Database Design

A4B KPI BizOps Organization Schema

Ryan Timbrook
IST659 Data Admin Concepts & DB Mgmt

Version 2.0
9/30/2018

Revision History

Date	Version	Description	Author(s)
8/13/2018	1.0	Part 1, Summary of project; Includes Conceptual Model; Includes Normalized Logical Model	Ryan Timbrook
9/30/18	2.0	Part 2, Includes Physical Database Design; Data Creation Examples; Data Manipulation Examples; Code snippets that provide examples of how the data answers questions posed in the summary	Ryan Timbrook

Table of Contents

1 Introduction

The Voice Services team at T-Mobile is developing a new set of Amazon Alexa Skills that are intended for Internal use by employees of T-Mobile. This team is leveraging Amazon's new 'Alexa for Business' AWS service which allows a company to publish Alexa Skills internally and has a management facility for inviting users to use these Skills. The Voice team sees Alexa Skills as the perfect tool for brining awareness to the skeptics of how powerful and efficient voice solutions can be.
Wanting to make a real impression on their users, they've selected a topic which all of the leadership at T-Mobile and other companies spend the majority of their time asking, being questioned on, and researching. These are Key Performance Indicators (KPIs). It's the thing that keeps most business people up at night and is the first thing they are trying to get caught up on every morning. With hundreds, if not thousands, of variabilities that make up KPIs and how they are reported, this is an opportunity to great to ignore.

To be successful in changing their leaders, as well as others, habits and perceptions of voice systems, the Voice team understands that their Skills must be simple to use and personal to the end user. Through investigation, the team has found there's no data source system that maps users to KPIs they own or have responsibility of, nor is there a way of assigning themselves to them. Based on this and the below bullets they have decided to develop a new database for this purpose.

Subset of challenges found:

- No logical mappings exist where a system would know what KPI's a user would be responsible for or interested in knowing about
- Users often aren't aware of the KPIs they are responsible for or what operational parameters these KPIs are composed of
- Organizations change regularly and with it KPIs change as well as People change roles
- KPIs often have the same or similar names across organizations, but are very different in their meaning and data definitions that comprise them
- KPIs are often aggregates at each hierarchy jump in the organization. An example would be organization performance goal metrics. Each team has the same KPI name, but as the hierarchy steps up, the KPIs values are summed at each level.

1.1 Purpose

The Voice Services team wishes to create a database that maintains organizational hierarchy data which links a user (employee) to KPIs they are responsible for or have registered to receive reports on. Employees of the DTD organization are software developers who own the applications they build. These employees all belong to well organized teams that are managed by an organization. Each team in the organization has its own sub-hierarchies with varying supervisory levels. The applications these teams build and manage are often referred to as Capabilities and a collection of Capabilities is referred to as a Product. To make it simpler for users of the new Alexa Skill to find the KPIs they are most interested in, KPIs need to be findable based on Product or Capability name searches as well as by Teams within an Organization.

The Voice Services team would like this database to be able to answer questions such as:

- What Employee's belong to a given Organization?
- What Employee's belong to a given Team?
- Who is the Supervisor of a given Organization?
- What KPIs are a given Organization responsible for?
- In an Organization, what Team is responsible for a given KPI?
- For a given KPI, in a Team, who is the primary point of contact and what is their preferred contact channel?
- For a given Employee, what KPIs are they associated to given their Role in the Organization?
- What KPIs are associated to a given Product
- What KPIs are associated to a given Capability
- What KPIs are associated to a given Application
- What Products, Capabilities, and Applications is a given Organization responsible for?

Additional data requirements include:

- An Employee can have one or more Roles
- An Employee can be a Supervisor of zero or more Employees
- An Employee belongs to one or more Teams; A Team has one or more Employees
- A Department has many teams; A Team belongs to one Department
- A Team owns zero or more Applications; An Application is owned by one Team
- Team Names can be duplicated, but not within the same Department
- An Employee can self-register for one or more KPIs
- KPIs are Key Performance Indicators that measure the success of a given Application
- An Employee can search for KPIs that are associated to a Product
- An Employee can search for KPIs that are associated to an Application
- An Employee can search for KPIs that a Team owns
- An Employee can search for KPIs that an Organization owns

1.2 Scope

The Database Design will consist of the following:

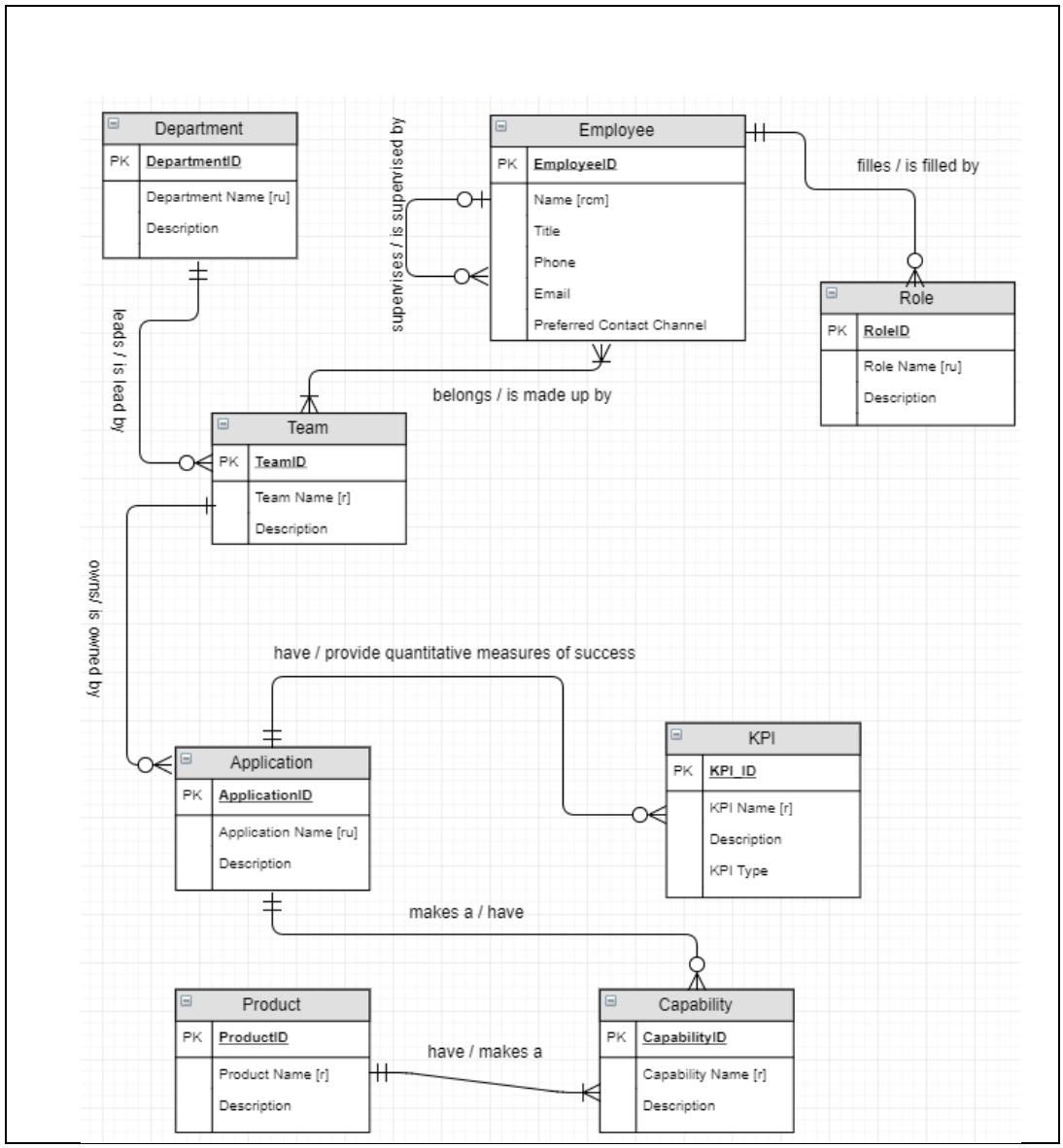
- Conceptual Model
- Normalized Logical Model
- Physical Database Design
- Data Creation
- Data Manipulation
- Answering Data Questions
- Implementation Screen Shots

2 Conceptual Model

2.1 Description

Below is an entity-relationship model that logically represents the data in which the Voice Services team would like to maintain so that their Alexa KPI Skill will have a personal connection to the Skill user.

2.2 E-R Diagram

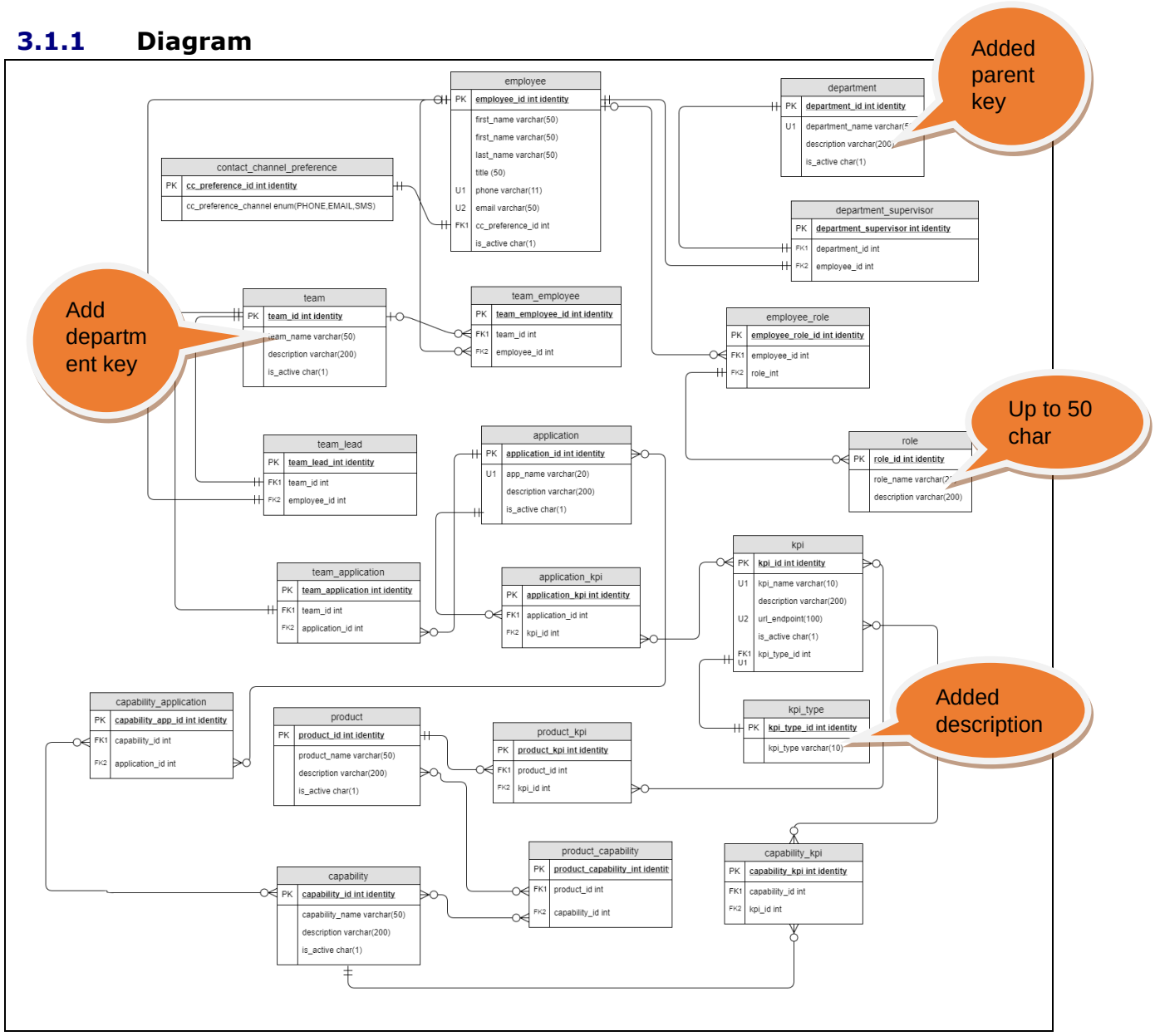


3 Normalized Logical Model

3.1 Description

Below is the logical data model which was crafted after reviewing the A4B Alexa KPI data and normalizing the model. This is the final logical model to be implemented.

3.1.1 Diagram

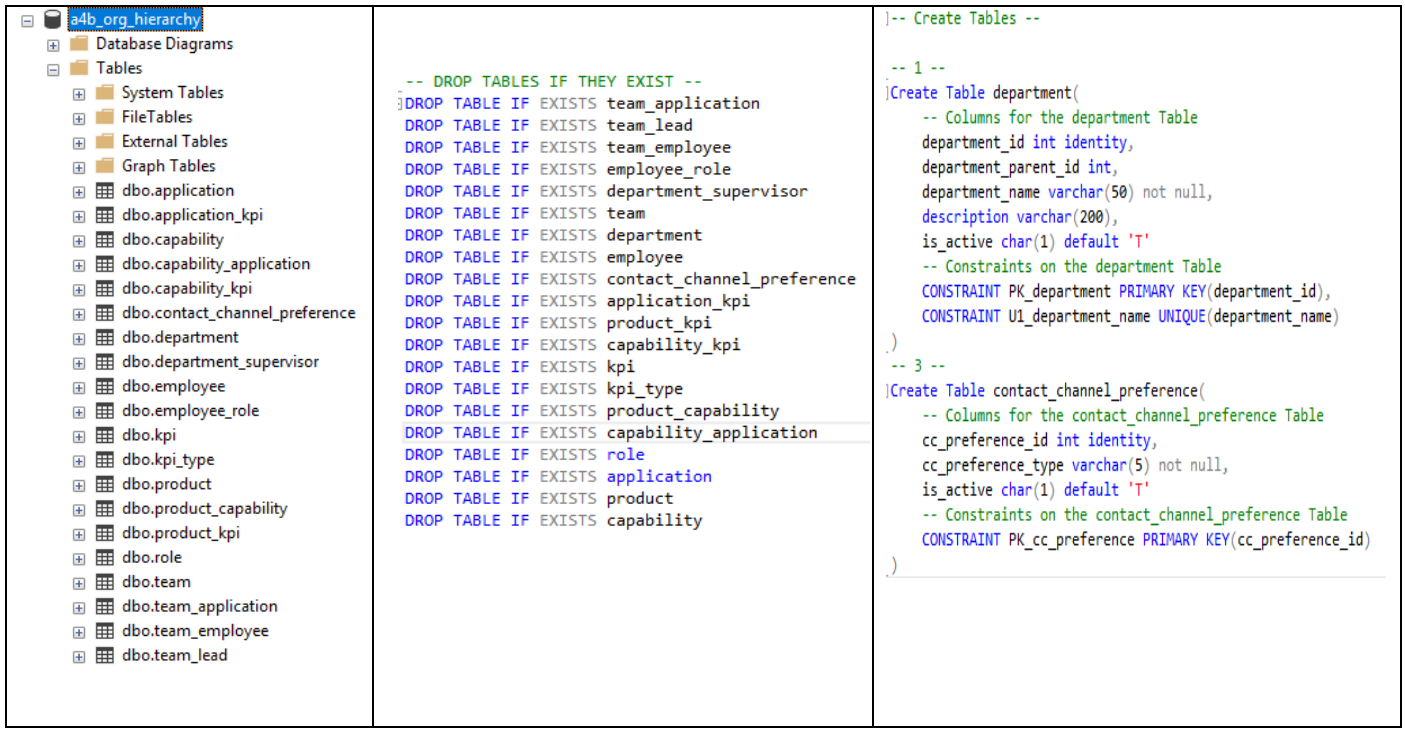


4 Physical Database Design (SQL DDL Commands)

4.1 Description

Is the SQL CREATE TABLE statements needed to CREATE all tables in the database. This includes DROP TABLE statements at the start of the script for clean execution.

4.2 Screen Print Examples



4.3 Table Definitions SQL

Complete SQL File Attached Here

4.3.1.1 SQL DDL Commands

```
/*
       IST 659 Data Admin Concepts &Db Mgmt
       Date: 9/30/2018
       Project Deliverable 2: DDL

*/

-- DROP TABLES IF THEY EXIST --
```

```
DROP TABLE IF EXISTS team_application
DROP TABLE IF EXISTS team_lead
DROP TABLE IF EXISTS team_employee
DROP TABLE IF EXISTS employee_role
DROP TABLE IF EXISTS department_supervisor
DROP TABLE IF EXISTS team
DROP TABLE IF EXISTS department
DROP TABLE IF EXISTS employee
DROP TABLE IF EXISTS contact_channel_preference
DROP TABLE IF EXISTS application_kpi
DROP TABLE IF EXISTS product_kpi
DROP TABLE IF EXISTS capability_kpi
DROP TABLE IF EXISTS kpi
DROP TABLE IF EXISTS kpi_type
DROP TABLE IF EXISTS product_capability
DROP TABLE IF EXISTS capability_application
DROP TABLE IF EXISTS role
DROP TABLE IF EXISTS application
DROP TABLE IF EXISTS product
DROP TABLE IF EXISTS capability



-- Create Tables --

-- 1 --
Create Table department(
       -- Columns for the department Table
       department_id int identity,
       department_parent_id int,
       department_name varchar(50) not null,
       description varchar(200),
       is_active char(1) default 'T'
       -- Constraints on the department Table
       CONSTRAINT PK_department PRIMARY KEY(department_id),
       CONSTRAINT U1_department_name UNIQUE(department_name)
)
-- 3 --
Create Table contact_channel_preference(
       -- Columns for the contact_channel_preference Table
       cc_preference_id int identity,
       cc_preference_type varchar(5) not null,
       is_active char(1) default 'T'
       -- Constraints on the contact_channel_preference Table
       CONSTRAINT PK_cc_preference PRIMARY KEY(cc_preference_id)
)
-- 4 --
Create Table employee(
       -- Columns for the employee Table
       employee_id int identity,
       first_name varchar(50) not null,
       first_name2 varchar(50),
       last_name varchar(50) not null,
       title varchar(100),
```

```
	phone varchar(13) not null,
	email varchar(50) not null,
	cc_preference_id int,
	is_active char(1) default 'T'
	-- Constraints on the employee Table
	CONSTRAINT PK_employee PRIMARY KEY(employee_id),
	CONSTRAINT FK1_cc_preference FOREIGN KEY(cc_preference_id) REFERENCES
contact_channel_preference(cc_preference_id),
	CONSTRAINT U1_phone UNIQUE(phone),
	CONSTRAINT U2_email UNIQUE(email)
)
-- 5 --
-- Composit Primary Key, different departments can have the same team names, but team names
must be unique within the same department
Create Table team(
	-- Columns for the team Table
	team_id int identity,
	team_name varchar(50) not null,
	department_id int not null,
	team_parent_id int,
	description varchar(200),
	is_active char(1) default 'T'
	-- Constraints on the team Table
	CONSTRAINT PK_team PRIMARY KEY(team_id),
	CONSTRAINT U1_team UNIQUE(team_name,department_id)

)
-- 6 --
Create Table role(
	-- Columns for the role Table
	role_id int identity,
	role_name varchar(50) not null,
	description varchar(200),
	is_active char(1) default 'T'
	-- Constraints on the role Table
	CONSTRAINT PK_role PRIMARY KEY(role_id)
)

-- 7 --
Create Table application(
	-- Columns for the application Table
	application_id int identity,
	app_name varchar(20) not null,
	description varchar(200),
	is_active char(1) default 'T'
	-- Constraints on the application Table
	CONSTRAINT PK_application PRIMARY KEY(application_id)
)

-- 8 --
Create Table product(
	-- Columns for the product Table
	product_id int identity,
	product_name varchar(50) not null,
```

```
	description varchar(200),
	is_active char(1) default 'T'
	-- Constraints on the product Table
	CONSTRAINT PK_product PRIMARY KEY(product_id)
)

-- 9 --
Create Table capability(
	-- Columns for the capability Table
	capability_id int identity,
	capability_name varchar(50) not null,
	description varchar(200),
	is_active char(1) default 'T'
	-- Constraints on the capability Table
	CONSTRAINT PK_capability PRIMARY KEY(capability_id)
)

-- 10 --
Create Table kpi_type(
	-- Columns for the x Table
	kpi_type_id int identity,
	kpi_type varchar(20) not null,
	description varchar(200),
	is_active char(1) default 'T'
	-- Constraints on the x Table
	CONSTRAINT PK_kpi_type PRIMARY KEY(kpi_type_id)
)

-- 11 --
Create Table team_application(
	-- Columns for the team_application Table
	team_application_id int identity,
	team_id int not null,
	application_id int not null,
	is_active char(1) default 'T'
	-- Constraints on the team_application Table
	CONSTRAINT PK_team_application PRIMARY KEY(team_application_id),
	CONSTRAINT U1_team_application UNIQUE(team_id,application_id),
	CONSTRAINT FK1_team_application FOREIGN KEY(team_id) REFERENCES team(team_id),
	CONSTRAINT FK2_team_application FOREIGN KEY(application_id) REFERENCES
application(application_id)
)

-- 12 --
Create Table team_lead(
	-- Columns for the team_lead Table
	team_lead_id int identity,
	team_id int not null,
	employee_id int not null,
	is_active char(1) default 'T'
	-- Constraints on the team_lead Table
	CONSTRAINT PK_team_lead PRIMARY KEY(team_lead_id),
	CONSTRAINT U1_team_lead UNIQUE(team_id,employee_id),
	CONSTRAINT FK1_team_lead FOREIGN KEY(team_id) REFERENCES team(team_id),
```

```
	CONSTRAINT FK2_team_lead FOREIGN KEY(employee_id) REFERENCES employee(employee_id)
)

-- 13 --
Create Table team_employee(
	-- Columns for the team_employee Table
	team_employee_id int identity,
	team_id int not null,
	employee_id int not null,
	is_active char(1) default 'T'
	-- Constraints on the team_employee Table
	CONSTRAINT PK_team_employee PRIMARY KEY(team_employee_id),
	CONSTRAINT U1_team_employee UNIQUE(team_id, employee_id),
	CONSTRAINT FK1_team_employee FOREIGN KEY(team_id) REFERENCES team(team_id),
	CONSTRAINT FK2_team_employee FOREIGN KEY(employee_id) REFERENCES employee(employee_id)
)

-- 14 --
Create Table employee_role(
	-- Columns for the employee_role Table
	employee_role_id int identity,
	employee_id int not null,
	role_id int not null,
	is_active char(1) default 'T'
	-- Constraints on the employee_role Table
	CONSTRAINT PK_employee_role PRIMARY KEY(employee_role_id),
	CONSTRAINT FK1_employee_role FOREIGN KEY(employee_id) REFERENCES
employee(employee_id),
	CONSTRAINT FK2_employee_role FOREIGN KEY(role_id) REFERENCES role(role_id)
)

-- 15 --
Create Table department_supervisor(
	-- Columns for the department_supervisor Table
	department_supervisor_id int identity,
	department_id int not null,
	employee_id int not null,
	is_active char(1) default 'T'
	-- Constraints on the department_supervisor Table
	CONSTRAINT PK_department_supervisor PRIMARY KEY(department_supervisor_id),
	CONSTRAINT U1_department_supervisor UNIQUE(department_id),
	CONSTRAINT FK1_department_supervisor FOREIGN KEY(department_id) REFERENCES
department(department_id),
	CONSTRAINT FK2_department_supervisor FOREIGN KEY(employee_id) REFERENCES
employee(employee_id)
)

-- 16 --
	Create Table kpi(
	-- Columns for the kpi Table
	kpi_id int identity,
	kpi_name varchar(20) not null,
	description varchar(200),
	url_endpoint varchar(200),
```

```
	kpi_type_id int not null,
	is_active char(1) default 'T'
	-- Constraints on the kpi Table
	CONSTRAINT PK_kpi PRIMARY KEY(kpi_id),
	CONSTRAINT FK1_kpi FOREIGN KEY(kpi_type_id) REFERENCES kpi_type(kpi_type_id)
)

-- 17 --
Create Table application_kpi(
	-- Columns for the application_kpi Table
	application_kpi_id int identity,
	application_id int not null,
	kpi_id int not null,
	is_active char(1) default 'T'
	-- Constraints on the application_kpi Table
	CONSTRAINT PK_application_kpi PRIMARY KEY(application_kpi_id),
	CONSTRAINT U1_application_KPI UNIQUE(application_id,kpi_id),
	CONSTRAINT FK1_application_kpi FOREIGN KEY(application_id) REFERENCES
application(application_id),
	CONSTRAINT FK2_application_kpi FOREIGN KEY(kpi_id) REFERENCES kpi(kpi_id)
)

-- 18 --
Create Table product_kpi(
	-- Columns for the product_kpi Table
	product_kpi_id int identity,
	product_id int not null,
	kpi_id int not null,
	is_active char(1) default 'T'
	-- Constraints on the product_kpi Table
	CONSTRAINT PK_product_kpi PRIMARY KEY(product_kpi_id),
	CONSTRAINT U1_product_kpi UNIQUE(product_id,kpi_id),
	CONSTRAINT FK1_product_kpi FOREIGN KEY(product_id) REFERENCES product(product_id),
	CONSTRAINT FK2_product_kpi FOREIGN KEY(kpi_id) REFERENCES kpi(kpi_id)
)

-- 19 --
Create Table capability_kpi(
	-- Columns for the capability_kpi Table
	capability_kpi_id int identity,
	capability_id int not null,
	kpi_id int not null,
	is_active char(1) default 'T'
	-- Constraints on the capability_kpi Table
	CONSTRAINT PK_capability_kpi PRIMARY KEY(capability_kpi_id),
	CONSTRAINT U1_capability_kpi UNIQUE(capability_id,kpi_id),
	CONSTRAINT FK1_capability_kpi FOREIGN KEY(capability_id) REFERENCES
capability(capability_id),
	CONSTRAINT FK2_capability_kpi FOREIGN KEY(kpi_id) REFERENCES kpi(kpi_id)
)

-- 20 --
Create Table product_capability(
	-- Columns for the product_capability Table
```

```
	product_capability_id int identity,
	product_id int not null,
	capability_id int not null,
	is_active char(1) default 'T'
	-- Constraints on the product_capability Table
	CONSTRAINT PK_product_capability PRIMARY KEY(product_capability_id),
	CONSTRAINT U1_product_capability UNIQUE(product_id,capability_id),
	CONSTRAINT FK1_product_capability FOREIGN KEY(product_id) REFERENCES
product(product_id),
	CONSTRAINT FK2_product_capability FOREIGN KEY(capability_id) REFERENCES
capability(capability_id)
)

-- 21 --
Create Table capability_application(
	-- Columns for the capability_application Table
	capability_app_id int identity,
	capability_id int not null,
	application_id int not null,
	is_active char(1) default 'T'
	-- Constraints on the capability_application Table
	CONSTRAINT PK_capability_application PRIMARY KEY(capability_app_id),
	CONSTRAINT U1_capability_application UNIQUE(capability_id,application_id),
	CONSTRAINT FK1_capability_application FOREIGN KEY(capability_id) REFERENCES
capability(capability_id),
	CONSTRAINT FK2_capability_application FOREIGN KEY(application_id) REFERENCES
application(application_id)
)
```

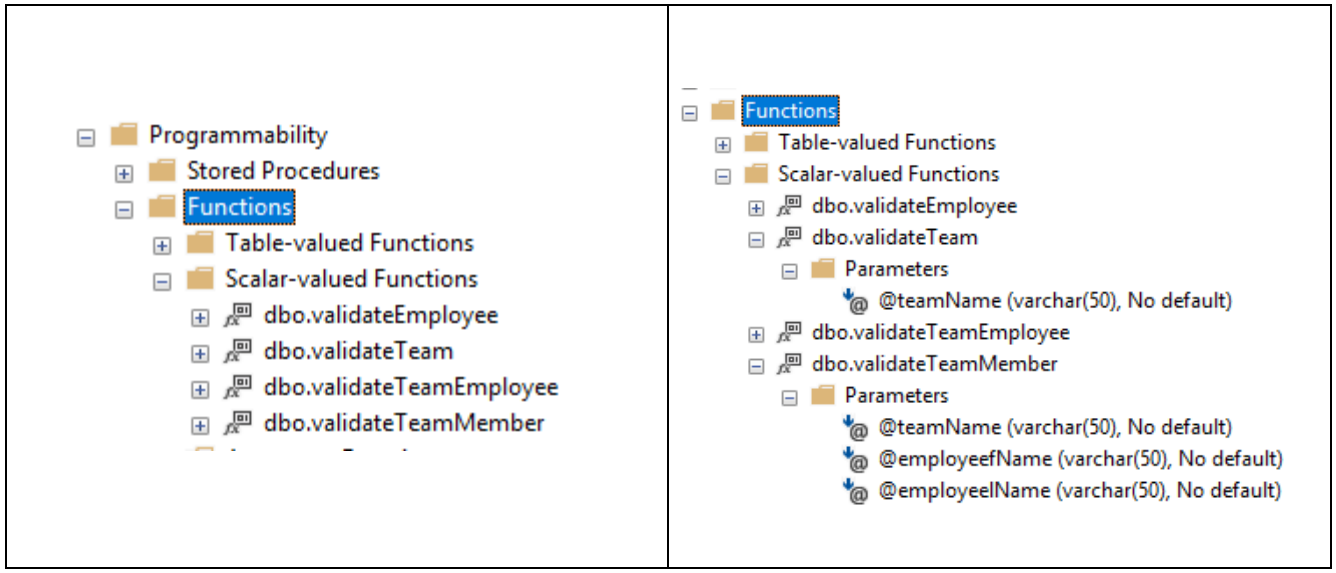
5 SQL DDL Programming Objects

5.1 Description

CREATE Statements for programming objects such as Views, User-Defined Functions and Stored Procedures

5.2 Object – Functions

5.2.1 *Screen Print Ex:*



5.2.2 *SQL DDL Script*

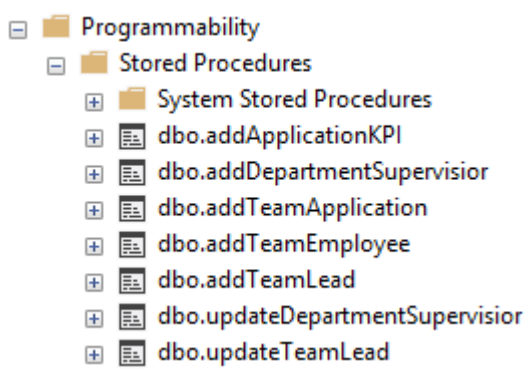
```
/*
       IST 659 Data Admin Concepts &Db Mgmt
       Date: 9/30/2018
       Project Deliverable 2: FUNCTIONS

*/

-- #### VALIDATE EMPLOYEE EXISTS ####
-- DROP FUNCTION IF EXISTS
IF EXISTS (SELECT * FROM sys.objects WHERE object_id=OBJECT_ID(N'dbo.validateEmployee') AND
type in (N'FN',N'IF',N'TF',N'FS',N'FT'))
DROP FUNCTION validateEmployee
GO
-- CREATE FUNCTION
CREATE FUNCTION validateEmployee(@employeefName varchar(50), @employeelName varchar(50))
RETURNS int AS
BEGIN
```

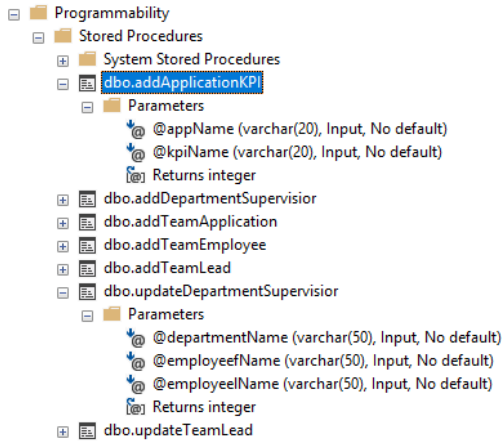
```
	DECLARE @success int
	SET @success = 0

	SELECT @success = ISNULL((SELECT 1 FROM employee where first_name = @employeefName AND
last_name = @employeelName), 0)

	RETURN @success
END
GO


-- #### VALIDATE A TEAM EXISTS
-- DROP FUNCTION IF EXISTS
IF EXISTS (SELECT * FROM sys.objects WHERE object_id=OBJECT_ID(N'dbo.validateTeam') AND type
in (N'FN',N'IF',N'TF',N'FS',N'FT'))
DROP FUNCTION validateTeam
GO
-- CREATE FUNCTION
CREATE FUNCTION validateTeam(@teamName varchar(50))
RETURNS int AS
BEGIN
	DECLARE @success int
	SET @success = 0

	SELECT @success = ISNULL((SELECT 1 FROM team where team_name = @teamName AND is_active
= 'T'), 0)

	RETURN @success
END
GO


-- #### VALIDATE AN EMPLOYEE IS A MEMBER OF A TEAM
-- DROP FUNCTION IF EXISTS
IF EXISTS (SELECT * FROM sys.objects WHERE object_id=OBJECT_ID(N'dbo.validateTeamEmployee')
AND type in (N'FN',N'IF',N'TF',N'FS',N'FT'))
DROP FUNCTION validateTeamEmployee
GO
-- CREATE FUNCTION
CREATE FUNCTION validateTeamEmployee(@teamName varchar(50), @employeefName varchar(50),
@employeelName varchar(50))
RETURNS int AS
BEGIN
	DECLARE @success int, @employee_id int, @team_id int
	SET @success = 0
	SET @employee_id = 0

	-- USE EXISTING FUNCTIONS TO VALIDATE EMPLOYEE AND TEAM INPUTS ARE VALID
	EXEC @success = validateEmployee @employeeFname, @employeelName
	IF @success <> 1
		RETURN @success
	EXEC @success = validateTeam @teamName
	IF @success <> 1
		RETURN @success
```

```
        --
        SELECT @success = ISNULL((SELECT 1 FROM team_employee
                WHERE employee_id = (SELECT employee_id FROM employee WHERE first_name =
@employeeFname AND last_name = @employeelName)
                        AND team_id = (SELECT team_id FROM team WHERE team_name = @teamName)
                ), 0)

        RETURN @success
END
GO
```

5.3 Object – Stored Procedures

5.3.1 *Screen Print Ex:*





5.3.2 *SQL DDL Script*

```
/*
        IST 659 Data Admin Concepts &Db Mgmt
        Date: 9/30/2018
        Project Deliverable 2: Stored Procedure

*/
```

```
-- #### ADD SUPERVISIOR TO A DEPARTMENT
-- Delete Procedure if exists
DROP PROCEDURE IF EXISTS addDepartmentSupervisior
GO
-- Function Make an Employee the Department Supervisior
-- TODO: Add conditional logic validating input parameters contain valid data; exception
handling for business rule constraints
CREATE PROCEDURE addDepartmentSupervisior(@departmentName varchar(50),@employeefName
varchar(50),@employeelName varchar(50)) AS
BEGIN
	DECLARE @departmentID int, @employeeID int, @success int
	SET @success = 0
	-- Validate that the Employee info entered is an existing Employee in the database
	EXEC @success = dbo.validateEmployee @employeefName, @employeelName
	PRINT(@success)
	-- IF a valid employee wasn't found, exit the procedure with an error
	IF @success <> 1
		RETURN @success


	SELECT @departmentID = department_id FROM department WHERE department_name =
@departmentName
	PRINT(@departmentID)
	SELECT @employeeID = employee_id FROM employee WHERE first_name = @employeefName AND
last_name = @employeelName
	PRINT(@departmentID)

	-- Add row to department_supervisor table
	INSERT INTO department_supervisor(department_id,employee_id)
		VALUES (@departmentID, @employeeID)
	SELECT @success = ISNULL((SELECT 1 FROM department_supervisor WHERE
department_supervisor_id = @@identity), 0)

	RETURN @success
END
GO

-- #### CHANGE SUPERVISOR OF A DEPARTMENT
-- Delete Procedure if exists
DROP PROCEDURE IF EXISTS updateDepartmentSupervisior
GO
-- Function Make an Employee the Department Supervisior
CREATE PROCEDURE updateDepartmentSupervisior(@departmentName varchar(50),@employeefName
varchar(50),@employeelName varchar(50)) AS
BEGIN
	--
	DECLARE @departmentID int, @employeeID int, @success int
	SET @success = 0
	-- Validate that the Employee info entered is an existing Employee in the database
	EXEC @success = dbo.validateEmployee @employeefName, @employeelName
	--PRINT(@success)
	-- IF a valid employee wasn't found, exit the procedure with an error
	IF @success <> 1
		RETURN @success
```

```
	SELECT @departmentID = department_id FROM department WHERE department_name = 
@departmentName
	--PRINT(@departmentID)
	SELECT @employeeID = employee_id FROM employee WHERE first_name = @employeefName AND 
last_name = @employeelName
	--PRINT(@departmentID)

	-- Update row to department_supervisor table
	UPDATE department_supervisor
	SET employee_id = @employeeID
	WHERE department_id = @departmentID

	RETURN @success
END
GO

-- ####  ADD TEAM LEAD
DROP PROCEDURE IF EXISTS addTeamLead
GO
-- Function Make an Employee the Team Lead
/*
	Business Rule Validations:
		-Sequence:1: Employee must exist in the employee table before they can be added 
as a team lead
		-Sequence:2: Team must exist in the team table before an employee can be added 
as a team lead
		-Sequence:3: Employee must be a member of the team they are being assigned lead 
over
		If any of the above rules are viloated, a success code of 0 will be returned
*/
CREATE PROCEDURE addTeamLead(@teamName varchar(50), @employeefName varchar(50),@employeelName 
varchar(50)) AS
BEGIN
	--
	DECLARE @teamID int, @employeeID int, @success int, @successTeam int, @successEmployee 
int, @successTeamEmployee int
	SET @success = 1
	-- Validate that the Employee info entered is an existing Employee in the database
	EXEC @successEmployee = validateEmployee @employeefName, @employeelName
	EXEC @successTeam = validateTeam @teamName
	EXEC @successTeamEmployee = validateTeamEmployee @teamName, @employeefName, 
@employeelName
	-- IF a valid employee wasn't found or a valid team wasn't found, exit the procedure 
with an error
	IF (@successEmployee <> 1 OR @successTeam <> 1 OR @successTeamEmployee <> 1)
		BEGIN
			SET @success = 0
			RETURN @success
		END
	ELSE
		BEGIN
			SELECT @teamID = team_id FROM team WHERE team_name = @teamName
			SELECT @employeeID = employee_id FROM employee WHERE first_name =
```

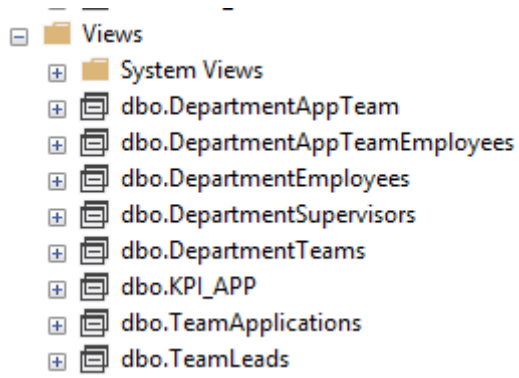
```
@employeefName AND last_name = @employeelName
                     -- Add row to team_lead table
                     INSERT INTO team_lead(team_id,employee_id)
                            VALUES (@teamID, @employeeID)
                     SELECT @success = ISNULL((SELECT 1 FROM team_lead WHERE team_lead_id =
@@identity), 0)
              END


       RETURN @success
END
GO

-- UPDATE TEAM LEAD
DROP PROCEDURE IF EXISTS updateTeamLead
GO
-- Function Update an Employee the Team Lead
CREATE PROCEDURE updateTeamLead(@teamName varchar(50), @employeefName
varchar(50),@employeelName varchar(50)) AS
BEGIN
       --
       DECLARE @teamID int, @employeeID int, @success int
       SET @success = 0
       -- Validate that the Employee info entered is an existing Employee in the database
       EXEC @success = dbo.validateEmployee @employeefName, @employeelName
       -- IF a valid employee wasn't found, exit the procedure with an error
       IF @success <> 1
              RETURN @success

       SELECT @teamID = team_id FROM team WHERE team_name = @teamName
       SELECT @employeeID = employee_id FROM employee WHERE first_name = @employeefName AND
last_name = @employeelName

       -- Update row to team_lead table
       UPDATE team_lead
       SET employee_id = @employeeID
       WHERE team_id = @teamID

       RETURN @success
END
GO


-- ADD Employee to a Team
DROP PROCEDURE IF EXISTS addTeamEmployee

GO
CREATE PROCEDURE addTeamEmployee(@teamName varchar(50), @employeefName
varchar(50),@employeelName varchar(50)) AS
BEGIN
       DECLARE @teamID int, @employeeID int, @success int, @successTeam int, @successEmployee
int
       SET @success = 1
       -- Validate that the Employee info entered is an existing Employee in the database
```

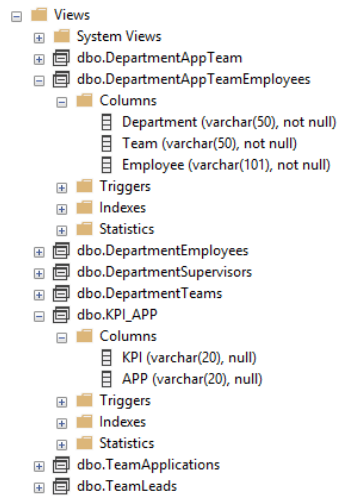
```
	EXEC @successEmployee = dbo.validateEmployee @employeefName, @employeelName
	EXEC @successTeam = dbo.validateTeam @teamName

	-- IF a valid employee wasn't found or a valid team wasn't found, exit the procedure
with an error
	IF (@successEmployee != 1 OR @successTeam != 1)
		BEGIN
		SET @success = 0
		RETURN @success
		END
	ELSE
		BEGIN
			SELECT @teamID = team_id FROM team WHERE team_name = @teamName
			SELECT @employeeID = employee_id FROM employee WHERE first_name =
@employeefName AND last_name = @employeelName

			-- Add row to team_employee table
			INSERT INTO team_employee(team_id,employee_id)
				VALUES (@teamID, @employeeID)

			SELECT @success = ISNULL((SELECT 1 FROM team_employee WHERE
team_employee_id = @@identity), 0)
		END

	RETURN @success
END
GO

-- ADD APPLICATION TO A TEAM
DROP PROCEDURE IF EXISTS addTeamApplication
GO
CREATE PROCEDURE addTeamApplication(@teamName varchar(50), @departmentName varchar(50),
@appName varchar(20)) AS
BEGIN
	DECLARE @success int, @teamID int, @departmentID int, @appID int

	BEGIN TRY
	SELECT @teamID = team_id FROM team WHERE team_name = @teamName AND department_id =
(Select department_id FROM department WHERE department_name = @departmentName)
	SELECT @appID = application_id FROM application WHERE app_name = @appName

		INSERT INTO team_application(team_id,application_id)
			VALUES (@teamID,@appID)
	SET @success = 1
	END TRY
	BEGIN CATCH
		PRINT(ERROR_NUMBER())
		PRINT(ERROR_MESSAGE())
		SET @success = 0
	END CATCH

	RETURN @success
END
GO
```

```
-- ADD APPLICATION KPI
DROP PROCEDURE IF EXISTS addApplicationKPI
GO
CREATE PROCEDURE addApplicationKPI(@appName varchar(20), @kpiName varchar(20)) AS
BEGIN

	DECLARE @success int, @appID int, @kpiID int
	SET @success = 0

	BEGIN TRY
		SELECT @appID = application_id FROM application WHERE app_name = @appName
		PRINT(@appID)
		SELECT @kpiID = kpi_id FROM kpi WHERE kpi_name = @kpiName
		PRINT(@kpiID)

		INSERT INTO application_kpi(application_id, kpi_id)
			VALUES(@appID,@kpiID)
		SET @success = 1
	END TRY

	BEGIN CATCH
		PRINT(ERROR_NUMBER())
		PRINT(ERROR_MESSAGE())
		SET @success = 0
	END CATCH

	RETURN @success

END
GO
```

5.4 Object – Views

5.4.1 *Screen Print Ex:*





5.4.2 *SQL DDL Script*

```
/*
	IST 659 Data Admin Concepts &Db Mgmt
	Date: 9/30/2018
	Project Deliverable 2: Views

*/

-- Show Employee's by Organization
-- Drop VIEW if it exists
IF EXISTS(SELECT * FROM sys.views WHERE name = 'DepartmentEmployees' AND schema_id =
SCHEMA_ID('dbo'))
	DROP VIEW DepartmentEmployees

GO
CREATE VIEW DepartmentEmployees AS

	SELECT
		department_name AS Department, team_name AS APP_Team, CONCAT(first_name, ' ',
last_name) AS Employee
	FROM team_employee
	JOIN team on team.team_id = team_employee.team_id
	JOIN employee on employee.employee_id = team_employee.employee_id
	JOIN department on department.department_id = team.department_id
	ORDER BY Department ASC OFFSET 0 ROWS
GO
-- Test the vc_MostProlificUsers VIEW
SELECT * FROM DepartmentEmployees
```

```
-- Show Applications By Team
-- Drop VIEW if it exists
IF EXISTS(SELECT * FROM sys.views WHERE name = 'TeamApplications' AND schema_id =
SCHEMA_ID('dbo'))
	DROP VIEW TeamApplications

GO
CREATE VIEW TeamApplications AS
	SELECT
		team_name AS Team, app_name AS APP
	FROM team_application
	JOIN team on team.team_id = team_application.team_id
	JOIN application on application.application_id = team_application.application_id
	ORDER BY TEAM ASC OFFSET 0 ROWS

GO
-- TEST VIEW
SELECT * FROM TeamApplications

-- Department Teams
-- Drop VIEW if it exists
IF EXISTS(SELECT * FROM sys.views WHERE name = 'DepartmentTeams' AND schema_id =
SCHEMA_ID('dbo'))
	DROP VIEW DepartmentTeams

GO
CREATE VIEW DepartmentTeams AS
	SELECT
		department_name AS Department, team_name AS Team
		FROM team_employee
		JOIN team on team.team_id = team_employee.team_id
		JOIN department on department.department_id = team.department_id
		ORDER BY Department ASC OFFSET 0 ROWS

GO
-- TEST VIEW
SELECT * FROM DepartmentTeams



-- Listing Department, App, Team
-- Drop VIEW if it exists
IF EXISTS(SELECT * FROM sys.views WHERE name = 'DepartmentAppTeamEmployees' AND schema_id =
SCHEMA_ID('dbo'))
	DROP VIEW DepartmentAppTeamEmployees

GO
CREATE VIEW DepartmentAppTeamEmployees AS
	SELECT
		department_name AS Department, team_name AS Team, CONCAT(first_name, ' ',
last_name) AS Employee
		FROM team_employee
		JOIN team on team.team_id = team_employee.team_id
```

```
                JOIN employee on employee.employee_id = team_employee.employee_id
                JOIN department on department.department_id = team.department_id
                ORDER BY Department ASC OFFSET 0 ROWS

GO
-- TEST VIEW
SELECT * FROM DepartmentAppTeamEmployees

-- Listing KPI by App
-- Drop VIEW if it exists
IF EXISTS(SELECT * FROM sys.views WHERE name = 'KPI_APP' AND schema_id = SCHEMA_ID('dbo'))
       DROP VIEW KPI_APP

GO
CREATE VIEW KPI_APP AS
       SELECT
              kpi_name AS KPI, app_name as APP
       FROM application_kpi
       LEFT OUTER JOIN kpi on kpi.kpi_id = application_kpi.kpi_id
       LEFT JOIN application on application.application_id = application_kpi.application_id
       ORDER BY KPI ASC OFFSET 0 ROWS
GO
-- TEST VIEW
SELECT * FROM KPI_APP

-- List Team Leads by Team
IF EXISTS(SELECT * FROM sys.views WHERE name = 'TeamLeads' AND schema_id = SCHEMA_ID('dbo'))
       DROP VIEW TeamLeads

GO
CREATE VIEW TeamLeads AS
       SELECT
       team_name AS Team, CONCAT(first_name, ' ', last_name) AS Team_Lead
       FROM team_lead
       JOIN team on team.team_id = team_lead.team_id
       JOIN employee on employee.employee_id = team_lead.employee_id
GO
-- TEST VIEW
SELECT * FROM TeamLeads

-- Department Supervisors
-- List Team Leads by Team
IF EXISTS(SELECT * FROM sys.views WHERE name = 'DepartmentSupervisors' AND schema_id =
SCHEMA_ID('dbo'))
       DROP VIEW DepartmentSupervisors

GO
CREATE VIEW DepartmentSupervisors AS
       SELECT
       department_name AS Department, CONCAT(first_name, ' ', last_name) AS
Department_Supervisor
       FROM department_supervisor
       JOIN department on department.department_id = department_supervisor.department_id
       JOIN employee on employee.employee_id = department_supervisor.employee_id
```

```
GO
-- TEST VIEW

SELECT * FROM DepartmentSupervisors
```

6 Data Load SQL DML INSERT

6.1 Description

Insert statements and data manipulation statements such as Updates

6.2 INSERT

6.2.1 *Screen Print Ex:*

Results | Messages

	employee_id	first_name	first_name2	last_name	title	phone	email	cc_preference_id	is_active
1	1	Ryan	NULL	Timbrook	Manager, Software Development	206-516-9956	Ryan.Timbrook1@T-Mobile.com	1	T
2	2	Kristy	NULL	Gillis	Sr Engineer, Software	612-799-8100	Kristy.Gillis1@T-Mobile.com	2	T
3	3	Edjan	NULL	Preut	Sr Engineer, Systems Reliability	206-696-0656	Edjan.Preut@T-Mobile.com	3	T
4	4	Phoebe	NULL	Parsons	Engineer, Software	425-499-7542	Phoebe.Parsons2@T-Mobile.com	3	T
5	5	Paul	NULL	James	Sr Analyst, Technical	206-819-9955	Paul.James@T-Mobile.com	1	T
6	6	David	NULL	Marshall	Manager, Project Management Technical	206-226-9532	David.Marshall@T-Mobile.com	1	T
7	7	Rob	NULL	Stamm	Sr Manager, Software Development	425-383-2919	Robert.Stamm@T-Mobile.com	3	T
8	8	Leigh	NULL	Gower	Sr Director, Software Development	206-940-0657	Leigh.Gower@T-Mobile.com	3	T
9	9	Matthew	NULL	Davis	VP, IT Development	425-383-6242	Matthew.Davis@T-Mobile.com	3	T
10	10	Trevor	NULL	Malmos	Sr Manager, Product Management	425-383-5309	Trevor.Malmos@T-Mobile.com	1	T
11	11	Ryan	NULL	Yokal	Sr Manager, Product Management	425-383-8366	Ryan.Yokel@T-mobile.com	3	T

Results | Messages

	kpi_id	kpi_name	description	url_endpoint	kpi_type_id	is_active
1	1	CPA	Frontline calls per account	https://dummy.com/kpi/care/voice/cpa/1	1	T
2	2	CPC	Frontline inbound calls per customer	https://dummy.com/kpi/care/voice/cpc/1	1	T
3	3	Call Length	duration of customer inbound call	https://dummy.com/kpi/care/voice/cl/1	1	T
4	4	Transfer Percent	percent of inbound customers transferred to an a...	https://dummy.com/kpi/care/voice/tp/1	1	T
5	5	Deflection Rate	SIVR inbound calls contained within the system	https://dummy.com/kpi/care/voice/dr/1	1	T
6	6	CSAT	Customer Satisfaction Score	https://dummy.com/kpi/care/voice/csat/1	1	T
7	7	CPH	Messenging Conversations Per Hour	https://dummy.com/kpi/care/messaging/cph/1	1	T
8	8	TTFR	Messenging Time to First Response by Care	https://dummy.com/kpi/care/messaging/ttfr/1	1	T
9	9	VOC	Voice of the Customer	https://dummy.com/kpi/care/voc/1	1	T

6.2.2 *SQL DDL Script*

```
/*
       IST 659 Data Admin Concepts &Db Mgmt
       Date: 9/30/2018
       Project Deliverable 2: INSERTS

*/

-- DELETE all the records from contact_channel_preference
--DELETE FROM contact_channel_preference
-- 1: Add Rows to the Contact Channel Preference Table
INSERT INTO contact_channel_preference(cc_preference_type)
       VALUES('EMAIL'),('PHONE'),('SMS')


-- Test Insert
SELECT * FROM contact_channel_preference
```

```
-- DELETE all the records then insert new set
--DELETE FROM employee
-- 2: Add Rows to the Employee Table
INSERT INTO employee(first_name,last_name,title,phone,email,cc_preference_id)
       VALUES
              ('Ryan','Timbrook','Manager, Software Development','206-516-
9956','Ryan.Timbrook1@T-Mobile.com',(SELECT cc_preference_id FROM contact_channel_preference
WHERE cc_preference_type='EMAIL')),
              ('Kristy','Gillis','Sr Engineer, Software','612-799-8100','Kristy.Gillis1@T-
Mobile.com',(SELECT cc_preference_id FROM contact_channel_preference WHERE
cc_preference_type='PHONE')),
              ('Edjan','Preut','Sr Engineer, Systems Reliability','206-696-
0656','Edjan.Preut@T-Mobile.com',(SELECT cc_preference_id FROM contact_channel_preference
WHERE cc_preference_type='SMS')),
              ('Phoebe','Parsons','Engineer, Software','425-499-7542','Phoebe.Parsons2@T-
Mobile.com',(SELECT cc_preference_id FROM contact_channel_preference WHERE
cc_preference_type='SMS')),
              ('Paul','James','Sr Analyst, Technical','206-819-9955','Paul.James@T-
Mobile.com',(SELECT cc_preference_id FROM contact_channel_preference WHERE
cc_preference_type='EMAIL')),
              ('David','Marshall','Manager, Project Management Technical','206-226-
9532','David.Marshall@T-Mobile.com',(SELECT cc_preference_id FROM contact_channel_preference
WHERE cc_preference_type='EMAIL')),
              ('Rob','Stamm','Sr Manager, Software Development','425-383-
2919','Robert.Stamm@T-Mobile.com',(SELECT cc_preference_id FROM contact_channel_preference
WHERE cc_preference_type='SMS')),
              ('Leigh','Gower','Sr Director, Software Development','206-940-
0657','Leigh.Gower@T-Mobile.com',(SELECT cc_preference_id FROM contact_channel_preference
WHERE cc_preference_type='SMS')),
              ('Matthew','Davis','VP, IT Development','425-383-6242','Matthew.Davis@T-
Mobile.com',(SELECT cc_preference_id FROM contact_channel_preference WHERE
cc_preference_type='SMS')),
              ('Trevor','Malmos','Sr Manager, Product Management','425-383-
5309','Trevor.Malmos@T-Mobile.com',(SELECT cc_preference_id FROM contact_channel_preference
WHERE cc_preference_type='EMAIL')),
              ('Ryan','Yokal','Sr Manager, Product Management','425-383-8366','Ryan.Yokel@T-
mobile.com',(SELECT cc_preference_id FROM contact_channel_preference WHERE
cc_preference_type='SMS'))

-- Test Insert
SELECT * FROM employee

-- 3: Add Rows to the department table
INSERT INTO department(department_name, description)
       VALUES
              ('Voice Services','Contact Center, Inbound Voice IVR Development Team'),
              ('SMPD','Social Media Product Development'),
              ('Connected Customer','Combined SMPD and Voice Services Teams'),
              ('Digital Business Connected Customer','Combined Connected Customer Teams and
Digital Business Teams'),
              ('Frontline Care','Parent Organization of all Frontline Care Applications'),
              ('B2B And Commissions','Business to Business Teams and Commissions Teams')
-- Test department
SELECT * FROM department
```

```
-- Update Department Parent Child Relationship
UPDATE department
SET department_parent_id = (SELECT department_id FROM department WHERE department_name =
'Connected Customer')
WHERE department_name = 'Voice Services'

UPDATE department
SET department_parent_id = (SELECT department_id FROM department WHERE department_name =
'Connected Customer')
WHERE department_name = 'SMPD'

UPDATE department
SET department_parent_id = (SELECT department_id FROM department WHERE department_name =
'Digital Business Connected Customer')
WHERE department_name = 'Connected Customer'

UPDATE department
SET department_parent_id = (SELECT department_id FROM department WHERE department_name =
'Digital Business Connected Customer')
WHERE department_name = 'B2B And Commissions'

UPDATE department
SET department_parent_id = (SELECT department_id FROM department WHERE department_name =
'Frontline Care')
WHERE department_name = 'Digital Business Connected Customer'

UPDATE department
SET department_parent_id = (SELECT department_id FROM department WHERE department_name =
'Frontline Care')
WHERE department_name = 'Frontline Care'

SELECT * FROM department


-- 4: Add Rows to the role table
INSERT INTO role(role_name, description)
       VALUES
              ('VP, IT Development','Vice Precident of IT Development'),
              ('Sr Director, Software Development','Senior Director of Software
Development'),
              ('Director, Software Development','Director of Software Development'),
              ('Sr Manager, Software Development','Senior Manager of Software Development'),
              ('Manager, Software Development','Manager of Software Development'),
              ('Sr Manager, Product Management','Senior Manager of Product Management'),
              ('Tech. Product Owner','Technical Product Owner'),
              ('AppOps Engineer','Application Operations Engineer'),
              ('Scrum Master','Agile Software Delivery Scrum Master'),
              ('Tech. Delivery Mgr','Technical Delivery Manager'),
              ('Technical Analyst','Technical Systems Analyst'),
              ('Software Engineer','Software Engineer'),
              ('SDET','Software Engineer in Test')
-- Test role
SELECT * FROM role
```

```sql
-- 5: Add Rows to the team table
INSERT INTO team(team_name, description, department_id)
       VALUES
              ('Voice Portal Product','SIVR Business Product Owner Team',(SELECT
department_id FROM department WHERE department_name='Voice Services')),
              ('Voice Portal Delivery','SIVR DevOps Software Delivery Team',(SELECT
department_id FROM department WHERE department_name='Voice Services')),
              ('Voice Services Gamma Delivery','SIVR Agile DevTeam that supports the SIVR
Postpaid voice experience application',(SELECT department_id FROM department WHERE
department_name='Voice Services')),
              ('Voice Services Gamma Support','SIVR Kanban Non-Dev team that supports the
SIVR Postpaid voice experience application',(SELECT department_id FROM department WHERE
department_name='Voice Services')),
              ('Voice Services Gamma AppOps','SIVR Application Opperations Production Support
team that supports the SIVR Postpaid voice',(SELECT department_id FROM department WHERE
department_name='Voice Services')),
              ('Voice Services Omega Delivery ','SIVR Agile DevTeam that supports the SIVR
Prepaid voice experience application',(SELECT department_id FROM department WHERE
department_name='Voice Services')),
              ('Voice Services Omega Support','SIVR Kanban Non-Dev team that supports the
SIVR Prepaid voice experience application',(SELECT department_id FROM department WHERE
department_name='Voice Services')),
              ('Voice Services Omega AppOps','SIVR Application Opperations Production Support
team that supports the SIVR Prepaid voice experience ',(SELECT department_id FROM department
WHERE department_name='Voice Services')),
              ('SpaceBear','SMPD DevOps Team, Focus area is Auth, Shopping, iFrame',(SELECT
department_id FROM department WHERE department_name='SMPD')),
              ('HuggyBear','SMPD DevOps Team, Focus area is OPEX',(SELECT department_id FROM
department WHERE department_name='SMPD'))

-- Test team
SELECT * FROM team
-- Update Team Parent Child Relationship
UPDATE team
SET team_parent_id = (SELECT team_id FROM team WHERE team_name = 'Voice Portal Product')
WHERE team_name = 'Voice Portal Product'

UPDATE team
SET team_parent_id = (SELECT team_id FROM team WHERE team_name = 'Voice Portal Product')
WHERE team_name = 'Voice Portal Delivery'

UPDATE team
SET team_parent_id = (SELECT team_id FROM team WHERE team_name = 'Voice Portal Delivery')
WHERE team_name = 'Voice Services Gamma Delivery'

UPDATE team
SET team_parent_id = (SELECT team_id FROM team WHERE team_name = 'Voice Services Gamma
Delivery')
WHERE team_name = 'Voice Services Gamma Support'

UPDATE team
SET team_parent_id = (SELECT team_id FROM team WHERE team_name = 'Voice Services Gamma
Delivery')
```

```
WHERE team_name = 'Voice Services Gamma AppOps'

UPDATE team
SET team_parent_id = (SELECT team_id FROM team WHERE team_name = 'Voice Portal Delivery')
WHERE team_name = 'Voice Services Omega Delivery'

UPDATE team
SET team_parent_id = (SELECT team_id FROM team WHERE team_name = 'Voice Services Omega
Delivery')
WHERE team_name = 'Voice Services Omega Support'

UPDATE team
SET team_parent_id = (SELECT team_id FROM team WHERE team_name = 'Voice Services Omega
Delivery')
WHERE team_name = 'Voice Services Omega AppOps'

UPDATE team
SET team_parent_id = (SELECT team_id FROM team WHERE team_name = 'SpaceBear')
WHERE team_name = 'SpaceBear'

UPDATE team
SET team_parent_id = (SELECT team_id FROM team WHERE team_name = 'HuggyBear')
WHERE team_name = 'HuggyBear'

SELECT * FROM team

-- 6: Add Rows to the application table

INSERT INTO application(app_name, description)
       VALUES
              ('SIVR Postpaid','Speech Self-Service Application supporting the Postpaid
product'),
              ('SIVR Prepaid','Speech Self-Service Application supporting the Prepaid
product'),
              ('SIVR U2 Prepaid','Speech Self-Service Application supporting the U2 Prepaid
product'),
              ('Lithium','SMPD - Social Messaging (Twitter, Facebook)'),
              ('Live Engage','SMPD - Messaging (Async, Sync, iMessage, SMS)'),
              ('CCS','Commissions')

-- Test application
SELECT * FROM application

-- 7: Add Rows to the kpi_type table
INSERT INTO kpi_type(kpi_type, description)
       VALUES
              ('Quantitative','indicators that can be presented with a number'),
              ('Qualitative','indicators that cant be presented as a number'),
              ('Leading','indicators that can predict the outcome of a process'),
              ('Lagging','indicators that present the success or failure post hoc'),
              ('Input','indicators that measure the amount of resources consumed during the
generation of the outcome'),
              ('Process','indicators that represent the efficiency or the productivity of the
process'),
```

```
              ('Output','indicators that reflect the outcome or results of the process
activities'),
              ('Practical','indicators that interface with existing company processes'),
              ('Directional','indicators specifying whether or not an organization is getting
better'),
              ('Actionable','indicators are sufficiently in an organizations control to
effect change'),
              ('Financial','indicators used in performance measurement and when looking at an
operating index')

-- Test kpi_type
SELECT * FROM kpi_type

-- 8: Add Rows to the kpi table
INSERT INTO kpi(kpi_name,description,url_endpoint,kpi_type_id)
       VALUES
              ('CPA','Frontline calls per
account','https://dummy.com/kpi/care/voice/cpa/1',(SELECT kpi_type_id from kpi_type WHERE
kpi_type='Quantitative')),
              ('CPC','Frontline inbound calls per
customer','https://dummy.com/kpi/care/voice/cpc/1',(SELECT kpi_type_id from kpi_type WHERE
kpi_type='Quantitative')),
              ('Call Length','duration of customer inbound
call','https://dummy.com/kpi/care/voice/cl/1',(SELECT kpi_type_id from kpi_type WHERE
kpi_type='Quantitative')),
              ('Transfer Percent','percent of inbound customers transferred to an
agent','https://dummy.com/kpi/care/voice/tp/1',(SELECT kpi_type_id from kpi_type WHERE
kpi_type='Quantitative')),
              ('Deflection Rate','SIVR inbound calls contained within the
system','https://dummy.com/kpi/care/voice/dr/1',(SELECT kpi_type_id from kpi_type WHERE
kpi_type='Quantitative')),
              ('CSAT','Customer Satisfaction
Score','https://dummy.com/kpi/care/voice/csat/1',(SELECT kpi_type_id from kpi_type WHERE
kpi_type='Quantitative')),
              ('CPH','Messenging Conversations Per
Hour','https://dummy.com/kpi/care/messaging/cph/1',(SELECT kpi_type_id from kpi_type WHERE
kpi_type='Quantitative')),
              ('TTFR','Messenging Time to First Response by
Care','https://dummy.com/kpi/care/messaging/ttfr/1',(SELECT kpi_type_id from kpi_type WHERE
kpi_type='Quantitative')),
              ('VOC','Voice of the Customer','https://dummy.com/kpi/care/voc/1',(SELECT
kpi_type_id from kpi_type WHERE kpi_type='Quantitative'))

-- Test kpi
SELECT * FROM kpi

-- 9: Add Rows to the product table
INSERT INTO product(product_name, description)
       VALUES
              ('SIVR Postpaid','Speech Self-Service IVR System for the Postpaid Customer'),
              ('SIVR Prepaid','Speech Self-Service IVR System for the Prepaid Customer'),
              ('SIVR U2 Prepaid','Speech Self-Service IVR System for the U2 Prepaid
Customer'),
              ('Messaging','Social Messaging Channel Connecting Customers to Care Agents
```

```
through SMS and Social Technology Channels'),
              ('Commissions','Commissions payment systems for Retaila and Care Agents')
-- Test product
SELECT * FROM product

-- 10: Add Rows to the capability table
INSERT INTO capability(capability_name, description)
       VALUES
              ('Make a Payment','Automation self-service feature allowing a customer to make
bill payment'),
              ('Make Payment Arrangements','Automation self-service feature allowing a
customer to configure automated scheduled payments to a bill'),
              ('Usage','Automated self-service feature allowing a customer to make inquires
about their phone usage'),
              ('Account Balance Lookup','Automated self-service feature allowing a customer
to lookup their account balance'),
              ('Order Status','Automated self-service feature allowing a customer to inquire
about the status of their purchase order'),
              ('Store Locator','Automated self-service feature allowing a customer to inquire
location of retail stores'),
              ('PIN Validation','Automated self-service feature allowing a customer to set or
update their security PIN'),
              ('Authentication','Automated self-service feature allowing a customer to
authenticate themselves'),
              ('Add A Line','Automated self-service feature allowing a customer to add
additional lines to their service')

-- Test capability
SELECT * FROM capability
```

6.3 EXECUTE Procedures

6.3.1 *Screen Print Ex:*

```
45  SELECT * FROM team_lead
46  -- Load Team Leads
47  EXEC addTeamLead 'Voice Portal Product','Trevor','Malmos'
48  EXEC addTeamLead 'Voice Portal Delivery','Ryan','Timbrook'
49  EXEC addTeamLead 'Voice Services Gamma Delivery','Kristy','Gillis'
50  EXEC addTeamLead 'Voice Services Gamma Support','David','Marshall'
51  EXEC addTeamLead 'Voice Services Gamma AppOps','Edjan','Preut'
52  EXEC addTeamLead 'Voice Services Omega Delivery ','Kristy','Gillis'
53  EXEC addTeamLead 'Voice Services Omega Support','David','Marshall'
54  EXEC addTeamLead 'Voice Services Omega AppOps','Edjan','Preut'
55  EXEC addTeamLead 'SpaceBear','Rob','Stamm'
56  EXEC addTeamLead 'HuggyBear','Ryan','Yokal'
57
58  SELECT * FROM team_lead
```

00 %

Results | Messages

	team_lead_id	team_id	employee_id	is_active
1	1	1	10	T
2	2	2	1	T
3	3	3	2	T
4	4	4	6	T
5	5	5	3	T
6	6	6	6	T
7	7	7	2	T
8	8	8	3	T
9	9	9	7	T
10	10	10	11	T

6.3.2 *SQL DDL Script*

```
/*
       IST 659 Data Admin Concepts &Db Mgmt
       Date: 9/30/2018
       Project Deliverable 2: Execute Procedures

*/

-- Load Department Supervisiors
EXEC addDepartmentSupervisior 'Voice Services', 'Ryan', 'Timbrook'
EXEC addDepartmentSupervisior 'SMPD', 'Rob', 'Stamm'
EXEC addDepartmentSupervisior 'Connected Customer', 'Rob', 'Stamm'
EXEC addDepartmentSupervisior 'Digital Business Connected Customer', 'Leigh', 'Gower'
EXEC addDepartmentSupervisior 'Frontline Care', 'Matthew', 'Davis'

SELECT * FROM department_supervisor

-- Update A Department Supervisior
DECLARE @newID int
EXEC @newID = updateDepartmentSupervisior 'Voice Services', 'Ryan', 'Timbrook'
SELECT * FROM department_supervisor WHERE department_supervisor_id = @newID
select * from employee
SELECT * FROM department_supervisor
```

```
-- End Test Execution

SELECT * from team_employee
-- Load Team Leads to Team Employee Table- Must be executed before Team Lead
EXEC addTeamEmployee 'Voice Portal Product','Trevor','Malmos'
EXEC addTeamEmployee 'Voice Portal Delivery','Ryan','Timbrook'
EXEC addTeamEmployee 'Voice Services Gamma Delivery','Kristy','Gillis'
EXEC addTeamEmployee 'Voice Services Gamma Support','David','Marshall'
EXEC addTeamEmployee 'Voice Services Gamma AppOps','Edjan','Preut'
EXEC addTeamEmployee 'Voice Services Omega Delivery ','Kristy','Gillis'
EXEC addTeamEmployee 'Voice Services Omega Support','David','Marshall'
EXEC addTeamEmployee 'Voice Services Omega AppOps','Edjan','Preut'
EXEC addTeamEmployee 'SpaceBear','Rob','Stamm'
EXEC addTeamEmployee 'HuggyBear','Ryan','Yokal'
-- Load Employees to Team Employee Table
-- TEAM -> 'Voice Services Gamma Delivery' Members
EXEC addTeamEmployee 'Voice Services Gamma Delivery', 'Paul', 'James'
-- TEAM -> 'Voice Services Omega Delivery' Members
EXEC addTeamEmployee 'Voice Services Omega Delivery','Phoebe','Parsons'

SELECT * FROM team_employee

SELECT * FROM team_lead
-- Load Team Leads
EXEC addTeamLead 'Voice Portal Product','Trevor','Malmos'
EXEC addTeamLead 'Voice Portal Delivery','Ryan','Timbrook'
EXEC addTeamLead 'Voice Services Gamma Delivery','Kristy','Gillis'
EXEC addTeamLead 'Voice Services Gamma Support','David','Marshall'
EXEC addTeamLead 'Voice Services Gamma AppOps','Edjan','Preut'
EXEC addTeamLead 'Voice Services Omega Delivery ','Kristy','Gillis'
EXEC addTeamLead 'Voice Services Omega Support','David','Marshall'
EXEC addTeamLead 'Voice Services Omega AppOps','Edjan','Preut'
EXEC addTeamLead 'SpaceBear','Rob','Stamm'
EXEC addTeamLead 'HuggyBear','Ryan','Yokal'

SELECT * FROM team_lead

-- Update a Team Lead
EXEC updateTeamLead 'Voice Services Omega Delivery','David','Marshall'
EXEC updateTeamLead 'Voice Services Omega Support','Kristy','Gillis'

SELECT * FROM team_lead
--

-- Load Team Applications team, department, application
EXEC addTeamApplication 'Voice Services Omega AppOps','Voice Services','SIVR Prepaid'
EXEC addTeamApplication 'Voice Services Omega Support','Voice Services','SIVR Prepaid'
EXEC addTeamApplication 'Voice Services Omega AppOps','Voice Services','SIVR U2 Prepaid'
EXEC addTeamApplication 'Voice Services Omega Support','Voice Services','SIVR U2 Prepaid'
EXEC addTeamApplication 'Voice Services Gamma AppOps','Voice Services','SIVR Postpaid'
EXEC addTeamApplication 'Voice Services Gamma Support','Voice Services','SIVR Postpaid'
EXEC addTeamApplication 'SpaceBear','SMPD','Lithium'
EXEC addTeamApplication 'HuggyBear','SMPD','Live Engage'
```

```sql
SELECT * FROM team_application

-- Load Application KPIs
-- APP -> SIVR Postpaid
EXEC addApplicationKPI 'SIVR Postpaid', 'Deflection Rate'
EXEC addApplicationKPI 'SIVR Postpaid', 'CPC'
EXEC addApplicationKPI 'SIVR Postpaid', 'Call Length'
EXEC addApplicationKPI 'SIVR Postpaid', 'Transfer Percent'

-- APP -> SIVR Prepaid
EXEC addApplicationKPI 'SIVR Prepaid', 'Deflection Rate'
EXEC addApplicationKPI 'SIVR Prepaid', 'Call Length'
EXEC addApplicationKPI 'SIVR Prepaid', 'VOC'

-- APP -> SIVR U2 Prepaid
EXEC addApplicationKPI 'SIVR U2 Prepaid', 'Deflection Rate'
EXEC addApplicationKPI 'SIVR U2 Prepaid', 'Call Length'
EXEC addApplicationKPI 'SIVR U2 Prepaid', 'Transfer Percent'

-- APP -> Lithium
EXEC addApplicationKPI 'Lithium', 'CPH'
EXEC addApplicationKPI 'Lithium', 'TTFR'
EXEC addApplicationKPI 'Lithium', 'CSAT'

-- APP -> Live Engage
EXEC addApplicationKPI 'Live Engage', 'TTFR'
EXEC addApplicationKPI 'Live Engage', 'CSAT'

select * from application_kpi
```

7 Answering Data Questions

7.1 Description

Answer specific data questions presented at the introduction section

7.2 Question 1

As a Department Supervisor, what Application KPIs do I Own?

7.2.1 *SQL SELECT Output Screen Print*

	Department
1	Voice Services

	Team
1	Voice Portal Product
2	Voice Portal Delivery
3	Voice Services Gamma Delivery
4	Voice Services Gamma Delivery
5	Voice Services Gamma Support
6	Voice Services Gamma AppOps
7	Voice Services Omega Delivery
8	Voice Services Omega Delivery

	APP
1	SIVR Postpaid
2	SIVR Postpaid
3	SIVR Prepaid
4	SIVR U2 Prepaid
5	SIVR Prepaid
6	SIVR U2 Prepaid

	KPI
1	Call Length
2	CPC
3	Deflectio...
4	Transfer ...
5	VOC

7.2.2 *SQL SELECT STATEMENTS*

```
SELECT Department FROM DepartmentSupervisors WHERE Department_Supervisor = 'Ryan Timbrook'
SELECT Team FROM DepartmentTeams WHERE Department = 'Voice Services'
SELECT APP FROM TeamApplications WHERE Team in(SELECT Team FROM DepartmentTeams WHERE
Department = 'Voice Services')
```

```
SELECT DISTINCT KPI FROM KPI_APP WHERE APP in(SELECT APP FROM TeamApplications WHERE Team
in(SELECT Team FROM DepartmentTeams WHERE Department = 'Voice Services'))
```

7.3 Question 2

As a Team Lead, what Application KPIs do I Own?

7.3.1 *SQL SELECT Output Screen Print*

Results | Messages

	Team
1	Voice Services Gamma Delivery
2	Voice Services Omega Support

	APP
1	SIVR Prepaid
2	SIVR U2 Prepaid

	KPI
1	Call Length
2	Deflection Rate
3	Transfer Percent
4	VOC

7.3.2 *SQL SELECT STATEMENTS*

```
-- As a Team Lead, what Application KPIs do I Own?
SELECT Team FROM TeamLeads WHERE Team_Lead = 'Kristy Gillis'
SELECT APP FROM TeamApplications WHERE Team in(SELECT Team FROM TeamLeads WHERE Team_Lead =
'Kristy Gillis')
SELECT DISTINCT KPI FROM KPI_APP WHERE APP in(SELECT APP FROM TeamApplications WHERE Team
in(SELECT Team FROM TeamLeads WHERE Team_Lead = 'Kristy Gillis'))
```

7.4 Question 3

Who are the Department Supervisors?

7.4.1 *SQL SELECT Output Screen Print*

Results | Messages

	Department	Department_Supervisor
1	Voice Services	Ryan Timbrook
2	SMPD	Rob Stamm
3	Connected Customer	Rob Stamm
4	Digital Business Connected Customer	Leigh Gower
5	Frontline Care	Matthew Davis

7.4.2 *SQL SELECT STATEMENTS*

```
SELECT * FROM DepartmentSupervisors
```

7.5 Question 4

Who are the Department Employees?

7.5.1 *SQL SELECT Output Screen Print*

Results | Messages

	Department	Department_Supervisor
1	Voice Services	Ryan Timbrook
2	SMPD	Rob Stamm
3	Connected Customer	Rob Stamm
4	Digital Business Connected Customer	Leigh Gower
5	Frontline Care	Matthew Davis

7.5.2 *SQL SELECT STATEMENTS*

```
SELECT * FROM DepartmentEmployees
```