# Database Basics (MSSQL) Demo Exam

# Colonial Journey

2000 years from now, the known space is colonized by the human race. However, the four Citadel Council races are planning to populate new home worlds in the SoftUnia Galaxy as part of a strategy called the SoftUnia Initiative. 20000 citizens are send aboard space transportation vessels. The Council has asked you to create a Colonization Management system so they can keep track of the colonists' journeys trough the stars.

## Database Overview

You have given an Entity / Relationship Diagram of the CJMS Database:



The **ColonialJourney** Database holds information about colonists, their travel cards, information about the journeys, types of space vessels and destination planets. Your task is to create a database called **ColonialJourney**. Then you will have to create several **tables**.

* **Planets** – contains information about **planets**.
* **Spaceports** – contains information about **space ports**.
* **Spaceships –** contains information about **space ships**.
* **Colonists –** contains information about **colonists**.
* **Journeys –** contains information about **journeys**.
* **TravelCards –** contains information about **travel cards**.

Make sure you implement the whole database correctly on your local machine, so that you could work with it.

The instructions you are given will be the minimal needed for you to implement the database.

# Section 1. DDL (30 pts)

You have been tasked to create the tables in the database by the following models:

### Planets

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Constraints** |
| Id | **Integer,** from **1** to **2,147,483,647.** | Unique table **identificator**, **Identity** |
| Name | **String** up to 30 symbols. Non Unicode | **NULL** is **not** allowed |

### Spaceports

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Constraints** |
| Id | **Integer,** from **1** to **2,147,483,647.** | Unique table **identificator**, **Identity** |
| Name | **String** up to 50 symbols. Non Unicode | **NULL** is **not** allowed |
| PlanetId | **Integer,** from **1** to **2,147,483,647.** | **NULL** is **not** allowed, Relationship with table Planets |

### Spaceships

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Constraints** |
| Id | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| Name | **String** up to 50 symbols. Non Unicode | **NULL** is **not** allowed |
| Manufacturer | **String** up to 30 symbols. Non Unicode | **NULL** is **not** allowed |
| LightSpeedRate | **Integer** from **0** to **2,147,483,647** | Has a **default value** of 0. |

### Colonists

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Constraints** |
| Id | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| FirstName | **String** up to 20 symbols. Non Unicode | **NULL** is **not** allowed |
| LastName | **String** up to 20 symbols. Non Unicode | **NULL** is **not** allowed |
| Ucn | **String** up to 10 symbols. Non Unicode | **NULL** is **not** allowed **UNIQUE** values. |
| BirthDate | **Date** | **NULL** is **not** allowed |

### Journeys

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Constraints** |
| Id | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| JourneyStart | **DateTime** | **NULL** is **not** allowed |
| JourneyEnd | **DateTime** | **NULL** is **not** allowed |
| Purpose | **String** up to 11 symbols. Non Unicode | Should **only** contain one of the following purposes: “**Medical**”, “**Technical**”, “**Educational**”, “**Military**” |
| DestinationSpaceportId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table Spaceports. |
| SpaceshipId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table Spaceships |

### TravelCards

|  |  |  |
| --- | --- | --- |
| **Column** | **Data Type** | **Constraints** |
| Id | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| CardNumber | A **string** containing exactly **10 characters**.Non Unicode | **NULL** is **not** allowed **UNIQUE** values. |
| JobDuringJourney | **String** up to 8 symbols. Non Unicode | Should **only** contain one of the following jobs: “**Pilot**”, “**Engineer**”, “**Trooper**”, “**Cleaner**”, “**Cook**” |
| ColonistId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table Colonists |
| JourneyId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table Journeys |

**SOLUTION:**

CREATE TABLE Planets

(

Id INT PRIMARY KEY IDENTITY(1,1),

[Name] VARCHAR(30) NOT NULL

)

CREATE TABLE Spaceports

(

Id INT PRIMARY KEY IDENTITY(1,1),

[Name] VARCHAR(50) NOT NULL,

PlanetId INT FOREIGN KEY REFERENCES Planets(Id) NOT NULL

)

CREATE TABLE Spaceships

(

Id INT PRIMARY KEY IDENTITY(0,1),

[Name] VARCHAR(50) NOT NULL,

Manufacturer VARCHAR(30) NOT NULL,

LightSpeedRate INT DEFAULT 0

)

CREATE TABLE Colonists

(

Id INT PRIMARY KEY IDENTITY(0,1),

FirstName VARCHAR(20) NOT NULL,

LastName VARCHAR(20) NOT NULL,

Ucn VARCHAR(10) UNIQUE NOT NULL,

BirthDate DATE NOT NULL

)

CREATE TABLE Journeys

(

Id INT PRIMARY KEY IDENTITY(0,1),

JourneyStart DATETIME NOT NULL,

JourneyEnd DATETIME NOT NULL,

Purpose VARCHAR(11) CHECK

(Purpose = 'Medical' OR

Purpose = 'Technical' OR

Purpose = 'Educational' OR

Purpose = 'Military'),

DestinationSpaceportId INT FOREIGN KEY REFERENCES Spaceports(Id) NOT NULL,

SpaceshipId INT FOREIGN KEY REFERENCES Spaceships(Id) NOT NULL

)

CREATE TABLE TravelCards

(

Id INT PRIMARY KEY IDENTITY(0,1),

CardNumber CHAR(10) UNIQUE NOT NULL,

JobDuringJourney VARCHAR(8) CHECK

(

JobDuringJourney = 'Pilot' OR

JobDuringJourney = 'Engineer' OR

JobDuringJourney = 'Trooper' OR

JobDuringJourney = 'Cleaner' OR

JobDuringJourney = 'Cook'

) ,

ColonistId INT FOREIGN KEY REFERENCES Colonists(Id) NOT NULL,

JourneyId INT FOREIGN KEY REFERENCES Journeys(Id) NOT NULL

)

## Database Design

Submit all of yours **create** **statements** to the **Judge** system.

# Section 2. DML (10 pts)

**Before you start, you must import “**DataSet-ColonialJourney.sql**”. If you have created the structure correctly, the data should be successfully inserted without any errors.**

In this section, you have to do some data manipulations:

## Insert

**Insert** sample data into the database. Write a query to add the following records into the corresponding tables. **All Ids should be auto-generated**.

**SOLUTION:**

INSERT INTO Planets ([Name])

VALUES

('Mars'),

('Earth'),

('Jupiter'),

('Saturn')

INSERT INTO Spaceships ([Name], Manufacturer, LightSpeedRate)

VALUES

('Golf', 'VW', 3),

('WakaWaka', 'Wakanda', 4),

('Falcon9', 'SpaceX', 1),

('Bed', 'Vidolov', 6)

**Planets**

|  |
| --- |
| **Name** |
| Mars |
| Earth |
| Jupiter |
| Saturn |

**Spaceships**

|  |  |  |
| --- | --- | --- |
| **Name** | **Manufacturer** | **LightSpeedRate** |
| Golf | VW | 3 |
| WakaWaka | Wakanda | 4 |
| Falcon9 | SpaceX | 1 |
| Bed | Vidolov | 6 |

## Update

Update all spaceships light speed rate with 1where the **Id** is between **8** and **12.**

**SOLUTION:**

UPDATE Spaceships

SET LightSpeedRate += 1

WHERE Id BETWEEN 8 AND 12

## Delete

Delete first three inserted **Journeys** (be careful with the relationships).

**SOLUTION:**

DELETE FROM TravelCards

WHERE JourneyId IN (1,2,3)

DELETE FROM Journeys

WHERE Id IN (1,2,3)

# Section 3. Querying (40 pts)

**You need to start with a fresh dataset, so recreate your DB and import the sample data again (**DataSet-ColonialJourney.sql**).**

## Select all travel cards

Extract from the database, all **travel cards.** Sort the results by **card number ascending.**

### Required Columns

* **CardNumber**
* **JobDuringJourney**

### Example

|  |  |
| --- | --- |
| **CardNumber** | **JobDuringJourney** |
| **0032031181** | Engineer |
| 0037637193 | Engineer |
| ... | ... |

**SOLUTION:**

SELECT CardNumber, JobDuringJourney

FROM TravelCards

ORDER BY CardNumber

## Select all colonists

Extract from the database, all **colonists.** Sort the results by **first name**, them by **last name**, and finally by **id** in **ascending** order.

### Required Columns

* **Id**
* **FullName**
* **Ucn**

### Example

|  |  |  |
| --- | --- | --- |
| **Id** | **FullName** | **Ucn** |
| 35 | Aigneis McConville | 9225403496 |
| 92 | Althea Kelinge | 9998159318 |
| ... | ... | ... |

**SOLUTION:**

SELECT Id, CONCAT(FirstName, ' ', LastName) AS FullName, Ucn

FROM Colonists

ORDER BY FirstName, LastName, Id

## Select all military journeys

Extract from the database, all **Military** journeys. Sort the results **ascending** by **journey start.**

### Required Columns

* **Id**
* **JourneyStart**
* **JourneyEnd**

### Example

|  |  |  |
| --- | --- | --- |
| **Id** | **JourneyStart** | **JourneyEnd** |
| 7 | 04/01/2019 | 09/12/2049 |
| 3 | 21/02/2019 | 03/01/2049 |
| ... | ... | ... |

**SOLUTION:**

SELECT Id,

CONVERT( VARCHAR, JourneyStart, 103) AS JourneyStart,

CONVERT( VARCHAR, JourneyEnd, 103 ) AS JourneyEnd

FROM Journeys

WHERE Purpose = 'Military'

ORDER BY JourneyStart

## Select all pilots

Extract from the database all colonists, which have a **pilot job.** Sort the result by **id, ascending.**

### `Required Columns

* **Id**
* **FullName**

### Example

|  |  |
| --- | --- |
| **id** | **full\_name** |
| 6 | Clark Cowan |
| 18 | Wald Bim |
| ... | ... |

**SOLUTION:**

SELECT c.Id AS id, CONCAT(c.FirstName, ' ', c.LastName) AS full\_name

FROM Colonists AS c

JOIN TravelCards AS tc ON tc.ColonistId = c.Id

WHERE tc.JobDuringJourney = 'Pilot'

ORDER BY c.Id

## Count colonists

Count all colonists that are on **technical journey.**

### Required Columns

* **Count**

### Example

|  |
| --- |
| **count** |
| 16 |

**SOLUTION:**

SELECT COUNT(\*) AS [count] FROM Colonists AS c

JOIN TravelCards AS tc ON tc.ColonistId = c.Id

GROUP BY tc.JobDuringJourney

HAVING tc.JobDuringJourney = 'Engineer'

## Select the fastest spaceship

Extract from the database the fastest **spaceship** and its destination **spaceport name**. In other words, the ship with the **highest** light speed rate.

### Required Columns

* **SpaceshipName**
* **SpaceportName**

### Example

|  |  |
| --- | --- |
| **SpaceshipName** | **SpaceportName** |
| SSE Priestess | Yggdrasil Station |

**SOLUTION:**

SELECT TOP(1) s.[Name] AS SpaceshipName,

sp.[Name] AS SpaceportName

FROM Spaceships AS s

JOIN Journeys AS j ON s.Id = j.SpaceshipId

JOIN Spaceports AS sp ON j.DestinationSpaceportId = sp.Id

ORDER BY s.LightSpeedRate DESC

## Select spaceships with pilots younger than 30 years

Extract from the database those **spaceships**, which have pilots, **younger** than 30 years old. In other words, 30 years from 01/01/2019. Sort the results **alphabetically** by spaceship **name**.

### Required Columns

* **Name**
* **Manufacturer**

**SOLUTION:**

SELECT DISTINCT s.[Name], s.Manufacturer FROM Spaceships AS s

JOIN Journeys AS j ON j.SpaceshipId = s.Id

JOIN TravelCards AS tc ON tc.JourneyId = j.Id

JOIN Colonists AS c ON tc.ColonistId = c.Id

WHERE DATEPART(YEAR, c.BirthDate) > 1988 AND tc.JobDuringJourney = 'Pilot'

ORDER BY s.[Name]

### Example

|  |  |
| --- | --- |
| **Name** | **Manufacturer** |
| Anarchy | Fivebridge |
| ... | ... |

## Select all educational mission planets and spaceports

Extract from the database names of all **planets** and their **spaceports**, which have **educational** missions. Sort the results by **spaceport name** in **descending** order.

### Required Columns

* **PlanetName**
* **SpaceportName**

### Example

|  |  |
| --- | --- |
| **PlanetName** | **SpaceportName** |
| Kascarth | Yggdrasil Station |
| Lescore | Tartarus |
| ... | ... |

**SOLUTION:**

SELECT p.[Name] AS PlanetName , sp.[Name] AS SpaceportName FROM Planets AS p

JOIN Spaceports AS sp ON sp.PlanetId = p.Id

JOIN Journeys AS j ON j.DestinationSpaceportId = sp.Id

WHERE j.Purpose = 'Educational'

ORDER BY sp.[Name] DESC

## Select all planets and their journey count

Extract from the database all **planets’ names** and their **journeys count**. Order the results by journeys **count**, **descending** and by **planet name ascending**.

### Required Columns

* **PlanetName**
* **JourneysCount**

### Example

|  |  |
| --- | --- |
| **PlanetName** | **JourneysCount** |
| Otroyphus | 4 |
| Eipra | 2 |
| ... | ... |

**SOLUTION:**

SELECT p.[Name] AS PlanetName , COUNT(j.Id) AS JourneysCount FROM Planets AS p

JOIN Spaceports AS sp ON sp.PlanetId = p.Id

JOIN Journeys AS j ON j.DestinationSpaceportId = sp.Id

GROUP BY p.[Name]

ORDER BY JourneysCount DESC , p.[Name]

## Select the shortest journey

Extract from the database the **shortest** **journey,** its destination **spaceport** **name**, **planet name** and **purpose**.

### Required Columns

* **Id**
* **PlanetName**
* **SpaceportName**
* **JourneyPurpose**

### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **PlanetName** | **SpaceportName** | **JourneyPurpose** |
| 3 | Casmadus | Minerva Station | Military |

**SOLUTION:**

SELECT TOP(1) j.Id,

p.[Name] AS PlanetName,

sp.[Name] AS SpaceportName,

j.Purpose AS JourneyPurpose

FROM Journeys AS j

JOIN Spaceports AS sp ON j.DestinationSpaceportId = sp.Id

JOIN Planets AS p ON sp.PlanetId = p.Id

ORDER BY DATEDIFF(SECOND, JourneyStart, JourneyEnd )

## Select the less popular job

Extract from the database the **less popular job** in the **longest journey**. In other words, the job with less assign colonists.

**SOLUTION:**

SELECT TOP(1) r.Id, r.JobDuringJourney FROM

( SELECT

j.Id, tc.JobDuringJourney ,

MAX(DATEDIFF(SECOND, JourneyStart, JourneyEnd)) AS Duration,

COUNT(tc.JobDuringJourney) AS [Count],

DENSE\_RANK()OVER(PARTITION BY j.Id ORDER BY COUNT(tc.JobDuringJourney))

AS Ranking

FROM TravelCards AS tc

JOIN Journeys AS j ON tc.JourneyId = j.Id

GROUP BY j.Id, tc.JobDuringJourney, JourneyStart, JourneyEnd ) AS r

ORDER BY r.Duration DESC , r.[Count]

### Required Columns

* **JourneyId**
* **JobName**

### Example

|  |  |
| --- | --- |
| **JourneyId** | **JobName** |
| 7 | Engineer |

## Select Second Oldest Important Colonist

Find all colonists and their job during journey with rank 2. Keep in mind that all the selected colonists with rank 2 must be the oldest ones. You can use ranking over their job during their journey.

### Required Columns

* **JobDuringJourney**
* **FullName**
* **JobRank**

### Example

|  |  |  |
| --- | --- | --- |
| **JobDuringJourney** | **FullName** | **JobRank** |
| Cleaner | Hale O'Doireidh | 2 |
| Cook | Laurie Askin | 2 |
| … | … | … |

**SOLUTION:**

SELECT \* FROM (

SELECT tc.JobDuringJourney ,

CONCAT(c.FirstName, ' ', c.LastName) AS FullName , DENSE\_RANK()OVER(PARTITION BY tc.JobDuringJourney

ORDER BY c.BirthDate) AS JobRank

FROM Colonists AS c

JOIN TravelCards AS tc ON tc.ColonistId = c.Id ) AS f

WHERE f.JobRank = 2

ORDER BY f.JobDuringJourney

## Planets and Spaceports

Find all planets and all of their spaceports. Select planet name and the count of the spaceports. Sort them by spaceports count (descending), then by name (ascending).

### Required Columns

* **Name**
* **Count**

### Example

|  |  |
| --- | --- |
| **Name** | **Count** |
| Kascarth | 4 |
| Jeayama | 3 |
| … | … |

**SOLUTION:**

SELECT p.[Name], COUNT(sp.Id) AS [Count] FROM Planets AS p

LEFT JOIN Spaceports AS sp ON sp.PlanetId = p.Id

GROUP BY p.[Name]

ORDER BY COUNT(sp.Id) DESC, p.[Name]

# Section 4. Programmability (20 pts)

## Get Colonists Count

Create a **user defined function** with the name **dbo.udf\_GetColonistsCount(PlanetName VARCHAR (30))** that receives **planet name** and returns the count of all colonists sent to that planet.

### Example

|  |  |
| --- | --- |
| **Query** | |
| SELECT dbo.udf\_GetColonistsCount('Otroyphus') | |
| PlanetName | Count |
| **Otroyphus** | **35** |

**SOLUTION:**

CREATE FUNCTION udf\_GetColonistsCount(@PlanetName VARCHAR(30))

RETURNS INT

AS

BEGIN

RETURN ( SELECT COUNT(\*) FROM Planets AS p

JOIN Spaceports AS sp ON p.Id = sp.PlanetId

JOIN Journeys AS j ON j.DestinationSpaceportId = sp.Id

JOIN TravelCards AS tc ON tc.JourneyId = j.Id

GROUP BY p.[Name]

HAVING p.[Name] = @PlanetName )

END

## Change Journey Purpose

Create a **user defined stored procedure**, named **usp\_ChangeJourneyPurpose(@JourneyId, @NewPurpose)**, that receives an **journey id** and **purpose**, and attempts to **change the purpose of that journey**. An purpose will only be changed if all of these conditions **pass**:

* If the **journey id** doesn’t exists, then it **cannot be changed.** **Raise an error** with the message “The journey does not exist!”
* If the **journey** has already that purpose, **raise an error** with the message “You cannot change the purpose!”

If all the above conditions pass, **change the purpose of that journey**.

### Example

|  |  |
| --- | --- |
| **Query** | **Output** |
| **EXEC usp\_ChangeJourneyPurpose 1, 'Technical'** **SELECT \* FROM Journeys** | **998**  **2455** |
| **EXEC usp\_ChangeJourneyPurpose 2, 'Educational'** | **You cannot change the purpose!** |
| **EXEC usp\_ChangeJourneyPurpose 196, 'Technical'** | **The journey does not exist!** |

**SOLUTION:**

CREATE PROCEDURE usp\_ChangeJourneyPurpose

(@JourneyId INT, @NewPurpose VARCHAR(20))

AS

BEGIN

IF((SELECT Id FROM Journeys

WHERE Id = @JourneyId) IS NULL )

BEGIN

SELECT 'The journey does not exist!'

END

ELSE

BEGIN

IF((SELECT Purpose FROM Journeys

WHERE Id = @JourneyId ) = @NewPurpose )

BEGIN

SELECT 'You cannot change the purpose!'

END

ELSE

BEGIN

ALTER TABLE Journeys

DROP CONSTRAINT CK\_\_Journeys\_\_Purpos\_\_1B0907CE

UPDATE Journeys

SET Purpose = @NewPurpose

WHERE Id = @JourneyId

END

END

END