HELP DESK SYSTEM

DATABASE PROJECT REPORT

Ezgi GÖKDEMİR, Melike KINIŞ, Murat SÜNGÜ, Zeynep TOPÇU

JANUARY / 2021

CONTENTS

LEGEND

PK	→	Primary key
FK	→	Foreign key
PKR	→	Primary key relation
FKR	→	Foreign key relation
N	→	Nullable
@>	→	Output
>@	→	Input

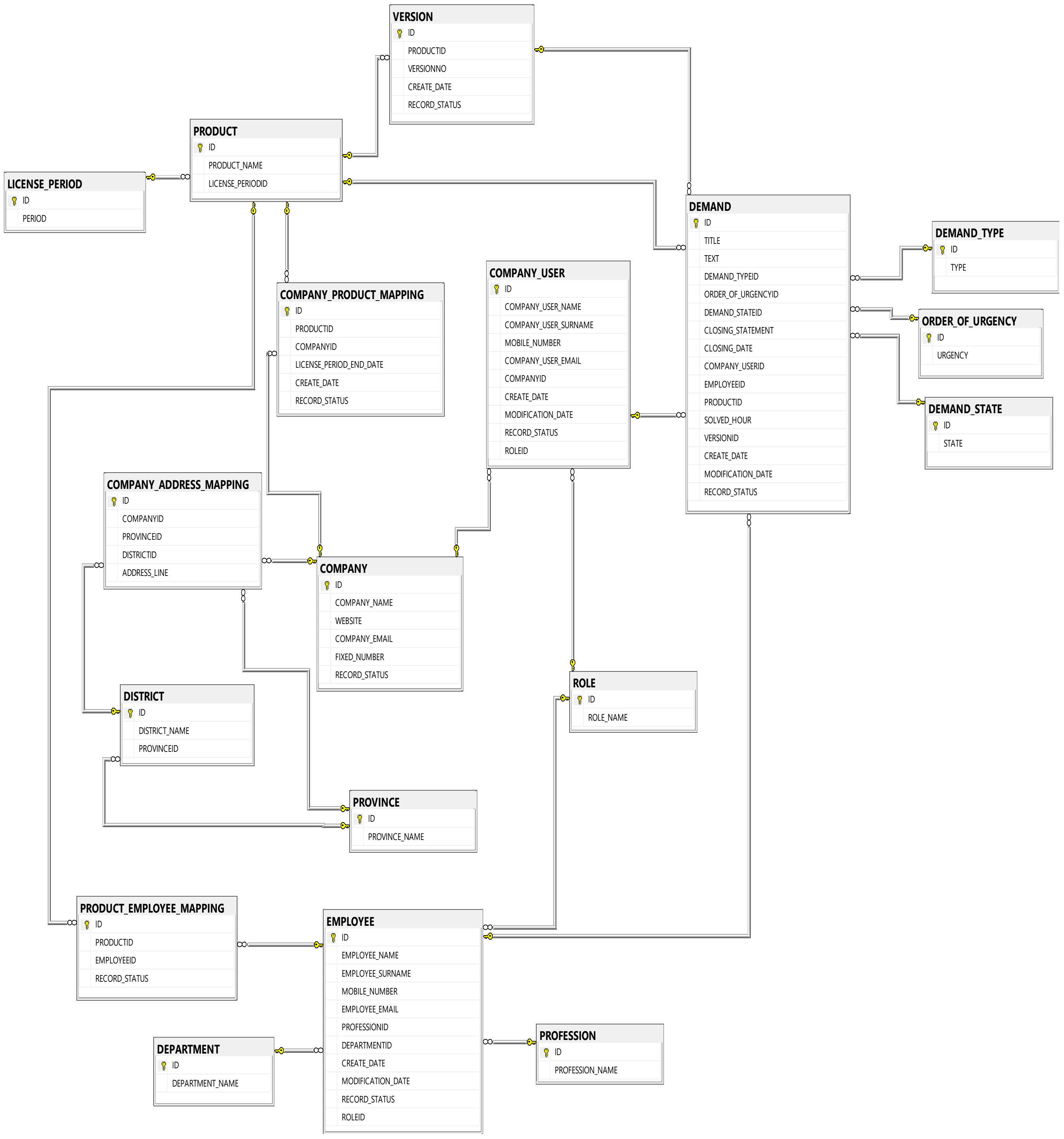
1.HELP DESK SYSTEM

Help desk is a process that aims to provide information and support to users who need technical support for a company's products or services. Help desk systems are needed to manage the process effectively and efficiently.

By using help desk systems which have a very important place in IT business process management, all employees and end users of the enterprise are gathered on the same platform and fast and practical solutions are offered for the demands and problems of the customers. The performance for customer support can be monitored through these systems. The data obtained are guiding in determining strategies in the long or short term.

In this project, a help desk system was designed for a company operating in the software industry. The queries and reporting required for database design and operations are included.

2.HELP DESK DATABASE DIAGRAM



3.TABLES

3.1.Table : EMPLOYEE

The table keeps the information of the software company employees using the system.

Columns

	NAME	DATA TYPE	N	DESCRIPTION / ATTRIBUTES
PK	ID	İnt		Primary key for EMPLOYEE records Identity / Auto increment column
	EMPLOYEE_NAME	nvarchar(50)		Name of employee
	EMPLOYEE_SURNAME	nvarchar(50)		Surname of employee
	MOBILE_NUMBER	nvarchar(15)	**N**	Phone number of employee
	EMPLOYEE_EMAIL	nvarchar(50)	**N**	E-mail address of employee
FK	PROFESSIONID	İnt		Unique identification number for profession Foreign key to PROFESSION table
FK	DEPARTMENTID	İnt		Unique identification number for deparment Foreign key to DEPARTMENT table
	CREATE_DATE	datetime		Create date of employee record
	MODIFICATION_DATE	datetime		Modification date of employee record
	RECORD_STATUS	Bit		Status for employee record / default 1
FK	ROLEID	İnt		Unique identification number for role Foreign key to ROLE table

Links to

	TABLE	JOIN	TITLE / NAME / DESCRIPTION
PKR	PROFESSION	**EMPLOYEE**.PROFESSIONID =PROFESSION.ID	FK_EMPLOYEE_PROFESSION_PROFESSIONID foreign key constraint referencing PROFESSION.ID
PKR	DEPARTMENT	**EMPLOYEE**.DEPARTMENTID = DEPARTMENT.ID	FK_EMPLOYEE_DEPARTMENT_DEPARTMENTID foreign key constraint referencing DEPARTMENT.ID
PKR	ROLE	**EMPLOYEE**.ROLEID= ROLE.ID	FK_EMPLOYEE_ROLE_ROLEID foreign key constraint referencing ROLE.ID

Linked from

	TABLE	JOIN	TITLE / NAME / DESCRIPTION
FKR	DEMAND	**EMPLOYEE**.ID = DEMAND. EMPLOYEEID	FK_DEMAND_EMPLOYEE_EMPLOYEEID foreign key constraint referencing EMPLOYEE.ID
FKR	PRODUCT_EMPLOYEE_MAPPING	**EMPLOYEE**. ID = PRODUCT_EMPLOYEE_MAPPING.EMPLOYEEID	FK_PRODUCTEMPLOYEEMAPPING_EM PLOYEE_EMPLOYEEID foreign key constraint referencing EMPLOYEE.ID

Unique keys

COLUMNS		NAME / DESCRIPTION
PK	ID	PK_EMPLOYEE_ID Primary key (clustred) constraint

Uses

NAME
EMPLOYEE
PROFESSION
DEPARMENT
ROLE

Used by

NAME
EMPLOYEE
DEMAND
PRODUCT_EMPLOYEE_MAPPING

3.2.Table : PROFESSION

The table keeps job information for software company employees.

Columns

NAME		DATA TYPE	N	DESCRIPTION / ATTRIBUTES
PK	ID	İnt		Primary key for PROFESSION records Identity / Auto increment column
	PROFESSION_NAME	nvarchar(50)		Name of profession

Linked from

TABLE		JOIN	TITLE / NAME / DESCRIPTION
FKR	EMPLOYEE	**PROFESSION**.ID = EMPLOYEE.PROFESSIONID	FK_EMPLOYEE_PROFESSION_PROFESSIONID foreign key constraint referencing PROFESSION.ID

Unique keys

COLUMNS		NAME / DESCRIPTION
PK	ID	PK_PROFESSION_ID Primary key (clustred) constraint

Used by

NAME
PROFESSION
EMPLOYEE

3.3.Table : DEPARTMENT

The table keeps department information of software company employees

Columns

	NAME	DATA TYPE	N	DESCRIPTION / ATTRIBUTES
PK	ID	İnt		Primary key for DEPARTMENT records Identity / Auto increment column
	DEPARTMENT_NAME	nvarchar(50)		Name of department

Linked from

	TABLE	JOIN	TITLE / NAME / DESCRIPTION
FKR	EMPLOYEE	**DEPARTMENT**.ID = EMPLOYEE.DEPARTMENTID	FK_EMPLOYEE_DEPARTMENT_DEPARMENTID foreign key constraint referencing DEPARTMENT.ID

Unique keys

	COLUMNS	NAME / DESCRIPTION
PK	ID	PK_DEPARTMENT_ID Primary key (clustred) constraint

Used By

NAME
DEPARTMENT
EMPLOYEE

3.4.Table : ROLE

The table keeps information about the roles of the actors in the system.

Columns

	NAME	DATA TYPE	N	DESCRIPTION / ATTRIBUTES
PK	ID	İnt		Primary key for ROLE records Identity / Auto increment column
	ROLE_NAME	nvarchar(30)		Name of role

Linked from

TABLE		JOIN	TITLE / NAME / DESCRIPTION
FKR	COMPANY_USER	**ROLE**.ID = COMPANY_USER.ROLEID	FK_COMPANYUSER_ROLE_ROLEID foreign key constraint referencing ROLE.ID
FKR	EMPLOYEE	**ROLE**.ID = EMPLOYEE.ROLEID	FK_EMPLOYEE_ROLE_ROLEID foreign key constraint referencing ROLE.ID

Unique keys

COLUMNS		NAME / DESCRIPTION
PK	ID	PK_ROLE_ID Primary key (clustred) constraint

Used by

NAME
ROLE
COMPANY_USER_ROLE_MAPPING
EMPLOYEE_ROLE_MAPPING

3.5.Table : PRODUCT_EMPLOYEE_MAPPING

The table keeps the information that which employee can be assigned to the demand about which product.

Columns

NAME		DATA TYPE	N	DESCRIPTION / ATTRIBUTES
PK	ID	İnt		Primary key for PRODUCT_EMPLOYEE_MAPPING records Identity / Auto increment column
FK	PRODUCTID	İnt		Unique identification number for product Foreign key to PRODUCT table
FK	EMPLOYEEID	İnt		Unique identification number for employee Foreign key to EMPLOYEE table
	RECORD_STATUS	Bit		Status for product employee mapping record / default 1

Links to

TABLE		JOIN	TITLE / NAME / DESCRIPTION
PKR	PRODUCT	**PRODUCT_EMPLOYEE_MAPPING**.PRODUCTID= PRODUCT.ID	FK_PRODUCTEMPLOYEEMAPPING_PRODUCT_PRODUCTID foreign key constraint referencing PRODUCT.ID
PKR	EMPLOYEE	**PRODUCT_EMPLOYEE_MAPPING**.EMPLOYEEID= EMPLOYEE.ID	FK_PRODUCTEMPLOYEEMAPPING_EMPLOYEE_EMPLOYEE ID foreign key constraint referencing EMPLOYEE.ID

Unique keys

COLUMNS		NAME / DESCRIPTION
PK	ID	PK_PRODUCT_EMPLOYEE_MAPPING _ID Primary key (clustred) constraint

Uses

NAME
COMPANY_PRODUCT_MAPPING
PRODUCT
EMPLOYEE

3.6.Table : COMPANY_USER

The table keeps information about the people who use the product of software company.

Columns

NAME		DATA TYPE	N	DESCRIPTION / ATTRIBUTES
PK	ID	İnt		Primary key for COMPANY_USER records Identity / Auto increment column
	COMPANY_USER_NAME	nvarchar(50)		Name of company user
	COMPANY_USER_SURNAME	nvarchar(50)		Surname of company user
	MOBILE_NUMBER	nvarchar(15)		Phone number of company user
	COMPANY_USER_EMAIL	nvarchar(50)		E-mail address of company user
FK	COMPANYID	İnt		Unique identification number for company. Foreign key to COMPANY table
	CREATE_DATE	Datetime		Create date of company user record
	MODIFICATION_DATE	Datetime		Modification date of company user record
	RECORD_STATUS	Bit		Status for company user record / default 1
FK	ROLEID	İnt		Unique identification number for role Foreign key to ROLE table

Links to

TABLE		JOIN	TITLE / NAME / DESCRIPTION
PKR	COMPANY	**COMPANY_USER**.COMPANYID= COMPANY.ID	FK_COMPANYUSER_COMPANY_COMPANYID foreign key constraint referencing COMPANY.ID
PKR	ROLE	COMPANY_USER.ROLEID= ROLE.ID	FK_COMPANYUSER_ROLE_ROLEID foreign key constraint referencing ROLE.ID

Linked from

TABLE		JOIN	TITLE / NAME / DESCRIPTION
FKR	DEMAND	**COMPANY_USER**.ID = DEMAND. COMPANY_USERID	FK_DEMAND_COMPANY_USER_COMPANY_USERID foreign key constraint referencing COMPANY_USER.ID

Unique keys

COLUMNS		NAME / DESCRIPTION
PK	ID	PK_COMPANY_USER_ID Primary key (clustred) constraint

Uses

NAME
COMPANY_USER
COMPANY
ROLE

Used by

NAME
COMPANY_USER
DEMAND

3.7.Table : COMPANY

The table keeps information about companies that are customers of the software company.

Columns

NAME		DATA TYPE	N	DESCRIPTION / ATTRIBUTES
PK	ID	İnt		Primary key for COMPANY records Identity / Auto increment column
	COMPANY_NAME	nvarchar(255)		Name of company
	WEBSITE	nvarchar(100)		Website of company
	COMPANY_EMAIL	nvarchar(100)		E-mail address of company
	FIXED_NUMBER	nvarchar(15)	**N**	Phone number of company
	RECORD_STATUS	Bit		Status for company record /default 1

Linked from

TABLE		JOIN	TITLE / NAME / DESCRIPTION
FKR	COMPANY_USER	**COMPANY.ID** = COMPANY_USER.COMPANYID	FK_COMPANYUSER_COMPANY_COMPANYID foreign key constraint referencing COMPANY.ID
FKR	COMPANY_ADDRESS_MAPPING	**COMPANY.ID** = COMPANY_ADDRESS_MAPPING. COMPANYID	FK_ADDRESS_COMPANY_COMPANYID foreign key constraint referencing COMPANY.ID
FKR	COMPANY_PRODUCT_MAPPING	**COMPANY.ID**= COMPANY_PRODUCT_MAPPING .COMPANYID	FK_COMPANYPRODUCTMAPPING_COMPANY_COMPAN YID foreign key constraint referencing COMPANY.ID

Unique keys

COLUMNS		NAME / DESCRIPTION
PK	ID	PK_COMPANY_ID Primary key (clustred) constraint

Used by

NAME
COMPANY
COMPANY_USER
COMPANY_ADDRESS_MAPPING
COMPANY_PRODUCT_MAPPING

3.8.Table : COMPANY_ADDRESS_MAPPING

A company can have more than one address. The table keeps the address information of the companies registered in the system.

Columns

NAME		DATA TYPE	N	DESCRIPTION / ATTRIBUTES
PK	ID	İnt		Primary key for COMPANY_ADDRESS_MAPPING records Identity / Auto increment column
FK	COMPANYID	İnt		Unique identification number for company Foreign key to COMPANY table
FK	PROVINCEID	İnt		Unique identification number for province Foreign key to PROVINCE table
FK	DISTRICTID	İnt		Unique identification number for district Foreign key to DISTRICT table
	ADDRESS_LINE	nvarchar(250)		Full address of the company

Links to

TABLE		JOIN	TITLE / NAME / DESCRIPTION
PKR	COMPANY	**COMPANY_ADDRESS_MAPPING**.COMPANYID =COMPANY.ID	FK_ADDRESS_COMPANY_COMPANYID foreign key constraint referencing COMPANY.ID
PKR	PROVINCE	**COMPANY_ADDRESS_MAPPING**.PROVINCEID = PROVINCE.ID	FK_ADDRESS_PROVINCE_PROVINCEID foreign key constraint referencing PROVINCE.ID
PKR	DISTRICT	**COMPANY_ADDRESS_MAPPING.**DISTRICTID = DISTRICT.ID	FK_ADDRESS_DISTRICT_DISTRICTID foreign key constraint referencing DISTRICT.ID

Unique keys

COLUMNS		NAME / DESCRIPTION
PK	ID	PK_COMPANY_ADDRESS_MAPPING_ID Primary key (clustred) constraint

Uses

NAME
COMPANY_ADDRESS_MAPPING
COMPANY
PROVINCE
DISTRICT

3.9.Table : PROVINCE

The table keeps the city information of the address.

Columns

NAME		DATA TYPE	N	DESCRIPTION / ATTRIBUTES
PK	ID	İnt		Primary key for PROVINCE records Identity / Auto increment column
	PROVINCE_NAME	nvarchar(30)		Name of province

Linked from

TABLE		JOIN	TITLE / NAME / DESCRIPTION
FKR	COMPANY_ADDRESS_MAPPING	**PROVINCE**.ID = COMPANY_ADDRESS_MAPPING.PROVINCEID	FK_ADDRESS_PROVINCE_PROVINCEID foreign key constraint referencing PROVINCE.ID
FKR	DISTRICT	**PROVINCE**.ID = DISTRICT.PROVINCEID	FK_DISTRICT_PROVINCE_PROVINCEID foreign key constraint referencing PROVINCE.ID

Unique keys

COLUMNS		NAME / DESCRIPTION
PK	ID	PK_PROVINCE_ID Primary key (clustred) constraint

Used by

NAME
PROVINCE
COMPANY_ADDRESS_MAPPING
DISTRICT

3.10.Table : DISTRICT

The table keeps the district information of the address.

Columns

NAME		DATA TYPE	N	DESCRIPTION / ATTRIBUTES
PK	ID	İnt		Primary key for DISTRICT records Identity / Auto increment column
	DISTRICT_NAME	nvarchar(50)		Name of district
FK	PROVINCEID	İnt		Unique identification number for province. Foreign key to PROVINCE table

Links to

TABLE		JOIN	TITLE / NAME / DESCRIPTION
PKR	PROVINCE	**DISTRICT**.PROVINCEID = PROVINCE.ID	FK_DISTRICT_PROVINCE_PROVINCEID foreign key constraint referencing PROVINCE.ID

Linked from

TABLE		JOIN	TITLE / NAME / DESCRIPTION
FKR	COMPANY_ADDRESS_MAPPING	**DISTRICT**.ID = COMPANY_ADDRESS_MAPPING.DISTRICTID	FK_ADDRESS_DISTRICT_DISTRICTID foreign key constraint referencing DISTRICTID

Unique keys

COLUMNS		NAME / DESCRIPTION
PK	ID	PK_DISTRICT_ID Primary key (clustred) constraint

Uses

NAME
DISTRICT
PROVINCE

Used by

NAME
DISTRICT
COMPANY_ADDRESS_MAPPING

3.11.Table : COMPANY_PRODUCT_MAPPING

The table keeps the information that which company buys which product.

Columns

	NAME	DATA TYPE	N	DESCRIPTION / ATTRIBUTES
PK	ID	İnt		Primary key for COMPANY_PRODUCT_MAPPING records Identity / Auto increment column
FK	PRODUCTID	İnt		Unique identification number for product Foreign key to PRODUCT table
FK	COMPANYID	İnt		Unique identification number for company Foreign key to COMPANY table
	LICENSE_PERIOD_END_DATE	Datetime		License ending date of product
	CREATE_DATE	Datetime		Create date of company product mapping record
	RECORD_STATUS	Bit		Status for company product mapping record / default 1

Links to

	TABLE	JOIN	TITLE / NAME / DESCRIPTION
PKR	PRODUCT	**COMPANY_PRODUCT_MAPPING**.PRODUCTID = PRODUCT.ID	FK_COMPANYPRODUCTMAPPING_PRODUCT_PRODUCTID foreign key constraint referencing PRODUCT.ID
PKR	COMPANY	**COMPANY_PRODUCT_MAPPING**.COMPANYID = COMPANY.ID	FK_COMPANYPRODUCTMAPPING_COMPANY_COMPANYID foreign key constraint referencing COMPANY.ID

Unique keys

	COLUMNS	NAME / DESCRIPTION
PK	ID	PK_COMPANY_PRODUCT_MAPPING_ID Primary key (clustred) constraint

Uses

NAME
COMPANY_PRODUCT_MAPPING
PRODUCT
COMPANY

3.12.Table : PRODUCT

The table keeps information about the products of software company

Columns

	NAME	DATA TYPE	N	DESCRIPTION / ATTRIBUTES
PK	ID	İnt		Primary key for PRODUCT records Identity / Auto increment column
	PRODUCT_NAME	nvarchar(100)		Name of product
FK	LICENCE_PERIODID	İnt		Unique identification number for license period Foreign key to LICENSE_PERIOD table

Links to

	TABLE	JOIN	TITLE / NAME / DESCRIPTION
PKR	LICENCE_PERIOD	**PRODUCT**.LICENCE_PERIODID = LICENSE_PERIOD.ID	FK_PRODUCT_LICENSE_PERIOD_LICENSE_PERIODID foreign key constraint referencing LICENSE_PERIOD.ID

Linked from

	TABLE	JOIN	TITLE / NAME / DESCRIPTION
FKR	PRODUCT_EMPLOYEE_MAPPING	**PRODUCT**.ID = PRODUCT_EMPLOYEE_MAPPING.PRODUCTID	FK_PRODUCTROLEMAPPING_PRODUCT_PRODUCTID foreign key constraint referencing PRODUCT.ID
FKR	COMPANY_PRODUCT_MAPPING	**PRODUCT**.ID = COMPANY_PRODUCT_MAPPING.PRODUCTID	FK_COMPANYPRODUCTMAPPING_PRODUCT_PRODUCTID foreign key constraint referencing PRODUCT.ID
FKR	DEMAND	**PRODUCT**.ID = DEMAND.PRODUCTID	FK_DEMAND_PRODUCT_PRODUCTID foreign key constraint referencing PRODUCT.ID
FKR	VERSION	**PRODUCT**.ID = VERSION.PRODUCTID	FK_VERSION_PRODUCT_PRODUCTID foreign key constraint referencing PRODUCT.ID

Unique keys

	COLUMNS	NAME / DESCRIPTION
PK	ID	PK_PRODUCT_ID Primary key (clustred) constraint

Uses

NAME
PRODUCT
LICENCE_PERIOD

Used by

NAME
PRODUCT
PRODUCT_EMPLOYEE_MAPPING
COMPANY_PRODUCT_MAPPING
DEMAND
VERSION

3.13.Table : LICENSE_PERIOD

The table keeps information about license period of products

Columns

NAME		DATA TYPE	N	DESCRIPTION / ATTRIBUTES
PK	ID	İnt		Primary key for LICENSE_PERIOD records Identity / Auto increment column
	PERIOD	nvarchar(50)		Period of license

Linked from

TABLE		JOIN	TITLE / NAME / DESCRIPTION
FKR	PRODUCT	**LICENSE_PERIOD**.ID = PRODUCT.LICENSE_PERIODID	FK_PRODUCT_LICENSE_PERIOD_LICENSE_PERIODID foreign key constraint referencing LICENSE_PERIOD.ID

Unique keys

COLUMNS		NAME / DESCRIPTION
PK	ID	PK_LICENSE_PERIOD_ID Primary key (clustred) constraint

Used by

NAME
LICENSE_PERIOD
PRODUCT

3.14.Table : VERSION

The table keeps the version information of the software products.

Columns

	NAME	DATA TYPE	N	DESCRIPTION / ATTRIBUTES
PK	ID	İnt		Primary key for VERSION records Identity / Auto increment column
FK	PRODUCTID	İnt		Unique identification number for product Foreign key to PRODUCT table
	VERSIONNO	nvarchar(20)		Number of version
	CREATE_DATE	Datetime		Create date of version record
	RECORD_STATUS	Bit		Status for version record / default 1

Links to

	TABLE	JOIN	TITLE / NAME / DESCRIPTION
PKR	PRODUCT	**VERSION**.PRODUCTID= PRODUCT.ID	FK_VERSION_PRODUCT_PRODUCTID foreign key constraint referencing PRODUCT.ID

Linked from

	TABLE	JOIN	TITLE / NAME / DESCRIPTION
FKR	DEMAND	**VERSION**.ID = DEMAND. VERSIONID	FK_DEMAND_VERSION_VERSIONID foreign key constraint referencing VERSION.ID

Unique keys

	COLUMNS	NAME / DESCRIPTION
PK	ID	PK_VERSION_ID Primary key (clustred) constraint

Uses

NAME
VERSION
PRODUCT

Used by

NAME
VERSION
DEMAND

3.15.Table : DEMAND

The table keeps information about demand created.

Columns

	NAME	DATA TYPE	N	DESCRIPTION / ATTRIBUTES
PK	ID	int		Primary key for DEMAND records Identity / Auto increment column
	TITLE	nvarchar(max)		Subject of demand
	TEXT	nvarchar(max)		Detailed explanation of demand
FK	DEMAND_TYPEID	int		Unique identification number for demand type Foreign key to DEMAND_TYPE table
FK	ORDER_OF_URGENCYID	int		Unique identification number for order of urgency Foreign key to ORDER_OF_URGENCY table
FK	DEMAND_STATEID	int		Unique identification number for order of demand state Foreign key to DEMAND_STATE table
	CLOSING_STATEMENT	nvarchar(max)		Closing statement of demand
	CLOSING_DATE			Closing date of demand
FK	COMPANY_USERID	int		Unique identification number for company user Foreign key to COMPANY_USER table
FK	EMPLOYEEID	int		Unique identification number for employee Foreign key to EMPLOYEE table
FK	PRODUCTID	int		Unique identification number for product Foreign key to PRODUCT table
	SOLVED_HOUR	int		Closing time of demand /as hour
FK	VERSIONID	int		Unique identification number for version Foreign key to VERSION table
	CREATE_DATE	datetime		Create date of demand record
	MODIFICATION_DATE	datetime		Modification date of demand record
	RECORD_STATUS	bit		Status for demand record / default 1

Links to

	TABLE	JOIN	TITLE / NAME / DESCRIPTION
PKR	DEMAND_TYPE	**DEMAND**.DEMAND_TYPEID = DEMAND_TYPE.ID	FK_DEMAND_DEMAND_TYPE_DEMAND_TYPEID foreign key constraint referencing DEMAND_TYPE.ID
PKR	ORDER_OF_URGENCY	**DEMAND**.ORDER_OF_URGENCYID = ORDER_OF_URGENCY.ID	FK_DEMAND_ORDER_OF_URGENCY_ORDER_OF_URGEN CYID foreign key constraint referencing ORDER_OF_URGENCY.ID
PKR	DEMAND_STATE	**DEMAND**.DEMAND_STATEID = DEMAND_STATE.ID	FK_DEMAND_DEMAND_STATE_DEMAND_STATEID foreign key constraint referencing DEMAND_STATE.ID
PKR	COMPANY_USER	**DEMAND**.COMPANY_USERID = COMPANY_USER.ID	FK_DEMAND_COMPANY_USER_COMPANY_USERID foreign key constraint referencing COMPANY_USER.ID
PKR	EMPLOYEE	**DEMAND**.EMPLOYEEID = EMPLOYEE.ID	FK_DEMAND_EMPLOYEE_EMPLOYEEID foreign key constraint referencing EMPLOYEE.ID
PKR	PRODUCT	**DEMAND**.PRODUCTID = PRODUCT.ID	FK_DEMAND_PRODUCT_PRODUCTID foreign key constraint referencing PRODUCT.ID
PKR	VERSION	**DEMAND**.VERSIONID= VERSION.ID	FK_DEMAND_VERSION_VERSIONID foreign key constraint referencing VERSION.ID

Unique keys

COLUMNS		NAME / DESCRIPTION
PK	ID	PK_DEMAND_ID Primary key (clustred) constraint

Uses

NAME
DEMAND
DEMAND_TYPE
ORDER_OF_URGENCY
DEMAND_STATE
COMPANY_USER
EMPLOYEE
PRODUCT
VERSION

3.16.Table : DEMAND _TYPE

The table keeps information about the type of demand generated by the company user.

Columns

NAME		DATA TYPE	N	DESCRIPTION / ATTRIBUTES
PK	ID	İnt		Primary key for DEMAND_TYPE records Identity / Auto increment column
	TYPE	nvarchar(20)		Type of demand

Linked from

TABLE		JOIN	TITLE / NAME / DESCRIPTION
FKR	DEMAND	**DEMAND_TYPE**.ID = DEMAND.DEMAND_TYPEID	FK_DEMAND_DEMAND_TYPE_DEMAND_TYPEID foreign key constraint referencing DEMAND_TYPE.ID

Unique keys

COLUMNS		NAME / DESCRIPTION
PK	ID	PK_DEMAND_TYPE_ID Primary key (clustred) constraint

Used by

NAME
DEMAND_TYPE
DEMAND

3.17.Table : ORDER_OF_URGENCY

The table keeps information on the urgency of the demand.

Columns

	NAME	DATA TYPE	N	DESCRIPTION / ATTRIBUTES
PK	ID	İnt		Primary key for ORDER_OF_URGENCY records Identity / Auto increment column
	URGENCY	nvarchar(20)		Urgency of the demand

Linked from

	TABLE	JOIN	TITLE / NAME / DESCRIPTION
FKR	DEMAND	**ORDER_OF_URGENCY**.ID = DEMAND.ORDER_OF_URGENCYID	FK_DEMAND_ORDER_OF_URGENCY_ORDER_OF_URGENCYID foreign key constraint referencing ORDER_OF_URGENCY.ID

Unique keys

	COLUMNS	NAME / DESCRIPTION
PK	ID	PK_ORDER_OF_URGENCY_ID Primary key (clustred) constraint

Used by

NAME
ORDER_OF_URGENCY
DEMAND

3.18.Table : DEMAND_STATE

The table keeps information about the status of the demand.

Columns

	NAME	DATA TYPE	N	DESCRIPTION / ATTRIBUTES
PK	ID	İnt		Primary key for DEMAND_STATE records Identity / Auto increment column
	STATE	nvarchar(15)		State of demand

Linked from

	TABLE	JOIN	TITLE / NAME / DESCRIPTION
FKR	DEMAND	**DEMAND_STATE**.ID = DEMAND.DEMAND_STATEID	FK_DEMAND_DEMAND_STATE_DEMAND_STATEID foreign key constraint referencing DEMAND_STATE.ID

Unique keys

COLUMNS		NAME / DESCRIPTION
PK	ID	PK_DEMAND_STATE_ID Primary key (clustred) constraint

Used by

NAME
DEMAND_STATE
DEMAND

4.FUNCTIONS

4.1.Function: f_GET_PRODUCTS_FOR_EMPLOYEE_ID

Table-valued function returning the product name for the given employee id.

Input/Output

	NAME	DATA TYPE	DESCRIPTION / ATTRIBUTES
@>	RETURNS	TABLE TYPE	
>@	EMPLOYEEID	INT	Input parameter for the table-valued function f_GET_PRODUCTS_BY_EMPLOYEE_ID. Enter a valid EMPLOYEEID from the EMPLOYEE table.

4.2.Function: f_GET_DEMANDS_FOR_COMPANY_USERID

Table-valued function returning the all columns of demand table for the given company user id.

Input/Output

	NAME	DATA TYPE	DESCRIPTION / ATTRIBUTES
@>	RETURNS	TABLE TYPE	
>@	COMPANY_USERID	INT	Input parameter for the table-valued function f_GET_DEMANDS_BY_COMPANY_USERID. Enter a valid COMPANY_USERID from the COMPANY_USER table.

4.3.Function: f_GET_ALL_DEMAND_URGENCY_FOR_EMPLOYEEID

Table-valued function returning the urgency types and total number of demands on the basis of urgency type for the given employee id.

Input/Output

	NAME	DATA TYPE	DESCRIPTION / ATTRIBUTES
@>	RETURNS	TABLE TYPE	
>@	EMPLOYEEID	INT	Input parameter for the table-valued function f_GET_ALL_DEMAND_URGENCY_BY_EMPLOYEEID. Enter a valid EMPLOYEEID from the EMPLOYEE table.

4.4.Function: f_GET_ALL_DEMAND_TYPES_FOR_EMPLOYEEID

Table-valued function returning the demand types and total number of demands on the basis of demand type for the given employee id.

Input/Output

	NAME	DATA TYPE	DESCRIPTION / ATTRIBUTES
@>	RETURNS	TABLE TYPE	
>@	EMPLOYEEID	INT	Input parameter for the table-valued function f_GET_ALL_DEMAND_TYPES_BY_EMPLOYEEID. Enter a valid EMPLOYEEID from the EMPLOYEE table.

4.5.Function: f_GET_ALL_DEMAND_STATES_FOR_EMPLOYEEID

Table-valued function returning the demand states and total number of demands on the basis of demand state for the given employee id.

Input/Output

	NAME	DATA TYPE	DESCRIPTION / ATTRIBUTES
@>	RETURNS	TABLE TYPE	
>@	EMPLOYEEID	INT	Input parameter for the table-valued function f_GET_ALL_DEMAND_STATES_BY_EMPLOYEEID. Enter a valid EMPLOYEEID from the EMPLOYEE table.

4.6.Function: f_GET_ALL_DEMAND_FOR_COMPANY

Table-valued function returning the all columns of demand table for the given company id.

Input/Output

	NAME	DATA TYPE	DESCRIPTION / ATTRIBUTES
@>	RETURNS	TABLE TYPE	

>@	COMPANYID	INT	Input parameter for the table-valued function f_GET_ALL_DEMAND_FOR_COMPANY. Enter a valid COMPANYID from the COMPANY table.

4.7.Function: f_COUNT_OF_DEMAND_FOR_COMPANY_BY_STATE

Table-valued function returning the demand state and count of demand on the basis of demand state for the given company id.

Input/Output

	NAME	DATA TYPE	DESCRIPTION / ATTRIBUTES
@>	RETURNS	TABLE TYPE	
>@	COMPANYID	INT	Input parameter for the table-valued function f_COUNT_OF_DEMAND_FOR_COMPANY_BY_STATE. Enter a valid COMPANYID from the COMPANY table.

4.8.Function: f_GET_ALL_DEMAND_FOR_EMPLOYEEID

Table-valued function returning the demands for the given employee id.

Input/Output

	NAME	DATA TYPE	DESCRIPTION / ATTRIBUTES
@>	RETURNS	TABLE TYPE	
>@	EMPLOYEEID	INT	Input parameter for the table-valued function f_GET_ALL_DEMAND_BY_EMPLOYEEID. Enter a valid EMPLOYEEID from the EMPLOYEE table.

4.9.Function: f_GET_ALL_DEMAND_AND_STATES_FOR_EMPLOYEEID

Table-valued function returning the all demands and demand states for the given employee id.

Input/Output

	NAME	DATA TYPE	DESCRIPTION / ATTRIBUTES
@>	RETURNS	TABLE TYPE	
>@	EMPLOYEEID	INT	Input parameter for the table-valued function f_GET_ALL_DEMAND_AND_STATES_BY_EMPLOYEEID. Enter a valid EMPLOYEEID from the EMPLOYEE table.

4.10.Function: f_COUNT_OF_DEMAND_FOR_COMPANY_USERID_BY_STATE

Table-valued function returning the demand state and count of demand on the basis of demand state for the given company user id.

Input/Output

	NAME	DATA TYPE	DESCRIPTION / ATTRIBUTES
@>	RETURNS	TABLE TYPE	
>@	COMPANY_USERID	INT	Input parameter for the table-valued function f_COUNT_OF_DEMAND_FOR_COMPANY_USERID_BY_STATE. Enter a valid COMPANY_USERID from the COMPANY_USER table.

4.11.Function: f_GET_DEMAND_BY_COMPANY_MONTHLY

Table-valued function returning company names and count of demand for the given year and month

Input/Output

	NAME	DATA TYPE	DESCRIPTION / ATTRIBUTES
@>	RETURNS	TABLE TYPE	
>@	YEAR	INT	Input parameter for the table-valued function f_GET_DEMAND_BY_COMPANY_MONTHLY. Enter a valid year.
>@	MONTH	INT	Input parameter for the table-valued function f_GET_DEMAND_BY_COMPANY_MONTHLY. Enter a valid month.

4.12.Function: f_GET_DEMAND_BY_PRODUCT_MONTHLY

Table-valued function returning product names and count of demand for the given year and month

Input/Output

	NAME	DATA TYPE	DESCRIPTION / ATTRIBUTES
@>	RETURNS	TABLE TYPE	
>@	YEAR	INT	Input parameter for the table-valued function f_GET_DEMAND_BY_PRODUCT_MONTHLY. Enter a valid year.
>@	MONTH	INT	Input parameter for the table-valued function f_GET_DEMAND_BY_PRODUCT_MONTHLY. Enter a valid month.

4.13.Function: f_GET_DEMAND_BY_PRODUCT_ANNUAL

Table-valued function returning product names and count of demand for given years.

	NAME	DATA TYPE	DESCRIPTION / ATTRIBUTES
@>	RETURNS	TABLE TYPE	

>@	Start_Year	INT	Input parameter for the table-valued function f_GET_DEMAND_BY_PRODUCT_ANNUAL. Enter a valid year.
>@	End_Year	INT	Input parameter for the table-valued function f_GET_DEMAND_BY_PRODUCT_ANNUAL. Enter a valid year.

4.14.Function: f_GET_DEMAND_BY_COMPANY_ANNUAL

Table-valued function returning company names and count of demand for given years

Input/Output

NAME		DATA TYPE	DESCRIPTION / ATTRIBUTES
@>	RETURNS	TABLE TYPE	
>@	Start_Year	INT	Input parameter for the table-valued function f_GET_DEMAND_BY_COMPANY_ANNUAL. Enter a valid year.
>@	End_Year	INT	Input parameter for the table-valued function f_GET_DEMAND_BY_COMPANY_ANNUAL. Enter a valid year.

4.15.Function: f_GET_ALL_DEMAND_ANNUAL

Table-valued function returning all demands between selected years.

Input/Output

NAME		DATA TYPE	DESCRIPTION / ATTRIBUTES
@>	RETURNS	TABLE TYPE	
>@	Start_Year	INT	Input parameter for the table-valued function f_GET_ALL_DEMAND_ANNUAL. Enter a valid year.
>@	End_Year	INT	Input parameter for the table-valued function f_GET_ALL_DEMAND_ANNUAL. Enter a valid year.

4.16.Function: f_COUNT_OF_COMPANY_PRODUCT_FOR_COMPANYID

Scalar-valued function returning count of product for the given company id.

Input/Output

NAME		DATA TYPE	DESCRIPTION / ATTRIBUTES
@>	RETURNS	INT	

>@	COMPANYID	INT	Input parameter for the scalar-valued function f_COUNT_OF_COMPANY_PRODUCT_FOR_COMPANYID. Enter a valid COMPANYID from the COMPANY table.

4.17.Function: f_FORMAT_PHONE_NUMBER

Scalar-valued function returning the true format for the given telephone number.

Input/Output

	NAME	DATA TYPE	DESCRIPTION / ATTRIBUTES
@>	RETURNS	VARCHAR(13)	
>@	PHONENUMBE R	VARCHAR(11)	Input parameter for the scalar-valued function [dbo].[Format_Phone_Number].

5.STORED PROCEDURES

5.1.Stored Procedure: sp_CREATE_COMPANY

Stored procedure using INSERT query to adding new company and address to company.

Input/Output

	NAME	DATA TYPE	DESCRIPTION
>@	RoleId	Int	Input parameter for the stored procedure sp_CREATE_COMPANY. Enter a valid RoleId from the ROLE table.
>@	Company_Name	nvarchar(255)	Input parameter for the stored procedure sp_CREATE_COMPANY. Enter a Company_Name for company
>@	Website	nvarchar(100)	Input parameter for the stored procedure sp_CREATE_COMPANY. Enter a Website for company
>@	Company_Email	nvarchar(100)	Input parameter for the stored procedure sp_CREATE_COMPANY. Enter a Company_Email for company
>@	Fixed_Number	nvarchar(15)	Input parameter for the stored procedure sp_CREATE_COMPANY. Enter a Fixed_Number for company
>@	Province_Name	nvarchar(30)	Input parameter for the stored procedure sp_CREATE_COMPANY. Enter a valid Province_Name from the PROVINCE table.
>@	District_Name	nvarchar(50)	Input parameter for the stored procedure sp_CREATE_COMPANY. Enter a valid District_Name from the DISTRICT table.
>@	Address_Line	nvarchar(255)	Input parameter for the stored procedure sp_CREATE_COMPANY. Enter a Address_Line for company

5.2.Stored Procedure: sp_ADD_COMPANY_ADDRESS

Stored procedure using INSERT query to adding new address to the company.

Input/Output

	NAME	DATA TYPE	DESCRIPTION
>@	RoleId	Int	Input parameter for the stored procedure sp_ADD_COMPANY_ADDRESS. Enter a valid RoleId from the ROLE table.
>@	Company_Name	nvarchar(255)	Input parameter for the stored procedure sp_ADD_COMPANY_ADDRESS. Enter a valid Company_Name from the COMPANY table.
>@	Province_Name	nvarchar(30)	Input parameter for the stored procedure sp_ADD_COMPANY_ADDRESS. Enter a valid Province_Name from the PROVINCE table.
>@	District_Name	nvarchar(50)	Input parameter for the stored procedure sp_ADD_COMPANY_ADDRESS. Enter a valid District_Name from the DISTRICT table.
>@	Address_Line	nvarchar(255)	Input parameter for the stored procedure sp_ADD_COMPANY_ADDRESS. Enter a Address_Line for company address

5.3.Stored Procedure: sp_CREATE_EMPLOYEE

Stored procedure using INSERT query to adding new employee and product to employee.

Input/Output

	NAME	DATA TYPE	DESCRIPTION
>@	RoleForCreator	Int	Input parameter for the stored procedure sp_CREATE_EMPLOYEE. Enter a valid ROLEID from the ROLE table.
>@	Employee_Name	nvarchar(50)	Input parameter for the stored procedure sp_CREATE_EMPLOYEE. Enter a valid Employee_Name from the EMPLOYEE table.
>@	Employee_Surname	nvarchar(50)	Input parameter for the stored procedure sp_CREATE_EMPLOYEE. Enter a valid Employee_Surname from the EMPLOYEE table.
>@	Mobile_Number	nvarchar(15)	Input parameter for the stored procedure sp_CREATE_EMPLOYEE. Enter a valid Mobile_Number from the EMPLOYEE table.
>@	Employee_Email	nvarchar(50)	Input parameter for the stored procedure sp_CREATE_EMPLOYEE. Enter a valid Employee_Email from the EMPLOYEE table.
>@	ProfessionId	Int	Input parameter for the stored procedure sp_CREATE_EMPLOYEE. Enter a valid ProfessionId from the PROFESSION table.
>@	DepartmentId	Int	Input parameter for the stored procedure sp_CREATE_EMPLOYEE. Enter a valid DepartmentId from the DEPARTMANT table.
>@	RoleId	Int	Input parameter for the stored procedure sp_CREATE_EMPLOYEE. Enter a valid RoleId from the ROLE table.
>@	ProductId	Int	Input parameter for the stored procedure sp_CREATE_EMPLOYEE. Enter a valid ProductId from the PRODUCT table.

5.4.Stored Procedure: sp_CREATE_PRODUCT

Stored procedure using INSERT query to adding new product.

Input/Output

	NAME	DATA TYPE	DESCRIPTION
>@	RoleId	Int	Input parameter for the stored procedure sp_CREATE_PRODUCT. Enter a valid ROLEID from the ROLE table.
>@	Product_Name	nvarchar(100)	Input parameter for the stored procedure sp_CREATE_PRODUCT. Enter a Product_Name for product
>@	License_PeriodId	Int	Input parameter for the stored procedure sp_CREATE_PRODUCT. Enter a valid License_PeriodId from the LICENCE_PERIOD table.
>@	VersionNo	nvarchar(20)	Input parameter for the stored procedure sp_CREATE_PRODUCT. Enter a version number for product.

5.5.Stored Procedure: sp_ADD_PRODUCT_TO_COMPANY

Stored procedure using INSERT query to adding new product to company.

Input/Output

	NAME	DATA TYPE	DESCRIPTION
>@	RoleId	int	Input parameter for the stored procedure sp_ADD_PRODUCT_TO_COMPANY. Enter a valid ROLEID from the ROLE table.
>@	ProductId	int	Input parameter for the stored procedure sp_ADD_PRODUCT_TO_COMPANY. Enter a valid ProductId from the PRODUCT table.
>@	CompanyId	int	Input parameter for the stored procedure sp_ADD_PRODUCT_TO_COMPANY. Enter a valid CompanyId from the COMPANY table.

5.6.Stored Procedure: sp_ASSIGN_PRODUCT_TO_EMPLOYEE

Stored procedure using INSERT query to adding new product to employee.

Input/Output

	NAME	DATA TYPE	DESCRIPTION
>@	RoleId	int	Input parameter for the stored procedure sp_ASSING_PRODUCT_TO_EMPLOYEE. Enter a valid ROLEID from the ROLE table.
>@	ProductId	int	Input parameter for the stored procedure sp_ASSING_PRODUCT_TO_EMPLOYEE. Enter a valid ProductId from the PRODUCT table.
>@	EmployeeId	int	Input parameter for the stored procedure sp_ASSING_PRODUCT_TO_EMPLOYEE. Enter a valid EmployeeId from the EMPLOYEE table.

5.7.Stored Procedure: sp_ASSIGN_PRODUCT_TO_EMPLOYEE_MULTIPLE

Stored procedure using INSERT query to adding new products to the company.

Input/Output

	NAME	DATA TYPE	DESCRIPTION
>@	RoleId	int	Input parameter for the stored procedure sp_ASSIGN_PRODUCT_TO_EMPLOYEE_MULTIPLE. Enter a valid RoleId from the ROLE table.
>@	Products	nvarchar(max)	Input parameter for the stored procedure sp_ASSIGN_PRODUCT_TO_EMPLOYEE_MULTIPLE. Enter Products

>@	EmployeeId	int	Input parameter for the stored procedure sp_ASSIGN_PRODUCT_TO_EMPLOYEE_MULTIPLE. Enter a valid EmployeeId from the EMPLOYEE table.

5.8.Stored Procedure: sp_CREATE_PROFFESION

Stored procedure using INSERT query to add profession.

Input/Output

	NAME	DATA TYPE	DESCRIPTION
>@	RoleId	Int	Input parameter for the stored procedure sp_CREATE_PROFESSION. Enter a valid RoleId from the ROLE table.
>@	Profession_Name	nvarchar(50)	Input parameter for the stored procedure sp_CREATE_PROFESSION. Enter a valid Profession_Name for profession

5.9.Stored Procedure: sp_CREATE DEPARTMENT

Stored procedure using INSERT query to add department.

Input/Output

	NAME	DATA TYPE	DESCRIPTION
>@	RoleId	Int	Input parameter for the stored procedure sp_CREATE_DEPARTMENT. Enter a valid RoleId from the ROLE table.
>@	Department_Name	nvarchar(50)	Input parameter for the stored procedure sp_CREATE_DEPARTMENT. Enter a Department_Name for department.

5.10.Stored Procedure: sp_ASSIGN_DEMAND

Stored procedure using UPDATE query to assign demand to employee.

Input/Output

	NAME	DATA TYPE	DESCRIPTION
>@	RoleId	int	Input parameter for the stored procedure sp_ASSIGN_DEMAND. Enter a valid RoleId from the ROLE table.
>@	ProductId	int	Input parameter for the stored procedure sp_ASSIGN_DEMAND. Enter a valid ProductId from the PRODUCT table.
>@	EmployeeId	int	Input parameter for the stored procedure sp_ASSIGN_DEMAND. Enter a valid EmployeeId from the EMPLOYEE table.

5.11.Stored Procedure: sp_CLOSE_DEMAND

Stored procedure using UPDATE query to closing demand.

Input/Output

	NAME	DATA TYPE	DESCRIPTION
>@	DemandId	int	Input parameter for the stored procedure sp_CLOSE_DEMAND. Enter a valid DemandId from the DEMAND table.
>@	EmployeeId	int	Input parameter for the stored procedure sp_CLOSE_DEMAND. Enter a valid EmployeeId from the EMPLOYEE table.
>@	RoleId	int	Input parameter for the stored procedure sp_CLOSE_DEMAND. Enter a valid RoleId from the ROLE table.

>@	Closing_statement	nvarchar(max)	Input parameter for the stored procedure sp_CLOSE_DEMAND. Enter a Closing_statement for demand
>@	Solved_hour	int	Input parameter for the stored procedure sp_CLOSE_DEMAND. Enter a Solved_hour for demand

5.12.Stored Procedure: sp_CREATE_VERSION

Stored procedure using INSERT query for adding new version.

Input/Output

	NAME	DATA TYPE	DESCRIPTION
>@	RoleId	int	Input parameter for the stored procedure sp_CREATE_VERSION. Enter a valid RoleId from the ROLE table.
>@	ProductId	int	Input parameter for the stored procedure sp_CREATE_VERSION. Enter a valid ProductId from the PRODUCT table.
>@	VersionNo	nvarchar(20)	Input parameter for the stored procedure sp_CREATE_VERSION. Enter a version number for VERSION.

5.13.Stored Procedure: sp_CREATE_COMPANY_USER

Stored procedure using INSERT query to add new company user.

Input/Output

	NAME	DATA TYPE	DESCRIPTION
>@	RoleIdForCreator	int	Input parameter for the stored procedure sp_CREATE_COMPANY_USER. Enter a valid RoleIdForCreator from the ROLE table.
>@	Company_User_Name	nvarchar(50)	Input parameter for the stored procedure sp_CREATE_COMPANY_USER. Enter a Company_User_Name for company user
>@	Company_User_Surname	nvarchar(50)	Input parameter for the stored procedure sp_CREATE_COMPANY_USER. Enter a Company_User_Surname for company user
>@	Mobile_Number	nvarchar(15)	Input parameter for the stored procedure sp_CREATE_COMPANY_USER. Enter a Mobile_Number for company user
>@	Company_User_Email	nvarchar(50)	Input parameter for the stored procedure sp_CREATE_COMPANY_USER. Enter a Company_User_Email for company user
>@	CompanyId	int	Input parameter for the stored procedure sp_CREATE_COMPANY_USER. Enter a valid CompanyId from the COMPANY table.
>@	RoleId	int	Input parameter for the stored procedure sp_CREATE_COMPANY_USER. Enter a valid RoleId from the ROLE table.

5.14.Stored Procedure: sp_CREATE_DEMAND

Stored procedure using INSERT query to adding new demand.

Input/Output

	NAME	DATA TYPE	DESCRIPTION
>@	RoleId	Int	Input parameter for the stored procedure sp_CREATE_DEMAND. Enter a valid RoleId from the ROLE table.

>@	Title	nvarchar(max)	Input parameter for the stored procedure sp_CREATE_DEMAND. Enter a title for demand.
>@	Text	nvarchar(max)	Input parameter for the stored procedure sp_CREATE_DEMAND. Enter a text for demand.
>@	Demand_TypeId	Int	Input parameter for the stored procedure sp_CREATE_DEMAND. Enter a valid Demand_TypeId from the DEMAND_TYPE table.
>@	Order_Of_UrgencyId	Int	Input parameter for the stored procedure sp_CREATE_DEMAND. Enter a valid Order_Of_UrgencyId from the ORDER_OF_URGENCY table.
>@	Company_UserId	Int	Input parameter for the stored procedure sp_CREATE_DEMAND. Enter a valid Company_UserId from the COMPANY_USER table.
>@	ProductId	Int	Input parameter for the stored procedure sp_CREATE_DEMAND. Enter a valid ProductId from the PRODUCT table.
>@	VersionId	Int	Input parameter for the stored procedure sp_CREATE_DEMAND. Enter a valid VersionId from the VERSION table.

5.15.Stored Procedure: sp_UPDATE_DEMAND

Stored procedure using UPDATE query to update demand.

Input/Output

NAME		DATA TYPE	DESCRIPTION
>@	RoleId	int	Input parameter for the stored procedure sp_UPDATE_DEMAND. Enter a valid RoleId from the ROLE table.
>@	DemandId	int	Input parameter for the stored procedure sp_UPDATE_DEMAND. Enter a valid DemandId from the DEMAND table.
>@	NewText	nvarchar(max)	Input parameter for the stored procedure sp_UPDATE_DEMAND. Enter a new text for demand
>@	Demand_TypeId	int	Input parameter for the stored procedure sp_UPDATE_DEMAND. Enter a valid Demand_TypeId from the DEMAND_TYPE table.
>@	Order_Of_UrgencyId	int	Input parameter for the stored procedure sp_UPDATE_DEMAND. Enter a valid Order_Of_UrgencyId from the ORDER_OF_URGENCY table.

5.16.Stored Procedure: sp_DELETE_DEMAND

Stored procedure using UPDATE query to delete demand record.

Input/Output

NAME		DATA TYPE	DESCRIPTION
>@	DemandCreatorId	int	Input parameter for the stored procedure sp_DELETE_DEMAND. Enter a valid DemandCreatorId
>@	DemandId	int	Input parameter for the stored procedure sp_DELETE_DEMAND. Enter a valid DemandId from the DEMAND table.

5.17.Stored Procedure: sp_DELETE_EMPLOYEE

Stored procedure using UPDATE query to delete employee record

Input/Output

	NAME	DATA TYPE	DESCRIPTION
>@	RoleId	int	Input parameter for the stored procedure sp_DELETE_EMPLOYEE. Enter a valid RoleId from ROLE table
>@	EmployeeId	int	Input parameter for the stored procedure sp_DELETE_EMPLOYEE. Enter a valid EmployeeId from the EMPLOYEE table.

5.18.Stored Procedure: sp_DELETE_COMPANY_USER

Stored procedure using UPDATE query to company user record

Input/Output

	NAME	DATA TYPE	DESCRIPTION
>@	RoleId	int	Input parameter for the stored procedure sp_DELETE_COMPANY_USER. Enter a valid RoleId from ROLE table
>@	CompanyUserId	Int	Input parameter for the stored procedure sp_DELETE_COMPANY_USER. Enter a valid CompanyUserId from the COMPANY_USER table.

6.VIEWS

6.1.View: vw_GET_ALL_COMPANY_USER

Company User names and surnames.

NAME	DATA TYPE	N	DESCRIPTION
NAME	nvarchar(50)		Company user name from COMPANY_USER
SURNAME	nvarchar(50)		Company user surname from COMPANY_USER

6.2.View: vw_GET_ALL_COMPANY

Company name, email, website and phone number.

NAME	DATA TYPE	N	DESCRIPTION
ID	Int		Company ID from COMPANY
COMPANY_NAME	nvarchar(255)		Company Name from COMPANY
WEBSITE	nvarchar(100)		Company Website from COMPANY
EMAIL	nvarchar(100)		Company Email from COMPANY

FIXED_NUMBER	nvarchar(15)	**N**	Company Fixed Number from COMPANY

6.3.View: vw_GET_ALL_DEMAND

All Demand columns.

NAME	DATA TYPE	N	DESCRIPTION
ID	Int		Demand ID from DEMAND
TITLE	nvarchar(max)		Demand title from DEMAND
TEXT	nvarchar(max)		Demand text from DEMAND
DEMAND_TYPEID	Int		Demand type id from DEMAND
ORDER_OF_URGENCYID	Int		Demand order of urgency from DEMAND
DEMAND_STATEID	Int	**N**	Demand statement id from DEMAND
CLOSING_STATEMENT	nvarchar(max)	**N**	Demand closing statement from DEMAND
CLOSING_DATE	Datetime	**N**	Demand closing date from DEMAND
COMPANY_USERID	Int		Demand company user id from DEMAND
EMPLOYEEID	Int	**N**	Demand employee id from DEMAND
PRODUCTID	Int		Demand product id from DEMAND
SOLVEDHOUR	Int	**N**	Demand solved hour from DEMAND
VERSIONID	Int		Demand version id from DEMAND
CREATE_DATE	Datetime		Demand create date from DEMAND
MODIFICATION_DATE	Datetime		Demand modification date from DEMAND
RECORD_STATUS	Bit		Demand record status from DEMAND

6.4.View: vw_GET_ALL_PRODUCT

Return product id, name, license period and version with JOIN statements.

NAME	DATA TYPE	N	DESCRIPTION
ID	Int		Product id from PRODUCT
PRODUCT_NAME	nvarchar(100)		Product name from PRODUCT
PERIOD	nvarchar(50)		Product license period from LICENSE_PERIOD
VERSIONNO	nvarchar(20)		Product version no from VERSION

7.TRIGGER

NAME	WHEN	DESCRIPTION
trg_TICKET_ASSIGNMENT_NOTIFICATION	AFTER UPDATE	When the demand is completed the company user is notified.