	FK Ident	ity Default Value	Unique	Index	Computed Columns
\	partID	bigint						
	quantity	smallint						
	boxNumber	smallint						

SOLUTION_TYPE

VIEWS

We decided to implement 5 views about the tables. We tried to create different views to reach specified data. Here is the list of the views.

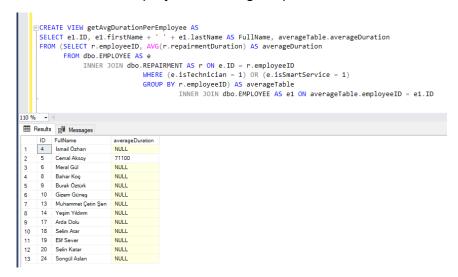
1-) employeesWithRepairmentCountsThisMonth

Definition: It selects all the repairments of the technicians in current month interval.

(Figure 1.1 Result and Query of the view)

2-) getAvgDurationPerEmployee

Definition: It lists all the employee's average repairment time.



(Figure 1.2 Result and Query of the view)

3-) getMostFreqRepairedModel

Definition: It list most frequent repaired model.

```
CREATE VIEW getMostFreqRepairedModel AS

SELECT model, COUNT(*) AS numOfTotalRepairments

FROM dbo.REPAIRMENT AS r

INNER JOIN dbo.CUSTOMER_DEVICE AS cd ON r.deviceID = cd.deviceID

GROUP BY model

HAVING (COUNT(*) = (SELECT MAX(modelCount) AS Expr1

FROM (SELECT cd1.model, COUNT(*) AS modelCount

FROM dbo.REPAIRMENT AS r

INNER JOIN dbo.CUSTOMER_DEVICE AS cd1 ON r.deviceID = cd1.deviceID

GROUP BY cd1.model) AS repairedTable))

I10 % 

Messages

model numOfTotalRepairments

SELECT MAX(model AS cd ON r.deviceID = cd1.deviceID and r.deviceID = cd1.deviceID = cd1.deviceID and r.deviceID = cd1.deviceID = cd1.deviceID
```

(Figure 1.3 Result and Query of the view)

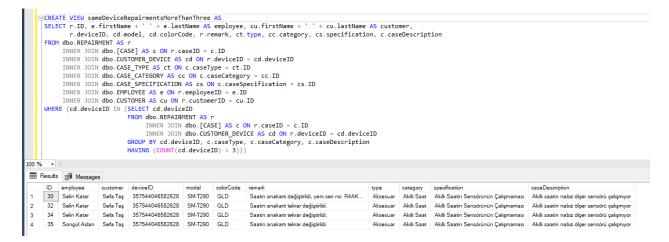
4-) numberOfChangedPartsDistinctModel

Definition: It list number of changed parts based on distinct model (Given number 3 as an example).

(Figure 1.4 Result and Query of the view)

5-) sameDeviceRepairmentsMoreThanThree

Definition: It list all related repairment information that devices which are repaired more than 3 times.



(Figure 1.5 Result and Query of the view)

TRIGGERS

We implemented 2 triggers for our REPAIRMENT and ORDERED_PARTS table. For the REPAIRMENT table we plan to update the **numOfLateRepairment** value by one when the repairmentDuration exceeds 1 hour. For the ORDERED_PARTS table we plan to update order's price when the technician makes the order. By the triggers we keep our database up-to-date when new insertions made. Here is the list of triggers.

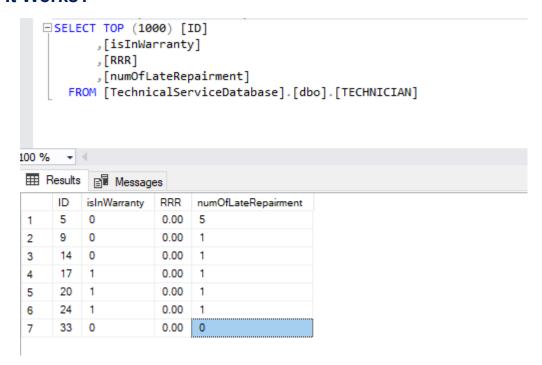
1-) trg_employee_numOfLateRepairment_Update

```
SET ANSI_NULLS ON
   SET QUOTED_IDENTIFIER ON
 ALTER TRIGGER [dbo].[trg_employee_numOfLateRepairment_Update]
  ON [dbo].[REPAIRMENT]
AFTER UPDATE, INSERT
  BEGIN
  declare @startDate datetime
declare @endDate datetime
declare @duration int
  declare @employeeID int
declare @isPartWaited bit
SELECT @startDate = repairmentStartDate, @endDate = repairmentEndDate, @duration=repairmentDuration, @employeeID=employeeID, @isPartWaited = isPartWaited FROM inserted
   declare @isTechnician bit
   SELECT @isTechnician=isTechnician, @isSmart=isSmartService FROM EMPLOYEE e WHERE e.ID=@employeeID
☐IF( @endDate is not NULL and @isPartWaited = 0)
       IF(@duration > 3600 and @isTechnician=1)
            BEGIN

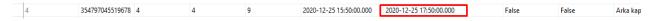
UPDATE dbo.[TECHNICIAN]
                SET numOfLateRepairment = numOfLateRepairment+1
                WHERE ID=@employeeID
      IF(@duration > 3600 and @isSmart=1)
                  JPDATE dbo.[SMART_SERVICE]
                 SET numOfLateRepairment = numOfLateRepairment+1
                WHERE ID=@employeeID
```

(Figure 2.1 Trigger of the REPAIRMENT table)

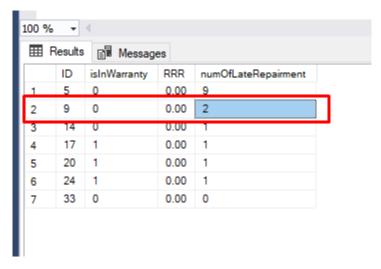
How It Works?



(Figure 2.1.1 List all the Technicians with numOfLateRepairment)



(Figure 2.1.2 Update the employee's repairmentEndDate with 2-hour delay)



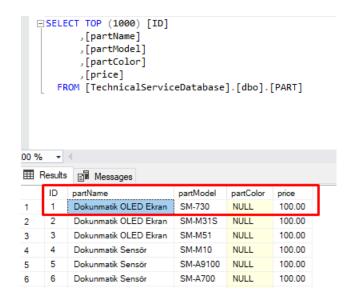
(Figure 2.1.3 See the changed numberOfLateRepairment value)

2-) TrgUpdateTotalCost

```
USE [TechnicalServiceDatabase]
 SET ANSI_NULLS ON
 SET QUOTED IDENTIFIER ON
□ ALTER TRIGGER [dbo].[TrgUpdateTotalCost] ON [dbo].[ORDERED_PART]
    AFTER INSERT, UPDATE
 AS
⊟ BEGIN
    declare @orderID int
    declare @partID bigint
    declare @quantity smallint
    SELECT @orderID=orderID, @partID=partID, @quantity=quantity FROM inserted
    declare @partCost smallmoney
    SELECT @partCost=price FROM dbo.[PART] p WHERE p.ID=@partID;
    --print before update
    SELECT * FROM dbo.[ORDER] o where o.orderID=@orderID;
    UPDATE dbo.[ORDER]
    SET totalCost = totalCost + @partCost * @quantity
    WHERE dbo.[ORDER].orderID=@orderID
    --print after update
    SELECT * FROM dbo.[ORDER] o where o.orderID=@orderID;
 END
```

(Figure 2.2 Trigger of the ORDERED_PART table)

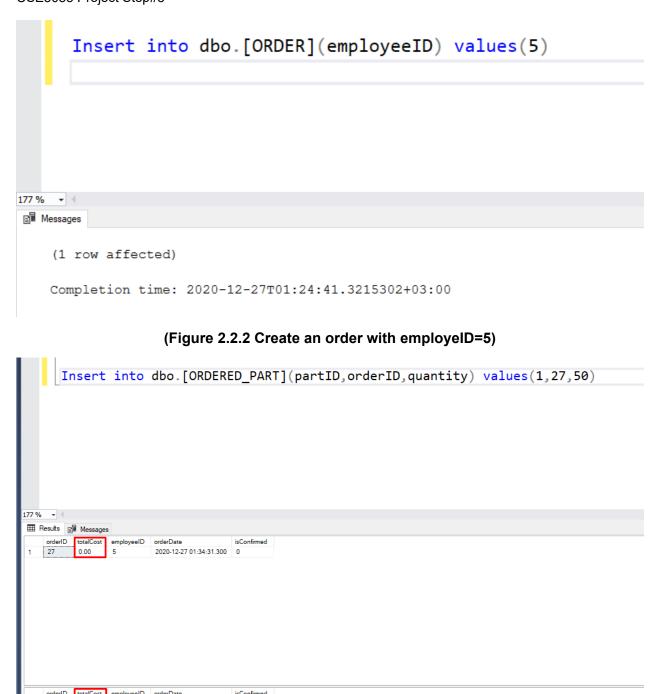
How It Works?



(Figure 2.2.1 List all the parts from PART table (Selected ID=1))

5000.00

2020-12-27 01:34:31.300 0



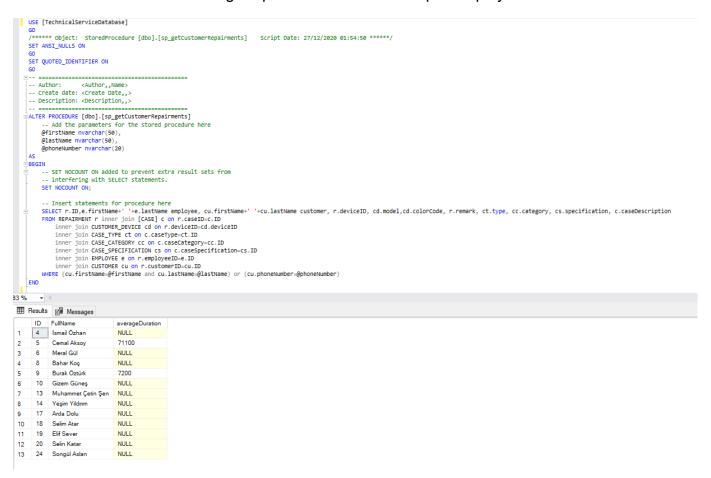
(Figure 2.2.3 See the total cost when a new ordered part created with selected partID, created orderID and quantity value)

Stored Procedures

We implemented 22 stored procedures to handle multiple different jobs. We tried to select most essential jobs to handle with stored procedures. Our stored procedures contain select, update and delete operations. Here is the list of the stored procedures.

1-) sp_getAvgRepairmentDurationPerEmployee

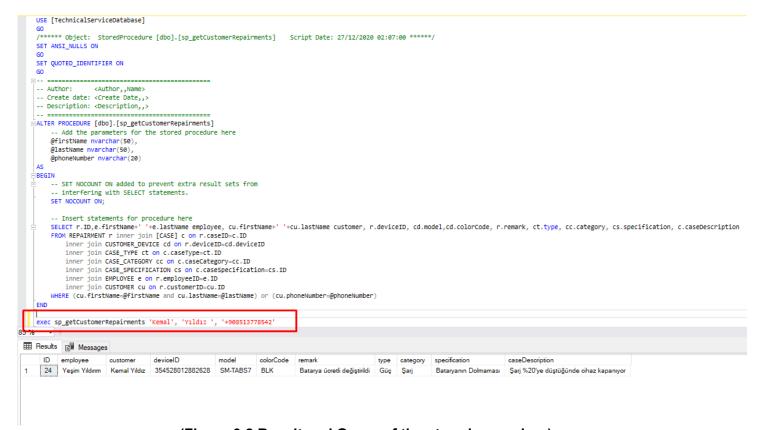
Definition: Get the average repairment duration time per employee.



(Figure 3.1 Result and Query of the stored procedure)

2-) sp_getCustomerRepairments

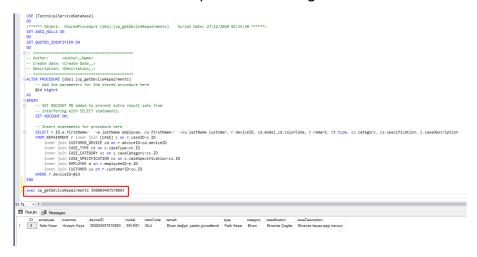
Definition: It lists all the customer's repairments with given customer firstName, lastName and phoneNumber variables.



(Figure 3.2 Result and Query of the stored procedure)

3-) sp_getDeviceRepairments

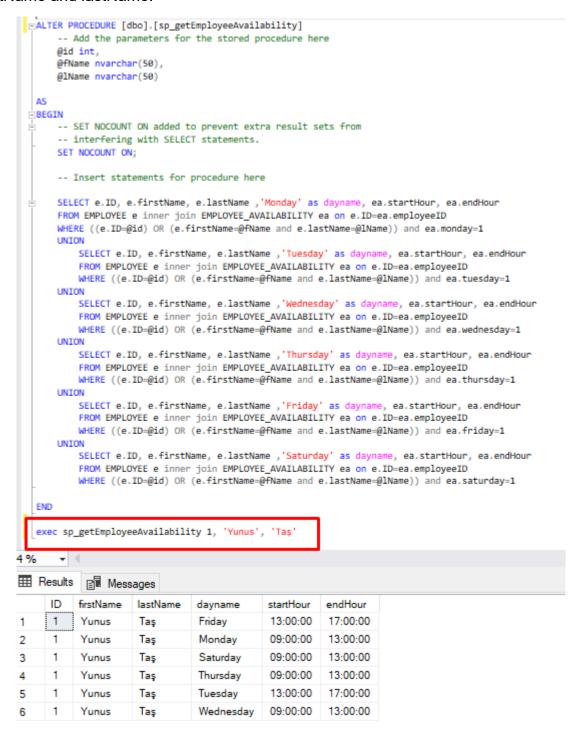
Definition: It lists all the device's repairments with given deviceID variables.



(Figure 3.3 Result and Query of the stored procedure)

4-) sp_getEmployeeAvailability

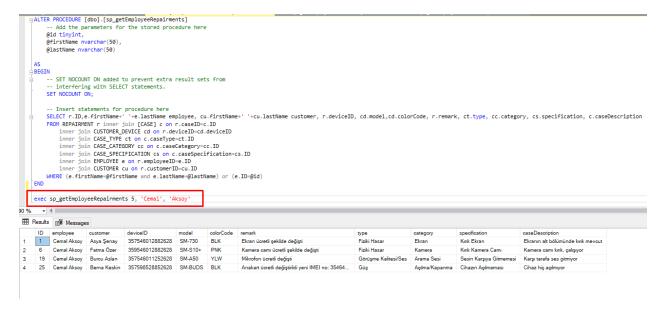
Definition: It lists the employee's available days and hours with given employeeID, firstName and lastName.



(Figure 3.4 Result and Query of the stored procedure)

5-) sp_getEmployeeRepairments

Definition: It list all repairments made by the employee with given employeeID, firstName and lastName.



(Figure 3.5 Result and Query of the stored procedure)

6-) sp_getMostChangedPart

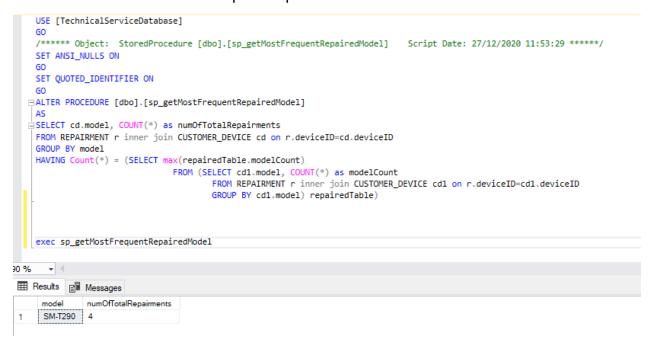
Definition: It lists the part that most changed.

```
USE [TechnicalServiceDatabase]
    /****** Object: StoredProcedure [dbo].[sp_getMostChangedParts] Script Date: 27/12/2020 11:51:02 ******/
    SET ANSI_NULLS ON
    SET QUOTED IDENTIFIER ON
   □ ALTER PROCEDURE [dbo].[sp_getMostChangedParts]
    SELECT p.partName, count(*) as partCount
    FROM PART p inner join PARTS_NEED_CHANGE pt on p.ID=pt.partID
    GROUP BY partName
    HAVING Count(*) = (SELECT max(partName) from (SELECT count(*) as partName
                        FROM PART p inner join PARTS_NEED_CHANGE pnc on p.ID=pnc.partID
                        GROUP BY partName) tableParts)
    exec sp_getMostChangedParts
90 %
Results Messages
     partName
               partCount
     Mikrafon
               5
```

(Figure 3.6 Result and Query of the stored procedure)

7-) sp_getMostFrequentRepairedModel

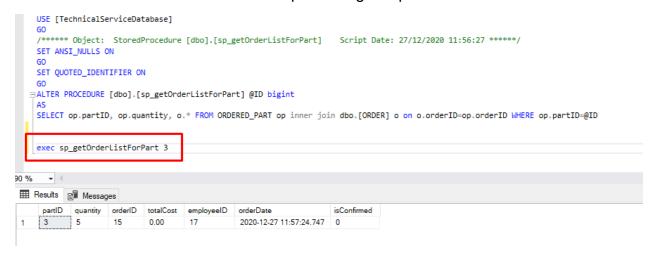
Definition: It lists most frequent repaired model.



(Figure 3.7 Result and Query of the stored procedure)

8-) sp_getOrderListForPart

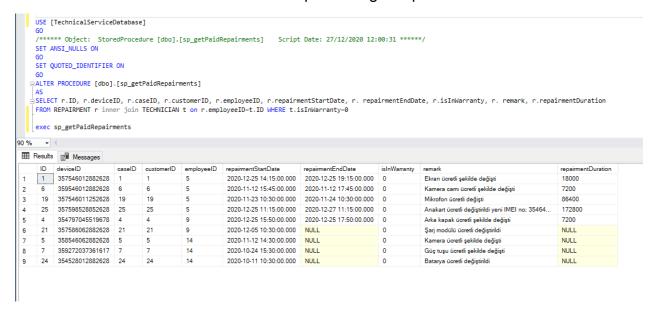
Definition: It lists order list for that part with given partID.



(Figure 3.8 Result and Query of the stored procedure)

9-) sp_getPaidRepairments

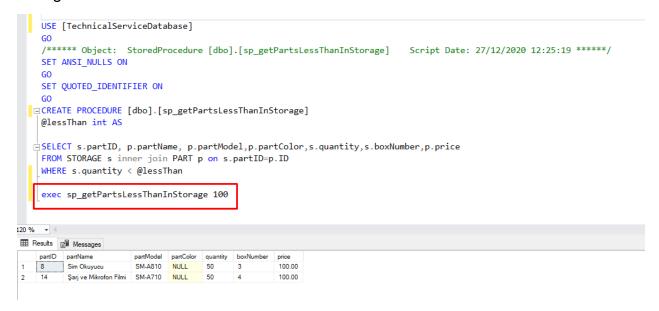
Definition: It lists order list for that part with given partID.



(Figure 3.9 Result and Query of the store procedure)

10-) sp_getPartsLessThanInStorage

Definition: It lists the parts which has less than given quantity parameter in storage.



(Figure 3.10 Result and Query of the stored procedure)

11-) sp_getRepairmentCost

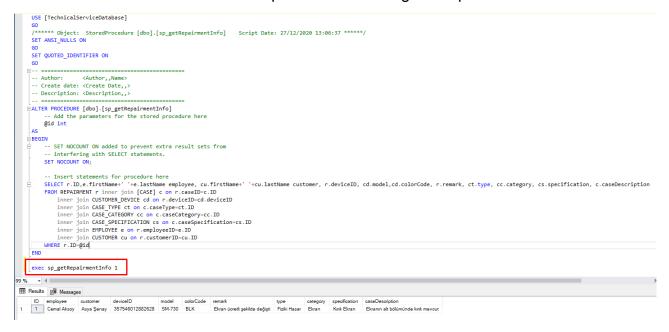
Definition: It lists repairment cost with given repairmentID.

```
USE [TechnicalServiceDatabase]
   SET ANSI_NULLS ON
   SET QUOTED_IDENTIFIER ON
   -- Author:
                <Author,,Name>
   -- Create date: <Create Date,,>
-- Description: <Description,,>
  ALTER PROCEDURE [dbo].[sp_getRepairmentCost]
       -- Add the parameters for the stored procedure here
       @id int
  BEGIN
       -- SET NOCOUNT ON added to prevent extra result sets from -- interfering with SELECT statements.
       SET NOCOUNT ON;
       -- Insert statements for procedure here
       SELECT p.totalCost
       FROM REPAIRMENT r inner join PAYMENT p on r.ID=p.repairmentID
       WHERE r.ID=@id
  exec sp_getRepairmentCost 1
100.00
```

(Figure 3.11 Result and Query of the stored procedure)

12-) sp_getRepairmentInfo

Definition: It lists all related repairment info with given repairmentID.



(Figure 3.12 Result and Query of the stored procedure)

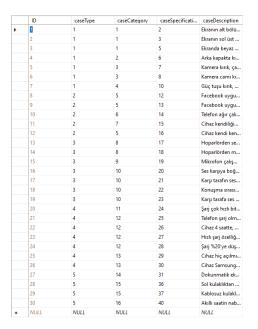
13-) sp_insertNewCase

Definition: It inserts new case to the CASE table with given caseType, caseCategory, caseSpec and caseDesc variables.

```
USE [TechnicalServiceDatabase]
 SET ANSI_NULLS ON
 SET QUOTED_IDENTIFIER ON
-- Author:
           <Author,,Name>
 -- Create date: <Create Date,,>
 -- Description: <Description,,>
 ALTER PROCEDURE [dbo].[sp_insertNewCase]
    -- Add the parameters for the stored procedure here
    @caseType tinyint,
    @caseCategory tinyint,
    @caseSpec tinyint,
    @caseDesc nvarchar(100)
 AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;
    -- Insert statements for procedure here
   INSERT INTO dbo.[CASE](caseType, caseCategory, caseSpecification, caseDescription) VALUES
       (@ case Type, \ @ case Category, \ @ case Spec, \ @ case Desc)\\
END
```

(Figure 3.13 Query of the stored procedure)

How It Works?



(Figure 3.13.1 Before the new case insertion)

```
⊟exec sp_insertNewCase 1, 1, 2, 'Here a description for adding a new case'
       Select * from dbo.[CASE]
.32 % + 4
Results Messages
                      caseCategory caseSpecification
           caseType
                                                     caseDescription
                                    2
      1
                                                      Ekranın alt bölümünde kırık mevcut
2
      2
                                    3
                                                      Ekranın sol üst köşesinde, leke mevcut
      3
                                    5
                                                      Ekranda beyaz çizgi mevcut
      4
                      2
                                    6
                                                      Arka kapakta kink mevcut
      5
                      3
                                     7
5
                                                      Kamera kırık, çalışmıyor
                      3
                                    8
      6
6
                                                      Kamera camı kırık, çalışıyor
                      4
                                     10
                                                      Güç tuşu kırık, çalışmıyor
                      5
      8
           2
                                     12
8
                                                      Facebook uygulaması açılmıyor
                      5
      9
           2
                                     13
9
                                                      Facebook uygulaması güncellenmiyor
10
     10 2
                                     14
                                                      Telefon ağır çalışıyor, donuyor
     11 2
                      7
                                     15
                                                      Cihaz kendiliğinden kapanıyor
11
12
     12 2
                                                      Cihaz kendi kendine reset atıyor
                                     17
13
     13 3
                                                      Hoparlörden ses çıkmıyor
     14 3
                                                      Hoparlörden müzik dinlerken cızırtılı ses çıkıyor
14
                      8
                                     18
     15 3
                                     19
                                                      Mikrofon çalışmıyor
15
      16 3
                      10
                                     20
16
                                                      Ses karşıya boğuk, cızırtılı gidiyor
17
      17
                      10
                                     21
                                                      Karşı tarafın sesi çok az geliyor veya hiç gelmi...
      18
                      10
                                     22
                                                      Konuşma sırasında ses kesikli geliyor
18
      19
                      10
                                     23
                                                      Karşı tarafa ses gitmiyor
19
      20 4
                                     24
                      11
                                                      Şarj çok hızlı bitiyor
20
     21 4
                      12
                                     25
                                                      Telefon şarj olmuyor
21
     22 4
                      12
                                     26
                                                      Cîhaz 4 saatte, çok yavaş şarj oluyor
22
     23 4
                      12
                                     27
                                                      Hızlı şarj özelliği çalışmıyor
23
24
     24 4
                      12
                                     28
                                                      Şarj %20'ye düştüğünde oihaz kapanıyor
     25 4
                      13
                                     29
                                                      Cihaz hiç açılmıyor
     26 4
                      13
                                     30
                                                      Cihaz Samsung logosunda takılıyor, açılmıyor
26
     27 5
                      14
                                     31
27
                                                      Dokunmatik ekran kalemi algılamıyor
     28 5
                      15
                                     36
                                                      Sol kulaklıktan çok az ses geliyor
28
      29 5
                                     37
                                                      Kablosuz kulaklık telefona bağlanmıyor
                      15
29
                       16
                                     40
                                                      Akıllı saatin nabız ölçer sensörü çalışmıyor
31 33 1
                                     2
                                                      Here a description for adding a new case
```

(Figure 3.13.2 After the new case insertion)

14-) sp_insertNewCustomer

Definition: It inserts new customer to the CUSTOMER table with given firstName, lastName, phoneNumber, streetName, streetNumber, city, country and zipcode variables. At the end customer's address information will send to ADDRESS table.

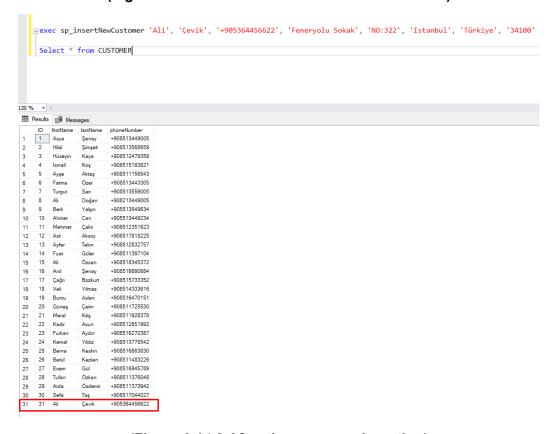
```
USE [TechnicalServiceDatabase]
 /***** Object: StoredProcedure [dbo].[sp_insertNewCustomer] Script Date: 27/12/2020 13:37:11 ******/
 SET ANSI_NULLS ON
 SET QUOTED_IDENTIFIER ON
---
 -- Author: <Author,,Name>
 -- Create date: <Create Date,,>
 -- Description: <Description,,>
 -- -----
□ ALTER PROCEDURE [dbo].[sp_insertNewCustomer]
     -- Add the parameters for the stored procedure here
     @firstName nvarchar(50),
     @lastName nvarchar(50),
     @phoneNumber nvarchar(20),
     @streetName nvarchar(50),
     @streetNumber nvarchar(50),
     @city nvarchar(50),
     @county nvarchar(50),
     @zipcode nvarchar(10)
 AS
ĖBEGIN
     -- SET NOCOUNT ON added to prevent extra result sets from
     -- interfering with SELECT statements.
     SET NOCOUNT ON;
     -- Insert statements for procedure here
     INSERT INTO dbo.[CUSTOMER](firstName, lastName, phoneNumber) VALUES
        (@firstName, @lastName, @phoneNumber)
     DECLARE @id int
     SELECT @id=c.ID FROM CUSTOMER c WHERE c.firstName=@firstName and c.lastName=alastName and c.phoneNumber=@phoneNumber
     INSERT INTO dbo.[ADDRESS](customerID, streetName, streetNumber, city, country, zipcode) VALUES
        (@id, @streetName, @streetNumber, @city, @county, @zipcode)
 END
```

(Figure 3.14 Query of the stored procedure)

How It Works?

ID	firstName	lastName	phoneNumber
1	Asya	Şenay	+908513449005
2	Hilal	Şimşek	+908513568659
3	Hüseyin	Kaya	+908512479358
4	İsmail	Koç	+908515183821
5	Ayşe	Aktaş	+908511156543
6	Fatma	Özer	+908513443305
7	Turgut	Sarı	+908513559005
8	Ali	Doğan	+908213449005
9	Berk	Yalçın	+905513549634
10	Ahmet	Can	+905513449234
11	Mehmet	Çakır	+908512351623
12	Aslı	Aksoy	+908517818225
13	Ayfer	Tekin	+908512832757
14	Fuat	Güler	+908511397104
15	Ali	Özcan	+908518345372
16	Anıl	Şenay	+908516690684
17	Çağrı	Bozkurt	+908515733352
18	Veli	Yılmaz	+908514333616
19	Burcu	Aslan	+908516470151
20	Güneş	Çetin	+908511725530
21	Meral	Kılıç	+908511928378
22	Kadir	Acun	+908512851992
23	Furkan	Aydın	+908516270387
24	Kemal	Yıldız	+908513778542
25	Berna	Keskin	+908516863830
26	Betül	Kaplan	+908511483226
27	Ecem	Gül	+908516945789
28	Tufan	Özkan	+908511376048
29	Arda	Özdemir	+908511373942
30	Sefa	Taş	+908517044027

(Figure 3.14.1 Before the new customer insertion)



(Figure 3.14.2 After the new case insertion)

15-) sp_insertNewEmployee

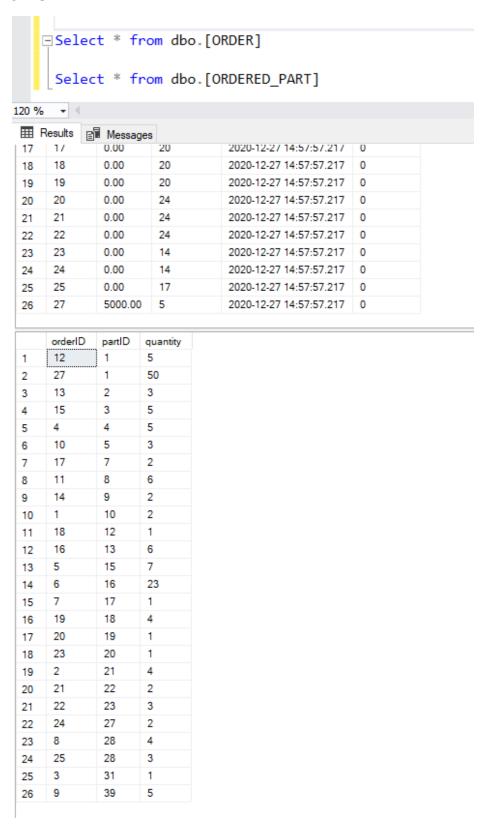
Definition: It inserts new order with given employeeID. It can also add maximum 10 part with their quantities to that order.

```
CREATE PROCEDURE [dbo].[sp_insertNewOrder]

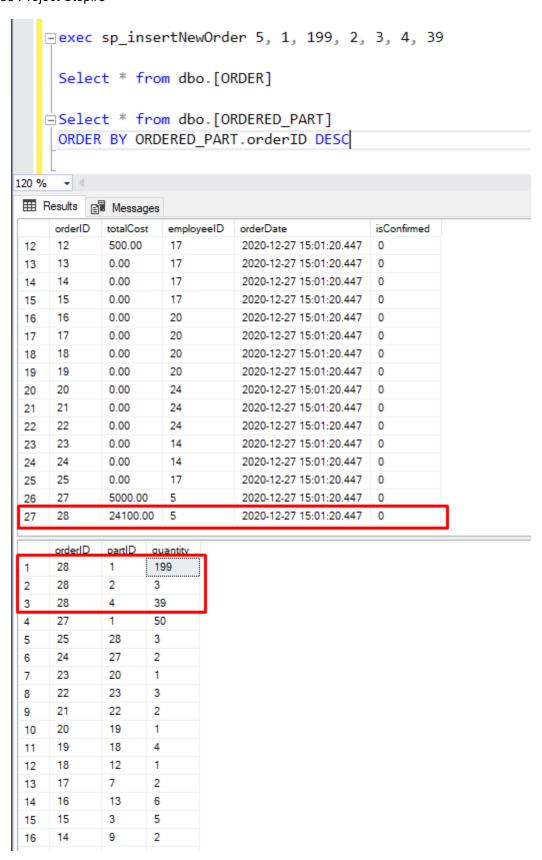
-- Add the parameters for the stored procedure here
@employeeID tinyint,
           MpartID 1 bigint - NULL
           @partID_1_quantity smallint = NULL,
          @partID 2 bigint = NULL,
@partID 2 quantity smallint = NULL,
@partID 3 bigint = NULL,
           @partID 3 quantity smallint = NULL,
@partID 4 bigint = NULL,
          @partID 4 quantity smallint = NULL,
@partID 5 bigint = NULL,
          MartID_S igant = NULL,
MpartID_S duantity smallint = NULL,
MpartID_S quantity smallint = NULL,
MpartID_7 bigint = NULL,
MpartID_7 quantity smallint = NULL,
MpartID_8 bigint = NULL,
MpartID_8 
           @partID_8_quantity smallint = NULL,
          @partID 9 bigint = NULL,
@partID 9 quantity smallint = NULL,
@partID 18 bigint = NULL,
@partID 10 quantity smallint = NULL
 BEGIN
           -- SET NOCOUNT ON added to prevent extra result sets from
                 interfering with SELECT statements.
          SET NOCOUNT ON:
                 Insert statements for procedure here
           INSERT INTO dbo.[ORDER](employeeID) VALUES
                  (@employeeID)
           DECLARE GorderID int
          SELECT @orderID = (SELECT SCOPE_IDENTITY())
SELECT CONCAT('Order with id ', @orderID) + ' is added.';
           IF(@partID_1 is not MULL and @partID_1_quantity is not MULL)
                   INSERT INTO dbo.[ORDERED_PART](orderID, partID, quantity) VALUES (@orderID, @partID_1, @partID_1_quantity)
           IF(@partID_2 is not NULL and @partID_2_quantity is not NULL)
                   INSERT INTO dbo.[ORDERED_PART](orderID, partID, quantity) VALUES (@orderID, @partID_2, @partID_2_quantity)
           IF(@partID_3 is not NULL and @partID_3_quantity is not NULL)
                  INSERT INTO dbo.[ORDERED_PART](orderID, partID, quantity) VALUES (@orderID, @partID_3, @partID_3 quantity)
           IF(@partID_4 is not NULL and @partID_4_quantity is not NULL)
                  INSERT INTO dbo.[ORDERED PART](orderID, partID, quantity) VALUES (@orderID, @partID 4, @partID 4 quantity)
           IF(@partID_5 is not NULL and @partID_5_quantity is not NULL)
                   INSERT INTO dbo.[ORDERED_PART](orderID, partID, quantity) VALUES (@orderID, @partID_5, @partID_5_quantity)
           IF(@partID_6 is not NULL and @partID_6_quantity is not NULL)
                   INSERT INTO dbo.[ORDERED_PART](orderID, partID, quantity) VALUES (@orderID, @partID_6, @partID_6_quantity)
           IF(@partID_7 is not NULL and @partID_7_quantity is not NULL)
                   INSERT INTO dbo.[ORDERED_PART](orderID, partID, quantity) VALUES (@orderID, @partID_7, @partID_7_quantity)
           IF(@partID_8 is not NULL and @partID_8_quantity is not NULL)
                   INSERT INTO dbo.[ORDERED_PART](orderID, partID, quantity) VALUES (@orderID, @partID_8, @partID_8_quantity)
           IF(@partID 9 is not MULL and @partID 9 quantity is not MULL)
                  INSERT INTO dbo.[ORDERED_PART](orderID, partID, quantity) VALUES (@orderID, @partID_9, @partID_9 quantity)
           IF(@partID_10 is not NULL and @partID_10_quantity is not NULL)
                  INSERT INTO dbo.[GRDERED_PART](orderID, partID, quantity) VALUES (@orderID, @partID_10, @partID_10_quantity)
```

(Figure 3.15 Query of the stored procedure)

How It Works?



(Figure 3.15.1 Before the insert operation)



(Figure 3.15.2 After the insert operation (orderID=28))

16-) sp_insertNewEmployee

Definition: It inserts new employee with given, firstName, lastName, phoneNumber, username, password, email, address, dateOfBirth, startDate and type variables. At the end with type variable, the employee's branch and manager will be identified.

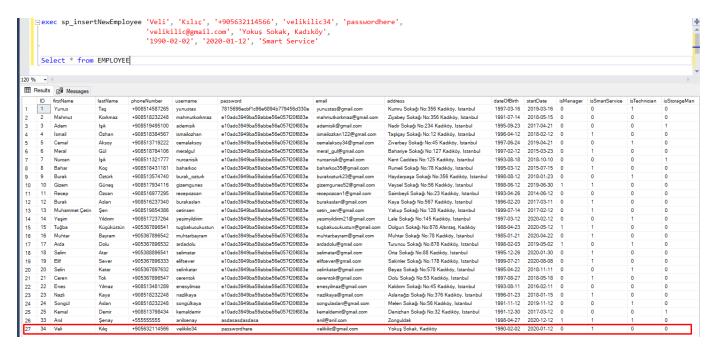
```
□ALTER PROCEDURE [dbo].[sp_insertNewEmployee]

-- Add the parameters for the stored proc
@firstName nvarchar(50),
          @lastName nvarchar(50),
@phoneNumber nvarchar(20),
          @username nvarchar(30),
          @password nvarchar(32)
         @email nvarchar(100),
          Maddress nvarchar(150)
          @dateOfBirth date,
          @startDate date.
         @type nvarchar(32)
         -- SET NOCOUNT ON added to prevent extra result sets from
         -- interfering with SELECT statements.
SET NOCOUNT ON;
         DECLARE @isSmartService bit
DECLARE @isTechnician bit
DECLARE @isStorageMan bit
          DECLARE @isTester bit
         DECLARE @isAccountant bit
         SET @isManager = CASE WHEN LOWER(@type) like '%manager%' THEN 1 ELSE 0 END
SET @isSmartService = CASE WHEN LOWER(@type) Like '%smart%' THEN 1 ELSE 0 END
SET @isTechnician = CASE WHEN LOWER(@type) like '%tech%' THEN 1 ELSE 0 END
SET @isStorageMan = CASE WHEN LOWER(@type) like '%storage%' THEN 1 ELSE 0 END
SET @isSester = CASE WHEN LOWER(@type) like '%test%' THEN 1 ELSE 0 END
SET @isAccountant = CASE WHEN LOWER(@type) like '%account%' THEN 1 ELSE 0 END
           -- Insert statements for procedure here
         INSERT INCO dob. [EMPLOYEE](firstName, lastName, phoneNumber, username, password, email, address, dateOfBirth, startDate, isManager, isSmartService, isTechnician, isStorageMan, isTester, isAccountant) VALUES (@firstName, @lastName, @phoneNumber, @username, @password, @email, @address, @dateOfBirth, @startDate, @isManager, @isSmartService, @isTechnician, @isStorageMan, @isTester, @isAccountant)
         SELECT @id=ID FROM EMPLOYEE e WHERE e.firstName=@firstName and e.lastName=@lastName and e.email=@email
         IF (@isManager=1)
               DECLARE @title nvarchar(30)
               SET @title = CASE WHEN @isSmartService=1 THEN 'Smart Manager' ELSE '' END
SET @title = CASE WHEN @isTechnician=1 THEN 'Service Manager' ELSE '' END
                INSERT INTO dbo.[MANAGER](ID, title) VALUES
                (@id, @title)
                INSERT INTO dbo.[SMART_SERVICE](ID) VALUES
                IF (@isTechnician=1)
         REGIN
                INSERT INTO dbo.[TECHNICIAN](ID) VALUES
                (@id)
```

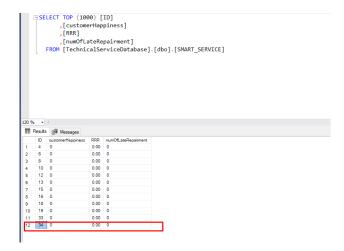
(Figure 3.16 Query of the stored procedure)

ID	firstName	lastName	phoneNumber	username	password	email	address	dateOfBirth	startDate	isManager	isSmartService	isTechnician	isStorageMa
1	Yunus	Taş	+908514587265	yunustas	7815696ecbf1c9	yunustas@gma	Kumru Sokağı	1997-03-16	2019-03-16	False	False	True	False
2	Mahmut	Korkmaz	+908518232248	mahmutkorkmaz	e10adc3949ba5	mahmutkorkm	Ziyabey Sokağı	1991-07-14	2018-05-15	False	False	False	False
3	Adem	lşık	+908519495100	ademisik	e10adc3949ba5	ademisik@gma	Nadir Sokağı N	1995-09-23	2017-04-21	False	False	False	True
4	İsmail	Özhan	+908518384567	ismailozhan	e10adc3949ba5	ismailozkan122	Taşlıçay Sokağı	1996-04-12	2018-02-12	False	True	False	False
5	Cemal	Aksoy	+908513719222	cemalaksoy	e10adc3949ba5	cemalaksoy34	Ziverbey Sokağı	1997-06-24	2019-04-21	False	False	True	False
6	Meral	Gül	+908518784106	meralgul	e10adc3949ba5	meral_gul@gm	Bahariye Sokağı	1997-02-12	2015-03-23	False	True	False	False
7	Nurcan	lşık	+908511321777	nurcanisik	e10adc3949ba5	nurcanisik@gm	Kent Caddesi N	1993-08-18	2018-10-10	False	False	False	True
8	Bahar	Koç	+908518431181	baharkoc	e10adc3949ba5	baharkoc35@g	Rumeli Sokağı	1995-03-12	2015-07-15	False	True	False	False
9	Burak	Öztürk	+908513574740	burak_ozturk	e10adc3949ba5	burakozturk23	Haydarpaşa So	1998-08-12	2018-01-23	False	False	True	False
10	Gizem	Güneş	+908517934116	gizemgunes	e10adc3949ba5	gizemgunes52	Veysel Sokağı N	1998-06-12	2019-06-30	True	True	False	False
11	Recep	Özcan	+908516977295	recepozcan	e10adc3949ba5	recepozcan1@	Saimbeyli Soka	1993-04-26	2014-06-12	False	False	False	False
12	Burak	Aslan	+908516237340	burakaslan	e10adc3949ba5	burakaslan@g	Kaya Sokağı No	1996-02-20	2017-03-11	False	True	False	False
13	Muhammet Çe	Şen	+908519854386	cetinsen	e10adc3949ba5	cetin_sen@gm	Yakup Sokağı N	1999-07-14	2017-02-12	False	True	False	False
14	Yeşim	Yıldırım	+908517237264	yesimyildirim	e10adc3949ba5	yesimyildirim21	Lale Sokağı No:	1997-03-12	2020-02-12	False	False	True	False
15	Tuğba	Küçüküstün	+905367896541	tugbakucukust	e10adc3949ba5	tugbakucukust	Dolgun Sokağı	1988-04-23	2020-05-12	True	True	False	False
16	Muhtar	Bayram	+905367896542	muhtarbayram	e10adc3949ba5	muhtarbayram	Muhtar Sokağı	1985-01-21	2020-04-22	False	True	False	False
17	Arda	Dolu	+905367896532	ardadolu	e10adc3949ba5	ardadolu@gma	Turuncu Sokağı	1998-02-03	2019-05-02	True	False	True	False
18	Selim	Atar	+905388896541	selimatar	e10adc3949ba5	selimatar@gma	Orta Sokağı No:	1995-12-26	2020-01-30	False	True	False	False
19	Elif	Sever	+905367896333	elifsever	e10adc3949ba5	elifsever@gmai	Sakinler Sokağı	1999-07-21	2020-08-08	False	True	False	False
20	Selin	Katar	+905367897632	selinkatar	e10adc3949ba5	selinkatar@gm	Beyaz Sokağı N	1995-04-22	2018-11-11	False	False	True	False
21	Ceren	Tok	+905367896547	cerentok	e10adc3949ba5	cerentok@gma	Dolu Sokağı No	1997-08-27	2018-05-18	False	True	False	False
22	Enes	Yılmaz	+908513481289	enesyilmaz	e10adc3949ba5	enesyilmaz@g	Kaldırım Sokağı	1993-08-11	2016-02-11	False	False	False	False
23	Nazlı	Kaya	+908518232248	nazlikaya	e10adc3949ba5	nazlikaya@gma	Aslanağzı Soka	1996-01-23	2018-01-15	False	True	False	False
24	Songül	Aslan	+908518232248	songülkaya	e10adc3949ba5	songulaslan@g	Melen Sokağı N	1991-11-12	2019-11-12	False	False	True	False
25	Kemal	Demir	+908513798434	kemaldemir	e10adc3949ba5	kemaldemir@g	Denizhan Soka	1991-12-30	2017-03-12	False	False	False	True
33	Anil	Şenay	+55555555	anilsenay	asdasasdasdasa	anil@anil.com	Zonguldak	1998-04-27	2020-12-12	True	True	True	False
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

(Figure 3.16.1 Before the new employee insertion)



(Figure 3.16.2 After the new employee insertion)



(Figure 3.16.3 After the new employee insertion (Smart Service Insertion))

17-) sp_insertNewRepairment

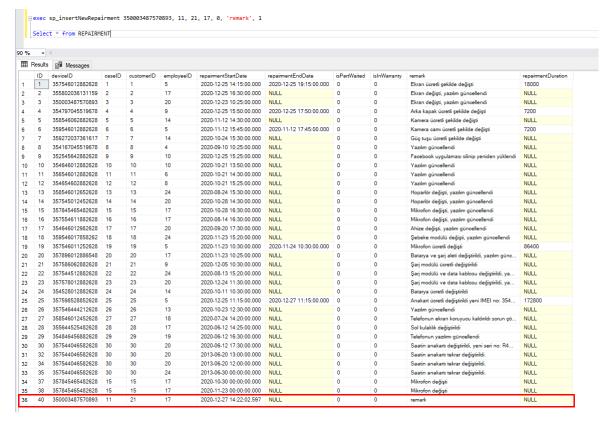
Definition: It inserts new repairment with deviceID, caseID, customerID, employeeID, isInWarranty, remark, and value. Based on value and employee type it also inserts SMART SERVICE and TECHNICAL REPAIRMENT table.

```
ALTER PROCEDURE [dbo].[sp_insertNewRepairment]
     -- Add the parameters for the stored procedure here
     @deviceID bigint,
     @caseID int,
     @customerID int.
     @employeeID tinyint,
     @isInWarranty bit,
     @remark_nyarchar(100).
     @value tinyint
BEGIN
     -- SET NOCOUNT ON added to prevent extra result sets from
     -- interfering with SELECT statements.
     SET NOCOUNT ON;
    INSERT INTO dbo.[REPAIRMENT](deviceID, caseID, customerID, employeeID, isInWarranty, remark, repairmentStartDate) VALUES
         (@deviceID, @caseID, @customerID, @employeeID, @isInWarranty, @remark, GETDATE())
     DECLARE @repairmentID int
     SELECT @repairmentID=r.ID FROM REPAIRMENT r WHERE r.deviceID=@deviceID and r.caseID=@caseID and r.customerID=@customerID and r.employeeID=@employeeID
     DECLARE @isTech bit
     DECLARE @isSmart bit
    SELECT @isTech=isTechnician, @isSmart=isSmartService FROM EMPLOYEE e WHERE e.ID=@employeeID
     IF(@isTech=1 \ and \ @value \ in \ (0,1,2,3))
         INSERT INTO dbo.[TECHINCAL_REPAIRMENT](ID,repairmentDegree) VALUES (@repairmentID, @value)
     END
     IF(@isSmart=1 \ and \ @value \ in(1,2))
         INSERT INTO dbo.[SMART_SERVICE_REPAIRMENT](ID, solutionType) VALUES (@repairmentID, @value)
     END
 END
```

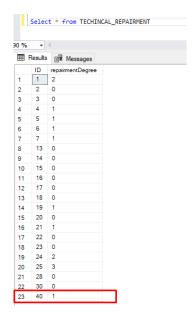
(Figure 3.17 Query of the stored procedure)

	ID	devicelD	caseID	customerID	employeelD	repairmentStar	repairmentEnd	isPartWaited	isInWarranty	remark	repairmentDur
	1	357546012882628	1	1	5	2020-12-25 14:1	2020-12-25 19:1	False	False	Ekran ücretli şe	18000
	2	355802036131159	2	2	17	2020-12-25 16:3	NULL	False	False	Ekran değişti, y	NULL
	3	350003487570893	3	3	20	2020-12-23 10:2	NULL	False	False	Ekran değişti, y	NULL
	4	354797045519678	4	4	9	2020-12-25 15:5	2020-12-25 17:5	False	False	Arka kapak ücr	7200
	5	358546062882628	5	5	14	2020-11-12 14:3	NULL	False	False	Kamera ücretli	NULL
	6	359546012882628	6	6	5	2020-11-12 15:4	2020-11-12 17:4	False	False	Kamera camı ü	7200
	7	359272037361617	7	7	14	2020-10-24 15:3	NULL	False	False	Güç tuşu ücretli	NULL
	8	354167045519678	8	8	4	2020-09-10 10:2	NULL	False	False	Yazılım güncell	NULL
	9	352545642882628	9	9	10	2020-12-25 15:2	NULL	False	False	Facebook uygu	NULL
	10	354646012882628	10	10	10	2020-10-21 13:5	NULL	False	False	Yazılım güncell	NULL
	11	356546012882628	11	11	6	2020-10-21 14:3	NULL	False	False	Yazılım güncell	NULL
	12	354654602882628	12	12	8	2020-10-21 15:2	NULL	False	False	Yazılım güncell	NULL
	13	358546012652628	13	13	24	2020-08-24 15:3	NULL	False	False	Hoparlör değişt	NULL
	14	357545012452628	14	14	20	2020-10-28 14:3	NULL	False	False	Hoparlör değişt	NULL
	15	357845465482628	15	15	17	2020-10-28 16:3	NULL	False	False	Mikrofon değiş	NULL
	16	357554611882628	16	16	17	2020-08-14 16:3	NULL	False	False	Mikrofon değiş	NULL
	17	354646012982628	17	17	20	2020-09-20 17:3	NULL	False	False	Ahize değişti, y	NULL
	18	359546017858262	18	18	24	2020-11-23 15:2	NULL	False	False	Şebeke modülü	NULL
	19	357546011252628	19	19	5	2020-11-23 10:3	2020-11-24 10:3	False	False	Mikrofon ücretl	86400
	20	357896012886548	20	20	17	2020-11-23 10:2	NULL	False	False	Batarya ve şarj	NULL
	21	357586062882628	21	21	9	2020-12-05 10:3	NULL	False	False	Şarj modülü üc	NULL
	22	357544512882628	22	22	24	2020-08-13 15:2	NULL	False	False	Şarj modülü ve	NULL
	23	357578012882628	23	23	20	2020-12-24 11:3	NULL	False	False	Şarj modülü ve	NULL
	24	354528012882628	24	24	14	2020-10-11 10:3	NULL	False	False	Batarya ücretli	NULL
	25	357598528852628	25	25	5	2020-12-25 11:1	2020-12-27 11:1	False	False	Anakart ücretli	172800
	26	357546444212628	26	26	13	2020-10-23 12:3	NULL	False	False	Yazılım güncell	NULL
	27	358546012452628	27	27	18	2020-07-24 14:2	NULL	False	False	Telefonun ekra	NULL
	28	355644525482628	28	28	17	2020-06-12 14:2	NULL	False	False	Sol kulaklık değ	NULL
	29	354846456882628	29	29	19	2020-06-12 16:3	NULL	False	False	Telefonun yazılı	NULL
	30	357544046582628	30	30	20	2020-06-12 17:3	NULL	False	False	Saatin anakartı	NULL
	32	357544046582628	30	30	20	2013-06-20 13:0	NULL	False	False	Saatin anakartı	NULL
	34	357544046582628	30	30	20	2013-06-20 12:0	NULL	False	False	Saatin anakartı	NULL
	35	357544046582628	30	30	24	2013-06-30 00:0	NULL	False	False	Saatin anakartı	NULL
	37	357845465482628	15	15	17	2020-10-30 00:0	NULL	False	False	Mikrofon değişti	NULL
	38	357845465482628	15	15	17	2020-11-23 00:0	NULL	False	False	Mikrofon değişti	NULL
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

(Figure 3.17.1 Before the new repairment insertion)



(Figure 3.17.2 After the new case insertion)



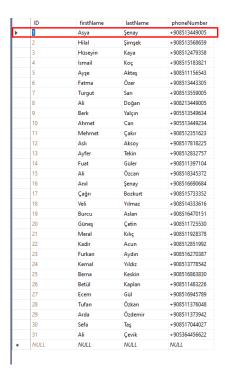
(Figure 3.17.3 After the new repairment insertion (Technical Repairment))

18-) sp_updateCustomer

Definition: It updates the customer's information based on customerID variable. If firstName, lastName and phoneNumber values are empty, it will update with old variables.

```
USE [TechnicalServiceDatabase]
 SET QUOTED_IDENTIFIER ON
 -- Author:
              <Author,,Name>
 -- Create date: <Create Date,,>
 -- Description: <Description,,>
☐ ALTER PROCEDURE [dbo].[sp_updateCustomer]
     -- Add the parameters for the stored procedure here
    @id int,
    @firstName nvarchar(50).
    @lastName nvarchar(50),
    @phoneNumber nvarchar(20)
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;
    DECLARE @fName nvarchar(50)
    DECLARE @lName nvarchar(50)
    DECLARE @pNumber nvarchar(20)
    SELECT @fName=firstName, @lName=lastName, @pNumber=phoneNumber FROM CUSTOMER c WHERE c.ID=@id
    SET @firstName = CASE WHEN @firstName='' THEN @fName ELSE @firstName END
    SET @lastName = CASE WHEN @lastName='' THEN @lName ELSE @lastName END
    SET @phoneNumber = CASE WHEN @phoneNumber='' THEN @pNumber ELSE @phoneNumber END
     -- Insert statements for procedure here
     UPDATE CUSTOMER SET firstName=@firstName, lastName=@lastName, phoneNumber=@phoneNumber WHERE ID=@id
```

(Figure 3.18 Query of the stored procedure)



(Figure 3.18.1 Before the update operation (Selected ID=1))



(Figure 3.18.2 After the update operation)

19-) sp_updatePassword

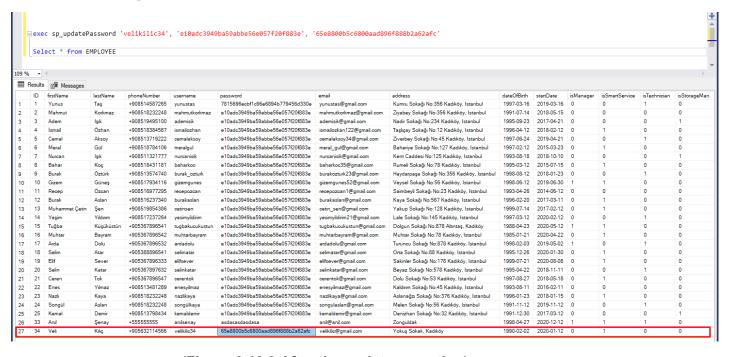
Definition: It updates the employee's password with given username oldPassword and newPassword variables. It also handles basic password creation procedure.

```
USE [TechnicalServiceDatabase]
 SET ANSI_NULLS ON
 SET QUOTED_IDENTIFIER ON
 -- Author: <Author,,Name>
 -- Create date: <Create Date,,>
 -- Description: <Description,,>
 -- ------
□ ALTER PROCEDURE [dbo].[sp_updatePassword]
    -- Add the parameters for the stored procedure here
    @username nvarchar(30),
    @oldPassword nvarchar(32),
    @newPassword nvarchar(32)
 AS
⊨BEGIN
   -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;
    -- Insert statements for procedure here
    DECLARE @pass nvarchar(32)
    SELECT @pass=e.password FROM EMPLOYEE e WHERE e.username=@username
    IF(@pass=@oldPassword)
-0-0-0-
    BEGIN
       IF(@pass=@newPassword)
           SELECT 'New password can not be the same as old!'
       END
        ELSE
₽
           UPDATE EMPLOYEE SET password=@newPassword WHERE username=@username
    END
    ELSE
    BEGIN
      SELECT 'Wrong Password!'
    END
 END
```

(Figure 3.19 Query of the stored procedure)

ID	firstName	lastName	phoneNumber	username	password	email	address	dateOfBirth	startDate	isManager	isSmartService	isTechnician	isStorageMan	isTes
1	Yunus	Taş	+908514587265	yunustas	7815696ecbf1c9	yunustas@gma	Kumru Sokağı	1997-03-16	2019-03-16	False	False	True	False	False
2	Mahmut	Korkmaz	+908518232248	mahmut	e10adc3949ba5	mahmutkorkm	Ziyabey Sokağı	1991-07-14	2018-05-15	False	False	False	False	True
3	Adem	lşık	+908519495100	ademisik	e10adc3949ba5	ademisik@gma	Nadir Sokağı N	1995-09-23	2017-04-21	False	False	False	True	False
4	İsmail	Özhan	+908518384567	ismailozh	e10adc3949ba5	ismailozkan122	Taşlıçay Sokağı	1996-04-12	2018-02-12	False	True	False	False	False
5	Cemal	Aksoy	+908513719222	cemalaks	e10adc3949ba5	cemalaksoy34	Ziverbey Sokağı	1997-06-24	2019-04-21	False	False	True	False	False
6	Meral	Gül	+908518784106	meralgul	e10adc3949ba5	meral_gul@gm	Bahariye Sokağı	1997-02-12	2015-03-23	False	True	False	False	False
7	Nurcan	lşık	+908511321777	nurcanisik	e10adc3949ba5	nurcanisik@gm	Kent Caddesi N	1993-08-18	2018-10-10	False	False	False	True	False
8	Bahar	Koç	+908518431181	baharkoc	e10adc3949ba5	baharkoc35@g	Rumeli Sokağı	1995-03-12	2015-07-15	False	True	False	False	False
9	Burak	Öztürk	+908513574740	burak_ozt	e10adc3949ba5	burakozturk23	Haydarpaşa So	1998-08-12	2018-01-23	False	False	True	False	False
10	Gizem	Güneş	+908517934116	gizemgu	e10adc3949ba5	gizemgunes52	Veysel Sokağı N	1998-06-12	2019-06-30	True	True	False	False	False
11	Recep	Özcan	+908516977295	recepozcan	e10adc3949ba5	recepozcan1@	Saimbeyli Soka	1993-04-26	2014-06-12	False	False	False	False	True
12	Burak	Aslan	+908516237340	burakaslan	e10adc3949ba5	burakaslan@g	Kaya Sokağı No	1996-02-20	2017-03-11	False	True	False	False	False
13	Muhammet Çe	Şen	+908519854386	cetinsen	e10adc3949ba5	cetin_sen@gm	Yakup Sokağı N	1999-07-14	2017-02-12	False	True	False	False	False
14	Yeşim	Yıldırım	+908517237264	yesimyild	e10adc3949ba5	yesimyildirim21	Lale Sokağı No:	1997-03-12	2020-02-12	False	False	True	False	False
15	Tuğba	Küçüküstün	+905367896541	tugbakuc	e10adc3949ba5	tugbakucukust	Dolgun Sokağı	1988-04-23	2020-05-12	True	True	False	False	False
16	Muhtar	Bayram	+905367896542	muhtarb	e10adc3949ba5	muhtarbayram	Muhtar Sokağı	1985-01-21	2020-04-22	False	True	False	False	False
17	Arda	Dolu	+905367896532	ardadolu	e10adc3949ba5	ardadolu@gma	Turuncu Sokağı	1998-02-03	2019-05-02	True	False	True	False	False
18	Selim	Atar	+905388896541	selimatar	e10adc3949ba5	selimatar@gma	Orta Sokağı No:	1995-12-26	2020-01-30	False	True	False	False	False
19	Elif	Sever	+905367896333	elifsever	e10adc3949ba5	elifsever@gmai	Sakinler Sokağı	1999-07-21	2020-08-08	False	True	False	False	False
20	Selin	Katar	+905367897632	selinkatar	e10adc3949ba5	selinkatar@gm	Beyaz Sokağı N	1995-04-22	2018-11-11	False	False	True	False	False
21	Ceren	Tok	+905367896547	cerentok	e10adc3949ba5	cerentok@gma	Dolu Sokağı No	1997-08-27	2018-05-18	False	True	False	False	False
22	Enes	Yılmaz	+908513481289	enesyilmaz	e10adc3949ba5	enesyilmaz@g	Kaldırım Sokağı	1993-08-11	2016-02-11	False	False	False	False	True
23	Nazlı	Kaya	+908518232248	nazlikaya	e10adc3949ba5	nazlikaya@gma	Aslanağzı Soka	1996-01-23	2018-01-15	False	True	False	False	False
24	Songül	Aslan	+908518232248	songülkaya	e10adc3949ba5	songulaslan@g	Melen Sokağı N	1991-11-12	2019-11-12	False	False	True	False	False
25	Kemal	Demir	+908513798434	kemalde	e10adc3949ba5	kemaldemir@g	Denizhan Soka	1991-12-30	2017-03-12	False	False	False	True	False
33	Anıl	Şenay	+55555555	anilsenay	be56e057f20f883	anil@anil.com	Zonguldak	1998-04-27	2020-12-12	True	True	True	False	False
34	Veli	Kılıç	+905632114566	velikilic34	e10adc3949ba5	velikilic@gmail	Yokuş Sokak, K	1990-02-02	2020-01-12	False	True	False	False	False
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

(Figure 3.19.1 Before the update operation (Selected ID=34))



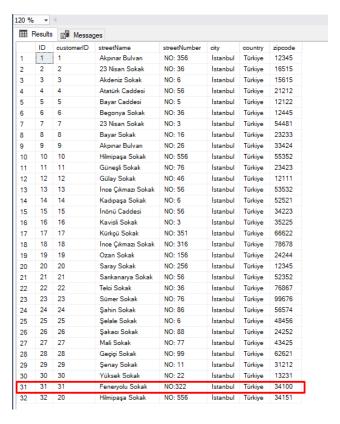
(Figure 3.19.3 After the update operation)

20-) sp_deleteAddress

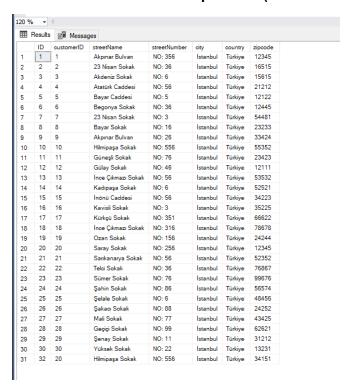
Definition: It deletes the customer's address based on id variable.

```
USE [TechnicalServiceDatabase]
 G0
 SET ANSI_NULLS ON
 GO
 SET QUOTED_IDENTIFIER ON
-- Author: <Author,,Name>
 -- Create date: <Create Date,,>
 -- Description: <Description,,>
 ☐ ALTER PROCEDURE [dbo].[sp_deleteAddress]
    -- Add the parameters for the stored procedure here
   @id int
 AS
⊟ BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
   SET NOCOUNT ON;
    -- Insert statements for procedure here
    DELETE FROM dbo.[ADDRESS] WHERE ID=@id
 END
```

(Figure 3.20 Query of the stored procedure)



(Figure 3.20.1 Before the delete operation (Selected ID=31))



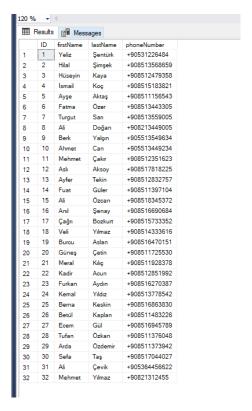
(Figure 3.20.2 After the delete operation)

21-) sp_deleteCustomer

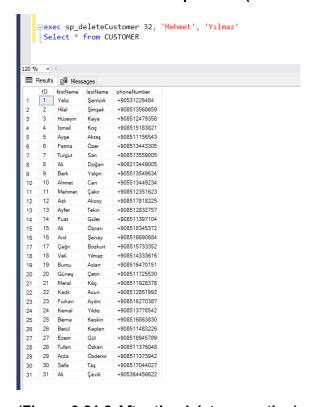
Definition: It deletes the customer's record based on customerID, firstName and lastName variable.

```
USE [TechnicalServiceDatabase]
 SET ANSI_NULLS ON
 SET QUOTED_IDENTIFIER ON
=--=======================
 -- Author: <Author,,Name>
 -- Create date: <Create Date,,>
 -- Description: <Description,,>
 □ ALTER PROCEDURE [dbo].[sp deleteCustomer]
    -- Add the parameters for the stored procedure here
    @id int,
    @firstName nvarchar(50),
    @lastName nvarchar(50)
 AS
⊟BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;
    -- Insert statements for procedure here
    DELETE FROM dbo.[ADDRESS] WHERE customerID=@id
    DELETE FROM dbo.[CUSTOMER] WHERE ID=@id and firstName=@firstName and lastName=@lastName
 END
```

(Figure 3.21 Query of the stored procedure)



(Figure 3.21.1 Before the delete operation (Selected ID=32))



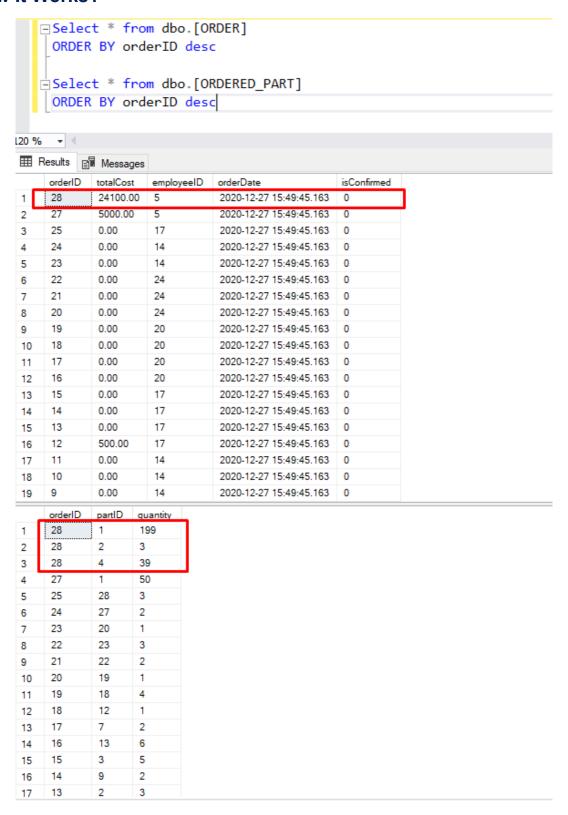
(Figure 3.21.2 After the delete operation)

22-) sp_deleteWrongOrder

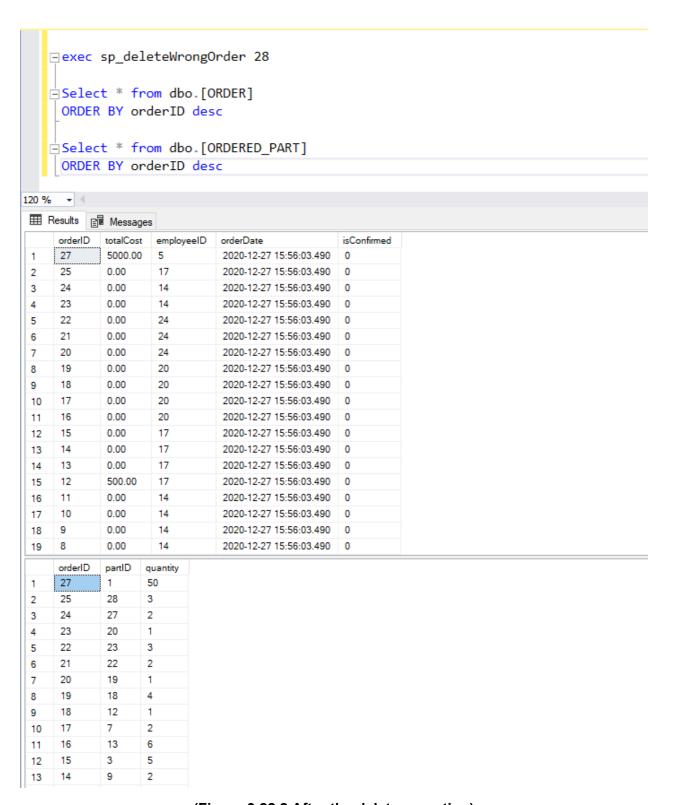
Definition: It deletes the all-related order information with given orderID.

```
USE [TechnicalServiceDatabase]
                                                       Script Date: 27/12/2020 15:52:45 ******/
 /***** Object: StoredProcedure [dbo].[sp_deleteWrongOrder]
 SET ANSI_NULLS ON
 GO
 SET QUOTED IDENTIFIER ON
 GO
-- Author: <Author,,Name>
 -- Create date: <Create Date,,>
 -- Description: <Description,,>
 -- -----
□ALTER PROCEDURE [dbo].[sp_deleteWrongOrder]
     -- Add the parameters for the stored procedure here
    @orderID int
 AS
BEGIN
     -- SET NOCOUNT ON added to prevent extra result sets from
     -- interfering with SELECT statements.
    SET NOCOUNT ON;
     -- Insert statements for procedure here
    DELETE FROM dbo.[ORDERED_PART] WHERE orderID=@orderID
    DELETE FROM dbo.[ORDER] WHERE orderID=@orderID
 END
```

(Figure 3.22 Query of the stored procedure)



(Figure 3.22.1 Before the update operation (Selected ID=28))



(Figure 3.22.2 After the delete operation)