MOCK HANDSON ANSWERS

```
--Table Name : Doctor
create table Doctor
(
DoctorID int primary key identity(1001,1),
DoctorName varchar(25) not null
)

--Insert into Doctor
insert into Doctor values('Susan')

select * from Doctor

--Table Name : Specialization
Create table Specialization
(
SpecializationCode char(3) primary key,
SpecializationName varchar(20) not null
)

--Insert into Specialization
insert into Specialization values('GYN','Gynecologiest')
insert into Specialization values('CAR','Cardiologiest')
insert into Specialization values('ANE','Anesthesiologiest')

select * from Specialization


--Table Name : DoctorSpecialization
create table DoctorSpecialization
(
DoctorId int constraint fk_DoctorId references Doctor(DoctorId),
SpecializationCode char(3) constraint fk_SpecializationCode references
Specialization(SpecializationCode),
SpecializationDate date not null,
constraint pk_DoctorSpeId primary key(DoctorId,SpecializationCode)
)

--Insert into DoctorSpecialization
insert into DoctorSpecialization values(1001,'ANE','2010-01-01')
insert into DoctorSpecialization values(1002,'CAR','2010-01-01')
insert into DoctorSpecialization values(1003,'CAR','2010-01-01')

select * from DoctorSpecialization


--Table Name : Surgery
create table Surgery
(
SurgeryId int Primary key identity(5000,1),
DoctorId int constraint fk_DctorId references Doctor(DoctorID),
SurgeryDate date not null,
StartTime decimal(4,2) not null,
EndTime decimal(4,2) not null,
```

```
SurgeryCategory char(3) constraint fk_SurgeryCategory references
Specialization(SpecializationCode)
)

--Insert into Surgery
insert into Surgery values(1001,'2011-01-01',09.00,14.00,'ANE')

insert into Surgery values(1002,'2015-01-01',10.00,16.00,'CAR')

select * from Surgery
```

--1)Procedure 1 Answer

```
alter procedure usp_AddSurgeryDetails
(
@DoctorId int,
@SurgeryDate date,
@StartTime decimal(4,2),
@EndTime decimal(4,2),
@SurgeryCategory char(3),
@SurgeryId int out
)
as
begin
begin try

if not exists(select DoctorID from Doctor where DoctorID=@DoctorId)
return -1

if not exists(select [DoctorId],SpecializationCode from [dbo].[DoctorSpecialization]
                                    where [DoctorId]=@DoctorId and
SpecializationCode=@SurgeryCategory)
return -2

if not exists(select StartTime from Surgery where StartTime<@StartTime)
return -3
if not exists(select EndTime from Surgery where EndTime>@EndTime)
return -3

else
 insert into Surgery values(@DoctorId,@SurgeryDate,@StartTime,@EndTime,@SurgeryCategory)
 SET @SurgeryId = IDENT_CURRENT('Surgery')
end try
begin catch
return -4
end catch
end


DECLARE @SurgryId INT, @ReturnValue int
EXEC @ReturnValue = usp_AddSurgeryDetails 1001,'2020-01-01',10.00,12.00,'ANE',@SurgryId
OUT
SELECT @ReturnValue AS ReturnValue, @SurgryId AS SurgeryId

select * from Surgery
```

--2)Procedure 2 Answer

```
create procedure usp_AddDoctorDetails
(
@DoctorName varchar(25),
@SpecializationCode char(3),
@SpecializationName varchar(20)
)
as
begin
begin try
declare @newDoctorId int
if not exists(Select [SpecializationCode] from [dbo].[Specialization] where
[SpecializationCode]=@SpecializationCode)
insert into Specialization values(@SpecializationCode,@SpecializationName)

insert into Doctor values(@DoctorName)
set @newDoctorId=IDENT_CURRENT('Doctor')

insert into DoctorSpecialization values(@newDoctorId,@SpecializationCode,GETDATE())
return 1

end try
begin catch
return -1
end catch
end


DECLARE @ReturnValue INT
EXEC @ReturnValue = usp_AddDoctorDetails 'DR.ABRHAM','LOC','Specialist'
SELECT @ReturnValue AS ReturnValue

select * from DoctorSpecialization

select * from Specialization
```

--3)Function 1 Answer

```
create function ufn_fetchSurveryDetails(
)
returns table
as
return
(select
d.[DoctorName],s.[SpecializationName],sg.[SurgeryDate],sg.[StartTime],sg.[EndTime]
from [dbo].[Doctor] d,[dbo].[Specialization] s,[dbo].[Surgery] sg
where d.[DoctorID]=sg.[DoctorId] and s.SpecializationCode=sg.[SurgeryCategory] and
sg.[SurgeryDate]<GETDATE())


select * from dbo.ufn_fetchSurveryDetails()
```

--4)Function 2 Answer

```
alter function ufn_fetchSurgeryDetails(
@SpecializationName varchar(20),
@SurgeryDate date
)
returns int
begin
declare @result int
select @result=COUNT([SurgeryId]) from Surgery sg,[dbo].[Specialization] s
where s.[SpecializationCode]=sg.[SurgeryCategory] and
SpecializationName=@SpecializationName and SurgeryDate=@SurgeryDate

return @result
end

select dbo.ufn_fetchSurgeryDetails('Anesthesiologiest','2011-01-01')


select * from Surgery
select * from Specialization
```

Database development using SQL Server

Hands-On

Duration: 1 hours **Max Marks: 20**

Instructions to candidates:

1. Read the problem statement provided carefully and implement the solution
2. The coding standards are mandatory wherever applicable
3. You must stop working when the invigilator asks you to do so
4. No additional database object other than those mentioned in the question paper should be created
5. Code submitted with compilation errors will not get evaluated

********** *WISH YOU ALL THE BEST* **********

Coding Standards:

The following coding standards are to be followed for database during the implementation of the system.

Object Type	Guide Lines	Example
Table Name	Pascal casing	Customer
Column Name	Pascal casing	CustomerName, TotalPrice
Constraints	<constraint type>_<column name>	pk_CustomerId, fk_ManagerId
Procedure	usp_<procedure name >	usp_GetCustomerName
Function	ufn_<function name>	ufn_CalculateTax
Literals	Pascal casing	CustomerId

General guidelines on Naming Conventions:

- Procedure / Function / Variable names should not be more than 30 characters long.

Type here to search

Exam Data Submit Help

0 : 59 :

iterals	Pascal casing	CustomerId

General guidelines on Naming Conventions:

- Procedure / Function / Variable names should not be more than 30 characters long.

The coding standards to be followed in data access layer are as follows:

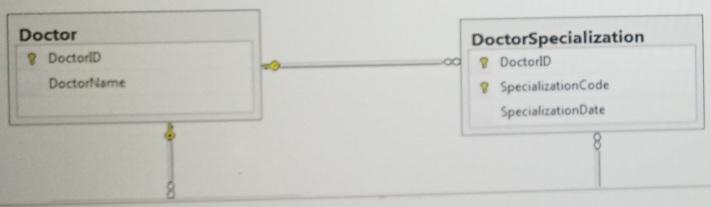
)ject Type	Guide Lines	Examples
riables	Camel casing	qtyOnShelf
ethods	Pascal casing	GetInsertDetails
)perties	Pascal casing	Context

Problem Statement:

"CureHospital" is a well renowned hospital in Sydney. They require an application to plan surgeries based on the specialization and availability of doctors. You have been **provided with table scripts** and are expected to develop the database objects.

DATABASE

Database Design:

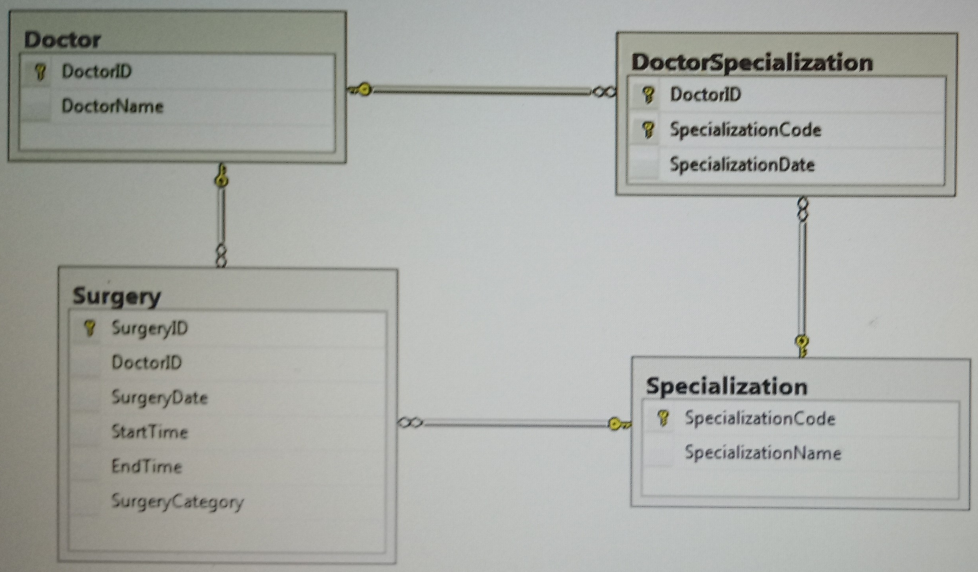


Type here to search

"CureHospital" is a well renowned hospital in Sydney. They require an application to plan surgeries based on the specialization and availability of docto
develop the database objects.

DATABASE

Database Design:



a. **Table Name: Doctor**

This table contains the details of doctors.

Fields:

Field Name	Data Type	Description

Type here to search

a. **Table Name: Doctor**
This table contains the details of doctors.

Fields:

Field Name	Data Type	Description
DoctorID	INT	Primary key, IDENTITY, starts from 1001 and incremented by 1
DoctorName	VARCHAR(25)	Name of the doctor, Mandatory

Sample Data:

DoctorID	DoctorName
1001	Albert
1002	Olivia
1003	Susan

b. **Table Name: Specialization**
This table contains the details of all specializations.

Fields:

Field Name	Data Type	Description
SpecializationCode	CHAR(3)	Primary key
SpecializationName	VARCHAR(20)	Name of the specialization, Mandatory

Sample Data:

SpecializationCode	SpecializationName
GYN	Gynecologist
CAR	Cardiologist

This table contains the details of doctors.

Fields:

Field Name	Data Type	Description
DoctorID	INT	Primary key, IDENTITY, starts from 1001 and incremented by 1
DoctorName	VARCHAR(25)	Name of the doctor, Mandatory

Sample Data:

DoctorID	DoctorName
1001	Albert
1002	Olivia
1003	Susan

b. **Table Name: Specialization**

This table contains the details of all specializations.

Fields:

Field Name	Data Type	Description
SpecializationCode	CHAR(3)	Primary key
SpecializationName	VARCHAR(20)	Name of the specialization, Mandatory

Sample Data:

SpecializationCode	SpecializationName
GYN	Gynecologist
CAR	Cardiologist
ANE	Anesthesiologist

c. **Table Name: DoctorSpecialization**
This table contains the specialization details of doctors.

Fields:

Field Name	Data Type	Description
DoctorID	INT	Should be an existing doctor id in **'Doctor'** table
SpecializationCode	CHAR(3)	Should be an existing specialization code in **'Specialization'** table
SpecializationDate	DATE	Date on which specialization is added, Mandatory

Note: Combination of DoctorID and SpecializationCode is the primary key of the table

Sample Data:

DoctorID	SpecializationCode	SpecializationDate
1001	ANE	2010-01-01
1002	CAR	2010-01-01
1003	CAR	2010-01-01

d. **Table Name: Surgery**
This table contains the details of all the surgeries.

Fields:

Field Name	Data Type	Description
SurgeryID	INT	Primary key, IDENTITY, starts with 5000 and auto-incremented by 1
DoctorID	INT	Should be an existing doctor id in **'Doctor'** table
SurgeryDate	DATE	Date of the surgery, Mandatory

d. Table Name: Surgery

This table contains the details of all the surgeries.

Fields:

Field Name	Data Type	Description
SurgeryID	INT	Primary key, IDENTITY, starts with 5000 and auto-incremented by 1
DoctorID	INT	Should be an existing doctor id in **'Doctor'** table
SurgeryDate	DATE	Date of the surgery, Mandatory
StartTime	DECIMAL(4,2)	Start time of the surgery, Mandatory
EndTime	DECIMAL(4,2)	End time of the surgery, Mandatory
SurgeryCategory	CHAR(3)	Should be an existing specialization code in **'Specialization'** table

Sample Data:

SurgeryID	DoctorID	SurgeryDate	StartTime	EndTime	SurgeryCategory
5000	1001	2011-01-01	09.00	14.00	ANE
5001	1002	2015-01-01	10.00	16.00	CAR

Note: StartTime and EndTime are in HH.MM (24 hours) format.

1. Stored Procedure: usp_AddSurgeryDetails

Create a stored procedure named **usp_AddSurgeryDetails** to insert the surgery details into '**Surgery**' table and provide the newly g
handling.

Input Parameters:

- DoctorID INT
- SurgeryDate DATE

Note: StartTime and EndTime are in HH.MM (24 hours) format

1. **Stored Procedure: usp_AddSurgeryDetails** [1

Create a stored procedure named **usp_AddSurgeryDetails** to insert the surgery details into **'Surgery'** table and provide the newly generated **'SurgeryID'** as OUTPUT parameter. Implement appropriate exception handling.

Input Parameters:

- DoctorID INT
- SurgeryDate DATE
- StartTime DECIMAL(4,2)
- EndTime DECIMAL(4,2)
- SurgeryCategory CHAR(3)

Output Parameters:

- SurgeryID INT

Functionality:

- Check if the **'DoctorID'** exists in **'Doctor'** table
- Check if the combination of **'DoctorID'** and **'SurgeryCategory'** exists in **'DoctorSpecialization'** table
- Check if the doctor is available between the start time and end time for the given surgery date
- If all the validations are successful, insert the data into **'Surgery'** table and set the OUT parameter to the newly generated **'SurgeryID'**
- In all other cases, set the OUT parameter to -1

Return Values:

- 1, in case of successful insertion
- -1, if **'DoctorID'** does not exist in **'Doctor'** table
- -2, if the combination of **'DoctorID'** and **'SurgeryCategory'** does not exist in **'DoctorSpecialization'** table
- -3, if doctor is not available for the given date and time period
- -4, in case of any exception

2. Stored Procedure: usp_AddDoctorDetail

2. Stored Procedure: usp_AddDoctorDetails

Create a stored procedure named **usp_AddDoctorDetails** to insert the details of doctors and their specialization. Implement appropriate exception handling.

Input Parameters:

- DoctorName VARCHAR(25)
- SpecializationCode CHAR(3)
- SpecializationName VARCHAR(20)

Functionality:

- Check if the **'SpecializationCode'** exists in **'Specialization'** table
  - If it does not exist, insert the **'SpecializationCode'** and **'SpecializationName'** in **'Specialization'** table
- Insert the doctor details in the **'Doctor'** table
- Insert the newly generated **'DoctorID'** and the given **'SpecializationCode'** in **'DoctorSpecialization'** table with **'SpecializationDate'** as current date

Return Values:

- 1, in case of successful insertion
- -1, in case of any exception

3. Function: ufn_FetchSurgeryDetails

Create a function named **ufn_FetchSurgeryDetails** to fetch the details of surgeries.

Functionality:

- Fetch the details of all the surgeries which have already been performed *(i.e. **'SurgeryDate'** should be less than today's date).*

Return Values:

A table containing following fields:

DoctorName	SpecializationName	SurgeryDate	StartTime	EndTime

3. Function: ufn_FetchSurgeryDetails

Create a function named **ufn_FetchSurgeryDetails** to fetch the details of surgeries.

Functionality:

- Fetch the details of all the surgeries which have already been performed *(i.e. **'SurgeryDate'** should be less than today's date)*.

Return Values:
A table containing following fields:

DoctorName	SpecializationName	SurgeryDate	StartTime	EndTime

4. Function: ufn_FetchSurgeryCount

Create a function named **ufn_FetchSurgeryCount** to fetch the number of surgeries done for a given specialization on a given date.

Input Parameters:

- SpecializationName VARCHAR(20)
- SurgeryDate DATE

Functionality:

- Fetch the number of the surgeries done from 'Surgery' table for a given specialization on a given date.

Return Value:
An integer value specifying the count of surgeries

********** *END* **********